Russ. J. Numer. Anal. Math. Modelling, Vol. 23, No. 5, pp. 469–485 (2008)
DOI 10.1515/ RJNAMM.2008.027

Free surface flow modelling on dynamically refined hexahedral meshes

K. NIKITIN* and Yu. VASSILEVSKI*

Abstract — An efficient method for modelling incompressible free surface flows is presented. The method unites the projection method for solving the Navier–Stokes equations and the particle level set method for free surface evolution. The method uses adaptively refined hexahedral meshes built on an enhanced octree data structure.

Free surface flow simulation has increasingly attracted attention of the scientific community. It has applications in diverse disciplines, such as geophysics, petroleum engineering, biology, etc. The method discussed in this paper is designed for applications in computer graphics. It targets modelling, animating, and controlling viscous liquids in a 3D environment.

The objectives of practical computer animation impose several partly controversial requirements on simulation. It should be efficient enough to run various scenarios for acceptable time. However, it should retain enough detail to give a realistic behaviour and the possibility of flow control. Direct numerical simulation provides the desirable realism, since it is based on accurate simulation of physical processes. However, it is computationally expensive and based on the assumption that after setting an initial state the fluid is left to flow freely. This approach reduces capabilities to control the flow locally and globally. A feasible compromise between complexity, realistic behaviour, and flow control is provided by a computational technology which has been developed in the last decade by many researchers [1, 8, 14, 15, 18]. In this paper we present our version of the numerical method forming such technology. We do not address the issues of flow control here. We just note that this technology opens several possibilities to control the fluid flow [5].

The general idea of the method is to use fractional steps in order to unite the projection method for solving the Navier–Stokes equations and the particle level set method for free surface evolution. The computational efficiency of the method is based on two points. First, the pressure projection method is one of the most efficient solvers for the unsteady Navier–Stokes equations [22]. Second, the particle level set method is one of the most convenient and efficient approaches in simulation of free surface evolution [18]. The framework of fractional steps allows us to combine both approaches and gives additional control over liquid motion.

*Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow 119333, Russia
This work has been supported in part by RFBR grant 08-01-00159-a.

The projection scheme traces back to the 1960s [3, 21, 23]. It splits the time integration into two substeps. At the first substep the momentum equation is solved in order to advance the velocity field disregarding the fluid incompressibility. The momentum equation can be solved in many different ways [22]. We mention here the semi-Lagrangian approach [19, 20], since it reduces numerical dissipation and is simple in implementation. At the second substep, the velocity vector field is projected onto the subspace of divergence-free vector functions. This projection is performed by the solution of the Poisson equation for pressure correction, which is used for both pressure update and the projection. It should be stressed here that the Poisson equation appears in the scheme as a formal product of the velocity divergence and the pressure gradient operator. This remark removes the difficulties of setting boundary conditions for that equation.

Similarly to the projection technique, the level set method has a long history [15]. The method suggests representing a dynamic surface by zero level set of a special function. The surface evolves in time together with the level set function. If the function has the signed distance property, it provides a lot of useful information, such as the signed distance from a point to the surface, the normals and curvature of the surface. At each update of the level set function, it looses the signed distance property. Therefore, it should be recovered by a redistancing or reinitialization procedure, which changes the function to satisfy the signed distance property, while freezing its zero isolevel. In mathematical terms, reinitialization reduces to the solution of the Eikonal equation. The reinitialization procedure is the most difficult step of the entire technology. Among several methods proposed for reinitialization, we choose the fast sweeping method [2] due to its computational efficiency. Another technical difficulty of the level set method is a possible loss of mass in unsteady simulations. The accuracy of free surface representation decreases, as the surface elements become comparable with the mesh size. For instance, when a region of liquid breaks away during splashing, it may disappear due to insufficient resolution by the discrete level set function. In order to improve the surface resolution, we adopt the particle level set method [4]. It suggests using massless particles in order to correct the position of the free surface where the grid resolution is not sufficient. The strategy of reseeding the particles and their interaction with the level set is the cornerstone of the method.

Although the level set provides the evolution of the free surface, the dynamics of the liquid body is based on the velocity field, which satisfies the Navier–Stokes equations. The computation of the velocity uses a rough representation of the liquid body by voxels, where each grid cell is assumed to be either empty or filled completely with the liquid. A voxelized (MAC, Marker-And-Cell) grid was proposed in [6]. Our discretization of the Navier–Stokes equations uses the concept of staggered grids [11]. It suggests to assign different points of a cubic grid cell to degrees of freedom for the pressure and velocity components. The requirement of the resolution of the dynamic surface leads us to the use of dynamic grids, which may be refined or coarsened according to the surface motion. The requirement of data interpolation and an efficient mesh update at each time step is satisfied by hierarchical locally

refined grids. We adopt hexahedral meshes called octrees [7]. Octrees can exploit a special data structure, which provides a set of efficient local and global operations. Being combined with the staggered grid concept, they provide an economic distribution of the degrees of freedom in the computational domain.

These approaches form the basis of our computational technology. The details of the numerical technique are considered in the rest of the paper. In Section 2 we present the basic equations of the model. In Section 3 we discuss the fractional steps method. In Section 4 we consider discretizations of differential operators on octree meshes. Numerical results demonstrating the efficiency of the computational technology are presented in Section 5.

1. Basic equations

The flow of a viscous incompressible liquid is described by the Navier–Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \mathbf{f} \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{1.2}$$

where t is the time, $\mathbf{u} = (u, v, w)$ is the vector velocity field, p is the pressure, $\mathbf{f}$ is the external force (gravity), and $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)^T$.

The boundary $\partial\Omega$ of the domain Ω occupied by the liquid consists of two parts: the solid borders Γ_D and the moving free surface Γ. On the solid part of the boundary Γ_D the velocity field satisfies the Dirichlet boundary condition. It may be homogeneous (no-slip condition) or inhomogeneous (e.g., an inflow velocity profile). No particular boundary condition for the velocity is imposed on the free surface Γ. From the mathematical standpoint, the pressure function is the Lagrange multiplier, which is introduced in order to compensate the additional constraint (1.2) and, therefore, does not satisfy any boundary condition. On the other hand, in the pressure projection scheme presented below, the projection to the discrete divergence-free space imposes an artificial Dirichlet boundary condition for p on the free surface. More precisely, the liquid pressure should balance the air pressure and the free surface tension caused by the surface curvature.

The free liquid surface Γ is defined by the zero isolevel of the level set function: $\varphi(x, y, z) = 0$. The domain Ω occupied by the liquid is given by $\varphi < 0$, while the air domain Ω_{air} is given by $\varphi > 0$. The evolution of the free surface in space and time is set by the level set equation [15]:

$$\frac{\partial \varphi}{\partial t} + \mathbf{u} \cdot \nabla \varphi = 0. \tag{1.3}$$

If the level set function possesses the signed distance property $|\nabla \varphi| = 1$, it provides useful geometric characteristics of Γ. The distance from a point $\mathbf{x}$ to the surface is $|\varphi|$, the outward unit normal for Ω is $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$, and the local interface curvature is $\varkappa = \nabla \cdot \mathbf{n}$. The latter will be used for modelling surface tension.

2. Method of fractional steps

Numerical time integration of equations (1.1),(1.2),(1.3) can be performed implicitly or semi-implicitly via fractional steps. The fully implicit integration provides unconditional stability at the expense of several nested iterative processes [9]. This increases considerably the arithmetic complexity of the scheme. The semi-implicit integration combines faster algorithms with feasible restrictions on the time step. An example of the semi-implicit scheme is the fractional step method [8, 14]. It adopts several numerical techniques targeting efficient computation of the physically relevant discrete solution of (1.1)–(1.3). Each time step of the method consists of the following fractional steps:

1. Update of the velocity field via
   (a) Prediction of the velocity via the solution of the momentum equation (advection, diffusion and body forces),
   (b) Projection of the predicted velocity onto the divergence-free subspace;
2. Update of the level set function via
   (a) Advection along the updated velocity field,
   (b) Error correction,
   (c) Reinitialization;
3. Update of the liquid volume;
4. Adaptive remeshing of the computational domain and data interpolation.

A detailed description of the fractional steps will be given in the next sections.

2.1. Momentum equation

Momentum equation (1.1) is solved in two substeps. At the first substep we advect the velocity field ($\partial\mathbf{u}/\partial t = -(\mathbf{u}\cdot\nabla)\mathbf{u}$) and at the second substep we apply diffusion and the body forces ($\partial\mathbf{u}/\partial t = \nu\Delta\mathbf{u} - \nabla p + \mathbf{f}$).

We use a semi-Lagrangian method for the advection substep. Self-advection of the velocity would be easy to implement if the velocity were modelled as a set of particles. In this case we would simply have to trace the particles through the given velocity field shown in Fig.1a. Assume that each grid cell center is a particle and trace it through the velocity field as shown in Fig.1b. In order to avoid the conversion of moved particles back to the grid values, we search for particles which over a single time step end up exactly at grid cell centers as shown in Fig.1c. The values of the velocity components that these particles carry are interpolated from the neighbouring grid cell centers. In our discretization, the velocity components

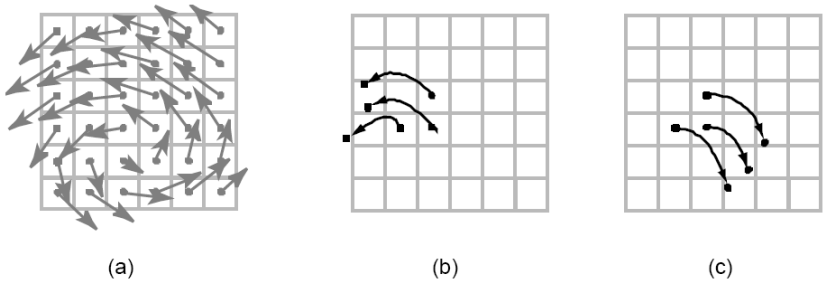

Figure 1. Interpretation of semi-Lagrangian advection: (a) the given velocity field, (b) moving the cell centers forward in time, (c) tracing the cell centers backward in time.

are stored in grid cell face's centers and we trace them backwards to find the points which end up exactly in these face centers.

At the second substep, diffusion and body forces are added via the standard explicit scheme.

2.2. Projection step

The velocity field $\mathbf{u}^*$ generated by the solution of the momentum equation is not divergence-free, and, therefore, does not characterize the incompressibility of the liquid. The idea of the projection step is to use a pressure correction (δp) for the projection of the velocity onto the divergence-free subspace:

$$\mathbf{u} = \mathbf{u}^* - \Delta t \nabla(\delta p)$$

$$p = p^* + \delta p$$

where δp satisfies the Poisson equation

$$-\nabla \cdot (\nabla \delta p) = -\frac{1}{\Delta t}(\nabla \cdot \mathbf{u}^*). \tag{2.1}$$

Equation (2.1) is to be considered as a constituent of the projection operator; at the discrete level, it may appear from algebraic arguments rather than from an approximation of a boundary value problem. The discrete Laplacian operator in (2.1) is defined as a product of two sparse matrices: discrete divergence and discrete gradient. The details of these discretizations are presented in the subsequent sections. It is well known that the finite difference discretization on uniform grids results in the product matrix which approximates the Laplacian with Neumann boundary conditions. In general, any pressure projection scheme provides a weak convergence towards the exact pressure function. Therefore, any 'boundary condition' for the pressure correction is applicable if the projection to the discrete divergence-free subspace is valid. Modifications of the product matrix related to the change of the 'boundary condition' for pressure can give additional control of the velocity field [1].

The symmetry of the product matrix depends on the methods of discretization of the divergence and gradient operators. In general, the matrix of the linear system for δp may be nonsymmetric. For the iterative solution we adopt the stabilized bi-conjugate gradient method [16] with the preconditioner based on the second-order incomplete factorization [10].

2.3. Evolution of free surface

Once the discrete divergence-free velocity field $\mathbf{u}$ is computed, the level set function is evolved according to (1.3). To this end, we adopt the semi-Lagrangian method discussed earlier. The location of the free surface evolves with the level set function φ. Using only the level set advection, however, can cause a noticeable loss of the liquid volume. This is observed when regions of liquid break away during splashing and then disappear because they are too small to be resolved by the level set function.

The loss of volume can be reduced in several ways: refining the computational grid, a more accurate time integration of equation (1.3), or introduction of particles. The first and the second approaches may increase considerably the arithmetic complexity and/or memory requirements. We adopt the third approach known as the particle level set method [4, 8]. The idea of the method is to use special massless marker particles near the interface in order to correct the level set function when it is needed.

The position of the particles evolves over time by simple advection, $\mathrm{d}\mathbf{x}_p/\mathrm{d}t = \mathbf{u}_p$ where $\mathbf{u}_p$ is the fluid velocity at point $\mathbf{x}_p$. The particle velocity is computed by interpolation from the velocity grid. The second-order Runge–Kutta integration is used to move particles in the velocity field.

To each particle we assign its coordinates $\mathbf{x}_p$, sign s_p and radius r_p. The particle sign is the sign of the level set function in the region where the particle was initially placed. Since particles can move in the close-to-interface region occupied by air, the velocity field has to be defined there. The particle radius depends on the level set function:

$$r_p = \begin{cases} r_{\max}, & s_p\varphi(\mathbf{x}_p) > r_{\max} \\ s_p\varphi(\mathbf{x}_p), & r_{\min} \leqslant s_p\varphi(\mathbf{x}_p) \leqslant r_{\max} \\ r_{\min}, & s_p\varphi(\mathbf{x}_p) < r_{\min} \end{cases}$$

and assigns a spherical level set function

$$\varphi_p(\mathbf{x}) = s_p(r_p - |\mathbf{x} - \mathbf{x}_p|).$$

Here $r_{\min}$ and $r_{\max}$ are equal to one tenth and one half of the local grid size. Note that the radius of a particle changes in time, since the particle moves in the φ-field. The radius of the particle is chosen so that particle boundary be tangent to the free surface whenever possible. In other words, the surface Γ is represented by the multiscale sampling by particles. Particles do not interact each other and thus may overlap.

A particle is called *escaped*, if it has been brought to the region where φ has the opposite sign, and the distance from the particle to the surface is greater than

its radius. For all escaped particles we initiate the *error correction* procedure rebuilding the $\varphi > 0$ region and $\varphi < 0$ region. We compare φ_p values at the eight grid points of the cell containing the particle, with the current values of φ and take the maximum value (in magnitude) as the corrected level set function. In other words, we initialize two functions $\varphi^+ = \varphi^-$ by φ and calculate separately for the escaped positive particles E^+ and the escaped negative particles E^-

$$\varphi^+ = \max_{p \in E^+}(\varphi_p, \varphi^+)$$

$$\varphi^- = \min_{p \in E^-}(\varphi_p, \varphi^-).$$

These calculations provide *two* corrected level set functions φ^+, φ^- representing the corrected regions $\varphi > 0$, $\varphi < 0$. These functions are merged to a single level set function according to

$$\varphi = \begin{cases} \varphi^+, & |\varphi^+| \leqslant |\varphi^-| \\ \varphi^-, & |\varphi^+| \geqslant |\varphi^-|. \end{cases}$$

This formula provides the minimum deviation from the original surface position due to the error correction procedure. In other words, the error correction procedure updates the free surface by dragging it to match locally φ_p for selected escaped particles.

The efficiency of the particle level set method is based on the compromise between the accuracy of the surface resolution due to the error correction and the arithmetical complexity of the particle motion. Particles should be used in the vicinity of the surface patches where the curvature is large enough. In the neighbourhood of low curvature patches the accuracy of the level set representation is high and there is no need to use particles there. The value of the curvature is used as the weighting factor for particle density. The maximum number of particles in a grid cell is set equal to 30. Particles are reseeded in the vicinity of the surface after every 20 time steps.

Both the level set advection and the error correction can cause the function to loose its signed distance property. For its recovery we should perform a redistancing step, or reinitialization. The signed distance property is formalized by a specialized Eikonal equation:

$$|\nabla \varphi(\mathbf{x})| = 1, \quad \mathbf{x} \in \Omega \cup \Omega_{\text{air}} \tag{2.2}$$

with the boundary condition on the interface Γ:

$$\varphi(\mathbf{x}) = 0, \quad \mathbf{x} \in \Gamma.$$

Prior to the solution of (2.2), the interface Γ should be determined explicitly. We recover Γ cellwise. For each cell a local internal surface triangulation is built using the marching cubes technique [12]. Remarkably, the triangulated approximation Γ^h of Γ turns to be a conformal triangulation in space.

In all grid cells intersecting Γ^h the equation (2.2) is solved explicitly: the level set φ at the cell nodes is defined as a minimal distance to Γ^h and marked with the proper sign.

In the other grid cells we apply the fast sweeping method [2]. The method is based on the nature of the Eikonal equation: at any grid cell the solution depends on the neighbours with smaller values of φ. For a cubic cell there can be only eight possible combinations of neighbours with smaller values of φ. Each combination corresponds to three cell's faces sharing a vertex and thus forms a stencil direction emanating from that vertex. Since the solution of (2.2) gives the minimum distance to Γ^h, we can solve all eight stencil directions and take the minimum of the potential solutions. If we were able to sweep over a sequence of cells such that each of them had three neighbours with the known solution, this would define a direct method of the solution of (2.2). Since such sequence is not available, the fast sweeping method suggests iteration over all cells and stencil directions. We define a sweep as the cellwise solution of (2.2), assuming a stencil direction is given. The outer iterative loop is the loop over stencil directions. The order in which the cells are visited can be arbitrary, although it may impact the convergence of the iterations.

In spite of its usual fast convergence, the fast sweeping method may stagnate. Therefore, the stopping criterion for the above iterations is not based on the residual of (2.2): the method is terminated if $\max_{\mathbf{x}} |\varphi^{n+1}(\mathbf{x}) - \varphi^n(\mathbf{x})| \leqslant \varepsilon,\ \varepsilon = 10^{-3}$.

In practice, the fast sweeping method can be complemented with another method of the solution of (2.2). The idea behind this method is to rewrite (2.2) as a time-dependent partial differential equation in the form

$$\frac{\partial \varphi}{\partial \tau} + \operatorname{sgn}(\varphi_0)(|\nabla \varphi| - 1) = 0 \tag{2.3}$$

with the homogeneous boundary condition on the interface and sgn being a smeared sign function:

$$\operatorname{sgn}(\varphi_0) = \frac{\varphi_0}{\sqrt{\varphi_0^2 + (\Delta x)^2}}.$$

This equation is assumed to converge towards a stable state where $|\nabla \varphi(\mathbf{x})| = 1$. Therefore, one can solve (2.3) by an explicit scheme to get the solution of (2.2). However, the duration of stabilization and stability restrictions on pseudo-time steps make the method far more expensive compared to the fast sweeping method.

2.4. Update of the liquid volume

The fractional step method has two representations of the liquid domain Ω. The level set zero isosurface is used for the accurate description of the air-water interface Γ. It forms the basis for the geometric characteristics of the interface and related physical effects (surface tension) and the visualization techniques (distance to the surface, normals).

The second representation is approximation by voxels, where each grid cell is assumed to be either empty or filled completely with the liquid. The voxelized grid is called MAC (Marker-And-Cell) grid [6]. In the case of cubic grid cells, this discretization gives a crude (first order) approximation of the interface. The advantage of such approach is the simple treatment of equations (1.1) and (1.2). The details of their discretization on staggered MAC grids will be given below. Here we just remark that the velocity components are associated with the cell faces of a staggered grid.

The combination of both representations allows us to obtain and track a smooth, temporally coherent liquid surface on the basis of simple spatial discretizations.

The update of the liquid volume is thus reduced to the update of the voxels and setting the velocity in newly filled voxels. A voxel is set to be filled with liquid if its intersection with the domain $\varphi < 0$ is non-empty. The velocities in the newly filled voxels have to be set in such a way that the normal and tangential stresses be zero on the faces between empty and liquid cells. In other words, the air should not inhibit the motion of the liquid, which can flow freely into empty cells. This is achieved by explicitly enforcing incompressibility [1] within each newly filled cell. In order to define the unknown velocities at cell faces, we have to assume that in some cell faces the velocities are known. This leads us to the CFL restriction onto the motion of the free surface: within a time step only one-cell layer of voxels may be newly filled with the liquid.

As it was mentioned earlier, the use of positive particles requires velocities in the air domain Ω_{air}. Particles may fly away from the free surface up to a few grid cells. Therefore, the velocity field has to be defined in this vicinity of the free surface. By analogy with (2.3), the extrapolated velocities $\mathbf{u}_{\text{air}}$ can be computed via the solution of unsteady partial differential equation

$$\frac{\partial \mathbf{u}_{\text{air}}}{\partial \tau} + \mathbf{n} \cdot \nabla \mathbf{u}_{\text{air}} = 0$$

where τ is fictitious time.

2.5. Adaptive remeshing and data interpolation

The accuracy of the free surface representation and numerical properties of the scheme (numerical dissipation) require a small enough mesh size h. Our experience shows that visually acceptable results can be obtained on a mesh with h smaller than one per cent of the characteristic size of the liquid domain. The use of uniform grids is prohibitively expensive in this case. Locally refined meshes demand less computational resources. The dynamic behaviour of the free surface requires adaptive remeshing in the course of simulation. On the other hand, the use of cubic cells is appealing due to the economic distribution of the degrees of freedom on staggered meshes. These considerations motivate the use of octree meshes [7], which are non-conformal hexahedral hierarchical meshes composed of cubic cells. The octree meshes can be locally refined and derefined. Data interpolation between

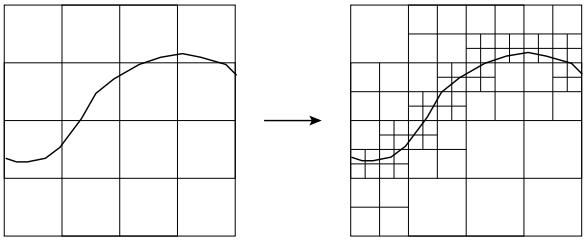

Figure 2. Simple adaptive mesh refinement.

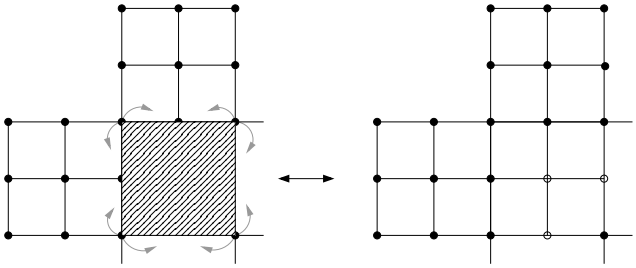

Figure 3. Coarsening and refinement of the mesh cell.

two consecutive meshes is efficient and straightforward. Moreover, due to the hierarchical structure, efficient multilevel techniques can be adopted in the solution of linear systems [13].

Our strategy of adaptive refinement and derefinement is based on three principles. First, we assume that at each time step the octree mesh becomes refined towards the free surface. The degree of refinement is controlled by the user-defined parameter $h_{\min}$. Cells located far from the surface can be coarsened up to the size $h_{\max}$. Second, the ratio of the sizes of any two neighbouring cells cannot exceed 2. This requirement stems from discretization methods discussed in the next section. Third, the refinement has priority over coarsening. This means that surface-driven splitting of a cell is imperative and thus it can cause splitting of the cell's neighbours. However, surface-driven merging cells can be cancelled if the coarsening neighbouring cells violate the second principle.

The three principles of mesh adaptation form the general framework of refinement. Several strategies may be suggested within this framework. The simplest one is to split a cell with different signs of φ in the cell vertices, provided that its size is larger than $h_{\min}$. Such aggressive refinement demonstrates a fast transition to the coarsest mesh size $h_{\max}$ (see Fig. 2). This saves a lot of degrees of freedom, but deteriorates the quality of discretization in the close-to-surface layers.

An alternative to the aggressive refinement is to split cells in a narrow layer surrounding the surface. We take advantage of the property of the level set function that $|\varphi(\mathbf{x})|$ is the distance to the surface. The average value of $|\varphi(\mathbf{x})|$ in a cell provides a sound criterion for cell splitting.

A complementary method for both strategies may be additional refinement in

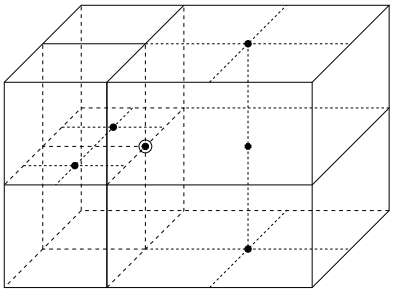

Figure 4. Interpolation of w into the marked node.

the neighbourhood of surface patches with a high local curvature.

The conventional data structure for octree meshes holds only a cell tree without connections of cells to nodes. For the sake of interpolation of nodal data in the fractional step method, the data structure has to be enhanced. In addition to the tree of cells, we also store nodes and all node-to-cell and cell-to-node connections. The interpolation between two meshes at the consecutive time steps uses this additional information (see Fig. 3). In the remeshing step all the connections are changed locally, so that the complexity of both the data structure update and the interpolation procedure is negligible.

We briefly illustrate the efficiency of the interpolation procedure using the example of velocity interpolation. As it was mentioned already, the velocity components are stored at cell mid-faces. In particular, the vertical component w is assigned to horizontal faces. The interpolation of w to the node shown in Fig. 4 uses the barycenters of the horizontal faces surrounding the node and lying in the same horizontal plane. These faces are easy to find, since one has a fast access from any node to the surrounding cells. Then the nodal velocity component is calculated by inverse distance weighting [17].

3. Discretization of differential operators

The staggered grid discretization implies that pressure degrees of freedom are assigned to cell centers, the velocity component u is assigned to the centers of faces which are orthogonal to x-axis, the velocity components v and w are assigned to faces orthogonal to y- and z-axes, respectively. Within each cell, the component u is assumed to be linear in the x-direction and constant in the y- and z-directions. The components v and w satisfy similar assumptions. The level set function φ is assumed to be a continuous, cellwise trilinear function with nodal degrees of freedom.

Discretization of the divergence operator is based on the Gauss–Ostrogradskii formula applied to a cell V

$$\int_V \nabla \cdot \mathbf{u} \, \mathrm{d}V = \int_{\partial V} (\mathbf{u} \cdot \mathbf{n}) \, \mathrm{d}s \tag{3.1}$$

where $\mathbf{n}$ represents the outward unit normal to the cell boundary. If a grid face is

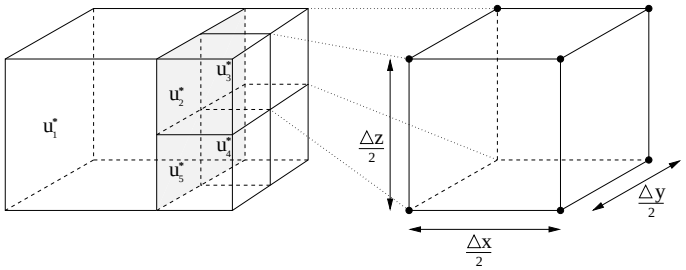

Figure 5. Bordering different sized cells.

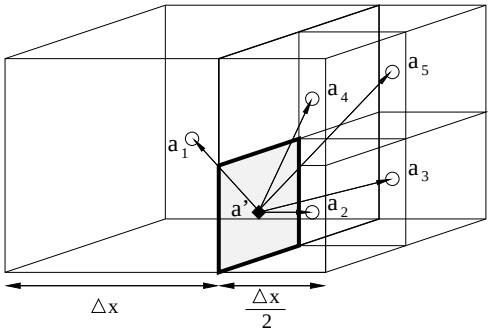

Figure 6. Discretization stencil for pressure gradient.

shared by cells of different sizes, the corresponding velocity component may be piecewise constant on this face. This should be taken into account by the formula (3.1). For instance, in the case shown in Fig. 5, one writes for $\mathbf{u} = (u,0,0)^T$

$$\Delta x\Delta y\Delta z\,\nabla\mathbf{u} = u_2A_2 + u_3A_3 + u_4A_4 + u_5A_5 - u_1A_1$$
$$A_2 = A_3 = A_4 = A_5 = A_1/4, \quad A_1 = \Delta y\Delta z.$$

The pressure gradient discretization is based on the Taylor formula. Due to the second principle of mesh refinement, for any interior face there can be only two topological cases. In the first case the face is shared by two equal cells, and we use the standard central finite difference for the corresponding gradient component. In the second case the sizes of the cells sharing the face are different. For the sake of simplicity, we discuss the discretization of the *x*-component of the gradient operator at the face center $\mathbf{a}'$ shown in Fig. 6. We consider the centers of five surrounding cells $\mathbf{a}_1,\ldots,\mathbf{a}_5$ and expand the pressure value $p(\mathbf{a}_i)$ with respect to $p(\mathbf{a}')$:

$$p(\mathbf{a}_i) = p(\mathbf{a}') + \nabla p(\mathbf{a}')\cdot(\mathbf{a}_i - \mathbf{a}') + O(|\mathbf{a}_i - \mathbf{a}'|^2).$$

Neglecting the second-order term we obtain a system of 5 linear equations with 4

unknowns

$$\begin{pmatrix} 1 & -\frac{\Delta}{2} & \frac{\Delta}{4} & \frac{\Delta}{4} \\ 1 & \frac{\Delta}{4} & 0 & 0 \\ 1 & \frac{\Delta}{4} & \frac{\Delta}{2} & 0 \\ 1 & \frac{\Delta}{4} & 0 & \frac{\Delta}{2} \\ 1 & \frac{\Delta}{4} & \frac{\Delta}{2} & \frac{\Delta}{2} \end{pmatrix} \begin{pmatrix} p(\mathbf{a}') \\ \frac{\partial p}{\partial x}(\mathbf{a}') \\ \frac{\partial p}{\partial y}(\mathbf{a}') \\ \frac{\partial p}{\partial z}(\mathbf{a}') \end{pmatrix} = \begin{pmatrix} p(\mathbf{a}_1) \\ p(\mathbf{a}_2) \\ p(\mathbf{a}_3) \\ p(\mathbf{a}_4) \\ p(\mathbf{a}_5) \end{pmatrix} \tag{3.2}$$

where $\Delta \equiv \Delta x = \Delta y = \Delta z$. The least squares solution of (3.2) is

$$\begin{pmatrix} p(\mathbf{a}') \\ \frac{\partial p}{\partial x}(\mathbf{a}') \\ \frac{\partial p}{\partial y}(\mathbf{a}') \\ \frac{\partial p}{\partial z}(\mathbf{a}') \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} & \frac{1}{6} & \frac{1}{6} & -\frac{1}{3} \\ -\frac{4}{3\Delta} & \frac{1}{3\Delta} & \frac{1}{3\Delta} & \frac{1}{3\Delta} & \frac{1}{3\Delta} \\ 0 & -\frac{1}{\Delta} & \frac{1}{\Delta} & -\frac{1}{\Delta} & \frac{1}{\Delta} \\ 0 & -\frac{1}{\Delta} & -\frac{1}{\Delta} & \frac{1}{\Delta} & \frac{1}{\Delta} \end{pmatrix} \begin{pmatrix} p(\mathbf{a}_1) \\ p(\mathbf{a}_2) \\ p(\mathbf{a}_3) \\ p(\mathbf{a}_4) \\ p(\mathbf{a}_5) \end{pmatrix}.$$

Therefore, the stencil for x-component of the gradient is

$$\frac{\partial p}{\partial x}(\mathbf{a}') \approx \frac{1}{3\Delta}(p_2 + p_3 + p_4 + p_5 - 4p_1).$$

Discretization of the Laplacian operator for the velocity components is based on the discrete Laplacian for the pressure. The latter is defined as the product of the discrete divergence and gradient operators. Interpolation from cell centers to face centers and its transpose operation define the Galerkin projection of the discrete pressure Laplacian onto the velocity space.

The numerical model includes a differential operator related to the free surface curvature $\varkappa$. The free surface tension $\sigma\varkappa$ depends on the surface tension coefficient and the second derivatives of a parametric representation of Γ. However, the use of the signed distance φ provides a simple formula for estimation of $\varkappa$, $\varkappa = \nabla \cdot (\nabla\varphi/|\nabla\varphi|)$. The curvature $\varkappa$ is obtained by computing the derivatives of φ in the cell nodes to obtain the normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$, averaging $\mathbf{n}$ components on the cell faces, averaging these values to the cell center to get $\varkappa$. The surface tension is incorporated into the model via the 'boundary condition' for the pressure function. In our implementation we set Dirichlet values $p_{\text{air}} + \sigma\varkappa$ in the air cells bordering the water.

The presented numerical model suggests several ways to control the fluid motion. It may be boundary conditions for the velocity field and even pressure (in the sense of the pressure projection scheme). The body force term conventionally represented by gravity and the pressure gradient may be modified for forcing liquid motion. Another approach is to set explicitly the velocity field of particular small pockets of liquid before the mass conservation (Poisson equation) calculation. This allows us to smooth out physical incorrectness in the applied velocities before using them in equations (1.3), (1.1).

Table 1.
Mass loss depending on the mesh size and mesh type. N_{total} and N_{water} denote the total number of cells and the number of cells filled with water, respectively.

$h_{\max}$	$h_{\min}$	δ	loss	N_{total}	N_{water}
16^{-1}	32^{-1}	0	4 %	12 524	1 168
32^{-1}	32^{-1}	0	3.86 %	51 920	1 532
16^{-1}	64^{-1}	0	0.84 %	19 160	5 012
32^{-1}	64^{-1}	0	0.82 %	57 408	5 168
16^{-1}	64^{-1}	32^{-1}	0.81 %	20 868	5 712
16^{-1}	128^{-1}	0	0.42 %	45 760	21 764
16^{-1}	128^{-1}	64^{-1}	0.40 %	54 160	25 308
16^{-1}	256^{-1}	0	0.14 %	154 428	91 052
16^{-1}	256^{-1}	128^{-1}	0.12 %	186 320	106 256
16^{-1}	512^{-1}	0	0.05 %	585 236	375 706
16^{-1}	512^{-1}	256^{-1}	0.04 %	713 441	436 620

4. Numerical experiments

In this section, we consider several test cases which demonstrate the properties of the method.

In the first test we examine the mass losses in the scheme. We consider an inviscid liquid drop of radius 0.2 centered at $(0.5, 0.5, 0.6)$ which falls in a unit cube $(0,1)^3$. The time of the motion is 0.1, the initial velocity is zero, and the acceleration is $(0,0,-1)$. The surface tension coefficient $\sigma = 0.0001$. We performed 20 time steps with $\Delta t = 0.005$ and measured the mass loss during the simulation. We examined meshes with different extents of refinement ($h_{\max}, h_{\min}$) and different strategies of refinement: $\delta = 0$ implies the aggressive refinement, $\delta > 0$ means that the mesh is refined towards the interior δ-layer of the drop surface. The data presented in Table 1 show that the parameter $h_{\min}$ is the most critical factor for the mass loss. The type of refinement makes a small impact on the mass loss, and the effect of the size of the coarsest cells is negligible. We observe that fine enough meshes may provide mass loss which is acceptable in the simulation.

In the second test case we model the same drop falling in the same unit cube. However, 20% of the cube is filled with the same liquid, and the initial position of the drop center is $(0.5, 0.5, 0.45)$. In Table 2 we present the typical CPU time measurements of a single time step and fractional substeps for meshes with different extents and strategies of refinement. The simulation was performed on a PC Pentium D, 2.8 GHz. The following notations for the CPU time measurement are used here: momentum equation solution t_{mom}, projection to divergence-free subspace t_{proj}, velocity extrapolation and evolution of the level set t_{adv}, particles integration and error correction t_{part}, reinitialization by the fast sweeping method t_{reinit}, update of the liquid volume and remeshing t_{remesh}, total time of the time step t_{total}. The table proves the arithmetic scalability of each fractional substep with respect to the number of cells filled with water N_{water}. The most expensive procedure ($\sim 40\%$) is the reinitialization step. The solution of the stiff problem (2.1) with the ILU preconditioner

Table 2.
CPU time (in seconds) of a single time step and individual substeps against the number of cells.

$h_{\max}$ $h_{\min}$	16^{-1} 64^{-1}	16^{-1} 64^{-1}	16^{-1} 128^{-1}	16^{-1} 128^{-1}	16^{-1} 256^{-1}	16^{-1} 512^{-1}
δ	0	32^{-1}	0	64^{-1}	0	0
N_{total}	31 592	41 567	96 440	141 919	352 997	1 379 239
N_{water}	11 636	19 572	63 994	67 277	243 534	729 624
t_{mom} (%)	0.06 (5.8%)	0.08 (6.2%)	0.25 (6.3%)	0.30 (5.7%)	1.03 (6.3%)	3.56 (5.7%)
t_{proj} (%)	0.17 (17.6%)	0.28 (20.9%)	1.02 (25.0%)	1.12 (21.1%)	4.00 (24.3%)	11.90 (19.1%)
t_{adv} (%)	0.05 (4.8%)	0.06 (4.5%)	0.17 (4.2%)	0.23 (4.4%)	0.73 (4.5%)	2.96 (4.7%)
t_{part} (%)	0.16 (16.7%)	0.16 (12.3%)	0.74 (18.3%)	0.71 (13.4%)	2.96 (17.9%)	12.97 (20.8%)
t_{reinit} (%)	0.44 (44.6%)	0.62 (46.3%)	1.5 (38.2%)	2.43 (45.8%)	6.34 (38.5%)	24.32 (38.9%)
t_{remesh} (%)	0.10 (10.6%)	0.13 (9.8%)	0.33 (8.0%)	0.51 (9.6%)	1.40 (8.5%)	6.75 (10.8%)
t_{total}	0.98	1.34	4.07	5.30	16.47	62.46

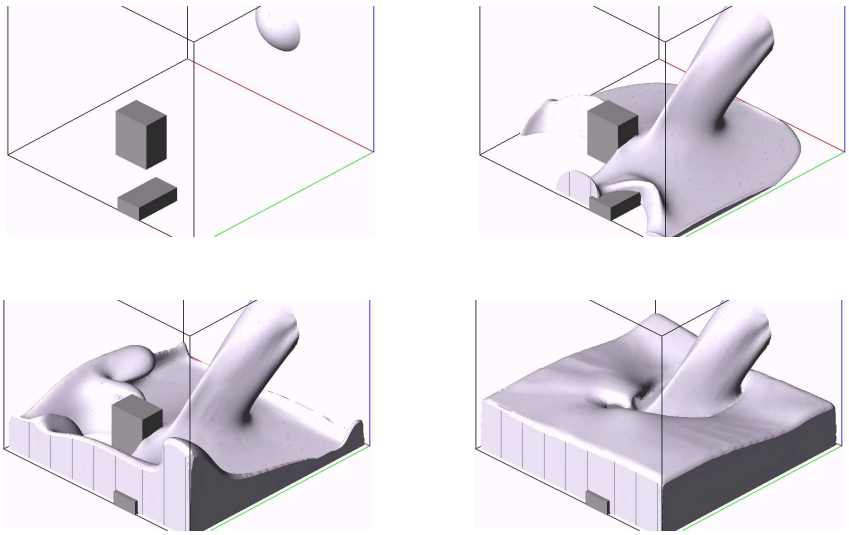

Figure 7. A box being filled with liquid. Time steps: $t_1 = 0$, $t_2 = 1.5$, $t_3 = 3$, $t_4 = 7$.

does not exceed 25% of the overall overheads. Therefore, there is no need to use the asymptotically optimal multilevel preconditioner [13] in this particular simulation. Nevertheless, the use of the multilevel methods may further accelerate the solution process.

In the third test case we simulate filling an empty container with an almost inviscid liquid. Several obstacles are put on the floor of the container. Figure 7 exhibits a few frames of the corresponding animation movie. The evolution of the free surface is complicated by the presence of obstacles.

5. Conclusions

We have presented an efficient computational technology for free surface flow simulation. Its arithmetical scalability is based on the fractional steps method and the use of octrees, i.e., dynamically refined and coarsened hexahedral meshes. The technology provides a toolkit which may be useful in computer graphics applications.

Acknowledgement

The authors are thankful to V. Leschinkskii, M. Olshanskii and A. Danilov for fruitful discussions and technical help.

References

1. M. Carlson, Rigid, melting, and flowing fluid. *PhD thesis.* Georgia Institute of Technology, 2004.
2. T. C. Cecil, S. J. Osher, and J. Qian, Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension. *J. Comp. Phys.* (2006) **213**, 458–473.
3. A. Chorin, Numerical solution of the Navier–Stokes equations. *Math. Comp.* (1968) **22**, 745–762.
4. D. Enright, F. Losasso, and R. Fedkiw, A fast and accurate semi-Lagrangian particle level set method. *Comp. Struct.* (2005) **83**, 479–490.
5. N. Foster and R. Fedkiw, Practical animation of liquids. *SIGGRAPH*, 2001, pp. 15–22.
6. F. Harlow and J. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids* (1965) **8**, 2182–2189.
7. F. Losasso, F. Gibou, and R. Fedkiw, Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics (TOG)*, August 2004, Vol. 23.
8. D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, A hybrid particle level set method for improved interface capturing. *J. Comp. Phys.* (2002) **183**, 83–116.
9. S. Gross, V. Reichelt, and A. Reusken, A finite element based level set method for two-phase incompressible flows. *Computing and Visualization in Science* (2006) **9**, 239–257.
10. I. Kaporin, High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition. *Numer. Linear Algebra Appl.* (1998) **5**, 483–509.
11. V. Lebedev, Difference analogies of orthogonal decompositions of basic differential operators

and some boundary value problems, I. *Sov. Comp. Math. Math. Phys.* (1964) **4**, 449–465 (in Russian).

12. W. Lorensen and H. Cline, Marching cubes: a high resolution 3D surface construction algorithm. *Comp. Graphics* (1987) **21**, No. 4, 163–169.
13. S. Nepomnyaschikh, Fictitious space method on unstructured meshes. *East-West J. Numer. Math.* (1995) **3**, No. 1, 71–79.
14. S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces.* Springer-Verlag, New York, Applied Mathematical Sciences, 2002, Vol. 153.
15. S. Osher and J. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi equations. *J. Comp. Phys.* (1988) **79**, 12–49.
16. Y. Saad, *Iterative methods for sparse linear systems, Second Edition.* Philadelphia, 2003.
17. D. Shepard, A two dimensional interpolation function for regularly spaced data. In: *Proc. 23d National Conf. of the Association for Computing Machinery*, Princeton, NJ, ACM, 1968, pp. 517–524.
18. J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, Cambridge, 1999.
19. A. Staniforth and J. Cotè, Semi-Lagrangian integration schemes for atmospheric models – a review. *Mont. Weath. Review* (1991) **119**, 2206–2223.
20. J. Strain, Semi–Lagrangian methods for level set equations. *J. Comp. Phys.* (1999) **151**, 498–533.
21. R. Temam, Sur l'approximation des équations de Navier–Stokes. *C.R.Acad. Sci. Paris, Série A* (1966) **262**, 219–221.
22. S. Turek, *Efficient Solvers for Incompressible Flow Problems: an Algorithm Approach in View of Computational Aspects.* Springer-Verlag, Berlin-Heidelberg, 1999.
23. N. N. Yanenko, *The Method of Fractional Steps (The Solution of Problems of Mathematical Physics in Several Variables)*. Nauka, Novosibirsk, 1967 (in Russian).