Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 219 (2006) 912–929
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A second order accurate projection method for the incompressible Navier–Stokes equations on non-graded adaptive grids

Chohong Min [a], Frédéric Gibou [b,c,*]

[a] Mathematics Department, University of California, Santa Barbara, CA 93106, United States
[b] Mechanical Engineering Department, University of California, Santa Barbara, CA 93106, United States
[c] Computer Science Department, University of California, Santa Barbara, CA 93106, United States

Received 13 November 2005; received in revised form 1 July 2006; accepted 4 July 2006
Available online 11 September 2006

Abstract

We present an unconditionally stable second order accurate projection method for the incompressible Navier–Stokes equations on non-graded adaptive Cartesian grids. We employ quadtree and octree data structures as an efficient means to represent the grid. We use the supra-convergent Poisson solver of [C.-H. Min, F. Gibou, H. Ceniceros, A supra-convergent finite difference scheme for the variable coefficient Poisson equation on fully adaptive grids, CAM report 05-29, J. Comput. Phys. (in press)], a second order accurate semi-Lagrangian method to update the momentum equation, an unconditionally stable backward difference scheme to treat the diffusion term and a new method that guarantees the stability of the projection step on highly non-graded grids. We sample all the variables at the grid nodes, producing a scheme that is straightforward to implement. We propose two and three-dimensional examples to demonstrate second order accuracy for the velocity field and the divergence free condition in the L^1 and L^∞ norms.

1. Introduction

The incompressible Navier–Stokes equations describe the motion of fluid flows and are therefore used in countless applications in science and engineering. In non-dimensional form these equations read

$$U_t + (U \cdot \nabla)U + \nabla p = \mu \Delta U + F \quad \text{in } \Omega,$$
$$\nabla \cdot U = 0 \quad \text{in } \Omega,$$
$$U|_{\partial\Omega} = U_b \quad \text{on } \partial\Omega,$$

* Corresponding author. Address: Mechanical Engineering Department, University of California, Santa Barbara, CA 93106, United States. Tel.: +1 7230338.
E-mail address: fgibou@engineering.ucsb.edu (F. Gibou).

0021-9991/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.07.019

where p is the pressure, F is the sum of the external forces and μ is the viscosity coefficient. Ω represents the domain in which the velocity field U is to be found and $\partial\Omega$ denotes the boundary of the domain, where the velocity field can be prescribed. In this paper, we consider the case where $U \cdot n = 0$ on $\partial\Omega$. These equations lack an evolution equation for pressure, which thus only plays a role in ensuring that the velocity field is divergence free. As a consequence, most numerical methods in the primitive variables are fractional methods, i.e. they first solve the momentum equation ignoring the effects of pressure, and then project the velocity onto the divergence free vector space. Starting with the seminal work of Chorin [8], several projection methods have been introduced, see e.g. the work of Kim and Moin [17], Kan [16], Bell et al. [3] and the references therein. The MAC grid configuration [14] used in finite volume methods, where the pressure is stored at the cells' center and where the velocity components are stored at their respective cells' faces, is often the preferred arrangement. This is mainly due to the fact that it produces methods that offer a straightforward mechanism to enforce *discretely* the incompressibility condition $\nabla \cdot u = 0$. However, other arrangements have been shown to produce high order accurate schemes for the velocity field, without enforcing the incompressibility condition at the discrete level (see e.g. the work of E et al. [10], Almgren et al. [2], the review by Brown et al. [6] and the references therein).

Physical phenomena have differences in length scales and numerical approximations on uniform grids are in such cases extremely inefficient in terms of C.P.U. and memory requirement. This stems from the fact that only a small fraction of the domain needs high grid resolution to correctly approximate the solution, while other parts of the domain can produce accurate solutions on coarser grids (for example in regions where the solution experiences smooth variations). As a consequence adaptive mesh refinement strategies, starting with the work of Berger and Oliger [5] for compressible flows, have been proposed in order to concentrate the computational effort where it is most needed. In the original work of Berger et al. [5,4], a fine Cartesian grid is hierarchically embedded into a coarser grid. Almgren et al. [1] then introduced a projection method for the variable density incompressible Navier–Stokes equations on nested grids. Sussman et al. extended this method to two-phase flows [30]. Within this block structured grid approach, a multigrid approach was used to efficiently solve the Poisson equation. The methods on quadtrees/octrees presented in [21,20,22,24] build one linear system of equations that was solved with standard iterative linear solvers [25].

One of the main difficulties in solving the Navier–Stokes equations on irregular grids is in solving the Poisson equation associated with the incompressibility condition. Rather recently, Popinet [24] introduced a Navier–Stokes solver using an octree data structure. In this work, the discretization of the Poisson equation at one cell's center involves cells that are not necessarily adjacent to it. As a consequence, a non-symmetric linear system of equations was obtained and graded octrees only were considered in order to ease the implementation. In this case the linear system was efficiently solved using a multigrid method. Later, Losasso et al. [21] introduced a symmetric discretization of the Poisson equation in the context of free surface flows. In this case, the discretization at one cell's center only involves adjacent cells, therefore producing a symmetric linear system of equations, which is straightforward to solve with a standard preconditioned conjugate gradient method. Moreover, this method is straightforward to implement and does not require any constraint on the grid. This approach produces first order accurate solutions in the case of a non-graded adaptive mesh and is found to be second order accurate in the case of a graded mesh. In this case, the pressure fluxes defined at the faces are the same for a large cell and its adjacent smaller cells. Using ideas introduced in [19], Losasso et al. then extended this method to second order accuracy. In [22], Min et al. introduced a second order accurate method to solve the Poisson equation on non-graded adaptive grids as well. A hallmark of this approach is that the solution's gradients are found to second order accuracy as well. In this case, the linear system is non-symmetric but is proven to be diagonally dominant. In this paper, we propose a second order accurate finite difference Navier–Stokes solver on non-graded adaptive grids, making use of the Poisson solver introduced in [22].

2. Spatial discretization

The physical domain in two (resp. three) spatial dimensions is discretized into squares (resp. cubes), and we use a standard quadtree (resp. octree) data structure to represent this partitioning. For example, consider the case depicted in Fig. 1 in the case of two spatial dimensions: The root of the tree is associated with the entire domain that is then split into four cells of equal sizes, called the children of the root. The discretization

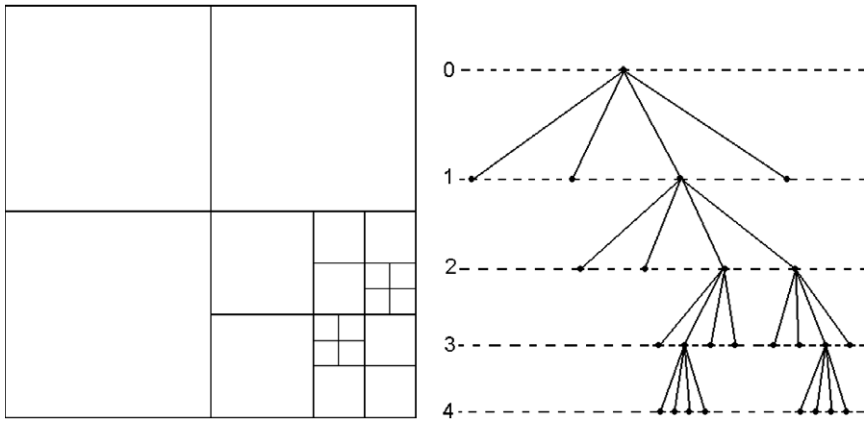

Fig. 1. Discretization of a two-dimensional domain (left) and its quadtree representation (right). The entire domain corresponds to the root of the tree (level 0). Then each cell can be recursively subdivided further to four children. In this example, this tree is ungraded, since the difference of level between cells exceeds one.

proceeds recursively, i.e. each cell can be in turn split into four children until the desired level of detail is achieved. In three spatial dimensions, the domain (root) is split in eight cubes (children) and each cell can be recursively split in the same manner. We refer the interested reader to the books of Samet [27,26] for more details on quadtree/octree data structures.

The level of a cell is set to be zero if it is associated with the root and is incremented by one for each new generation of children. A tree in which the difference of level between adjacent cells is at most one is called a graded tree. Meshes associated with graded trees are often used in the case of finite element methods in order to produce procedures that are easier to implement. In [24], Popinet also uses a graded grid to simplify the finite difference formulas associated with his discretizations. As a consequence, such methods may introduce extra grid cells in regions where they are not necessarily needed, consuming some computational resources that cannot be spent elsewhere, eventually limiting the highest level of detail that can be achieved. In fact, Moore [23] demonstrates that the cost of transforming an arbitrary quadtree into a graded quadtree could involve 8 times as many grid nodes in the worst case. Weiser [31] proposed a rough estimate for the three-dimensional case and concluded that as much as 71 times as many grid nodes could be needed for balancing octrees in the worst case. Even more important than the amount of grid cells necessary, the ease of implementation is a factor to consider. In this work, we do not impose any constraint on the difference of level between two adjacent cells in the proposed method, allowing for a non-graded adaptive mesh generation.

3. Numerical methods

In this section, we present an unconditionally stable second order accurate projection method for the incompressible Navier–Stokes equations. All the variables are stored at the nodes, producing a scheme that is straightforward to implement. We use the quadtree and octree data structures described in Section 2 and we allow for non-graded adaptive grids, hence removing the difficulties associated with grid generations. We use the supra-convergent Poisson solver of Min et al. [22], a second order accurate semi-Lagrangian method to update the momentum equation and an unconditionally stable backward difference scheme to treat the diffusion term.

3.1. Second order accurate semi-lagrangian method

Semi-Lagrangian schemes are extensions of the Courant–Isaacson–Rees [9] method for hyperbolic equations. They are unconditionally stable and therefore allow for large time steps, which is a particularly desirable feature in an adaptive setting since for standard explicit schemes the time step restriction imposed by the CFL

condition is proportional to the smallest grid cell. The general idea behind semi-Lagrangian methods is to reconstruct the solution by integrating numerically the equation along characteristic curves, starting from any grid point x_i and tracing back the departure point x_d in the upwind direction. Interpolation formulas are then used to recover the value of the solution at such points. Consider for example the linear advection equation

$$\phi_t + U \cdot \nabla \phi = 0,$$

where U is an externally generated velocity field (i.e. does not depend on ϕ). Then $\phi^{n+1}(x_i) = \phi^n(x_d)$, where x_i is any grid point and x_d is the corresponding departure point from which the characteristic curve originates from. In this work, we use the following second order explicit mid-point rule for locating the departure point, as in [32]

$$\hat{x} = x^{n+1} - \frac{\Delta t}{2} \cdot U^n(x^{n+1}),$$
$$x_d^n = x^{n+1} - \Delta t \cdot U^{n+\frac{1}{2}}(\hat{x}),$$

where we define the velocity at the mid time step $t^{n+1/2}$ as a linear combination of the velocities at the two previous time steps t^n and t^{n-1}, i.e. $U^{n+\frac{1}{2}} = \frac{3}{2}U^n - \frac{1}{2}U^{n-1}$.

Since $\hat{x}$ is not guaranteed to be on a grid node, a procedure must be provided to interpolate the value of $U^{n+\frac{1}{2}}(\hat{x})$ from the values of $U^{n+1/2}$ defined at the nodes. Likewise, $\phi^n(x_d^n)$ must be interpolated from the values of ϕ^n defined at the nodes. Piecewise multilinear interpolation schemes on non-uniform grids are often used in conjunction with semi-Lagrangian methods (see e.g. [21,28]). In this work, we use a quadratic Hermite interpolation [18], which is constructed from the solution's values at nine distinct nodes in two spatial dimensions (27 in three spatial dimensions). Since the local structure of a non-uniform cell is arbitrary, we use the four children of the parent cell to select a uniform grid of 3×3 nodes in two spatial dimensions ($3 \times 3 \times 3$ in three spatial dimensions) as illustrated in Fig. 2. Similarly, the discretization of the momentum equation in the projection method of Section 3.6 requires the definition of x_d^{n-1}, which is given by

$$\hat{x} = x^{n+1} - \Delta t \cdot U^n(x^{n+1}),$$
$$x_d^{n-1} = x^{n+1} - 2\Delta t \cdot U^n(\hat{x}).$$

3.2. Basic finite differences on non-uniform Cartesian grids

We derived in Min et al. formulas to obtain second order accurate discretizations for the first order derivatives and first order accurate discretizations for the second order derivatives on a non-uniform mesh. In order to discretize the derivatives in one direction, these formulas use the derivatives in the transversal directions to

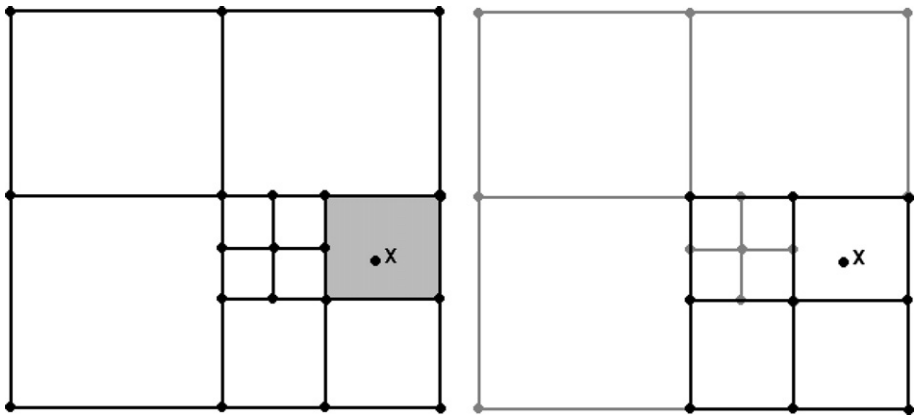

Fig. 2. Quadratic interpolation in quadtree: the shaded cell is the smallest cell containing the location x where the data must be interpolated at. The parent cell of the shaded cell has a 3×3 locally uniform grid that enables a straightforward quadratic Hermite interpolation.

increase the accuracy. Here, we recall the formulas and refer the interested reader to [22] for more details in the derivation.

3.2.1. Two spatial dimensions

Consider a node v_0 in a two-dimensional non-uniform grid as depicted in Fig. 3. Denoting $f_i = f(v_i)$, the discretizations for the first and second order derivatives in the x direction are given by

$$D_x f(v_0) = \widetilde{D}_x f(v_0) - \frac{s_1 s_5 s_6}{2 s_4 (s_1 + s_4)} D_{yy} f(v_0)$$

and

$$D_{xx} f(v_0) = \widetilde{D}_{xx} f(v_0) - \frac{s_5 s_6}{s_4 (s_1 + s_4)} D_{yy} f(v_0),$$

where $\widetilde{D}_x$ and $\widetilde{D}_{xx}$ are given by the standard finite difference approximations for the first and second order derivatives

$$\widetilde{D}_x f(v_0) = \frac{f_4 - f_0}{s_4} \frac{s_1}{s_1 + s_4} + \frac{f_0 - f_1}{s_1} \frac{s_4}{s_1 + s_4}$$

and

$$\widetilde{D}_{xx} f(v_0) = \frac{f_4 - f_0}{s_4} \frac{2}{s_1 + s_4} - \frac{f_0 - f_1}{s_1} \frac{2}{s_1 + s_4},$$

where $f(v_4)$ is defined by linear average between $f(v_5)$ and $f(v_6)$. The y-direction is treated similarly.

3.2.2. Three spatial dimensions

Consider a node v_0 in a three-dimensional non-uniform grid as depicted in Fig. 4. The discretizations for the first and second order derivatives are given by

$$D_x f(v_0) = \widetilde{D}_x f(v_0) + \frac{s_7 s_8}{2} \frac{s_1}{s_4 (s_1 + s_4)} D_{zz} f(v_0),$$

$$D_y f(v_0) = \widetilde{D}_y f(v_0) + \frac{s_{10} s_{11}}{2} \frac{s_2}{s_5 (s_2 + s_5)} D_{zz} f(v_0) + \frac{s_9 s_{12}}{2} \frac{s_2}{s_5 (s_2 + s_5)} D_{xx} f(v_0),$$

$$D_{xx} f(v_0) = \widetilde{D}_{xx} f(v_0) - \frac{s_7 s_8}{s_4 (s_1 + s_4)} D_{zz} f(v_0),$$

$$D_{yy} f(v_0) = \widetilde{D}_{yy} f(v_0) - \frac{s_{10} s_{11}}{s_5 (s_2 + s_5)} D_{zz} f(v_0) - \frac{s_9 s_{12}}{s_5 (s_2 + s_5)} D_{xx} f(v_0),$$

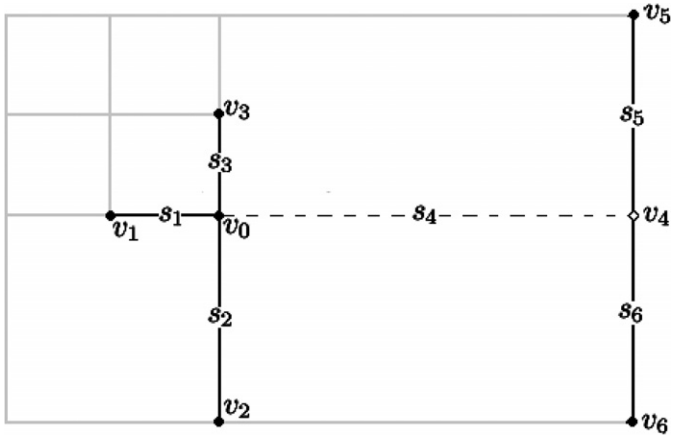

Fig. 3. Local structure around a node v_0 in a quadtree mesh: at most one node in the two Cartesian directions might not exist. In this case, we define a ghost node (here v_4) to be used in the discretizations.

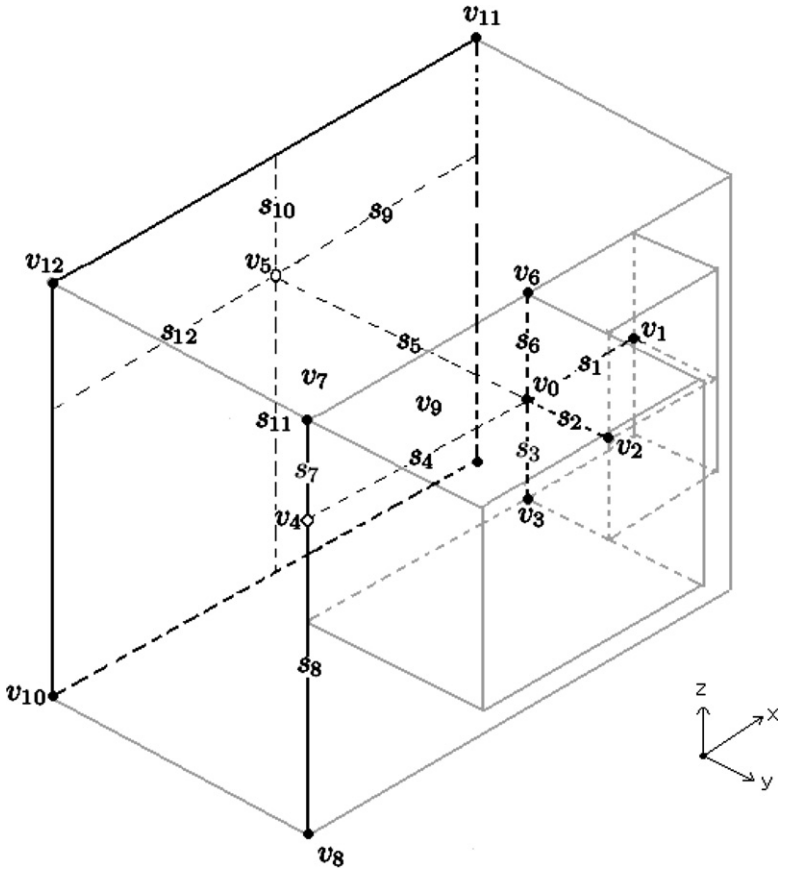

Fig. 4. Neighboring vertices of a vertex three spatial dimensions.

where $\widetilde{D}_x$, $\widetilde{D}_y$, $\widetilde{D}_{xx}$ and $\widetilde{D}_{yy}$ are given by

$$\widetilde{D}_x f(v_0) = \frac{f_4 - f_0}{s_4}\frac{s_1}{s_1 + s_4} + \frac{f_0 - f_1}{s_1}\frac{s_4}{s_1 + s_4},$$

$$\widetilde{D}_y f(v_0) = \frac{f_2 - f_0}{s_2}\frac{s_5}{s_2 + s_5} + \frac{f_0 - f_5}{s_5}\frac{s_2}{s_2 + s_5},$$

$$\widetilde{D}_{xx} f(v_0) = \frac{f_4 - f_0}{s_4}\frac{2}{s_1 + s_4} - \frac{f_0 - f_1}{s_1}\frac{2}{s_1 + s_4}$$

and

$$\widetilde{D}_{yy} f(v_0) = \frac{f_2 - f_0}{s_2}\frac{2}{s_2 + s_5} - \frac{f_0 - f_5}{s_5}\frac{2}{s_2 + s_5},$$

with

$$f(v_4) = \frac{s_7 f_8 + s_8 f_7}{s_7 + s_8}$$

and

$$f(v_5) = \frac{s_{11}s_{12}f_{11} + s_{11}s_9 f_{12} + s_{10}s_{12}f_9 + s_{10}s_9 f_{10}}{(s_{10} + s_{11})(s_9 + s_{12})}.$$

3.3. Interpolation procedures

Some reserve must be provided to define data anywhere in a cell, for example in order to use semi-Lagrangian methods (see Section 3.1). As pointed out in Strain [29], the most natural choice of interpolation in quadtree (resp. octree) data structures is the piecewise bilinear (resp. trilinear) interpolation: Consider a cell C with dimensions $[0,1]^2$, the bilinear interpolation at a point $x \in C$ using the values at the nodes reads

$$\begin{aligned}\phi(x,y) = {} & \phi(0,0)(1-x)(1-y) \\ & + \phi(0,1)(1-x)(\quad y) \\ & + \phi(1,0)(\quad x)(1-y) \\ & + \phi(1,1)(\quad x)(\quad y).\end{aligned} \tag{1}$$

Quadratic interpolation can also easily be constructed using the data from the parent cell: since the parent cell of any current cell of a quadtree (resp. octree) owns 2×2 children cells (resp. $2\times 2\times 2$) and 3×3 nodes (resp. $3\times 3\times 3$), one can defined the Hermite quadratic interpolation on the parent cell. For example in the case of a cell $[-1,1]^2$ in a quadtree, we can define the Hermite interpolation as

$$\begin{aligned}\phi(x,y) = {} & \phi(-1,-1)\frac{x(x-1)}{2}\frac{y(y-1)}{2} + \phi(0,-1)(x^2-1)\frac{y(y-1)}{2} + \phi(1,-1)\frac{x(x+1)}{2}\frac{y(y-1)}{2} \\ & + \phi(-1,0)\frac{x(x-1)}{2}(y^2-1) + \phi(0,0)(x^2-1)(y^2-1) + \phi(1,0)\frac{x(x+1)}{2}(y^2-1) \\ & + \phi(-1,1)\frac{x(x-1)}{2}\frac{y(y+1)}{2} + \phi(0,1)(x^2-1)\frac{y(y+1)}{2} + \phi(1,1)\frac{x(x+1)}{2}\frac{y(y+1)}{2}.\end{aligned}$$

However, this interpolation procedure is ill-advised in the case of high Reynolds number flows since such flows present rapid change in velocity and Hermite interpolations overshoot the data. We therefore prefer to define a quadratic interpolation by correcting Eq. (1) using second order derivatives. For a cell $[0,1]^2$, we have

$$\begin{aligned}\phi(x,y) = {} & \phi(0,0)(1-x)(1-y) \\ & + \phi(0,1)(1-x)(\quad y) \\ & + \phi(1,0)(\quad x)(1-y) \\ & + \phi(1,1)(\quad x)(\quad y) - \phi_{xx}\frac{x(1-x)}{2} - \phi_{yy}\frac{y(1-y)}{2},\end{aligned} \tag{2}$$

where we define

$$\begin{aligned}\phi_{xx} &= \min_{v\in \text{vertices}(C)}(|D^0_{xx}\phi_v|), \\ \phi_{yy} &= \min_{v\in \text{vertices}(C)}(|D^0_{yy}\phi_v|).\end{aligned} \tag{3}$$

3.4. Hodge decomposition

Projection methods are based on the Hodge decomposition that states that a vector field U^* on a domain Ω with $U^*\cdot n = 0$ on the domain's boundary $\partial\Omega$ can be uniquely decomposed into the sum of a divergence-free vector field U and a gradient field $\nabla\phi$ satisfying

$$\begin{aligned}&U^* = U + \nabla\phi, \\ &\nabla\cdot U = 0, \\ &U\cdot\nabla\phi = 0.\end{aligned}$$

In this section, we propose a numerical implementation of the Hodge decomposition that guarantees L^2 stability for the projection step: we sample the variables U^*, U, ϕ and $\nabla\phi$ at the nodes of the grid and approximate the gradient and the divergence operators with the second order finite differences described in Section 3.2, which we denote by G and D, respectively. For standard projection methods, one assumes that

$$U^* = U + G\phi,$$
$$DU = 0.$$

Taking a divergence of the above equation defines ϕ through the solution of the Laplace equation

$$DG\phi = DU^*.$$

However, this approximation of the Laplace operators (DG) decouples the solution into even and odd nodes, which is well known to produce spurious errors. For this reason, instead of using DG one prefers the standard Laplace discretization. In our case, we use the approximation described in Section 3.2, which we denote by L. As a result, we obtain $\widetilde{\phi}$ defined by

$$L\widetilde{\phi} = DU^*.$$

Note that $\widetilde{\phi}$ is different from ϕ, and $U^* = U + G\widetilde{\phi}$ is not a Hodge decomposition anymore at the discrete level. Since DG and L are both second order approximations of the Laplace operator, one traditionally uses $\widetilde{\phi}$ and defines

$$P_{\text{appr}}(U^*) = U^* - G\widetilde{\phi}.$$

However, an alternative approach is to impose the orthogonality property between U and $G\widetilde{\phi}$ in order to preserve the L^2 stability of the projection. We thus define

$$P_{\text{orth}}(U^*) = U^* - \frac{U^* \cdot G\widetilde{\phi}}{G\widetilde{\phi} \cdot G\widetilde{\phi}} G\widetilde{\phi},$$

where the inner product of two functions f and g is computed cell-wise by multiplying the average value for $f \times g$ using the nodes of the cell with the volume of the cell.

In the both cases, DP_{appr} and DP_{orth} are not zero at the discrete level, but near zero within the truncation error. This is typical of the approximate projection and is the source of the divergence free constraint being satisfied within the truncation error (finite difference) instead of exactly (finite volume). Likewise, $P_{\text{appr}}(U^*) \cdot G\widetilde{\phi}$ is approximately satisfied at the discrete level, whereas $P_{\text{orth}}(U^*) \cdot G\widetilde{\phi} = 0$ is satisfied *exactly*, thus guaranteeing the following L^2 stability for the projection step:

$$\|U^*\|_2^2 = \|P_{\text{orth}}(U^*)\|_2^2 + \|U^* \cdot G\widetilde{\phi}\|_2^2,$$
$$\|U^*\|_2^2 \geqslant \|P_{\text{orth}}(U^*)\|_2^2.$$

Remark:

- We show in the example section that the approximate projection does not have such a stability property.
- In the case where $U \cdot n \neq 0$, the splitting $U^* = U + \nabla\phi$ is not an orthogonal decomposition as mentioned in [7,12] so it is not clear how to extend the orthogonal projection presented above to this case. We are currently investigating this issue.

3.5. Neumann boundary condition

Let $Ax = b$ denoted the linear system associated with the Poisson equation in the projection step. Since a Neumann boundary condition makes the linear system singular, we consider the following augmented matrix [15]:

$$\begin{pmatrix} A & r \\ r^{\mathrm{T}} & 0 \end{pmatrix} \begin{pmatrix} x \\ \alpha \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix},$$

where r denotes the right null eigenvector of A. In this case, r is the constant unit vector. Though A is singular, the augmented matrix is not. As mentioned in [13], the preconditioning of the augmented matrix follows from that of A: let M be a preconditioner of A, one defines

$$\widetilde{M} = \begin{pmatrix} M & r \\ r^{\mathrm{T}} & 0 \end{pmatrix} \simeq \begin{pmatrix} A & r \\ r^{\mathrm{T}} & 0 \end{pmatrix}.$$

The inverse of $\widetilde{M}$ can be exactly calculated as

$$\widetilde{M}^{-1}\begin{pmatrix} f \\ \beta \end{pmatrix} = \begin{pmatrix} M^{-1}f - \frac{r\cdot M^{-1}f-\beta}{r\cdot M^{-1}r}\cdot M^{-1}r \\ \frac{r\cdot M^{-1}f-\beta}{r\cdot M^{-1}r} \end{pmatrix}. \tag{4}$$

In the example section, we chose M to be the symmetric Gauss–Seidel preconditioner, i.e. $M = LD^{-1}U$, where L, D and U denote the lower, diagonal and upper parts of A, respectively. Note that L, D and U are all invertible since A is an M-matrix. The stabilized Bi-Conjugate Gradient method was very effective in our calculations.

3.6. Projection method for the Navier–Stokes equations

Consider the momentum equation

$$U_t + (U\cdot\nabla)U + \nabla p = \mu\Delta U + F.$$

The Crank–Nicholson scheme has often been used for discretizing implicitly the viscosity term [3,17]. However, in the case where the convection term is treated with a semi-Lagrangian method, a difficulty arises: The corresponding pressure is not defined at the grid nodes, making the projection step slightly more complicated to implement in conjunction with a Crank–Nicholson scheme. The backward differentiation formula offers a more convenient choice, since in this case the corresponding pressure is defined *at* the grid nodes [32]. The discretization of the momentum equation using a backward differentiation formula and a semi-Lagrangian method for the convection term can be written as

$$\frac{1}{\Delta t}\left(\frac{3}{2}U^{n+1} - 2U_{\mathrm{d}}^{n} + \frac{1}{2}U_{\mathrm{d}}^{n-1}\right) + \nabla p^{n+1} = \mu\Delta U^{n+1} + F^{n+1}.$$

This equation is solved using the pressure-free three-step projection method approach of Brown et al. [6]: first, given the velocity field U^n at time t^n, an intermediate velocity U^* is calculated by ignoring the pressure component

$$\frac{1}{\Delta t}\left(\frac{3}{2}U^{*} - 2U_{\mathrm{d}}^{n} + \frac{1}{2}U_{\mathrm{d}}^{n-1}\right) = \mu LU^{*} + F^{n+1}.$$

Second, in order for the velocity U^{n+1} at time t^{n+1} to satisfy the incompressibility condition $\nabla\cdot U^{n+1} = 0$ the second step defines a potential function $\widetilde{\phi}^{n+1}$ through the solution of the following Poisson equation:

$$L\widetilde{\phi}^{n+1} = DU^{*} \tag{5}$$

In the last step, the fluid velocity U^{n+1} at the new time step is projected to the divergence free field

$$U^{n+1} = P_{\mathrm{appr}}(U^{*}) = U^{*} - G\widetilde{\phi},$$

or

$$U^{n+1} = P_{\mathrm{orth}}(U^{*}) = U^{*} - \frac{U^{*}\cdot G\widetilde{\phi}}{G\widetilde{\phi}\cdot G\widetilde{\phi}}G\widetilde{\phi}.$$

Following the approach of [6,17], the following boundary conditions for U^* and ϕ^{n+1} are sufficient to ensure second order accuracy for the velocity field:

$$N\cdot U^{*}|_{\partial\Omega} = N\cdot U^{n+1}|_{\partial\Omega},$$
$$N\cdot G\phi|_{\partial\Omega} = 0,$$
$$T\cdot U^{*}|_{\partial\Omega} = T\cdot U^{n+1}|_{\partial\Omega} + \begin{cases} T\cdot G\widetilde{\phi}^{n} & \text{if } U^{n+1} = P_{\mathrm{appr}}(U^{*}), \\ \frac{U^{*}\cdot G\widetilde{\phi}^{n}}{G\widetilde{\phi}^{n}\cdot G\widetilde{\phi}^{n}}T\cdot G\widetilde{\phi}^{n} & \text{if } U^{n+1} = P_{\mathrm{orth}}(U^{*}). \end{cases}$$

where N and T denote the normal and tangent vectors at the boundary, respectively.

We note that the first step of the projection method computes the intermediate velocity U^* by solving the following convection-diffusion equation:

$$\left(\frac{3}{2}Id - \Delta t\mu L\right)U^* = 2U_{\mathrm{d}}^n - \frac{1}{2}U_{\mathrm{d}}^{n-1} + \Delta t F^{n+1} \tag{6}$$

with Dirichlet boundary conditions at the domain's boundary.

In order to solve the Poisson equations (5) and (6), we build the corresponding linear systems using the central difference formulas for D_{xx}, D_{yy} and D_{zz} presented in Section 3.2. The solutions obtained with this linear system are second order accurate with second order accurate gradients as demonstrated in Min et al. [22]. The linear system obtained from the discretization of Eq. (5) with Neumann boundary conditions is singular. Thus, in this case, we use the augmented matrix described in Section 3.5. Both linear systems are solved using the BiCGSTAB method with the symmetric Gauss–Seidel preconditioner [25].

We also note that we use a starting routine to guess the initial value $G\widetilde{\phi}^0$ as described in Brown et al. [6]. All the derivatives are computed using the formulas of Section 3.2.

4. Examples

In this section, we present numerical evidences that the proposed projection method yields second order accuracy for the velocity field and the divergence free condition in the L_1 and the L_∞ norms. All the examples were tested on highly arbitrary grids to demonstrate that this scheme is applicable to non-graded adaptive grids. In the first example, we show the instability of the approximate projection on irregular grids and the stability of the proposed orthogonal projection. For this reason, all of our examples use the orthogonal projection.

4.1. Stability of the orthogonal projection

In order to demonstrate the stability of the projection methods, we consider a domain $\Omega = [0,\pi]^2$ and a vector field $U^* = (u^*, v^*)$ with $U^* \cdot n = 0$ on $\partial\Omega$ with

$$u^*(x,y) = \sin(x)\cos(y) + x(\pi - x)y^2\left(\frac{y}{3} - \frac{\pi}{2}\right),$$
$$v^*(x,y) = -\cos(x)\sin(y) + y(\pi - y)x^2\left(\frac{x}{3} - \frac{\pi}{2}\right).$$

This vector field can be written as $U^* = U + \nabla\phi$, where U is a divergence-free vector field and $\phi = \left(\frac{x^3}{3} - \frac{\pi x^2}{2}\right)\left(\frac{y^3}{3} - \frac{\pi y^2}{2}\right)$. We iteratively apply the approximate and the orthogonal projection methods on the highly non-graded depicted in Fig. 5, i.e. we successively compute $U^n_{\mathrm{appr}} = P^n_{\mathrm{appr}}(U^*)$ or $U^n_{\mathrm{orth}} = P^n_{\mathrm{orth}}(U^*)$. Fig. 6 depicts the results. In particular, note the monotonic decrease of $\|U^n_{\mathrm{orth}}\|_{L^2}$ as expected from Section 3.4.

4.2. Single vortex in two spatial dimensions

Consider a domain $\Omega = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]^2$ and a single vortex flow with a viscosity coefficient $\mu = 1$ and an exact solution of

$$u(x,y,t) = -\cos(x)\sin(y)\cos(t),$$
$$v(x,y,t) = \sin(x)\cos(y)\cos(t),$$
$$p(x,y,t) = -\frac{1}{4}\cos^2(t)(\cos(2x) + \cos(2y)).$$

We use the grid depicted in Fig. 7 and impose Dirichlet boundary conditions on the domain's boundary. We emphasize that the difference of level between some cells and their neighbors exceeds one, demonstrating the ability of our method to produce second order accurate solutions on arbitrary grids. The time step is chosen as $\Delta t = 5 \times \Delta x_s$, where Δx_s is the size of the finest grid cell and the final time is $t = \pi$. Table 1 demonstrates the

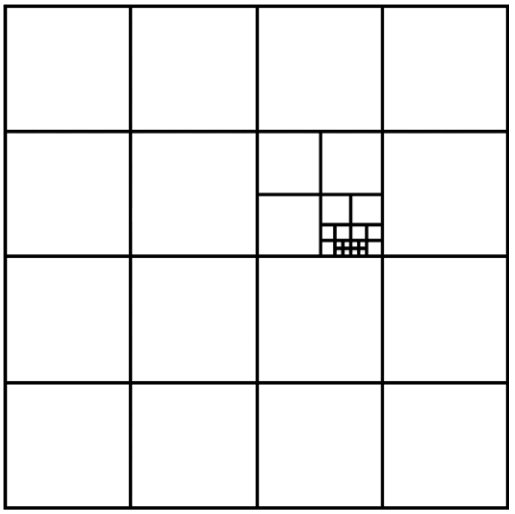

Fig. 5. Highly non-graded grid used in example 4.1.

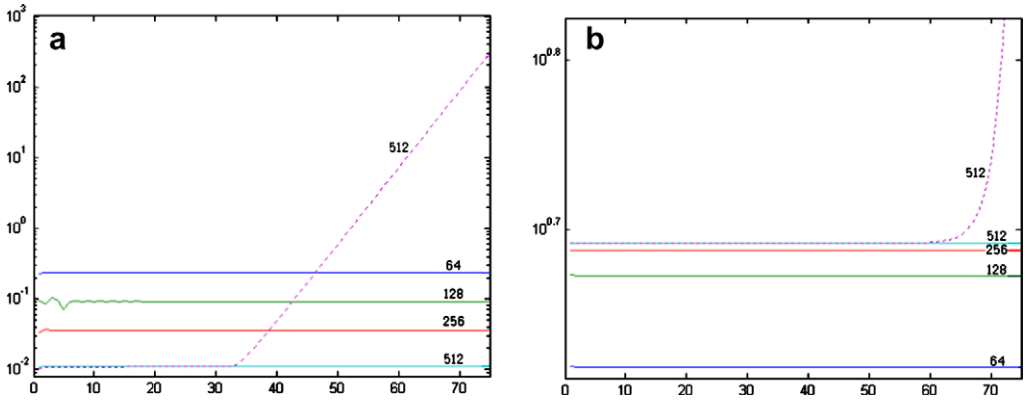

Fig. 6. Left: $\|U^n_{\text{orth}} - U\|_\infty$ (solid) and $\|U^n_{\text{appr}} - U\|_\infty$ (dotted). Right: $\|U^n_{\text{orth}}\|_2$ (solid) and $\|U^n_{\text{appr}}\|_2$ (dotted). The finest resolutions of the quadtrees are 64^2, 128^2, 256^2 and 512^2.

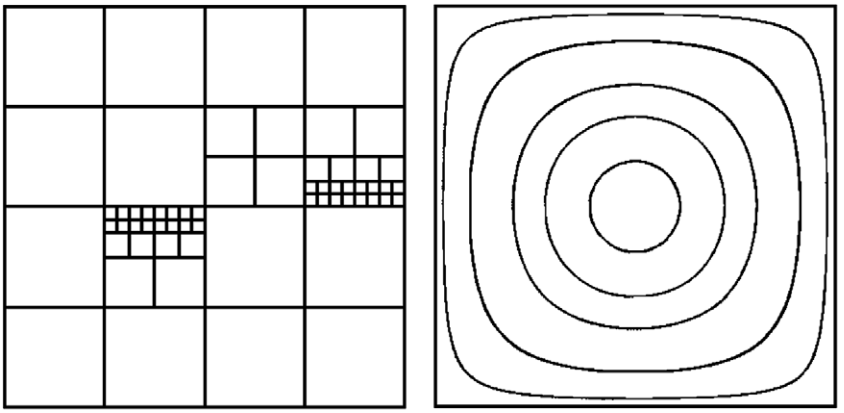

Fig. 7. Arbitrarily generated quadtree (left) and streamlines of the numerical solution (right) for example 4.2.

Table 1
Accuracy of the x-component of velocity and accuracy of the divergence free condition in the L^2 and L^∞ norms for example 4.2

Grid resolution (min–max)	x-Component of U				Divergence of U			
	L_2	Rate	L_∞	Rate	L_2	Rate	L_∞	Rate
4^2–32^2	6.26E−3		6.11E−2		1.78E−2		2.07E−1	
8^2–64^2	1.87E−3	1.73	1.49E−2	2.03	4.31E−3	2.05	5.08E−2	2.02
16^2–128^2	3.38E−4	2.47	2.47E−3	2.59	8.72E−4	2.30	1.16E−2	2.13
32^2–256^2	6.46E−5	2.38	4.42E−4	2.48	1.77E−4	2.29	3.36E−3	1.78
64^2–512^2	1.59E−5	2.01	1.74E−4	1.34	4.14E−5	2.10	9.24E−4	1.86

second order accuracy of the x-component of the velocity field as well as the accuracy of the divergence free condition in the L^2 and L^∞ norms.

4.3. Driven cavity

We test our Navier–Stokes solver on the well-known driven cavity problem of Ghia et al. [11]: Consider a domain $\Omega = [0,1]^2$, with the top wall moving with unit velocity, We impose no-slip boundary conditions on the

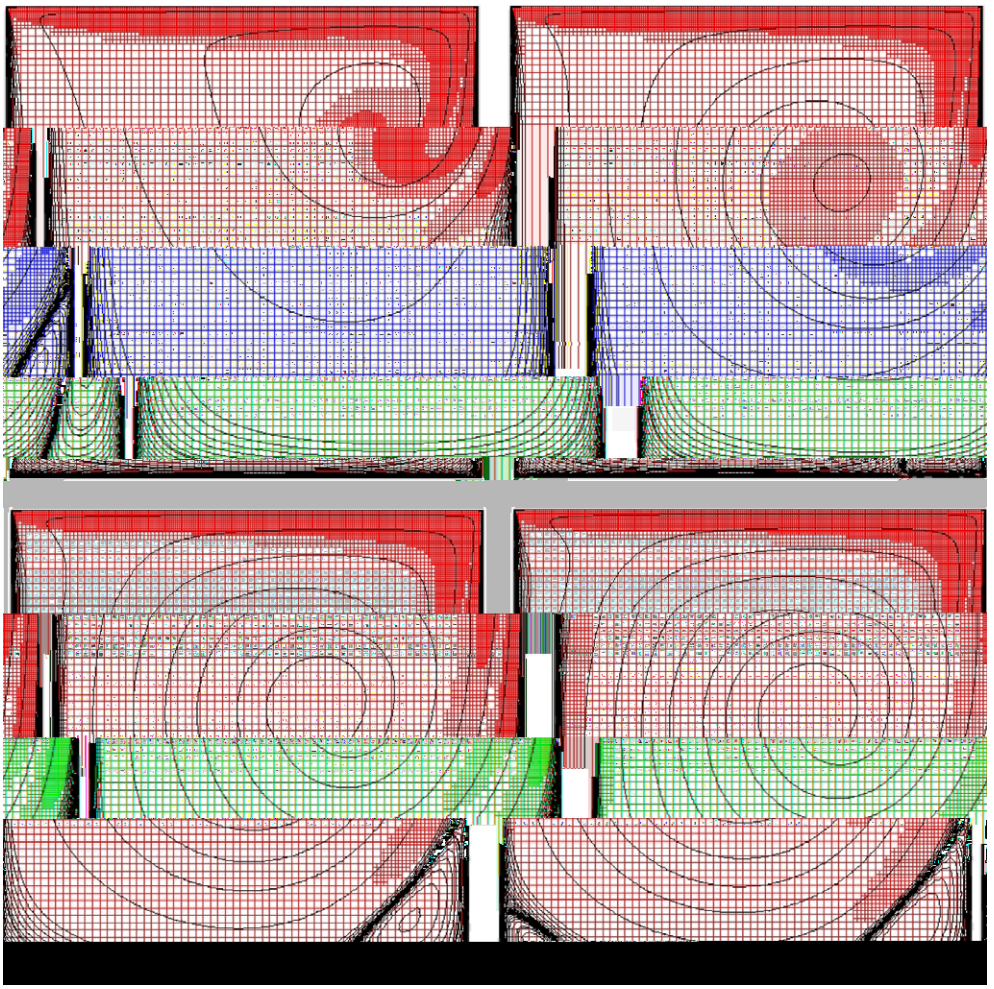

Fig. 8. Adaptive grids for the driven cavity example. From top to bottom and left to right: $t = 3.12$, 7.50, 13.75 and 37.50. The coarsest grid has level 6, and the finest has level 8.

four walls. In this example, we take a Reynolds number $Re = 1000$, i.e. the viscosity coefficient $\mu = 1/1000$. The criterion for mesh refinement we use is that proposed in [24], i.e. a cell C is refined whenever

$$\min(\Delta x, \Delta y)\frac{\max_{x\in C}\|\nabla \times U\|_2}{\max_{x\in\Omega}\|U\|_2} > \tau, \tag{7}$$

where τ is a chosen threshold taken to be 0.04. More precisely, consider a grid structure G^n at time t^n on which the velocity field is updated from U^n to U^{n+1}. The grid G^{n+1} at t^{n+1} is constructed in the following way: first, we compute $\nabla \times U^{n+1}$ at every nodes of G^n using the second order central difference formulas of Section 3.2. Second, starting from the root of G^{n+1} split the cell if (7) is satisfied using $\max_{x\in C}\|\nabla \times U^{n+1}\|_2 = \max_v\|(\nabla \times U^{n+1})(v)\|_2$, where v is a node of G^n and $v \in C$ and $\max_{x\in\Omega}\|U\|_2$ is computed by taking the maximum of $\|U\|_2$ over all nodes of G^n. Finally, U^{n+1} is defined on the new grid G^{n+1} from the values of U^{n+1} on G^n using the quadratic interpolation described in Section 3.3.

Fig. 8 depict the evolution of the streamlines and of the adaptive grid until steady state, while Fig. 9 demonstrates the convergence of the velocity at steady state to the benchmark solution of [11].

4.4. Three spatial dimensions

Consider a domain $\Omega = [-\frac{\pi}{2}, \frac{\pi}{2}]^3$ and a flow with viscosity $\mu = 1$ and with an exact solution defined by

$$\begin{aligned}
u(x, y, z, t) &= -2\cos(t)\cos(x)\sin(y)\sin(z),\\
v(x, y, z, t) &= \cos(t)\sin(x)\cos(y)\sin(z),\\
w(x, y, z, t) &= \cos(t)\sin(x)\sin(y)\cos(z),\\
p(x, y, z, t) &= \frac{1}{4}\cos^2(t)(2\cos(2x) + \cos(2y) + \cos(2z)).
\end{aligned}$$

The time step is chosen as $\Delta t = 5 \times \Delta x_s$, where Δx_s is the size of the finest grid cell and we run the simulation up to a final time of $t = \pi$. Fig. 10 depicts the grid used. In particular, the level difference between some cells and their neighbors is larger than one, illustrating the ability of our method to retain second order accuracy on non-graded adaptive grids. Table 2 demonstrates the second order accuracy of the x-component of the velocity field as well as the accuracy of the divergence free condition in the L^2 and L^∞ norms.

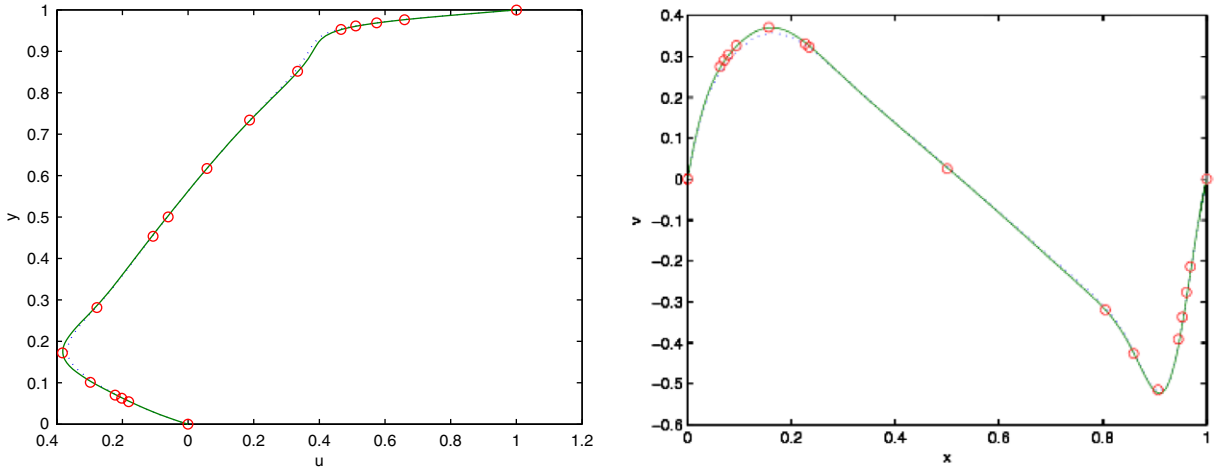

Fig. 9. x and y component of the velocity field in the driven cavity example of Ghia et al. [11]. The domain is $[0, 1]$ and we take $Re = 1000$ and a time step of $\Delta t = 2\Delta x_s$, where Δx_s is the size of the smallest grid cell. The symbols are the experiment results of [11], the dotted line depicts the numerical results obtained with an adaptive quadtree with level ranging from 6 to 8, the solid line depicts the numerical results obtained with an adaptive quadtree with level ranging from 7 to 9.

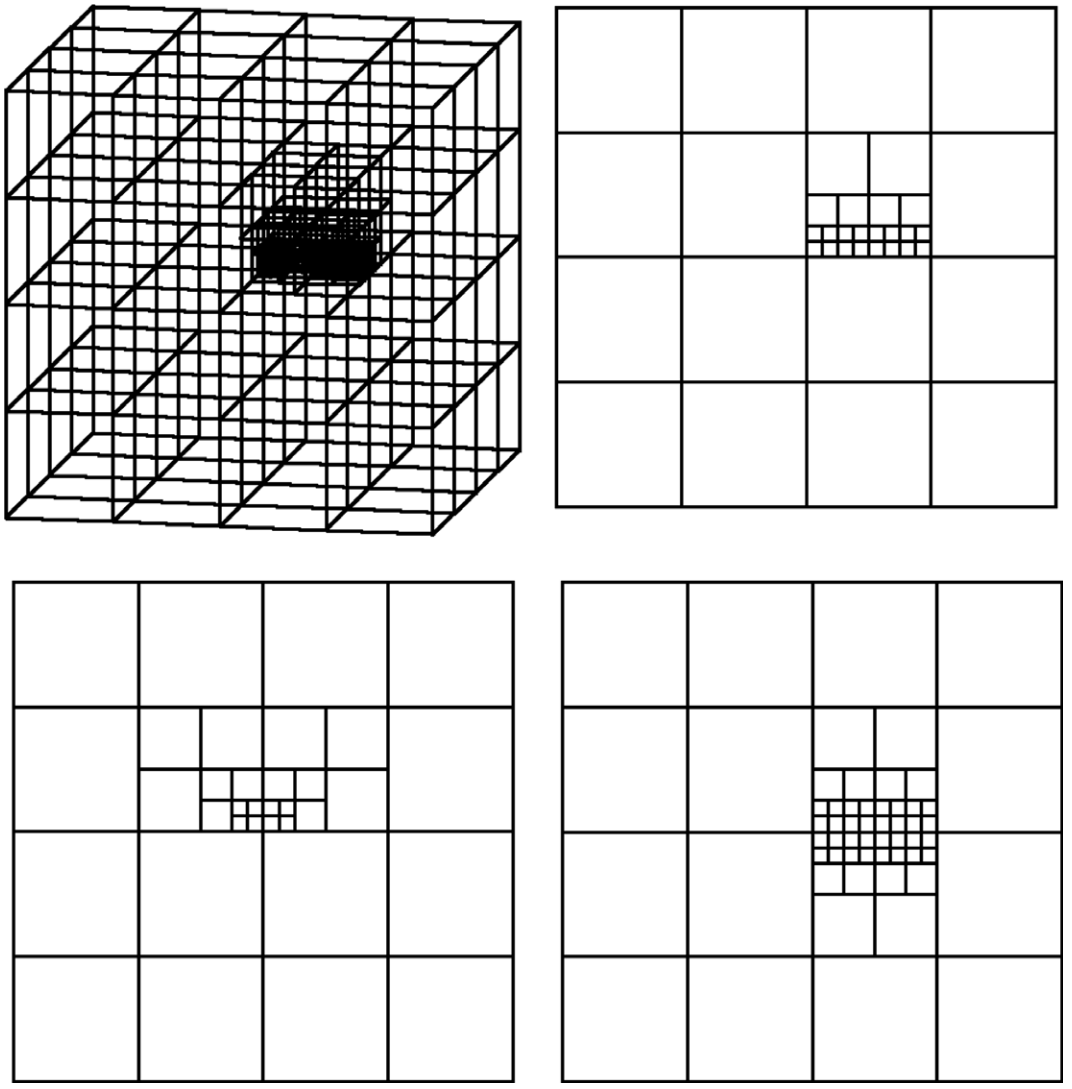

Fig. 10. From top to bottom and from left to right: Arbitrarily generated three-dimensional grid used in example 4.4, its front view, side view and top view. In particular, note that the difference of level between adjacent grid cells can exceed one.

Table 2
Accuracy of the x-component of velocity and accuracy of the divergence free condition in the L^2 and L^∞ norms for example 4.4

Grid resolution (min–max)	x-Component of U				Divergence of U			
	L_2	Rate	L_∞	Rate	L_2	Rate	L_∞	Rate
4^3–32^3	9.77E−3		6.52E−2		4.47E−2		3.45E−1	
8^3–64^3	3.19E−3	1.61	1.82E−2	1.84	9.65E−3	2.21	1.24E−1	1.47
16^3–128^3	5.85E−4	2.44	3.66E−3	2.31	2.00E−3	2.26	4.70E−2	1.40
32^3–256^3	1.23E−4	2.24	7.02E−4	2.38	3.27E−4	2.61	1.39E−2	1.75

4.5. Refinement test

As pointed out in [1], one important issue of adaptive grids is whether a refinement of one region improves the accuracy of the region and not worsen the accuracy of other regions. As noted in [1,24], in some cases, a refinement of one region not only worsen the other regions, but also the region itself. Here we test this issue for

the fluid solver presented in this article. To prevent the diffusion effect on error term, we consider an inviscid solution from [1]

$$u = 1 - 2\cos(2\pi(x-t))\sin(2\pi(y-t)),$$
$$v = 1 - 2\sin(2\pi(x-t))\cos(2\pi(y-t)),$$
$$p = -\cos(4\pi(x-t)) - \cos(4\pi(y-t)).$$

Consider a computational domain of $[0,1]^2$ with periodic boundary condition. A small square with dimensions $1/4 \times 1/4$ and bottom left corner located at $(1/4, 1/4)$ is patched with a grid with finest resolution, while the rest of the domain is patched with a grid with coarser resolution. We denote by r the level difference between the two regions. The calculations are run until $t = 0.5$ with CFL numbers of 0.75 for Tables 3 and 2 for Table 4. The results in the two tables demonstrate that the fluid solver proposed in this article improves the accuracy in the small square region, as well as the other regions. Our results in Table 3 show that the error are larger than those obtained in [1,24], but decrease with the refinement, while the errors in [1,24] increase.

Table 3
Convergence rate for example 4.5 with $CFL = 0.75$

Grid resolution (min–max)	x-Component of U in patch				x-Component of U in Ω			
	L_2	Rate	L_∞	Rate	L_2	Rate	L_∞	Rate
32^2	4.05E−2		7.40E−2		5.51E−2		1.24E−1	
64^2	9.63E−3	2.07	1.64E−2	2.16	1.37E−2	2.00	3.09E−2	2.00
128^2	2.34E−3	2.04	4.17E−3	1.98	3.37E−3	2.02	7.62E−3	2.01
32^2–64^2	2.64E−2		5.31E−2		3.97E−2		7.80E−2	
64^2–128^2	6.14E−3	2.10	1.30E−2	2.02	9.25E−3	2.10	1.72E−2	2.17
128^2–256^2	1.46E−3	2.07	3.33E−3	1.97	2.18E−3	2.08	3.89E−3	2.14
32^2–128^2	2.09E−2		4.48E−2		3.13E−2		6.04E−2	
64^2–256^2	5.18E−3	2.01	1.27E−2	1.81	7.75E−3	2.01	1.40E−2	2.10
128^2–512^2	1.27E−3	2.02	3.39E−3	1.90	1.84E−3	2.06	3.47E−3	2.01

Table 4
Convergence rate for example 4.5 with $CFL = 2$

Grid resolution (min–max)	x-Component of U in patch				x-Component of U in Ω			
	L_2	Rate	L_∞	Rate	L_2	Rate	L_∞	Rate
32^2	1.93E−1		4.36E−1		2.02E−1		4.60E−1	
64^2	4.74E−2	2.02	1.10E−1	1.97	5.60E−2	1.85	1.18E−1	1.95
128^2	1.16E−2	2.02	2.79E−2	1.98	1.47E−2	1.92	3.16E−2	1.95
32^2–64^2	5.10E−2		1.05E−1		6.12E−2		1.36E−1	
64^2–128^2	1.31E−2	1.95	2.75E−2	1.93	1.66E−2	1.88	3.83E−2	1.82
128^2–256^2	3.22E−3	2.03	6.64E−3	2.05	4.15E−3	2.00	9.70E−3	1.98
32^2–128^2	3.07E−2		6.23E−2		4.71E−2		9.98E−2	
64^2–256^2	7.17E−3	2.09	1.50E−2	2.05	1.06E−2	2.14	2.15E−2	2.21
128^2–512^2	1.74E−3	2.04	3.85E−3	1.96	2.51E−3	2.08	4.88E−3	2.14

Table 5
Convergence rate for example 4.6

Grid resolution (min–max)	x-Component of U			
	L_2	Rate	L_∞	Rate
16^2–64^2	1.17E−2		1.68E 0	
32^2–128^2	5.31E−2	1.14	9.68E−1	0.79
64^2–256^2	1.79E−2	1.57	3.40E−1	1.51
128^2–512^2	4.38E−3	2.03	8.17E−2	2.05

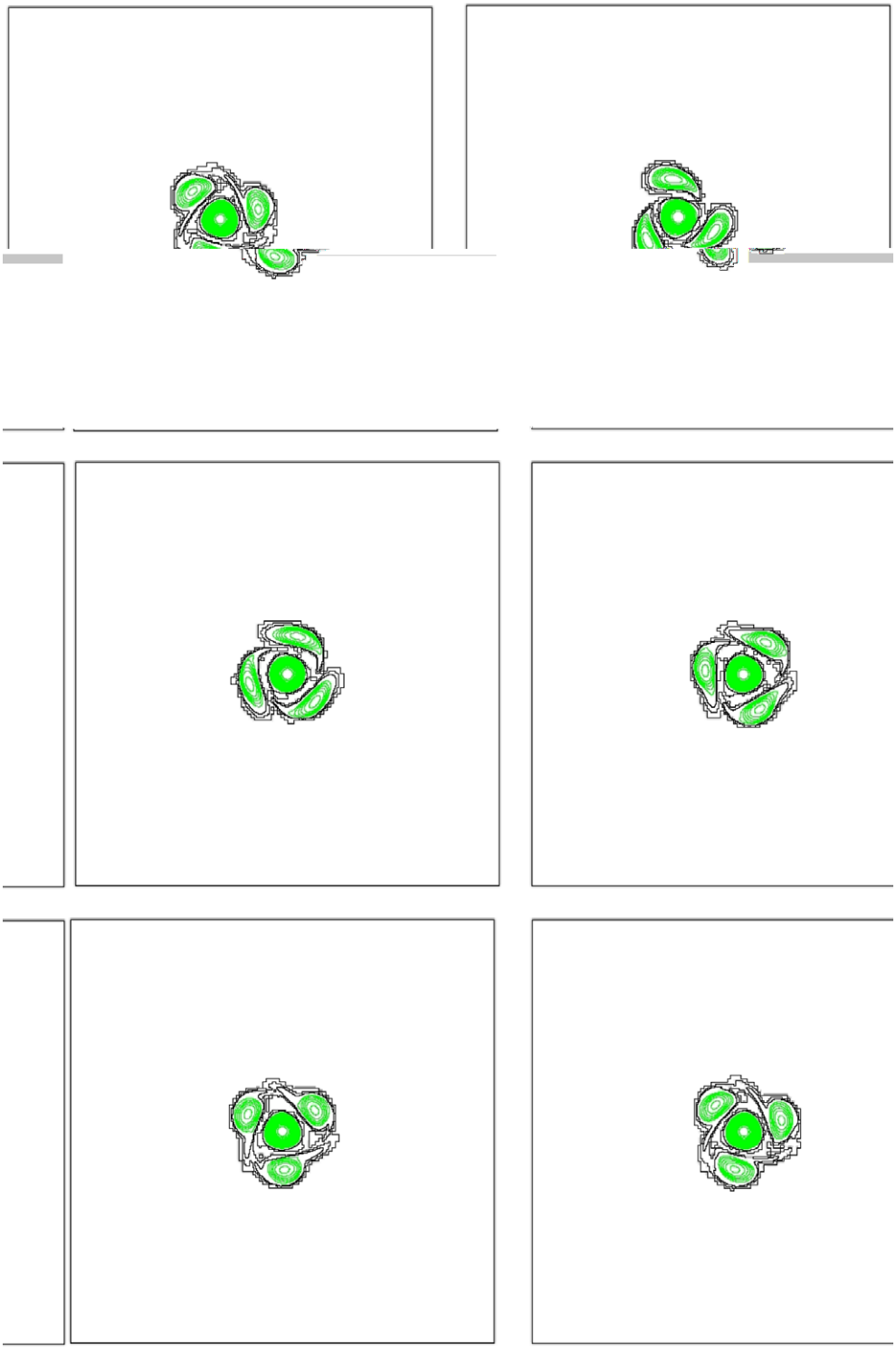

Fig. 11. Contour plots of the vorticity for example 4.6. From top to bottom and left to right: $t = 0$, 0.043, 0.096, 0.140, 0.184 and 0.219. The coarsest grid has level 7, and the finest has level 10. The lines represent the boundaries between levels of refinement.

4.6. Adaptive refinement

Here we consider the test problem proposed in [1], which requires an adaptive grid. Initially the velocity in the unit square is given by the vorticity produced by the sum of four vortices. The vortices are centered at $(0.5, 0.5)$, $(0.59, 0.5)$, $(0.455, 0.5 \pm 0.045\sqrt{3})$, with magnitudes -150, 50, 50 and 50, respectively. Each vortex has a profile of $\frac{1}{2}(1 + \tanh(100(0.03 - r)))$, where r is the distance to the center. The velocity is advected with viscosity 0.0001 and without a forcing term. The calculations are run until $t = 0.25$ with a CFL number of 0.9. Since the vorticity decays fast away from the four vortices, we assume the no-slip boundary condition for the velocity field. The refinement criteria is the same as that used for the driven cavity problem of Section 4.3 with $\tau = 0.004$. Since the exact solution is unknown, a numerical solution calculated on 1024^2 grid is used as the exact solution in the error analysis. Table 5 gives the accuracy results and Fig. 11 depicts the evolution of the vortices as well as the boundaries where the grid changes size. We find errors that are slightly larger than the errors reported in [1,24]. On the other hand, the time step we take is about three times larger.

5. Conclusions

We have presented an unconditionally stable second order accurate projection method for the incompressible Navier–Stokes equations on non-graded adaptive Cartesian grids. Quadtree and octree data structures are used to provide an optimal representation of the mesh. We use the supra-convergent Poisson solver of Min et al. [22] to account for the incompressibility condition, a second order accurate semi-Lagrangian method to update the momentum equation, a stiffly stable backward difference scheme to treat the diffusion term and a new method that guarantees the stability of the projection step on highly non-graded grids. All the variables are sampled at the nodes, producing a scheme that is straightforward to implement. Two- and three-dimensional examples have been presented to demonstrate second order accuracy for the velocity field and the divergence free condition in the L^1 and the L^∞ norms. Future work will seek to extend the present approach to the case where $U \cdot n \neq 0$ on $\partial\Omega$.

References

[1] A. Almgren, J. Bell, P. Colella, L. Howell, M. Welcome, A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations, J. Comput. Phys. 142 (1998) 1–46.
[2] A.S. Almgren, J.B. Bell, W.G. Syzmczak, A numerical method for the incompressible Navier–Stokes equations based on an approximate projection, SIAM J. Sci. Comput. 17 (1996) 358–369.
[3] 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.
[4] M. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.
[5] M. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[6] D. Brown, R. Cortez, M. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 168 (2001) 464–499.
[7] J. Cantarella, D. DeTurck, H. Gluck, Vector calculus and the topology of domains in 3-space, Am. Math. Mon. 109 (2002) 409–442.
[8] A. Chorin, A numerical method for solving incompressible viscous flow problems, J. Comput. Phys. 2 (1967) 12–26.
[9] R. Courant, E. Isaacson, M. Rees, On the solution of nonlinear hyperbolic differential equations by finite differences, Comm. Pure Appl. Math. 5 (1952) 243–255.
[10] W. E, J.G. Liu, Gauge method for viscous incompressible flows, Comm. Math. Sci. 1 (2003) 317–332.
[11] U. Ghia, K.N. Ghia, C.T. Shin, High-resolutions for incompressible flow using the Navier–Stokes equations and a multigrid method, J. Comput. Phys. 48 (1982) 387–411.
[12] R. Guy, A. Fogelson, Stability of approximate projection methods on cell-centered grids, J. Comput. Phys. 203 (2005) 517–538.
[13] M. Hagemann, O. Schenk, Weighted matchings for preconditioning symmetric indefinite linear systems, SIAM J. Sci. Comput. 28 (2006) 403–420.
[14] F. Harlow, J. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (1965) 2182–2189.
[15] W. Henshaw, A fourth-order accurate method for the incompressible Navier–Stokes equations on overlapping grids, J. Comput. Phys. 113 (1994) 13–25.
[16] J.V. Kan, A second-order-accurate pressure-correction scheme for viscous incompressible flow, SIAM J. Sci. Stat. Comput. 7 (1986) 870–891.
[17] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (1985) 308–323.

[18] D. Kincaid, W. Cheney, Numerical Analysis: Mathematics of Scientific Computing, Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 2002.
[19] K. Lipnikov, J. Morel, M. Shashkov, Mimetic finite difference methods for diffusion equations on non-orthogonal non-conformal meshes, J. Comput. Phys. 199 (2004) 589–597.
[20] F. Losasso, R. Fedkiw, S. Osher, Spatially adaptive techniques for level set methods and incompressible flow. Comput. Fluids (in press).
[21] F. Losasso, F. Gibou, R. Fedkiw, Simulating water and smoke with an octree data structure, ACM Trans. Graph. (SIGGRAPH Proc.) (2004) 457–462.
[22] C.-H. Min, F. Gibou, H. Ceniceros, A supra-convergent finite difference scheme for the variable coefficient Poisson equation on fully adaptive grids, CAM report 05-29, J. Comput. Phys. (in press).
[23] D. Moore, The cost of balancing generalized quadtrees, in: Proceedings of the Third ACM Symposium on Solid Modeling and Applications, 1995, pp. 305–312.
[24] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, J. Comput. Phys. 190 (2003) 572–600.
[25] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing, New York, NY, 1996.
[26] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, New York, 1989.
[27] H. Samet, Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS, Addison-Wesley, New York, 1990.
[28] J. Strain, Tree methods for moving interfaces, J. Comput. Phys. 151 (1999) 616–648.
[29] J. Strain, A fast modular semi-lagrangian method for moving interfaces, J. Comput. Phys. 161 (2000) 512–536.
[30] M. Sussman, A.S. Algrem, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, An adaptive level set approach for incompressible two-phase flow, J. Comput. Phys. 148 (1999) 81–124.
[31] A. Weiser, Local-mesh, local-order, adaptive finite element methods with a posteriori error estimators for elliptic partial differential equations, PhD thesis, Yale University, June 1981.
[32] D. Xiu, G. Karniadakis, A semi-lagrangian high-order method for Navier–Stokes equations, J. Comput. Phys. 172 (2001) 658–684.