ACADEMIC PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 184 (2003) 192–214

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

A moving unstructured staggered mesh method for the simulation of incompressible free-surface flows

Blair Perot*, Ramesh Nallapati

Department of Mechanical and Industrial Engineering, The University of Massachusetts at Amherst, Engineering Laboratory, Box 32210-219, Amherst, MA 01003-2210, USA

Received 7 February 2002; received in revised form 27 June 2002; accepted 7 October 2002

Abstract

A new moving staggered mesh discretization for the numerical simulation of incompressible flow problems involving free-surfaces is presented. The method uses the staggered mesh to obtain speed and conservation properties. Mesh motion provides a high quality mesh in the interior and detailed resolution of the free-surface motion on the surface. Mesh flipping allows for optimal mesh connectivity to be maintained. The method uses an exact projection procedure which reduces the number of unknowns as well as satisfying the continuity constraint without solving a pressure Poisson equation. The implementation of surface tension forces in the staggered mesh framework is discussed. The resulting method is tested against analytical solutions for liquid sloshing and free-surface channel flow. It is also demonstrated on the cases of droplet collision, three-dimensional sloshing, and turbulence next to a free-surface.

AMS: 65P10; 76D05; 76M25

Keywords: Free-surface; Incompressible; Unstructured; Staggered mesh; Navier–Stokes

1. Introduction

Free-surface flows encompass a wide range of engineering and environmental flows from very small-scale droplet dynamics to the flow around ships and offshore structures. The presence of a free-surface poses a challenging numerical problem because the location of the free-surface and hence the shape of the domain of interest may be time dependent. Additional physical degrees of freedom are also present with a free-surface due to the action of gravity and surface tension forces. These forces can lead to waves and additional nonlinearities in the governing Navier–Stokes equations. Numerical simulations of free-surface flows are particularly useful at large and small scales where it is frequently difficult to reformulate the problem on a laboratory scale while matching all the relevant dimensionless parameters.

* Corresponding author. Tel.: 1-413-545-3925; fax: 1-413-545-1027.
E-mail address: perot@ecs.umass.edu (B. Perot).

0021-9991/02/$ - see front matter
PII: S0021-9991(02)00027-X

There are two broadly different approaches to computing problems involving free-surfaces. The most common approach involves using a stationary fixed mesh and a method for identifying where the free-surface lies within that mesh. The level set method, the MAC method, and immersed boundary methods are schemes of this type. The other approach is to have a mesh that moves with the free-surface and therefore tracks it. Boundary element methods fall into this class of methods, but can only solve a restricted class of free-surface flows (unsteady potential flows and creeping flow). More general moving finite volume or finite element methods can be extremely computationally intensive (particularly in 3D) if the free-surface is unsteady and the mesh must be regenerated at each timestep. In this work, we use evolution equations for the mesh (as well as the flow) so that mesh regeneration is not performed and the solution procedure is efficient. The fact that the free-surface is tracked by the mesh means that its evolution can be very accurately tracked and that mesh refinement placed near the surface remains with the surface for arbitrarily large distortions of the free-surface. This makes the method well suited for applications in which the free-surface physics are of critical importance to the overall flow behavior. Moving mesh schemes using the finite element method and space-time finite elements are treated by, among others, [1–3]. Moving mesh methods based on finite volume techniques are treated in [4–6] where numerous prior references can be found. In this work, the focus is on generalizing the staggered mesh discretization so that it can be used in a moving mesh context and for free-surface simulations.

Cartesian staggered mesh methods date to the work of Harlow and Welch [7] and are very popular for the solution of incompressible flows. They are also frequently used in conjunction with level-set and MAC methods for free-surface flow problems (but using a fixed mesh). Staggered mesh methods have a number of attractive properties that make them popular in this context. They do not display any pressure modes such as those found in colocated finite volume methods, nor do they display the similar phenomena known as 'pressure locking' found in many finite element methods. They are known to locally conserve mass, momentum, kinetic energy, and vorticity. They are very computationally efficient and naturally conducive to using fractional step or projection methods for the pressure solution and satisfaction of incompressibility. Recently, staggered mesh methods have been generalized to unstructured (triangular and tetrahedral) meshes by Nicolaides [8–11], Porsching [12,13], and Perot [14–17]. They have been shown in these prior works to retain the properties of their Cartesian predecessor-lack of pressure modes, local conservation and high efficiency.

This paper presents a generalization of the unstructured staggered mesh method to moving meshes in Section 2. Unstructured staggered mesh methods do not have an obvious control volume associated with the normal velocity component and so the implementation of a moving mesh is a nontrivial task. In Section 3 the solution procedure is described, including the time advancement scheme and exact projection method for satisfying continuity. The evolution equations for the mesh and its reconnection strategy are presented in Section 4. A novel method for the implementation of surface tension in the context of unstructured staggered mesh methods is discussed in Section 5 along with more general boundary condition issues. In Section 6, the methodology is validated on a number of different free-surface problems including 2D and 3D sloshing in a tank, droplet collision with a wall at different Weber numbers, droplet oscillation, channel flow over a bump at different Froude numbers, and turbulence near a free-surface. A final discussion of the results is presented in Section 7.

2. Moving staggered mesh discretization

Staggered mesh methods are discretizations in which pressure is located at cell centers and the velocity is distributed to the cell faces, where only the normal component of the velocity at each cell face is known. A schematic diagram of the Cartesian and unstructured staggering arrangement is shown in Fig. 1. One major way in which the different unstructured staggered mesh methods differ is in how the

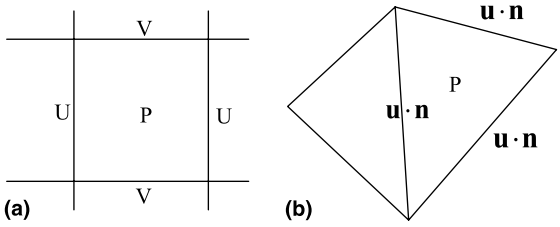


Fig. 1. (a) Cartesian staggered mesh variable locations. (b) Unstructured staggered mesh variable locations.

cell velocity vector is reconstructed or interpolated from the normal component at cell faces. The interpolation used in this work (Eq. (6a)) is known to result in local and global kinetic energy and momentum conservation [14].

The staggered mesh method requires a single evolution equation for the normal velocity component at each face. However, because we are interested in the face normal velocity equation on a moving mesh this equation is first derived from the equations governing the flow in a moving and distorting cell control volume,

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \int_{\mathrm{CS}(t)} \mathbf{u}_{\mathrm{surf}} \cdot \mathbf{n}\,\mathrm{d}A, \tag{1a}$$

$$\frac{\mathrm{d}(\rho V)}{\mathrm{d}t} + \int_{\mathrm{CS}(t)} \rho(\mathbf{u} - \mathbf{u}_{\mathrm{surf}}) \cdot \mathbf{n}\,\mathrm{d}A = 0, \tag{1b}$$

$$\frac{\mathrm{d}(\rho \mathbf{u} V)}{\mathrm{d}t} + \int_{\mathrm{CS}(t)} \rho\mathbf{u}(\mathbf{u} - \mathbf{u}_{\mathrm{surf}}) \cdot \mathbf{n}\,\mathrm{d}A = \int_{\mathrm{CV}(t)} (\rho\mathbf{g} - \nabla p)\,\mathrm{d}V + \int_{\mathrm{CS}(t)} \mu(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}}) \cdot \mathbf{n}\,\mathrm{d}A, \tag{1c}$$

where V is the cell volume and the flux through the control volume surface with an outwardly pointing unit normal $\mathbf{n}$ depends on the difference between the fluid velocity, $\mathbf{u}$, and the control surface velocity, $\mathbf{u}_{\mathrm{surf}}$. If the mesh moves in a Lagrangian manner ($\mathbf{u} = \mathbf{u}_{\mathrm{surf}}$) this flux term drops out of the mass (1b) and momentum (1c) equations. The first equation is unique to a distorting control volume and reflects the geometrical constraint that the change in the volume is a direct result of the net normal motion of the control volume surface.

Consider the following fully discrete version of these moving cell control volume equations:

$$\frac{V_{\mathrm{c}}^{n+1} - V_{\mathrm{c}}^{n}}{\Delta t} = \sum_{f}^{\text{cell faces}} U_{\mathrm{f}}^{\mathrm{mesh}}, \tag{2a}$$

$$\frac{\rho_{\mathrm{c}}^{n+1} V_{\mathrm{c}}^{n+1} - \rho_{\mathrm{c}}^{n} V_{\mathrm{c}}^{n}}{\Delta t} + \sum_{f}^{\text{cell faces}} \rho_{\mathrm{f}}(U_{\mathrm{f}} - U_{\mathrm{f}}^{\mathrm{mesh}}) = 0, \tag{2b}$$

$$\begin{aligned} &\frac{\rho_{\mathrm{c}}^{n+1} \mathbf{u}_{\mathrm{c}}^{n+1} V_{\mathrm{c}}^{n+1} - \rho_{\mathrm{c}}^{n} \mathbf{u}_{\mathrm{c}}^{n} V_{\mathrm{c}}^{n}}{\Delta t} + \sum_{f}^{\text{cell faces}} \rho_{\mathrm{f}} \mathbf{u}_{\mathrm{f}}(U_{\mathrm{f}} - U_{\mathrm{f}}^{\mathrm{mesh}}) \\ &\quad = -V_{\mathrm{c}} \nabla(p - \rho\mathbf{g} \cdot \mathbf{x}_{\mathrm{c}}^{\mathrm{CG}}) + \sum_{f}^{\text{cell faces}} \mu(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}}) \cdot \mathbf{n}_{\mathrm{f}} A_{\mathrm{f}} - V_{\mathrm{c}}(\mathbf{g} \cdot \mathbf{x}_{\mathrm{c}}^{\mathrm{CG}}) \nabla\rho, \end{aligned} \tag{2c}$$

where $U_f = \int_{face} \mathbf{u} \cdot \mathbf{n}_f \, dA$ is the velocity area integral on the face, and U_f^{mesh} is the equivalent mesh velocity of the face. More information on the mesh velocity can be obtained from Eq. (2a). This equation is not solved but is a constraint on the mesh velocity that should be satisfied exactly. Note that each face of an arbitrary distorting tetrahedron (or triangle in 2D) has a face normal and face area that only depend on time, and the mesh velocity is constant in time within a time interval but varies linearly in space since each node in the mesh is assumed to move along a straight line within each time interval. With this information the mesh flux is given exactly by,

$$U_f^{mesh} = \frac{1}{\Delta t} \int_n^{n+1} \mathbf{n}_f \cdot \left(\int_{face} \mathbf{u}_{mesh} \, dA \right) dt = \mathbf{u}_{mesh}^{CG} \cdot \left(\frac{1}{\Delta t} \int_n^{n+1} \mathbf{n}_f A_f \, dt \right), \tag{3a}$$

where $\mathbf{u}_{mesh}^{CG}$ is the velocity of the face centroid (or center of gravity). In 2D the product of the face normal and face area varies linearly in time, so

$$U_f^{mesh} = \mathbf{u}_{mesh}^{CG} \cdot \frac{1}{2} (\mathbf{n}_f^{n+1} A_f^{n+1} + \mathbf{n}_f^n A_f^n).$$

In 3D the time variation can be more complex and

$$U_f^{mesh} = \mathbf{u}_{mesh}^{CG} \cdot \left\{ \frac{1}{2} (\mathbf{n}_f^{n+1} A_f^{n+1} + \mathbf{n}_f^n A_f^n) - \frac{\Delta t^2}{12} \sum_e^{face\ edges} (\mathbf{v}_{n1} \times \mathbf{v}_{n2}) \right\}, \tag{3b}$$

where the $\mathbf{v}_n$ is the mesh velocity of the face nodes (corners), and each edge of the face contributes a cross product of its two end point velocities with the endpoints oriented in a counterclockwise fashion with respect to the face normal. Note that this mesh flux is the volume swept out by the face area during the time interval, and it is a constant during that interval.

Eq. (2b) is the mass equation. If the density values on all the cell faces and within the cell are equal and do not change with time and the constraint given by Eq. (2a) is satisfied, then Eq. (2b) returns the discrete incompressibility constraint

$$\sum_f^{cell\ faces} U_f = 0, \tag{4}$$

which states that incompressibility is enforced on average over the time interval and over the cell. The time advancement of the convection term in the mass equation has been left ambiguous in Eq. (2b). It can be discretized as either

$$\sum_f^{cell\ faces} \left\{ \alpha \rho_f^{n+1} (U_f^{n+1} - U_f^{mesh}) + (1-\alpha) \rho_f^n (U_f^n - U_f^{mesh}) \right\} \tag{5a}$$

or

$$\sum_f^{cell\ faces} \rho_f^{n+\alpha} (U_f^{n+\alpha} - U_f^{mesh}), \tag{5b}$$

where $\rho_f^{n+\alpha} = \{(1-\alpha)\rho_f^n + \alpha \rho_f^{n+1}\}$ and $U_f^{n+\alpha}$ is defined analogously. We believe that strict kinetic energy conservation requires Eq. (5b) with $\alpha = \frac{1}{2}$. However Eq. (5a) with $\alpha = \frac{1}{2}$ corresponds to the trapezoidal update scheme and in the case where density is constant in time (as in this work), both schemes are identical.

The momentum equation is given by Eq. (2c). We have chosen for the development of the staggered mesh scheme to split the gravity term into a hydrostatic pressure contribution and a density gradient

contribution. The density gradient is zero in the flows considered in this work and this term is therefore not included in future analysis. Eq. (2c) assumes that the hydrostatic pressure gradient within the cell can be approximated as a constant. A critical feature of the staggered mesh method, that distinguishes this particular approach from standard finite volume approaches is that the pressure gradient term is not approximated by a summation of the pressure over the various cell faces. The time advancement of the momentum convection, diffusion and pressure gradient are similar to Eq. (5a), and are discussed in more detail below.

In order to be considered a staggered mesh method the primary unknowns of the method should be the normal velocities at the faces, and there should be only one update equation for each normal velocity component. Eq. (2c) has the cell velocity vector as the unknown and three evolution equations (two in 2D) within the cell. This dilemma is rectified by defining the cell velocity in terms of the face normal-velocity components.

$$\mathbf{u}_c = \frac{1}{V_c} \sum_{f}^{\text{cell faces}} U_f (\mathbf{x}_f^{CG} - \mathbf{x}_c^{CG}), \tag{6a}$$

where the mass flux U_f points out of the cell, $\mathbf{x}_f^{CG}$ is the face center of gravity (centroid), and $\mathbf{x}_c^{CG}$ is the cell position. For a discretely incompressible flow (satisfying Eq. (4)) the cell position is arbitrary, but for numerical and analytical simplicity we take it to be the cell center of gravity. This interpolation or reconstruction procedure for the velocity has been shown to be second order accurate for arbitrary 2D meshes [14] and nearly second order accurate on arbitrary 3D meshes [15]. In 3D the first order error term is proportional to the change in the local mesh size from one cell to the next. On all practical meshes we have tested, this first order term is actually smaller than the standard second order error term due to the linear interpolation. This form of the velocity interpolation is critical to momentum and kinetic energy conservation of unstructured staggered mesh methods on fixed meshes [14,15].

In addition to Eq. (6a), the following average is performed on each face. This average involves the two neighboring discrete cell momentum equations.

$$\mathbf{e}_{e1} \cdot (\mathbf{x}_f^{CG} - \mathbf{x}_{c1}^{CG}) - \mathbf{e}_{c2} \cdot (\mathbf{x}_f^{CG} - \mathbf{x}_{c2}^{CG}), \tag{6b}$$

where $\mathbf{e}_c$ is Eq. (2c) for that cell. This is the discrete equivalent of integrating the momentum equation along two line segments joining two cell centers. Note that the line joining two cell centers is not necessarily normal to the face (unless cell circumcenters are used as the cell centers). Nonetheless, we get a single equation at each face as desired.

Eq. (6a) can be written in matrix notation as $\mathbf{u} = \mathbf{V}^{-1}\mathbf{R}U$, where $\mathbf{V}$ is a diagonal matrix of cell volumes and $\mathbf{R}$ is a nonsquare matrix of the cell-to-face vector, $\mathbf{r} = \mathbf{x}_f^{CG} - \mathbf{x}_c^{CG}$. By design, Eq. (6b) is the transpose operation, $\mathbf{R}^T$.

So the discrete momentum equation (Eq. (2c)) is rewritten as,

$$\begin{aligned}(\mathbf{R}^T)^{n+1} \frac{\rho}{\Delta t V^{n+1}} \mathbf{R}^{n+1} U^{n+1} - (\mathbf{R}^T)^{n+1} \frac{\rho}{\Delta t V^{n+1}} \mathbf{R}^n U^n + \alpha (\mathbf{R}^T)^{n+1} \mathbf{A}^{n+1} + (1-\alpha)(\mathbf{R}^T)^{n+1} \frac{V^n}{V^{n+1}} \mathbf{A}^n \\ = -\beta \mathbf{G} p_H^{n+1} - (1-\beta)(\mathbf{R}^T)^{n+1} \frac{V^n}{V^{n+1}} \nabla p_H^n,\end{aligned} \tag{7}$$

where α is a weighting between explicit and implicit time advancement with $\alpha = \frac{1}{2}$ giving the trapezoidal method. The parameter β is the weighting for the pressure gradient term. For stability it can be shown that $\beta \geqslant \frac{1}{2}$ so the pressure gradient is always partially implicit even if convection and diffusion are explicitly updated. The hydrostatic pressure is given by $p_H = p - \rho \mathbf{g} \cdot \mathbf{x}$, and the cell advection–diffusion vector $\mathbf{A}$ is given by

$$\mathbf{A}V_c = \rho_c \sum_f^{\text{cell faces}} \mathbf{u}_f (U_f - U_f^{\text{mesh}}) - \sum_f^{\text{cell faces}} \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T) \cdot \mathbf{n}_f A_f. \tag{8}$$

Note that the implicit pressure term results in a gradient operator defined by $\mathbf{G}p = p_{c2} - p_{c1}$, which is just the difference between the two neighboring cell values. This assumes that the pressure is a continuous function along the line joining the cell centers so $\int_{c1}^{c2} \nabla p \cdot d\mathbf{L} = p_{c2} - p_{c1}$. The matrices in the very first term can be collectively viewed as a type of mass matrix. This mass matrix is symmetric positive definite and connects each face with its nearest neighbors.

3. Solution procedure

The resulting system of equations (Eqs. (7) and (4)) can be written in the following concise form,

$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}^{n+1} - \alpha(\mathbf{R}^T)^{n+1}\mathbf{A}^{n+1} & \beta\mathbf{G} \\ \beta\mathbf{G}^T & 0 \end{bmatrix} \begin{bmatrix} U_f \\ p_H \end{bmatrix} = \begin{bmatrix} (\mathbf{R}^T)^{n+1} \frac{V^n}{V^{n+1}} \{ \frac{\rho}{\Delta t}\mathbf{u}_c^n - (1-\alpha)\mathbf{A}^n - (1-\beta)\nabla p^n \} \\ 0 \end{bmatrix}, \tag{9}$$

where $\mathbf{M}^{n+1} = (\mathbf{R}^T)^{n+1}(\rho/V^{n+1})\mathbf{R}^{n+1}$ is the mass matrix. Note that the summation in Eq. (4) is the negative transpose of the gradient operator, and this divergence constraint can be multiplied by any nonzero β, to obtain the last line of this matrix system.

The system given by Eq. (9) is symmetric if convection is explicit, but it is not positive or negative definite. It can be solved as a full system, though this is expensive, or an approximate solution can be obtained via a fractional step method. The fractional step method is attractive because it splits this system into an advection–diffusion solution and a separate Poisson solution. In this work we use a recently developed exact fractional step method [18] that solves this system exactly. Taking advantage of the easily constructed null space of the gradient operator, $\mathbf{C}$, such that $\mathbf{G}^T\mathbf{C} = 0$. This allows Eq. (9) to be rewritten as,

$$\mathbf{C}^T \left\{ \frac{1}{\Delta t}\mathbf{M}^{n+1} - \alpha\mathbf{R}^T\mathbf{A}^{n+1} \right\} \mathbf{C}s^{n+1} = \mathbf{C}^T\mathbf{R}^T \frac{V^n}{V^{n+1}} \left\{ \frac{\rho}{\Delta t}\mathbf{u}_c^n - (1-\alpha)\mathbf{A}^n - (1-\beta)\nabla p^n \right\} \tag{10}$$

for the unknown temporary variable s^{n+1}, and $U_f^{n+1} = \mathbf{C}s^{n+1}$ gives the final solution. This technique for solving Eq. (9) satisfies discrete incompressibility to machine precision even if Eq. (10) is only solved to some finite level of accuracy using an iterative method. It also reduces the number of unknowns in the system when the mesh is tetrahedral or triangular, thereby speeding solution times.

The pressure gradient can be recovered from the Eq. (2c) as a post-processing step once the final velocity is found

$$\beta V_c^{n+1}\nabla p_H^{n+1} = -(1-\beta)V_c^n\nabla p_H^n - \frac{\rho}{\Delta t}\mathbf{R}^{n+1}U_f^{n+1} + \frac{\rho}{\Delta t}\mathbf{R}^n U_f^n - \alpha V_c^{n+1}\mathbf{A}^{n+1} - (1-\alpha)V_c^n\mathbf{A}^n. \tag{11a}$$

The pressure gradient at the very first time step can be obtained from a Poisson equation, or assumed to be zero. The accuracy of the velocity solution is independent of the choice for β [19,20], and the fully implicit choice, $\beta = 1$, (used in Section 6) eliminates the need to ever compute the pressure gradient at all. It may be convenient to rearrange Eq. (11a) into the alternative form,

$$q^{n+1} = \frac{1}{\beta}V_c^{n+1}\left\{ \frac{\rho}{\Delta t}\mathbf{u}_c^{n+1} + (\alpha-\beta)\mathbf{A}^{n+1} \right\} - \left(\frac{1-\beta}{\beta} \right) q^n,$$

where

$$q^n = V_c^n \{ \frac{\rho}{\Delta t}\mathbf{u}_c^n - (1-\alpha)\mathbf{A}^n - (1-\beta)\nabla p_H^n \}. \tag{11b}$$

This variable can be used to eliminate the pressure gradient in Eqs. (9) and (10) when $\beta \neq 1$. In our work, the convection term is explicit, and the diffusion term is implicit and solved with the trapezoidal method ($\alpha = \frac{1}{2}$). The pressure is fully implicit so Eq. (10) becomes,

$$\mathbf{C}^{\mathrm{T}}\left\{\frac{1}{\Delta t}\mathbf{M}^{n+1}+\frac{1}{2}\mathbf{R}^{\mathrm{T}}\mathbf{L}^{n+1}\right\}\mathbf{C}s^{n+1}=\mathbf{C}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\frac{V^{n}}{V^{n+1}}\left\{\frac{\rho}{\Delta t}\mathbf{u}_{\mathrm{c}}^{n}+\mathbf{N}^{n+1/2}+\frac{1}{2}\mathbf{L}^{n}\right\}, \tag{12}$$

where $\mathbf{L}$ is the diffusion term and $\mathbf{N}$ is the nonlinear convection term. Implementing Eq. (10), (or Eq. (12)) requires knowing the expected mesh motion prior to solving for the velocity, because the final mesh positions and geometry appear implicitly in many of the terms, including the mass matrix and the mesh velocity. This means that even if all the terms in Eq. (10) are implicit, the free-surface flow solution will not necessarily be stable. Both surface tension and gravity can create surface waves which couple the fluid velocity and free-surface position. Simple extrapolation of the free-surface position from prior velocity knowledge is equivalent to updating the free-surface equation with explicit Euler or possibly Adams–Bashforth time advancement. These types of time advancement schemes are known to be unstable for pure wave solutions even if the velocity is updated implicitly. Fully implicit solution would require linearization of the geometry terms and coupled solution of velocity, pressure, and position variables. Alternatively we have chosen to use a three-step second-order, low storage, Runge–Kutta scheme for time advancement. This particular scheme is stable for pure wave solutions as long as the CFL based on the wave speed is less than 2. Since we are interested in resolving surface wave dynamics, this restriction is not overly burdensome and is equivalent to requiring that the temporal accuracy matches the spatial accuracy. Convection and diffusion can still be solved implicitly at each substep to avoid stability restrictions based on the fluid velocity or viscosity. The three-step scheme used in this work is

$$\begin{aligned}
\hat{u}^{n+1}-u^{n}&=\Delta t F(u^{n}),\\
\hat{\hat{u}}^{n+1}-u^{n}&=\Delta t\frac{1}{2}\{F(u^{n})+F(\hat{u}^{n+1})\},\\
u^{n+1}-u^{n}&=\Delta t\frac{1}{2}\{F(u^{n})+F(\hat{\hat{u}}^{n+1})\}.
\end{aligned} \tag{13}$$

This scheme is very easy to implement. It is second order, and the final substep can be left out if sufficient viscous damping is present. At each substep the new surface position is predicted and then Eq. (10) is solved for the normal velocity components.

4. Mesh motion and reconnection

The nodes on the free-surface move in a Lagrangian fashion and obey the equation.

$$\frac{\mathrm{d}\mathbf{x}_{\mathrm{node}}}{\mathrm{d}t}=\mathbf{u}_{\mathrm{node}} \tag{14a}$$

with the node velocity given by an average of the cell velocities.

$$\mathbf{u}_{\mathrm{node}}=\frac{1}{(\#\,\mathrm{of\,cells})}\sum_{c}^{\mathrm{node\ cells}}\mathbf{u}_{\mathrm{c}}+(\mathbf{x}_{\mathrm{node}}-\mathbf{x}_{\mathrm{c}})\cdot\nabla\mathbf{u}|_{\mathrm{c}}. \tag{14b}$$

The nodes in the interior of the domain do *not* move in a Lagrangian fashion since this can cause a highly distorted mesh to develop. Instead, the interior nodes obey the following relaxation equation,

$$\mathbf{x}_{\mathrm{node}}^{n+1}-\mathbf{x}_{\mathrm{node}}^{n}=\mathbf{G}_{\mathrm{n2}e}^{\mathrm{T}}k_{\mathrm{e}}\mathbf{G}_{\mathrm{n2}e}\mathbf{x}_{\mathrm{node}}^{n+1}. \tag{15}$$

This equation is equivalent to treating each edge of the mesh as a linear spring [21]. The spring force is proportional to the distance between the two nodes, $\mathbf{G}_{\mathrm{n2}e}\mathbf{x}^{n+1}_{\mathrm{node}} = \mathbf{x}^{n+1}_{\mathrm{node2}} - \mathbf{x}^{n+1}_{\mathrm{node1}}$, and has a spring constant k_{e}. The equilibrium length of the spring is zero. The forces from each spring are then added to the node via $\mathbf{G}^{\mathrm{T}}_{\mathrm{n2}e}$, and cause a net motion of the node towards equilibrium. This is illustrated in Fig. 2. In two dimensions the springs correspond to each face of the mesh. If $k_{\mathrm{e}} \gg 1$ the relaxation to equilibrium happens in one time step. If $k_{\mathrm{e}} \approx 1$ the mesh will not move more than the local mesh spacing in one time step. The implicit solution of this relation equation is carried out using a Jacobi preconditioned Conjugate Gradient solver. Smoothing an initial mesh can take some time depending on the quality of the mesh generator. However, once the simulation is evolving the mesh solver takes 2–10 iterations to converge and is extremely fast. This is far more cost efficient than regenerating the mesh after it becomes too distorted, and it maintains a very high quality mesh at every time step. This algorithm does not change the mesh connectivity so all control volumes retain the same neighbors as they move and distort. If a mesh starts with higher resolution in one area, such as near the free-surface, it retains that higher resolution as the simulation evolves.

Mesh smoothing can also be used to do adaptive mesh refinement [22]. The idea is to use a variable spring constant. By making the spring constant proportional to the second derivative of the solution, the mesh can be pulled into regions where the solution is changing rapidly and stretched where the solution is linear or constant. It is even possible to perform anisotropic mesh adaptation in this way, to resolve features such as boundary layers or shear layers. Anisotropic mesh relaxation takes the spring constant to be proportional to the second derivative of the solution in the direction of the spring. In this way, springs oriented along the boundary layer flow direction have small spring constants and stretch considerably, while springs oriented across the boundary layer see large second derivatives of the solution, have large spring constants, and cause a high resolution mesh to develop in the cross stream direction.

For adaptation on a single solution variable ϕ the anisotropic adaptive spring constant is given by,

$$k_{\mathrm{e}} = \frac{(\mathbf{x}_{\mathrm{node1}} - \mathbf{x}_{\mathrm{node2}}) \cdot (\nabla\phi|_{\mathrm{node1}} - \nabla\phi|_{\mathrm{node2}})}{(\mathbf{x}_{\mathrm{node1}} - \mathbf{x}_{\mathrm{node2}}) \cdot (\mathbf{x}_{\mathrm{node1}} - \mathbf{x}_{\mathrm{node2}})}. \tag{16}$$

This spring constant is then normalized so that the average value on the domain is equal to 1, and for stability reasons we frequently limit the extreme values of the spring constant to be at least 1/100 of the mean value of the spring constant, and less than 100 times the mean value. When multiple flow variables are present, the spring constant is taken to be the maximum of all the individual spring constants. This causes the mesh to adapt to the worst resolved flow variable at any location. The technique for adapting the mesh, has the advantage that it keeps the solution cost at each timestep constant. The user specifies the mesh size that they are willing or able to use, and the adaptation makes the best use of the available resources. Mesh adaptation schemes which insert points or recursively subdivide cells can lead to ex-

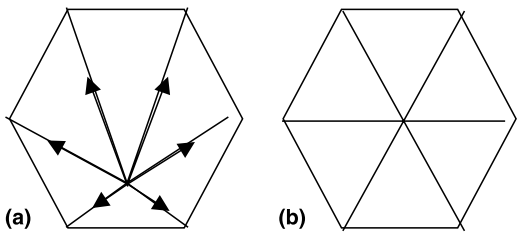


Fig. 2. Schematic representation of mesh smoothing in two dimensions. Arrows in the first picture represent spring forces acting on the central node. Boundary nodes are assumed fixed in this simple example. (a) The mesh and forces before relaxation. (b) The mesh after relaxation.

tremely unpredictable resource requirements and solution times. An example of mesh adaptation is shown in Fig. 3. The adapted mesh on the right contains the same number of mesh points but has over 20 times the mesh resolution in the critical shear layers. This vortex shedding from a square cylinder is unsteady, and the resolution adapts in time, maintaining high resolution on the moving shear layers at all times.

After a complete timestep, it is occasionally necessary to optimize the mesh connectivity. The Delaunay criterion, that no node lies within any cell circumsphere (circumcircles in 2D), is used to judge when the mesh connectivity can be improved. This criterion can be shown optimal in several senses for 2D meshes [23]. The criterion is still used in our 3D calculations but may no longer be optimal. Rather than completely rebuilding the mesh when it begins to loose optimal connectivity, a local connectivity correction algorithm called mesh flipping is used. An example of mesh flipping in 2D is shown in Fig. 4. In two dimensions mesh flipping removes a face and replaces it with a new face that improves the local quality of the mesh. The face that is removed can be sensed by observing the quantity

$$(\mathbf{x}_{c2}^{CG} - \mathbf{x}_{c1}^{CG}) \cdot (\mathbf{x}_{c2}^{CC} - \mathbf{x}_{c1}^{CC}), \tag{17}$$

where $\mathbf{x}_c^{CC}$ is the cell circumcenter position of the two neighboring cells. If this quantity is negative, the face needs to be flipped. It can be shown [23] that in 2D, flipping faces based on this criteria will eventually lead to a mesh that satisfies the Delaunay criteria everywhere. In practice, the Delaunay criteria is rarely

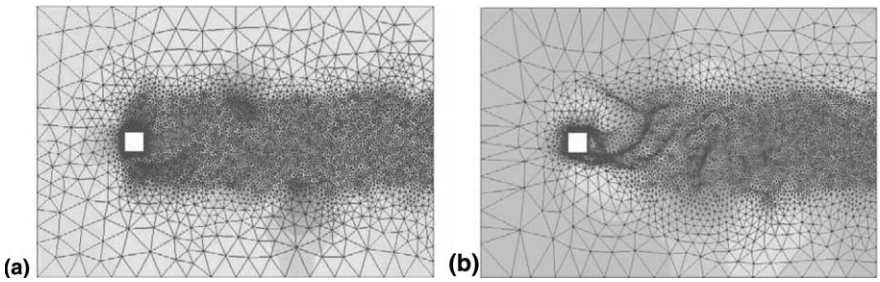


Fig. 3. Adaptive mesh refinement applied to the unsteady vortex shedding behind a square cylinder. (a) The best that can be achieved without a priori knowledge of the solution. (b) Solution adapted mesh using anisotropic springs. Adaptation is based on the flow kinetic energy.

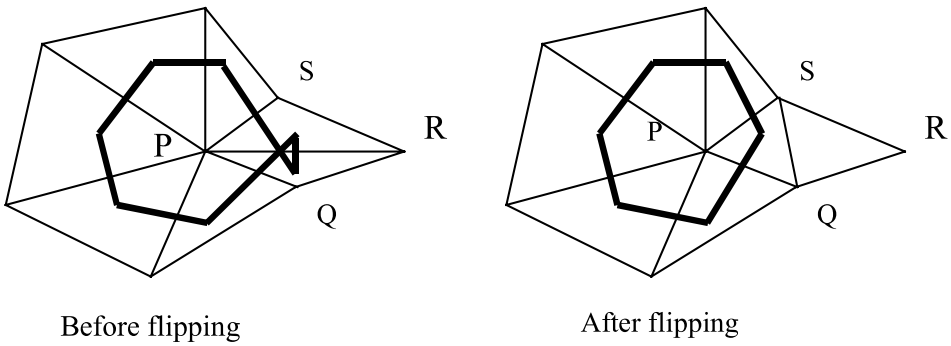


Fig. 4. Example of 2D flipping. The face PR is removed and replaced with face SQ to improve the mesh quality. The dark solid line is the Veronoi polygon formed by joining neighboring cell circumcenters. When the mesh is not Delaunay this polygon develops an inversion or 'fish tail'.

violated, and once the initial mesh is made Delaunay, the mesh flips in one or two places every ten or so timesteps. The local nature or this flipping algorithm makes it easy to implement and highly parallel.

Mesh flipping can also be performed in three dimensions [23]. However it is conceptually more difficult. The flipping either turns two tetrahedra into three tetrahedra by removing one face and adding one edge and three new faces, or the opposite. Other degenerate cases are also possible, such as two tetrahedra to two tetrahedra when two of the faces on each tetrahedron are coplanar, or multiple tetrahedra sharing a common edge flipping to fewer tetrahedra with the edge removed. However, we have determined that if all possible 2-to-3 flips are performed first, and zero volume cells are allowed, then these degenerate cases will automatically be taken care of. In three dimensions the criteria cited above (Eq. (17)) is necessary but not sufficient to indicate a 2–3 flip. In some cases the two tetrahedra involved do not form a convex hull. Flipping them would result in entanglement and negative volume cells. So in 3D the convexity of the two tetrahedra sharing a face is also tested. Nonconvex hulls are ultimately fixed by the 3-to-2 flip, which removes an edge and three faces and inserts a new face. Tests indicate that simply flipping faces which violate the Delaunay criteria is not sufficient in 3D to guarantee a globally Delaunay mesh. In 3D 'knots' can form. These have been successfully removed in all of our 3D simulations by flipping the 'worst' (closest to nonDelaunay, but still technically valid) face neighboring the knot, and then flipping the knot. However, there is no proof at this time that this approach, or some generalization of it, is guaranteed to work.

When a flip occurs the solution variables must be interpolated onto the new mesh. Because the flip is local (involving only two or three cells) it is relatively easy to develop interpolation schemes that conserve mass and momentum. For incompressible flows, mass is conserved when the volume is conserved, and this is always the case in a flipping operation. Our exact projection scheme [18] eliminates pressure as an unknown, and so it is not necessary to interpolate the pressure. The current interpolation scheme for the velocity takes advantage of the fact that the vector stream function component along each edge is known when using the exact projection method. The normal velocity at the faces (the primary unknown in staggered mesh methods) can be easily recovered from these stream function components (using Stokes Curl theorem). In two dimensions, the stream function is at cell nodes and is never altered by a flip-so the velocity interpolation is conservative and nondiffusive. In three dimensions, a 3-to-2 cell flip removes an edge and three faces and creates one new face. The normal velocity on the new face is obtained from the edge stream function values, and this interpolation is conservative and nondiffusive. A 2-to-3 cell flip creates a new edge, which requires a stream function value. The new stream function is assigned so that the vorticity along the new edge equals the average old vorticity in the original two cells. In this way the 2-to-3 interpolation conserves momentum and vorticity. It is not know if the 2-to-3 interpolation is diffusive or not. The stream function interpolation approach also guarantees incompressibility of the interpolated velocity field. Other interpolation schemes are certainly possible, and because the flipping is restricted to two or three cells at any one time, it is relatively easy to ensure that such interpolations conserve momentum, dilatation, and vorticity.

5. Boundary conditions

In flows with a free-surface or an outflow it is frequently desirable to leave the normal velocity at the boundary as an unknown. In the unstructured staggered mesh method, it is possible to prescribe the boundary pressure and leave the normal velocity as an unknown. Consider applying Eq. (7) at a boundary face. Normally, in the interior of the domain, this equation is an approximation for the momentum equation along the two line segments joining two neighboring cell centers. On the boundary, it has a very similar interpretation, but the integral is only along one line segment, joining the cell to the boundary face (see Fig. 5).

With this interpretation, the matrix operators $\mathbf{R}$, which interpolates the face normal velocity to construct cell velocity vectors, and $\mathbf{R}^{\mathrm{T}}$ which performs the line integration are still the transpose of each other. The

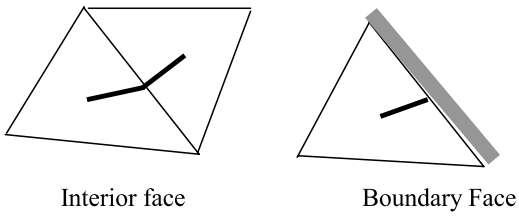


Fig. 5. Line integral for an interior face and a boundary face.

implication is that the boundary face normal velocity component is only used to construct one cell velocity vector, whereas in the interior, the normal velocity component is incorporated into the interpolation (reconstruction) of both neighboring cell velocity vectors.

With this interpolation, all the terms in Eq. (7) are well defined on boundary faces except the pressure gradient at time $n+1$. The pressure gradient term requires the hydrostatic pressure of the face to be known. Note that the exact projection scheme described by Eq. (10) eliminates the interior cell pressure unknowns at time $n+1$, but not the boundary pressure.

At an outflow, it is often a good assumption that the pressure is hydrostatic. This is equivalent to assuming that the hydrostatic pressure is constant along the outflow plane, and this is a very easy boundary condition to prescribe with this formulation. Note that the boundary faces have an implicit directional bias associated with them because they use information from only one cell. At an outflow, the interior cell is upwind, and this directional bias is known to be stable.

At a free-surface (without surface tension), the pressure can be assumed to be constant. The hydrostatic pressure is then calculated using this constant pressure (usually assumed to be zero) and subtracting the gravity term $\rho\mathbf{g}\cdot\mathbf{x}_{\mathrm{f}}^{\mathrm{CG}}$ which is based on the face center of gravity position and the gravity vector. At the free-surface, the mesh is required to move at the same speed as the fluid, so the convection term is set to zero. This allows the free-surface to move up or down and still remain stable. The downward motion looks somewhat like an unstable inflow condition, but is not-because the mesh moves at the same rate. Diffusion is often small near the free-surface but is not necessarily zero.

Interestingly, the pressure boundary condition can also be adapted to prescribe a natural and stable inflow condition. If a momentum source is present in the domain the fluid is sucked in through the inflow and should not be prescribed. Setting the dynamic hydrostatic pressure at such an inflow $p_{\mathrm{D}} = p - \rho\mathbf{g}\cdot\mathbf{x} + \frac{1}{2}\rho\mathbf{u}\cdot\mathbf{u}$ to be constant and forcing convection and diffusion to be zero is equivalent to assuming a quasi-steady potential flow is present at the inflow.

At a wall (either slip or no slip) or a standard inflow boundary, the normal velocity is usually prescribed. In this case, Eq. (7) is not evaluated on the boundary face, and no knowledge of the boundary pressure is required for these boundaries.

Surface tension is calculated on the free-surface using first principles. The force on a face is calculated as a sum over all its edges. Each edge pulls in the direction of the neighboring face, with the force proportional to the surface tension on that face. Fig. 6 shows the surface tension forces in both 2D and 3D.

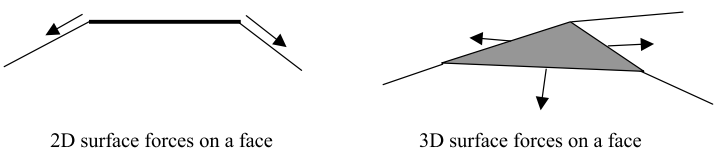


Fig. 6. Surface tension forces in 2D and 3D. Each force lies in the plane of the neighboring face and perpendicular to the edge on which it acts.

The force is given mathematically by the expression,

$$\mathbf{f}_{\text{tension}} = \sum_{e}^{\text{face edges}} L_{\text{e}}\sigma_{fi}(\mathbf{x}_{fi}^{\text{CC}} - \mathbf{x}_{\text{e}}^{\text{mid}})/|\mathbf{x}_{fi}^{\text{CC}} - \mathbf{x}_{\text{e}}^{\text{mid}}|, \tag{18}$$

where L_{e} is the length of the edge (which in 2D is just 1), σ_{fi} is the surface tension on the neighboring face sharing that edge, and $\mathbf{x}_{fi}^{\text{CC}}$ is the face circumcenter position (midpoint in 2D). The circumcenter position makes the resulting force at that edge perpendicular to that edge but still in the plane of the neighboring face.

This representation of the surface tension is independent of orientation and it does not require complex derivatives or curve fitting. It accounts for variable surface tension effects correctly, and it can be shown to be conservative [24]. Conservation means that the net surface tension force on an isolated liquid drop is always identically zero and will not cause the drop to drift randomly in the absence of gravity. This surface tension formulation, and the code in general, are validated in the following section.

6. Results

This section presents some of the results obtained from the numerical simulation of free-surface flows using the present numerical techniques. The first few cases validate the code by comparing the results of numerical simulations with analytical predictions or experimental data. The remaining cases present simulation results for a few complex free-surface problems that test the effectiveness of the numerical technique.

6.1. Free-surface sloshing in a rectangular tank

The first test case is 2D free surface sloshing in a rectangular tank. This problem consists of simulating a small amplitude gravity-induced standing wave in a tank containing an irrotational fluid. The free surface is given a small amplitude sinusoidal velocity profile and the resulting frequency of oscillation of the standing wave is computed numerically. The wavelength of the standing wave is taken to be twice the width of the tank. The size of the tank, L, and the initial depth of the liquid, D, are fixed at 1 unit each in these simulations. A two-dimensional triangular computational grid has been utilized for the problem as shown in Fig. 7. The side and bottom walls of the rectangular tank are assumed to be slip-walls and the free-surface is assigned a constant pressure boundary condition.

One cycle of the sloshing is shown in Fig. 7. The arrows in the pictures represent the local velocity vectors and are colored by the vertical component of the velocity with red standing for maximum upward velocity and blue for maximum downward velocity. While the maximum amplitude of the waves is only 10% of the initial fluid-level in the container, this simulation demonstrates the mesh flipping, mesh smoothing and surface tracking schemes, and the methods ability to accurately prediction free-surface deformation. We have computed up to 10 cycles of this oscillation with no perceptible variation from cycle to cycle.

Analytically, it has been demonstrated that the fundamental period of sloshing of infinitesimal standing waves with a wavelength $2L$ in a rectangular container of length L and initial depth D is given by [25],

$$T = \left[\frac{g}{4\pi L}\tanh\left(\frac{\pi D}{L}\right)\right]^{-1/2}, \tag{19}$$

where g is the acceleration due to gravity. The time period computed from the simulations are compared with the analytical predictions for the same problem for various values of gravity. Fig. 8 shows a comparison of numerical and analytical solutions as a function of gravity. The maximum error of the simulations is observed to be less than 2%. The presence of error can be attributed to the fact that the analytical

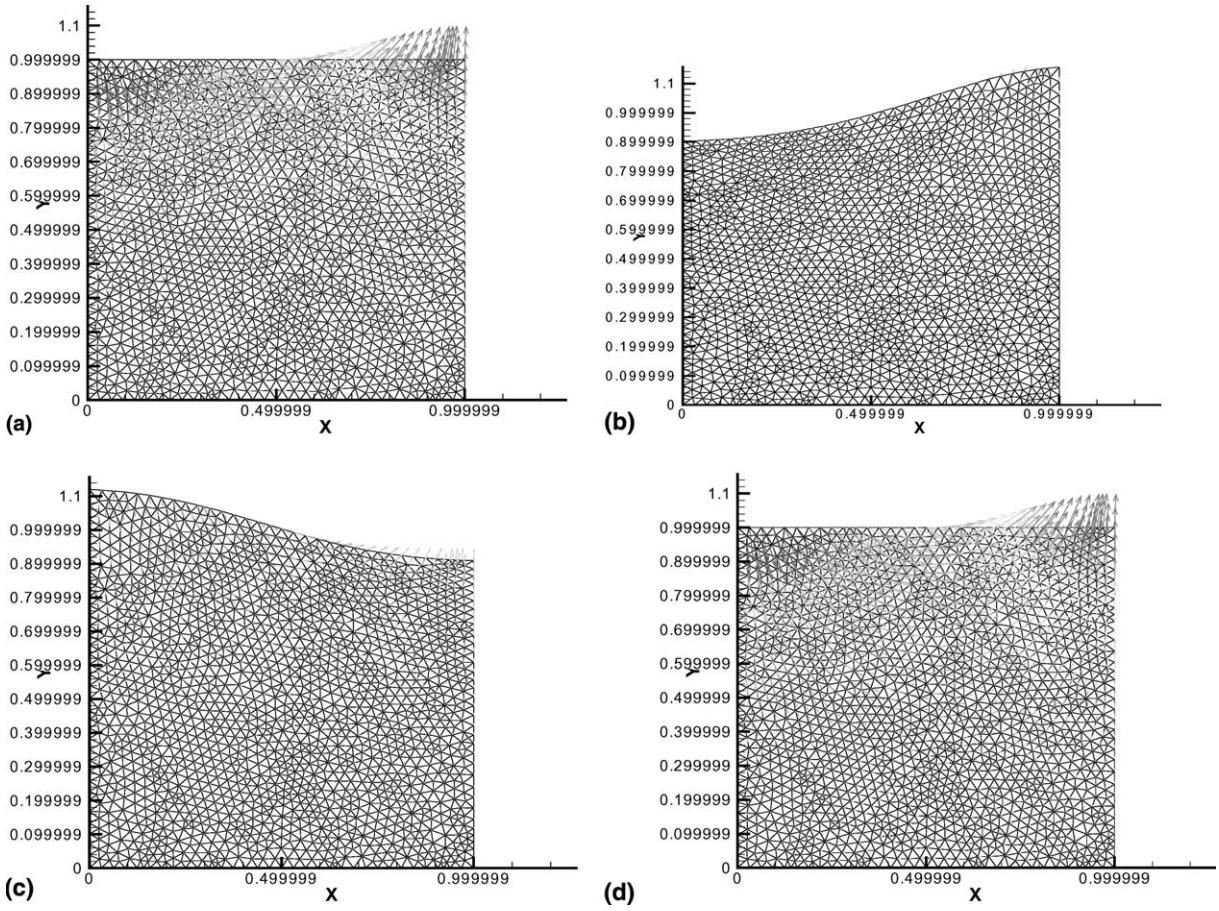


Fig. 7. Mesh and vectors colored by vertical velocity in the 2D sloshing tank. (a) $t = 0$, (b) $t = \frac{1}{4}$ cycle, (c) $t = \frac{3}{4}$ cycle, (d) $t = 7$ cycle.

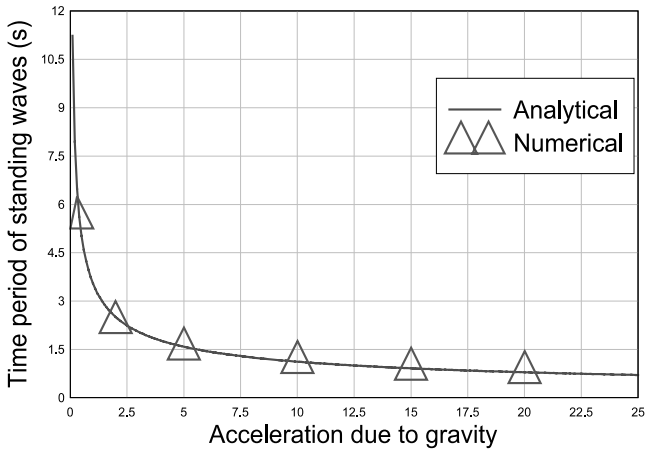


Fig. 8. Comparison of analytical and numerical results for the standing wave problem with dimensions $D = 1$ unit and $L = 1$ unit.

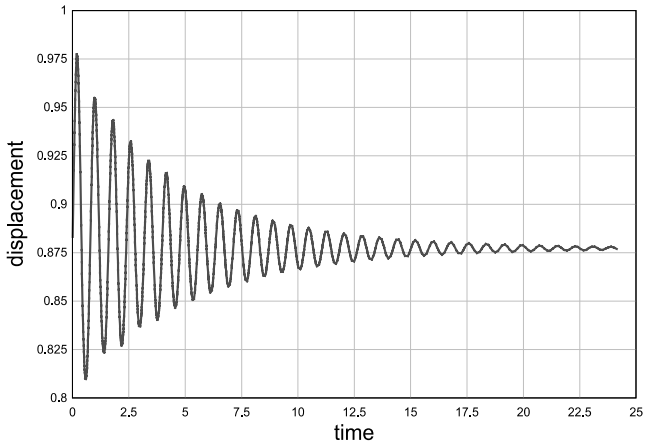


Fig. 9. Displacement of a point on the left wall (near the surface) as a function of time.

solution requires waves of infinitesimal amplitude. It was confirmed that as the initial amplitude of the wave is reduced, the error decreases.

The next case is viscous sloshing of an incompressible fluid contained in a two-dimensional rectangular tank. The computational setup is same as the one used for the problem above of inviscid sloshing. The amplitude of free-surface sloshing is expected to damp under the influence of viscosity. As an initial condition, the free-surface is assumed to be horizontal with a sinusoidal velocity field normal to the free-surface. The Reynolds number in the problem, based on the initial depth and maximum initial velocity is 100. The side walls are assumed to be slip walls while the bottom wall is assigned to be a no-slip wall. A constant pressure condition is imposed on the free-surface as before.

Fig. 9 traces the displacement of the left corner of the free-surface as a function of time. The plot clearly illustrates the viscous damping. The method is known to have zero numerical damping [15], and simulations of viscous decay of a Taylor vortex (in a periodic domain) are in excellent agreement with the analytical solution [16], so we expect that this damping rate is correct.

6.2. Droplet oscillation

Two frames from a simulation of a 3D droplet are shown in Fig. 10. The drop is subject to an initial perturbation which causes it to distort and then go into a damped oscillation. This droplet has a Weber number of 1.8 and a Reynolds number of 34 based on the drop diameter and initial perturbation velocity, this is equivalent to a 4.7 mm water drop subject to a 7 mm/s initial velocity perturbation. The initial condition and maximum distortion are shown in Fig. 10. Note that the numerical errors associated with many numerical methods will become obvious when applied to this simple problem. There can be mass loss (due to surface fitting), drift of the droplet (lack of momentum conservation), erroneous rotation (lack of vorticity conservation), or excessive damping (lack of kinetic energy conservation/numerical diffusion).

The period of oscillation of the drop is shown in Fig. 11 and is compared with the known exact solution for an infinitesimal perturbation ($T = D/U\frac{\pi}{4}We^{1/2}$) [26]. The discrepancy is due to the finite (2%) amplitude of the calculated perturbation. This solution method has been used to simulated over 15 periods of this oscillation with no discernable loss is mass, momentum, or energy (when the drop is inviscid) [27].

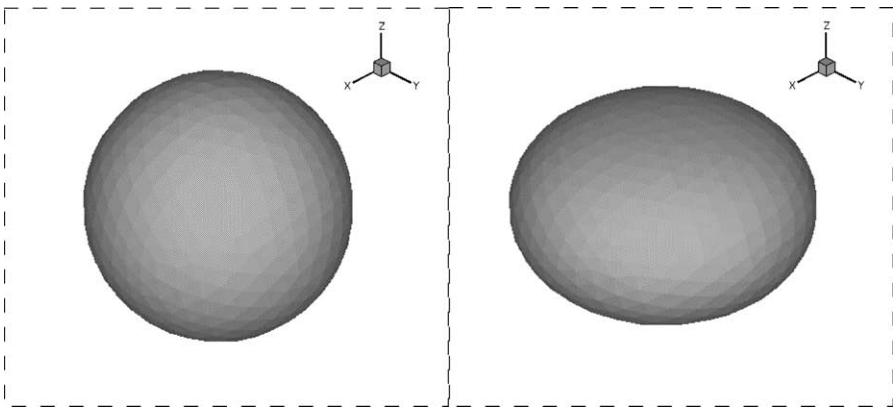


Fig. 10. Numerical simulation of a perturbed droplet. $Re = 34$, $We = 1.8$.

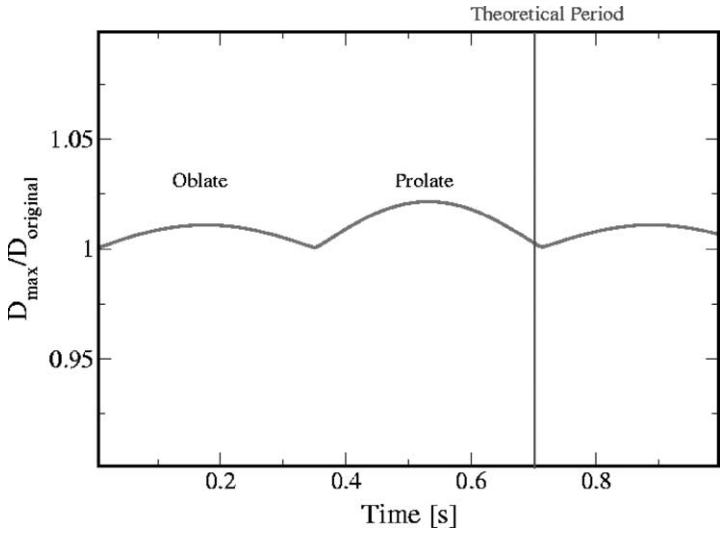


Fig. 11. Comparison of calculated and theoretical period. The calculation was performed for a single drop after a small perturbation.

6.3. Shallow water flow over a bump

Free-surface flow over a bump is the next test case. The problem consists of an open channel, and the uniform flow of an inviscid fluid over a slip-wall which has a circular arc shaped bump perpendicular to the direction of flow. The effect of the bump on the free-surface is investigated for different values of the problem parameters.

A two-dimensional triangular grid is used in this problem. The initial depth h_0 of the shallow flow is assumed to be 15.5 units and the height D of the circular arc is 3.5 units. An inflow boundary condition with a constant velocity is imposed on the left boundary. The right boundary is treated as an outflow with constant hydrostatic pressure. The bottom boundary is a slip-wall and the top boundary is a free-surface with a constant pressure boundary condition.

Analytical results are available for the present problem from shallow water equations assuming that the bump height changes very gradually. The height h of the free-surface, directly above the bump, from the bottom wall is given by, $h = h_0 + D/(1 - 1/Fr)$ where Fr, the Froude number, is given by $Fr = U_0^2/(gh_0)$

where g is the acceleration due to gravity and U_0 is the value of the constant velocity field at the inlet. One can infer from this analytical formula that when the Froude number is less than unity, the free-surface is expected to develop a depression above the bump. On the other hand, when the Froude number exceeds unity, the free surface is predicted to assume a bump-shaped elevation.

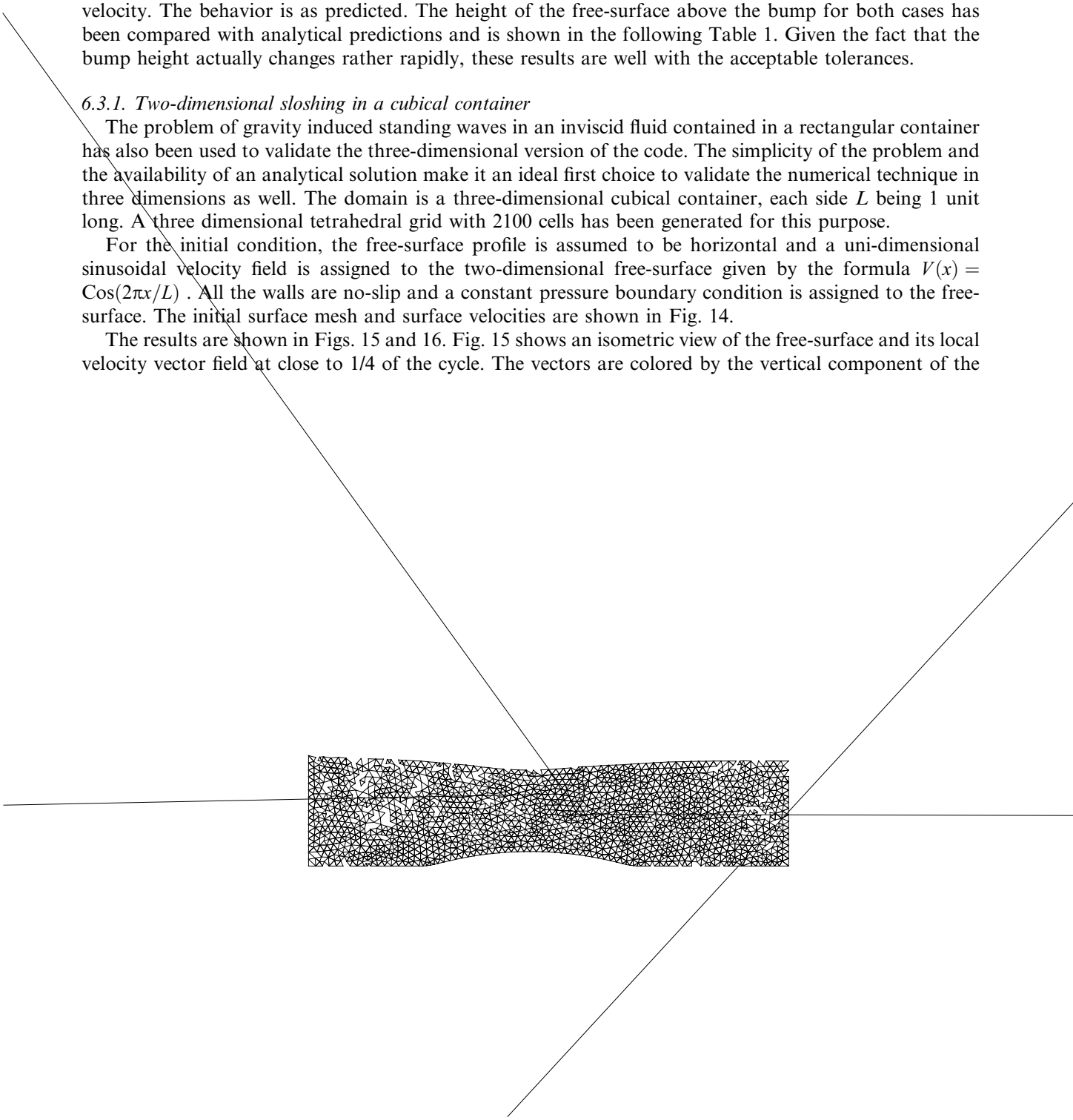
Numerical simulation is performed independently for the cases $Fr < 1$ and $Fr > 1$. Figs. 12 and 13 show the response of the free-surface to the two cases. As in the previous problems, the arrows shown in the picture represent the local velocity vectors and in this case are colored by the horizontal component of the velocity. The behavior is as predicted. The height of the free-surface above the bump for both cases has been compared with analytical predictions and is shown in the following Table 1. Given the fact that the bump height actually changes rather rapidly, these results are well with the acceptable tolerances.

6.3.1. Two-dimensional sloshing in a cubical container

The problem of gravity induced standing waves in an inviscid fluid contained in a rectangular container has also been used to validate the three-dimensional version of the code. The simplicity of the problem and the availability of an analytical solution make it an ideal first choice to validate the numerical technique in three dimensions as well. The domain is a three-dimensional cubical container, each side L being 1 unit long. A three dimensional tetrahedral grid with 2100 cells has been generated for this purpose.

For the initial condition, the free-surface profile is assumed to be horizontal and a uni-dimensional sinusoidal velocity field is assigned to the two-dimensional free-surface given by the formula $V(x) = \mathrm{Cos}(2\pi x/L)$. All the walls are no-slip and a constant pressure boundary condition is assigned to the free-surface. The initial surface mesh and surface velocities are shown in Fig. 14.

The results are shown in Figs. 15 and 16. Fig. 15 shows an isometric view of the free-surface and its local velocity vector field at close to 1/4 of the cycle. The vectors are colored by the vertical component of the



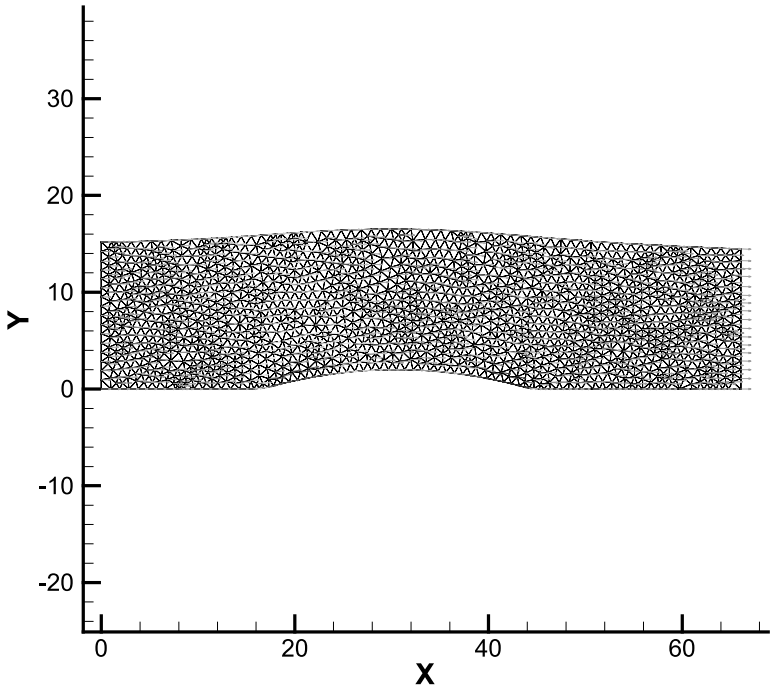


Fig. 13. Flow over a bump. $Fr = 4.5$.

Table 1
Comparison of theoretical and experimental results for the bump problem

Case	$h_{\text{theoretical}}$	$h_{\text{experimental}}$	% error
$Fr = 0.5$	12.0	12.45	3.75
$Fr = 4.5$	20.0	18.67	−6.65

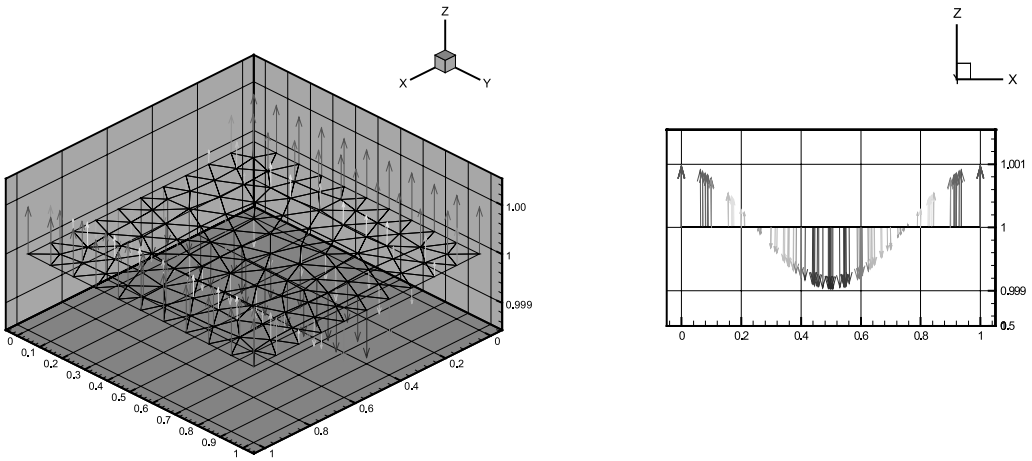


Fig. 14. Initial free-surface mesh and velocity vectors.

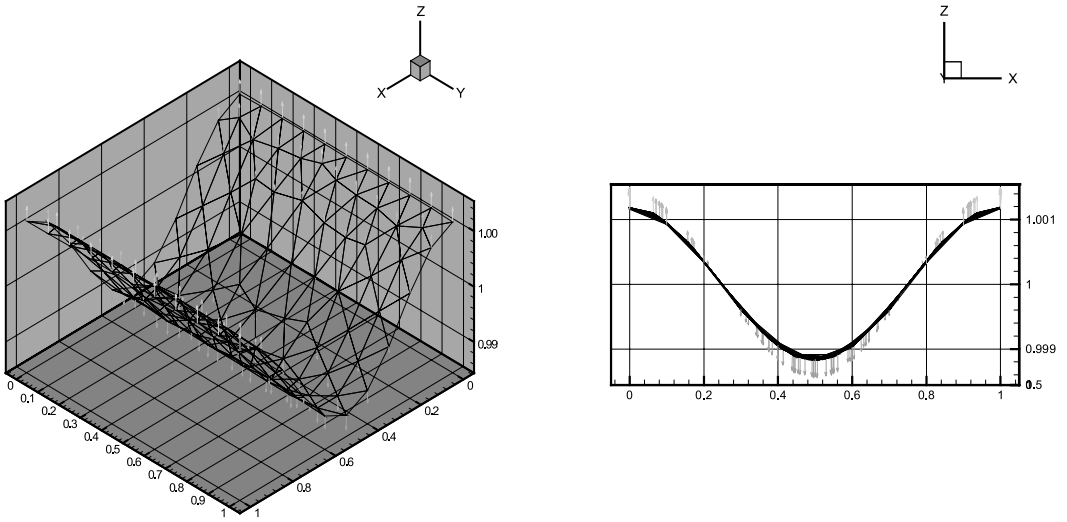


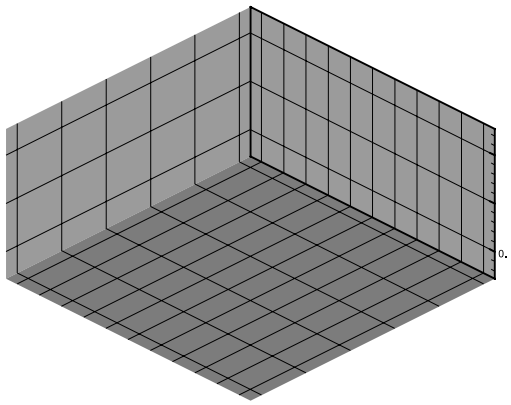
Fig. 15. Free-surface mesh and velocity vectors at one-fourth of the cycle.

velocity field. Fig. 16 is a similar view at close to 3/4 of the total cycle. The maximum percentage error in the period for this 3D simulation is 5%, which is attributed to the fact that a very coarse grid has been used in the present simulation.

6.4. Droplet impingement on a wall

This problem consists of a 2D droplet (cylinder) falling freely under gravity and colliding against a solid wall. This problem is complex since it involves changes in the boundary condition with time, significant surface tension forces, and large accelerations and deformations.

The study of droplet impingement finds its application in many industrial processes like spray painting and spray cooling, polymer applications and other industrial surfactant based processes. Although the



present problem is an over-simplified version of what in real life is a three-dimensional process, it helps us gain good insight into the numerical methods ability to capture the evolution of the droplet under the influence of a variety of forces such as gravity, surface tension and normal reaction.

A two-dimensional triangular grid is used to discretize the circular cylinder. The initial radius of the drop is assumed to be unity. The mesh near the periphery of the free-surface is made denser than the interior in order to obtain high accuracy when tracking the free surface. This is a form of user imposed mesh refinement. The spring constant for this simulation is constant and therefore the mesh adapts to the free-surface motion but not the solution (velocity gradients). The solution does not vary rapidly enough to make solution adaptation worth the effort.

As an initial condition, the droplet is assumed to be at rest. It is allowed to fall freely under the influence of gravity until it impinges on the solid wall. The boundary conditions are complicated in this problem in the sense that the boundary conditions on a few boundary nodes may vary with time. At the start of the simulation, the droplet is not in contact with the solid wall and hence the entire boundary has a constant-pressure boundary condition. However, as the droplet hits the wall, the part of the boundary that comes into contact with the wall is changed to a no-slip boundary condition. The area of this boundary and hence the number of nodes lying on it may change as the droplet deforms.

The problem is simulated for two different sets of values of the Weber number $We = V_0^2 D/\sigma$ and Reynolds number, $Re = V_0 D/v$. V_0 is the velocity of the droplet at the time of impact, D, its initial diameter, v is its kinematic viscosity, and σ is the coefficient of surface tension. In both cases, the drop falls in a vacuum and it is ensured that the Weber and Reynolds numbers are sufficiently low so that drop will not disintegrate into smaller droplets.

Fig. 17 illustrates the deformation characteristics of the droplet for a Reynolds number of 6.6 and Weber number of 2.0. The contours in the plot represent the vertical component of the local velocity vector. The figure shows that it takes some time for the influence of the impact to propagate to the upper surface of the droplet. As the droplet hits the wall it flows radially outward owing to the inertial forces. However, after a certain stage, the surface tension forces dominate and resist this radial motion. Eventually surface tension reverses the radial flow direction. In the last figure the upper surface is now moving upwards.

Fig. 18 illustrates the droplet deformation characteristics for the case where the Weber number is 5.2 and Reynolds number is 10.0. As the numbers indicate, the inertial forces are expected to play a more dominant role than the surface tension and viscous forces compared to the previous case. This is confirmed by the simulation results, and a rebound due to surface tension is not witnessed during the simulation.

6.5. Turbulence in a box

This test case involves preliminary results for turbulent fluid flow next to a free-surface boundary. This is one of the eventual applications for the method, and a difficult test case that tests both the efficiency of the method and its stability.

A grid of 12,000 tetrahedra was created in a cubical domain of unit size. The mesh is more refined based on its depth in order to adequately capture the free-surface motion. The initial stream function is prescribed by

$$\begin{aligned}\psi = 0.1\Bigg(&\left(\frac{1}{2\pi}\sin(2\pi x)\sin(2\pi y)\sin(2\pi z) + \frac{1}{3\pi}\sin(3\pi x)\sin(3\pi y)\sin(3\pi z)\right.\\ &\left.+\frac{1}{5\pi}\sin(5\pi x)\sin(5\pi y)\sin(5\pi z)\hat{j}\right) + \left(\frac{1}{2\pi}\sin(2\pi x)\sin(2\pi y)\sin(2\pi z)\right.\\ &\left.+\frac{1}{3\pi}\sin(3\pi x)\sin(3\pi y)\sin(3\pi z) + \frac{1}{5\pi}\sin(5\pi x)\sin(5\pi y)\sin(5\pi z)\right)\hat{k}\Bigg). \end{aligned} \tag{20}$$

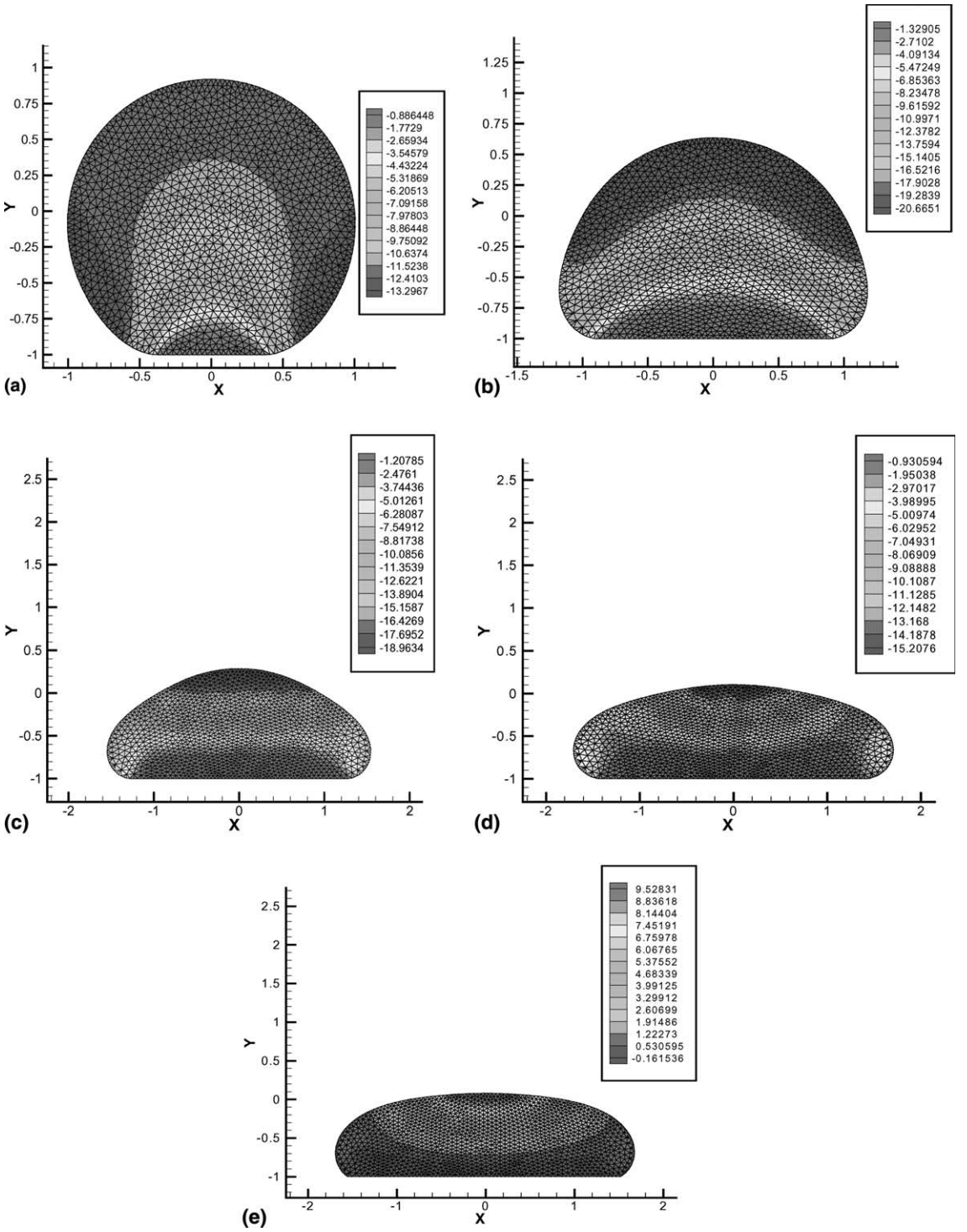


Fig. 17. Droplet profiles colored by vertical velocity ($Re = 6.6$; $We = 2.0$). (a) $T = 0.14$, (b) $T = 0.19$, (c) $T = 0.32$, (d) $T = 0.45$, (e) $T = 0.73$.

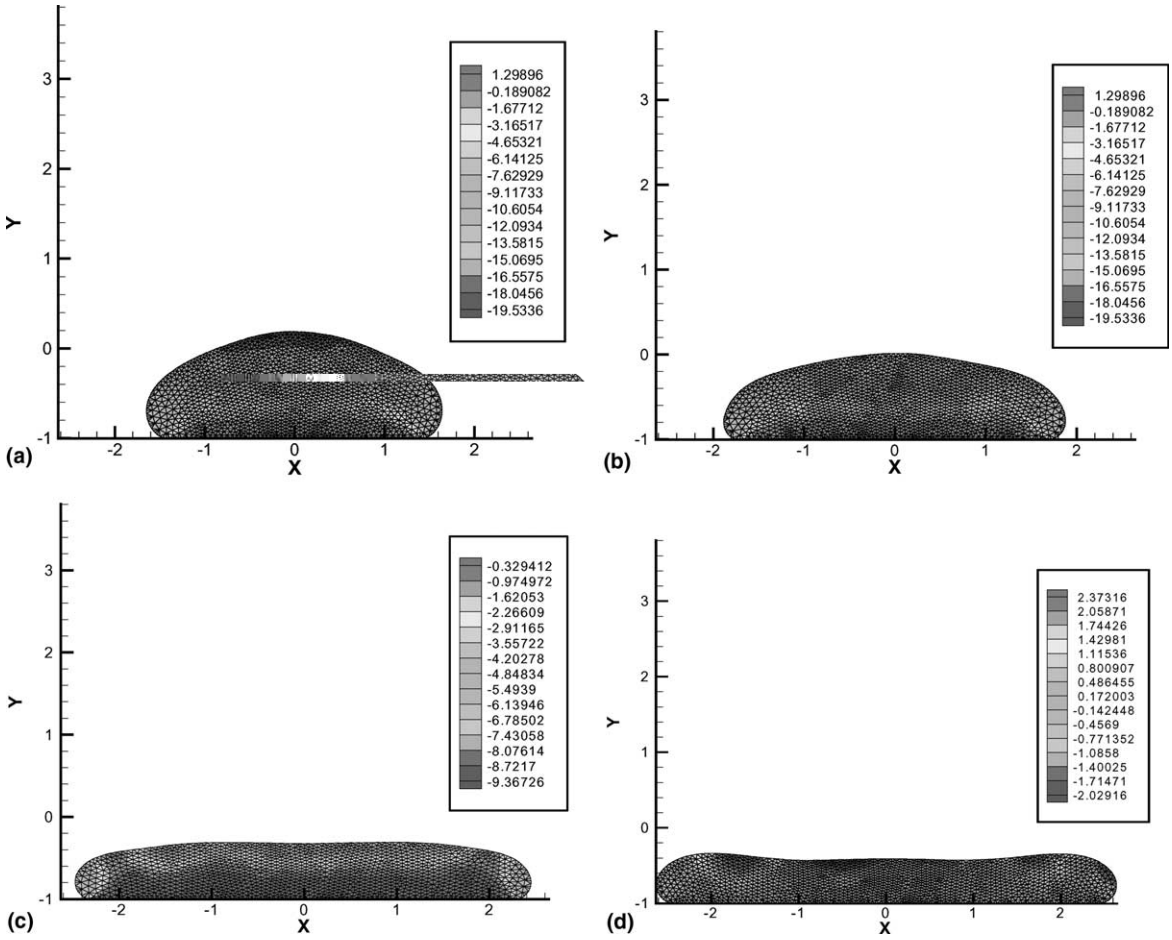


Fig. 18. Droplet profiles colored by vertical velocity ($Re = 10.0$; $We = 5.2$). (b) t = 0.47, (c) t = 0.78, (d) t = 0.91.

This initial condition is not true turbulence, but it does rapidly degenerate into a very chaotic and unpredictable flow similar to true turbulence. At the start of the simulation, the stream function is uniformly prescribed to zero on all the boundaries. The top face of the cube is assumed to be a free-surface, the side-faces are slip-walls, and the bottom face is a no-slip wall.

Fig. 19 presents the shape of the free-surface at various stages of the simulation. We notice that the disturbances initially spread to the free-surface from the interior. However, the turbulence soon decays due to the effect of the viscous forces and the free-surface slowly regains its original flat shape.

7. Conclusions

A moving unstructured staggered mesh approach has been described and evaluated for a variety of incompressible flows involving free-surfaces. The method takes advantage of the unstructured staggered mesh discretization to obtain computational efficiency, exact incompressibility, conservation of mass,

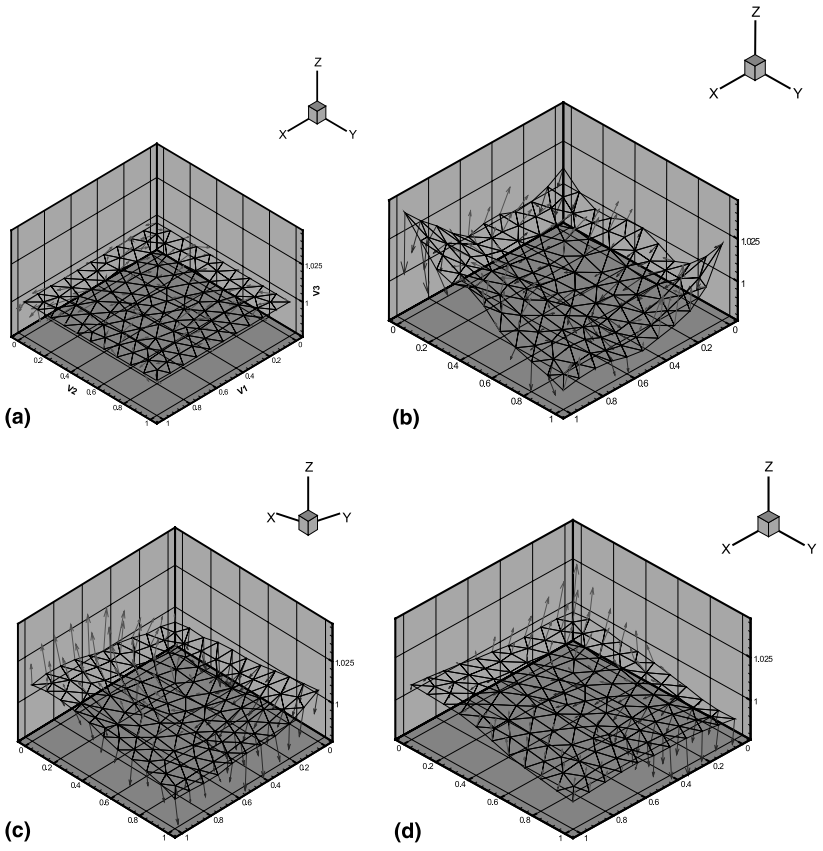


Fig. 19. Time sequence of free-surface shape at various stages during simulation.

momentum, and kinetic energy. The staggering also allows us to bypass a pressure Poisson equation and implement an exact projection method. Mesh motion is implemented in order to track the free surface motion with a very high degree of accuracy. Mesh motion also maintains a high quality mesh in the interior of highly distorting domains. The mesh motion is based on a spring analogy that allows for mesh adaptation, and which is cost effective and parallel. The adaptation algorithm maintains a fixed cost calculation, which is not always possible with point insertion algorithms. The mesh motion maintains optimal mesh quality for a fraction of the cost of global remeshing. Mesh flipping has been implemented to maintain optimal mesh connectivity. The flipping algorithm is also local, fast, and parallel.

Unstructured staggered mesh methods are not based on a control volume, and therefore their correct implementation on a moving and distorting mesh is a nontrivial problem. We have shown in this work, how to connect the classic staggered mesh discretization with a control volume formulation, and thereby include a moving mesh into the staggered mesh formulation. It is because of this formulation that the method continues to possess conservation properties even when the mesh moves.

The method was demonstrated on a number of two and three-dimensional flows. It was validated against analytical solutions and showed the ability to compute flows with a wide variety of boundary conditions and different surface physics.

Acknowledgements

This work was generously supported by a grant from The University of Massachusetts Research Council, the Office of Naval Research, and Aquasions Inc. The authors acknowledge the many useful comments of assistance of Prof. David Schmidt.

References

[1] Y. Tourigny, F. Hülsemann, A new moving mesh algorithm for the finite element solution of variational problems, SIAM J. Num. Anal. 35 (1998) 1416–1438.
[2] T.E. Tezduyar, M. Behr, J. Liou, A new strategy for finite element computations involving moving boundaries and interfaces – the DSD/ST procedure: I. The concept and the preliminary numerical tests, Comput. Meth. Appl. Mech. Eng. 94 (1992) 339–351.
[3] B. Koobus, C. Farhat, Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes, Comput. Meth. Appl. Mech. Eng. 170 (1999) 103–129.
[4] H.H. Hu, N.A. Patankar, M.Y. Zhu, Direct numerical simulations of fluid-solid systems using the arbitrary Lagrangian–Eulerian technique, J. Comput. Phys. 169 (2001) 427–462.
[5] S. Li, L. Petzold, Moving mesh methods with upwinding schemes for time-dependent PDEs, J. Comp. Phys. 131 (1997) 368–377.
[6] A. Harten, J.M. Hyman, A self-adjusting grid for the computation of weak solutions of hyperbolic conservation laws, J. Comput. Phys. 50 (1983) 235.
[7] F.H. Harlow, J.E. Welch, Numerical calculations of time dependent viscous incompressible flow of fluid with a free surface, Phys. Fluids 8 (1965) 2182.
[8] R.A. Nicolaides, The covolume approach to computing incompressible flows, in: M.Y. Hussaini, A. Kumar, M.D. Salas (Eds.), Algorithmic Trends in Computational Fluid Dynamics, Springer, Berlin, 1993, pp. 295–333.
[9] R.A. Nicolaides, Direct discretization of planar div-curl problems, ICASE Report 89–76, 1989.
[10] R.A. Nicolaides, The covolume approach to computing incompressible flow, in: M.D. Gunzburger, R.A. Nicolaides (Eds.), Incompressible Computational Fluid Dynamics, Cambridge University Press, Cambridge, 1993, pp. 234–295.
[11] R.A. Nicolaides, X. Wu, Covolume solutions of three-dimensional div-curl equations, ICASE Report 95-4, 1995.
[12] C.A. Hall, J.S. Peterson, T.A. Porsching, F.R. Sledge, The dual variable method for finite element discretizations of Navier/Stokes equations, Int. J. Num. Meth. Eng. 21 (1985) 883–898.
[13] J.C. Cavendish, C.A. Hall, T.A. Porsching, A complementary volume approach for modelling three-dimensional Navier–Stokes equations using dual Delaunay/Voronoi tessellations, Int. J. Num. Meth. Heat Fluid Flow 4 (1994) 329–345.
[14] J.B. Perot, Conservation properties of unstructured staggered mesh schemes, J. Comput. Phys. 159 (2000) 58–89.
[15] X. Zhang, D. Schmidt, B. Perot, Accuracy and conservation properties of a three-dimensional unstructured staggered mesh scheme for fluid dynamics, J. Comput. Phys. 175 (2002) 764–791.
[16] R. Nallapati, J.B. Perot, Numerical simulation of free surface flows using a moving mesh, in: Proceeedings of the 2000 American Society of Mechanical Engineers, Fluids Engineering Summer Conference, 2000.
[17] J.B. Perot, X. Zhang, Reformulation of the unstructured staggered mesh method as a classic finite volume method, in: Finite Volumes for Complex Applications, vol. II, Hermes Science Publications, 1999, pp. 263–270.
[18] W. Chang, F. Giraldo, J.B. Perot, Analysis of an exact fractional step method, J. Comput. Phys. 180 (2002) 183–199.
[19] J.C. Strikwerda, Y.S. Lee, The accuracy of the fractional step method, SIAM J. Num. Anal. 37 (1999) 37–47.
[20] J.B. Perot, An analysis of the fractional step method, J. Comput. Phys. 108 (1993) 51–58.
[21] W. Huang, R.D. Russel, Analysis of moving mesh partial differential equations with spatial smoothing, SIAM J. Num. Anal. 34 (1997) 1106–1126.
[22] W.G. Habashi, J. Dompierre, Y. Bourgault, D. Ait-Ali-Yahia, M. Fortin, M.-G. Vallet, Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD solutions: Part I: General principles, Int. J. Num. Meth. Fluids 32 (6) (2000) 725–744.
[23] B. Joe, Three-dimensional triangulations from local transformations, SIAM J. Sci. Statist. Comput. (1989) 718–741.
[24] R. Nallapati, Numerical simulation of free-surface flows using a moving unstructured staggered grid method, Masters Thesis, University of Massachussets, Amherst, 2001.
[25] F. Harlow, A. Amsden, Fluid Dynamics, an Introductory Text, Los Alamos Scientific Laboratory, LA-4700, 1980.
[26] L.D. Landau, E.M. Lifshitz, Fluid Mechanics, Pergamon Press, Oxford, 1959.
[27] M. Dai, H. Wang, J.B. Perot, D.P. Schmidt, Direct Interface Tracking of Droplets, Atomization and Sprays, (accepted March) 2002 (in press).