

Calculation of capillary jet

Kenn K. Q. Zhang^{*,†}

Department of Physics, University of Illinois at Chicago, 845 West Taylor Street, Chicago, IL 60607, U.S.A.

SUMMARY

The Arbitrary Lagrangian Eulerian (ALE) framework coupled with some boundary tracking techniques is proven to be an effective method for simulation of free-surface flows. In this paper, a special ALE framework is derived with clarification of three velocities, the notion of mesh-frozen and field-frozen, and the notion of tentatively inertial coordinates. A weighted integral ALE governing equations are formulated on generic coordinates and discretized with a finite element method and linear implicit time scheme. The system is solved with a discrete operator splitting technique and superposition-based logistic parallelization. The formulation and implementation are verified through several fixed-geometry problems and a reasonably good parallel performance is observed. Capillary jet flow is the main problem of the paper and the numerical techniques for boundary tracking are elaborated, which include an indirect boundary tracking of flux method and an iterative direct boundary tracking method. Also, a high-order compact scheme for dynamic boundary condition and a squeeze technique for kinematic boundary condition are adopted. The axisymmetric jet breakup is studied in detail and numerical results match with the published data very well. Numerical accuracy and sensitivity are studied, including effects of element type, time scheme, compact scheme, and boundary tracking techniques. Copyright © 2010 John Wiley & Sons, Ltd.

Received 9 March 2010; Revised 3 June 2010; Accepted 5 June 2010

KEY WORDS: discrete operator splitting; logistic parallelization; capillary; free-surface; ALE; linear implicit

1. INTRODUCTION

A large number of engineering problems and physical phenomena in areas such as fluid mechanics combustion and material science involve the motion of moving boundaries, which may be either captured on a fixed eulerian grid or tracked in a lagrangian manner. In the former approach, the background domain mesh is fixed and the boundary is moving over the mesh and is captured directly (as in the Immersed Boundary Method [1, 2]) or indirectly through domain (as in Volume Of Fluids [3] and the Level Set Method [4]). In the latter approach, the moving boundary is tracked in a lagrangian manner and the domain can be re-meshed as the boundary evolves, in which the conversion of data associated with the old mesh to the new one is necessary. More generally, especially for the latter approach, one may use the Arbitrary Lagrangian Eulerian (ALE) framework [5–12] for the domain. With ALE, governing equations are formulated on moving meshes so that mesh regeneration and interpolation can be avoided for the most of time and moving boundaries can be tracked more accurately. What needs to be first clarified here is that although ALE is closely associated with boundary tracking, it itself is not a tracking method. ALE deals with the formulation on moving meshes inside the domain, not on the boundary. With

^{*}Correspondence to: Kenn K. Q. Zhang, Department of Physics, University of Illinois at Chicago, 845 West Taylor Street, Chicago, IL 60607, U.S.A.

[†]E-mail: kenn.kq.zhang@gmail.com

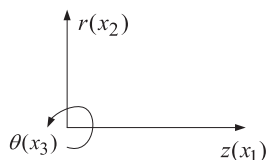


Figure 1. Generic coordinates.

ALE, boundary tracking is accomplished either by direct boundary tracking techniques (that is, the kinematic boundary condition is imposed directly on boundaries) or by indirect boundary tracking (IBT) techniques (that is, the kinematic boundary condition is imposed through the domain). In ALE, how the mesh moves is not driven by the underlying physics, but prescribed by selection. In other words, the motion of domain nodes has no necessary connection to the moving boundary, and as a corollary the ALE can be equally applied to fixed-geometry problems featuring rapid change of field quantities.

In this paper, the main task is on solving incompressible jet flows (especially the axisymmetric jet breakup) in a special ALE framework. One goal of calculating this flow is toward the more complex electrified jet breakup [13], which produces micro to nano encapsulation with important practical significance. The flow configuration and boundary tracking techniques will be elaborated later. The flow in the domain is governed by non-dimensional Navier–Stokes equations whose vector form reads

$$\nabla \cdot \vec{u} = 0, \quad (1a)$$

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = \nabla \cdot \vec{\sigma} + \vec{f}. \quad (1b)$$

To handle axisymmetric jet breakup, the generic coordinates are introduced for this purpose as shown in Figure 1. Equations (1a) and (1b) are then generalized to

$$\frac{\partial u_j}{\partial x_j} + c_a \frac{u_2}{x_2} = 0, \quad (2a)$$

$$\begin{aligned} \frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} - f_i = & -\frac{\partial p}{\partial x_i} + \frac{1}{Re} \frac{\partial}{\partial x_j} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \frac{c_a}{Re} \frac{1}{x_2} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) \delta_{1i} \\ & + \frac{2c_a}{Re} \frac{1}{x_2} \left(\frac{\partial u_j}{\partial x_j} + \frac{\partial u_2}{\partial x_2} \right) \delta_{2i}, \end{aligned} \quad (2b)$$

where $c_a = 0$ for cartesian coordinates and $c_a = 1$ for axisymmetric coordinates. A discrete operator splitting technique is used to handle incompressibility, together with a superposition-based logistic parallelization.

The paper is outlined as follows. In the next section, the discrete ALE form of the Navier–Stokes equations in generic coordinates is detailed. Then, some aspects of system solving are addressed, including handling of incompressibility and the logistic parallelization. After that, the tracking techniques for capillary jet are elaborated. Finally, numerical results and some numerical analysis are presented. The focus of this paper is on numerical aspects.

2. SYSTEM FORMULATION

2.1. Special ALE framework

Discrete equations that describe fluid particles are usually presented at nodes. This is true for both node-based finite difference method and element-based finite volume, finite element, and collocation/pseudo spectral element methods, but not true for galerkin spectral method and

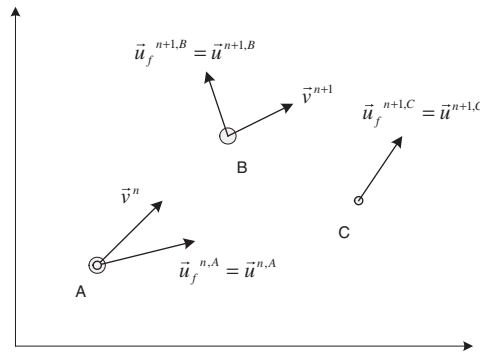


Figure 2. Illustration of node velocity \vec{v} , field velocity \vec{u} , and fluid velocity \vec{u}_f in universal fixed inertial coordinates.

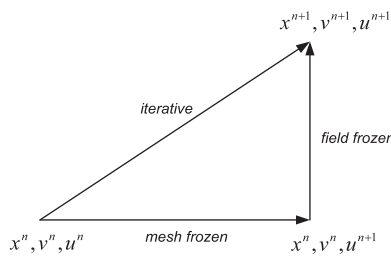


Figure 3. An illustration of time marching process under Arbitrary Lagrangian Eulerian framework.

boundary element method. In lagrangian description, nodes are always the moving material points; in eulerian description, nodes are fixed in space and not material points in general; in ALE description, nodes are moving in space and not material points in general. We shall define three ‘velocities’ in the current context. ‘Node velocity’ \vec{v} refers to the velocity of massless geometric nodes and the rule of node motion can be prescribed at will; ‘fluid velocity’ \vec{u}_f refers to the material velocity of fluid particles; ‘field velocity’ \vec{u} refers to the fluid velocity measured at nodes and it differs from fluid velocity in derivatives. Figure 2 illustrates three types of velocities. On time level n , a fluid particle resides on the top of a node at position A . On time level $n + 1$, the node arrives at the new position B whereas the fluid particle arrives at the new position C . The figure shows that the field velocity and the fluid velocity are always exactly the same in the zero derivative. However, due to the node-based nature of the field velocity

$$\frac{d\vec{u}}{dt} = \lim_{\Delta t \rightarrow 0} \frac{\vec{u}^{n+1,B} - \vec{u}^{n,A}}{\Delta t}, \quad \frac{d\vec{u}_f}{dt} = \lim_{\Delta t \rightarrow 0} \frac{\vec{u}_f^{n+1,C} - \vec{u}_f^{n,A}}{\Delta t},$$

hence $(d\vec{u}/dt) \neq (d\vec{u}_f/dt)$. In the lagrangian framework, node velocity, fluid velocity, and field velocity coincide. In the eulerian framework, field points coincide with nodes that are fixed in space; hence the field velocity is customarily called as ‘local velocity’. In the ALE framework the field velocity, measured at moving nodes, is no longer local.

Figure 3 shows one marching step of the field and the mesh from an old time level to a new one. This time marching step can be treated as an iterative process that in principle is exact. Alternatively, a divide-and-conquer methodology can be adopted. The marching step can be divided into two stages, a mesh-frozen stage and a field-frozen stage. During the mesh-frozen stage, \vec{x}^n and \vec{v}^n remain the same whereas \vec{u}^n evolves to \vec{u}^{n+1} ; during the field frozen stage, \vec{u}^{n+1} remains the same whereas \vec{x}^n and \vec{v}^n evolve. So far all velocities are defined in the universal fixed inertial coordinates, in which all nodes are moving. We have elaborated that field velocities are measured at nodes; hence, the universal coordinates system is convenient for eulerian description but not for ALE description. In convenience of the field velocity, *tentatively inertial coordinates* are

introduced. Each node is attached with such coordinates, which remain inertial and translating during the mesh-frozen stage of the time integration interval. At the end of the mesh-frozen stage, the coordinates abruptly jump to the next one. Since all nodes that are moving in universal fixed inertial coordinates always stay at rest in their own attached tentatively inertial coordinates, we can go back to eulerian description during each time interval. Each node remains at constant speed in the universal fixed inertial coordinates during the time interval. For the next time interval, the node remains at another constant speed. By analogy, the tentatively inertial coordinates are compared with the piecewise line segment used to linearize a curve.

Now we have two sets of coordinates, the universal fixed inertial coordinates x_j and the tentatively inertial coordinates x'_j moving at constant speed \vec{v}^n :

$$x_j = x'_j + v^n t.$$

Note that \vec{u} and \vec{v}^n are velocities defined in the universal fixed inertial coordinates, and are used with the same quantities in the tentatively inertial coordinates. The rate of change of any material $\phi = \phi(t, x_j)$ can be expressed in both coordinates:

$$\frac{\delta\phi}{\delta t} = \frac{\partial\phi}{\partial t} + \frac{\partial\phi}{\partial x_j} \frac{\delta x_j}{\delta t} = \frac{\partial\phi}{\partial t} + \frac{\partial\phi}{\partial x'_j} \frac{\delta x'_j}{\delta t}.$$

Some useful relations are

$$\frac{\partial\phi}{\partial x'_j} = \frac{\partial\phi}{\partial x_j}, \quad \frac{\delta x_j}{\delta t} = \vec{u}, \quad \frac{\delta x'_j}{\delta t} = \vec{u} - \vec{v}^n.$$

Therefore, in tentatively inertial coordinates moving at constant speed

$$\frac{\delta\phi}{\delta t} = \frac{\partial\phi}{\partial t} + (\vec{u} - \vec{v}^n) \cdot \nabla\phi.$$

Similarly, the momentum equation in tentatively inertial coordinates becomes

$$\frac{\partial\vec{u}}{\partial t} + (\vec{u} - \vec{v}^n) \cdot \nabla\vec{u} = \nabla \cdot \vec{\sigma} + \vec{f}. \quad (3)$$

The formulation here is restricted to the tentatively inertial coordinates; consequently, it bears the name *special ALE*.

2.2. Discrete weighted integral ALE equation

The weighted integral ALE governing equation for the mesh-frozen stage can be obtained from Equations (3) and (1a):

$$\int_{\Omega^{*,n}} q \nabla \cdot \vec{u} \, d\Omega^* + \int_{\Omega^{*,n}} \vec{w} \cdot \left[\frac{\partial\vec{u}}{\partial t} + (\vec{u} - \vec{v}^n) \cdot \nabla\vec{u} - \nabla \cdot \vec{\sigma} - \vec{f} \right] d\Omega^* = 0, \quad (4)$$

where \vec{w} and q are the weight functions for the velocity and the pressure, respectively, and Ω^* is the integration domain in generic coordinates. Owing to the independence of weight functions from time and the mesh-frozen nature

$$\int_{\Omega^{*,n}} \vec{w} \cdot \frac{\partial\vec{u}}{\partial t} \, d\Omega^* = \int_{\Omega^{*,n}} \frac{\partial}{\partial t} (\vec{w} \cdot \vec{u}) \, d\Omega^* = \frac{d}{dt} \int_{\Omega^{*,n}} \vec{w} \cdot \vec{u} \, d\Omega^*.$$

Therefore, the corresponding weighted integral ALE equation for Equations (2a) and (2b) is

$$\begin{aligned}
 &-(2\pi)^{ca} \int_{\Omega^n} q \left(\frac{\partial u_j}{\partial x_j} + c_a \frac{u_2}{x_2} \right) x_2^{ca} d\Omega + (2\pi)^{ca} \frac{d}{dt} \int_{\Omega^n} w_i u_i x_2^{ca} d\Omega \\
 &+ (2\pi)^{ca} \int_{\Omega^n} w_i (u_j - v_j^n) \frac{\partial u_i}{\partial x_j} x_2^{ca} d\Omega - (2\pi)^{ca} \int_{\Omega^n} w_i f_i x_2^{ca} d\Omega + (2\pi)^{ca} \int_{\Omega^n} w_i \frac{\partial p}{\partial x_i} x_2^{ca} d\Omega \\
 &- \frac{(2\pi)^{ca}}{Re} \int_{\Omega^n} w_i \frac{\partial}{\partial x_j} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) x_2^{ca} d\Omega - c_a \frac{(2\pi)^{ca}}{Re} \int_{\Omega^n} w_1 \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) d\Omega \\
 &- 2c_a \frac{(2\pi)^{ca}}{Re} \int_{\Omega^n} w_2 \left(\frac{\partial u_j}{\partial x_j} + \frac{\partial u_2}{\partial x_2} \right) d\Omega = 0,
 \end{aligned}$$

where Ω is the integration domain in cartesian coordinates. After integration by parts and some elementary operations, the weighted integral ALE equation is recast into

$$\begin{aligned}
 &-\int_{\Omega^n} q \left(\frac{\partial u_j}{\partial x_j} + c_a \frac{u_2}{x_2} \right) x_2^{ca} d\Omega + \frac{d}{dt} \int_{\Omega^n} w_i u_i x_2^{ca} d\Omega + \int_{\Omega^n} w_i (u_j - v_j^n) \frac{\partial u_i}{\partial x_j} x_2^{ca} d\Omega \\
 &-\int_{\Omega^n} \frac{\partial w_i}{\partial x_i} p x_2^{ca} d\Omega - c_a \int_{\Omega^n} w_2 p d\Omega + \frac{1}{Re} \int_{\Omega^n} \frac{\partial w_i}{\partial x_j} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) x_2^{ca} d\Omega - \frac{2c_a}{Re} \int_{\Omega^n} w_2 \frac{\partial u_j}{\partial x_j} d\Omega \\
 &= \oint_{\Gamma^n} w_i T_i x_2^{ca} d\Gamma + \int_{\Omega^n} w_i f_i x_2^{ca} d\Omega, \tag{5}
 \end{aligned}$$

where $T_i \equiv \sigma_{ij} n_j$ stands for the surface traction. Owing to discontinuous constant distribution for the pressure, Equation (5) is simplified to

$$\begin{aligned}
 &-q \oint_{\Gamma^n} n_j u_j x_2^{ca} d\Gamma + \frac{d}{dt} \int_{\Omega^n} w_i u_i x_2^{ca} d\Omega + \int_{\Omega^n} w_i (u_j - v_j^n) \frac{\partial u_i}{\partial x_j} x_2^{ca} d\Omega \\
 &- p \oint_{\Gamma^n} n_i w_i x_2^{ca} d\Gamma + \frac{1}{Re} \int_{\Omega^n} \frac{\partial w_i}{\partial x_j} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) x_2^{ca} d\Omega - \frac{2c_a}{Re} \int_{\Omega^n} w_2 \frac{\partial u_j}{\partial x_j} d\Omega \\
 &= \oint_{\Gamma^n} w_i T_i x_2^{ca} d\Gamma + \int_{\Omega^n} w_i f_i x_2^{ca} d\Omega. \tag{6}
 \end{aligned}$$

Let all quantities in Equation (6) be expressed in interpolation functions:

$$u_i = u_i^l(t) \psi_l(\vec{x}), \quad w_i = w_i^m \psi_m(\vec{x}), \quad u_j = u_j^r \psi_r, \quad f_i = f_i^l \psi_l, \quad T_i = T_i^l \psi_l,$$

then

$$\begin{aligned}
 &q \left(- \oint_{\Gamma^n} n_j \psi_l x_2^{ca} d\Gamma \right) u_j^l + w_i^m \left(\int_{\Omega^n} \psi_m \psi_l x_2^{ca} d\Omega \right) \frac{du_i^l}{dt} \\
 &+ w_i^m \left(\int_{\Omega^n} \psi_m \psi_r \frac{\partial \psi_l}{\partial x_j} x_2^{ca} d\Omega \right) (u_j^r - v_j^{r,(n)}) u_i^l + w_i^m \left(- \oint_{\Gamma^n} n_i \psi_m x_2^{ca} d\Gamma \right) p \\
 &- w_i^m \left(- \frac{1}{Re} \int_{\Omega^n} \frac{\partial \psi_m}{\partial x_j} \frac{\partial \psi_l}{\partial x_j} x_2^{ca} d\Omega \right) u_i^l - w_i^m \left(- \frac{1}{Re} \int_{\Omega^n} \frac{\partial \psi_m}{\partial x_j} \frac{\partial \psi_l}{\partial x_i} x_2^{ca} d\Omega \right) u_j^l \\
 &- c_a w_2^m \left(\frac{2}{Re} \int_{\Omega^n} \psi_m \frac{\partial \psi_l}{\partial x_j} d\Omega \right) u_j^l \\
 &= w_i^m \left(\oint_{\Gamma^n} \psi_m \psi_l x_2^{ca} d\Gamma \right) T_i^l + w_i^m \left(\int_{\Omega^n} \psi_m \psi_l x_2^{ca} d\Omega \right) f_i^l.
 \end{aligned}$$

A typical quadratic nonlinear term can be linearized as

$$\begin{aligned} \left[\frac{\partial}{\partial x_j} (u_j u_i) \right]^{n+1/2} &= \frac{1}{2} \left[\frac{\partial}{\partial x_j} (u_j u_i) \right]^n + \frac{1}{2} \left[\frac{\partial}{\partial x_j} (u_j u_i) \right]^{n+1} + \mathcal{O}[(\Delta t)^2] \\ &= \frac{1}{2} \frac{\partial}{\partial x_j} (u_j^n u_i^n) + \frac{1}{2} \frac{\partial}{\partial x_j} (u_j^{n+1} u_i^{n+1}) + \mathcal{O}[(\Delta t)^2] \\ &= \frac{1}{2} \frac{\partial}{\partial x_j} (u_j^n u_i^n) + \frac{1}{2} \frac{\partial}{\partial x_j} \left\{ \left[\left(1 + \frac{\Delta t^n}{\Delta t^{n-1}} \right) u_j^n - \frac{\Delta t^n}{\Delta t^{n-1}} u_j^{n-1} \right] u_i^{n+1} \right\} + \mathcal{O}[(\Delta t)^2], \end{aligned}$$

which is the key feature of *linear implicit* time scheme. Finally, we obtain the fully discrete equation

$$\begin{aligned} & q \left(- \oint_{\Gamma^n} n_j \psi_l x_2^{c_a} d\Gamma \right) u_j^{l,(n+1)} + w_i^m \left(\frac{1}{\Delta t^n} \int_{\Omega^n} \psi_m \psi_l x_2^{c_a} d\Omega \right) u_i^{l,(n+1)} \\ & + \frac{1}{2} w_i^m \left(1 + \frac{\Delta t^n}{\Delta t^{n-1}} \right) \left(\int_{\Omega^n} \psi_m \psi_r \frac{\partial \psi_l}{\partial x_j} x_2^{c_a} d\Omega \right) (u_j^{r,(n)} - v_j^{r,(n)}) u_i^{l,(n+1)} \\ & - \frac{1}{2} w_i^m \frac{\Delta t^n}{\Delta t^{n-1}} \left(\int_{\Omega^n} \psi_m \psi_r \frac{\partial \psi_l}{\partial x_j} x_2^{c_a} d\Omega \right) (u_j^{r,(n-1)} - v_j^{r,(n)}) u_i^{l,(n+1)} \\ & + w_i^m \left(- \oint_{\Gamma^n} n_i \psi_m x_2^{c_a} d\Gamma \right) p^{(n+1/2)} - \frac{1}{2} w_i^m \left(- \frac{1}{Re} \int_{\Omega^n} \frac{\partial \psi_m}{\partial x_j} \frac{\partial \psi_l}{\partial x_j} x_2^{c_a} d\Omega \right) u_i^{l,(n+1)} \\ & - \frac{1}{2} w_i^m \left(- \frac{1}{Re} \int_{\Omega^n} \frac{\partial \psi_m}{\partial x_j} \frac{\partial \psi_l}{\partial x_i} x_2^{c_a} d\Omega \right) u_j^{l,(n+1)} - \frac{1}{2} c_a w_2^m \left(\frac{2}{Re} \int_{\Omega^n} \psi_m \frac{\partial \psi_l}{\partial x_j} d\Omega \right) u_j^{l,(n+1)} \\ & = w_i^m \left(\frac{1}{\Delta t^n} \int_{\Omega^n} \psi_m \psi_l x_2^{c_a} d\Omega \right) u_i^{l,(n)} - \frac{1}{2} w_i^m \left[\left(\int_{\Omega^n} \psi_m \psi_r \frac{\partial \psi_l}{\partial x_j} x_2^{c_a} d\Omega \right) (u_j^{r,(n)} - v_j^{r,(n)}) \right] u_i^{l,(n)} \\ & + \frac{1}{2} w_i^m \left(- \frac{1}{Re} \int_{\Omega^n} \frac{\partial \psi_m}{\partial x_j} \frac{\partial \psi_l}{\partial x_j} x_2^{c_a} d\Omega \right) u_i^{l,(n)} + \frac{1}{2} w_i^m \left(- \frac{1}{Re} \int_{\Omega^n} \frac{\partial \psi_m}{\partial x_j} \frac{\partial \psi_l}{\partial x_i} x_2^{c_a} d\Omega \right) u_j^{l,(n)} \\ & + \frac{1}{2} c_a w_2^m \left(\frac{2}{Re} \int_{\Omega^n} \psi_m \frac{\partial \psi_l}{\partial x_j} d\Omega \right) u_j^{l,(n)} + w_i^m \left(\oint_{\Gamma^n} \psi_m \psi_l x_2^{c_a} d\Gamma \right) T_i^{l,(n+1/2)} \\ & + w_i^m \left(\int_{\Omega^n} \psi_m \psi_l x_2^{c_a} d\Omega \right) f_i^{l,(n+1/2)}. \end{aligned} \quad (7)$$

3. SYSTEM SOLVING

3.1. Discrete operator splitting

Equation (7) eventually produces a discrete system containing the following momentum and continuity equations:

$$Au + Gp = S_u, \quad (8a)$$

$$Du = S_p, \quad (8b)$$

where A is a velocity coefficient matrix, G is the pressure coefficient matrix, D is another velocity coefficient matrix as a consequence of incompressibility, and S_u and S_p are the right-hand-side

known vectors. It must be reminded that all boundary conditions have been incorporated into Equations (8a) and (8b) and that the system is ill-conditioned. In this paper, the original system of Equations (8a) and (8b) is transformed into a new system suited for source-term iteration. Matrix A is split into diagonal and off-diagonal parts:

$$\begin{aligned} (A - A^d + A^d)u + Gp &= S_u \\ \Leftrightarrow A^d u + Gp &= S_u - (A - A^d)u \\ \Leftrightarrow u + A^{-d}Gp &= A^{-d}[S_u - (A - A^d)u] \\ \Rightarrow Du + DA^{-d}Gp &= DA^{-d}[S_u - (A - A^d)u] \end{aligned} \tag{9}$$

$$\begin{aligned} \Leftrightarrow S_p + DA^{-d}Gp &= DA^{-d}[S_u - (A - A^d)u] \\ \Leftrightarrow DA^{-d}Gp &= -S_p + DA^{-d}S_u - DA^{-d}Au + Du, \end{aligned} \tag{10}$$

where A^{-d} stands for the inverse of A^d . Now the original continuity Equation (8b) is replaced by the derived Equation (10), which is a combination of momentum and continuity equations. In other words, the new discrete system is constituted by Equations (10) and (8a). All moves in the above forward derivation actually can be reversed directly, except the move leading to Equation (9). But the backward derivation is rather trivial; seeing that Equation (10) can be rearranged as

$$DA^{-d}(Au + Gp - S_u) = Du - S_p,$$

which immediately leads to continuity equation (8b). Therefore, the derived and the original discrete systems are equivalent to each other. For convenience, we define

$$D^* \equiv DA^{-d}, \quad L \equiv D^*G, \quad D^{**} \equiv D^*A,$$

then the derived discrete system is constituted by a pressure poisson equation and the momentum equation

$$Lp = -(S_p - D^*S_u) - D^{**}u + Du \equiv b_p(u), \tag{11a}$$

$$Au = S_u - Gp \equiv b_u(p). \tag{11b}$$

3.2. Logistic parallelization

The most widely used parallelization method is the juxtaposition-based domain decomposition [14], which is very efficient and in principle very easy. Unfortunately, the actual implementation is quite involved, mainly for inter processor data exchange. In this paper, a superposition-based logistic parallelization is adopted. A serial computation can be easily extended to logistic parallelization. All data in logistic parallelization are stored locally, except that each processor has the same copy of ‘solution vectors’ (which here by definition are excluded from ‘vectors’). Each processor retains the main iterative procedure of the serial computation, and all data immediately involved in the procedure are global. In other words, in term of the main structure, the logistic parallelization is the same as its serial counterpart. Exactly for this reason the method carries the name ‘logistic’, which produces digit-by-digit the same numerical results as its serial counterpart. However, these global data, except solution vectors, are the consequences of superposition of internal and external local data. To illustrate the whole idea we consider, on processor i , the multiplication of an arbitrary global matrix A by an arbitrary global solution vector x

$$Ax = \sum_{j \rightarrow i} (\hat{A}_j x),$$

where \hat{A}_j represents the local matrix on processor j , and $\sum_{j \rightarrow i}$ means: the product of \hat{A}_j and x (computed on processor j) is communicated to processor i and all external local data are superposed onto the internal one on processor i .

For convenience, let us here define three length scales: l^0 is defined as the order of the global number of discrete unknowns (NDU); l^1 is defined as the order of the local (that is, processor-level) NDU; l^2 is defined as the order of the NDU on interfaces shared by processors. As the second example we consider the product of a global matrix A and a global vector b , with two variations

$$Ab = \sum_{j \rightarrow i} \left[\hat{A}_j \left(\sum_{k \rightarrow j} \hat{b}_k \right) \right] = \sum_{j \rightarrow i} \left[\left(\sum_{k \rightarrow j} \hat{A}_k \right) \hat{b}_j \right].$$

In the first approach, the \hat{b} is l^1 -communicated, an l^1 -computation is carried out locally, then the product vector is l^1 -communicated; in the second approach, several l^1 -communications are made for \hat{A} , an l^1 -computation is carried out locally, then the product vector is l^1 -communicated. Obviously, the first approach is more efficient due to less communications.

The efficiency of the first approach can be enhanced further:

$$Ab = \sum_{j \rightarrow i} \left\{ \hat{A}_j \left[\sum_{k \rightarrow j} R^{jk} (\hat{b}_k) \right] \right\},$$

where the operator $R^{jk}(\cdot)$ filters out entries of \hat{b}_k irrelevant to processor j , since these irrelevant data will anyway be ignored when multiplied with matrix \hat{A}_j . This filtered version of matrix-vector multiplication consists of an l^2 -communication, l^1 -computation, and l^1 -communication. To be able to do this, the correlation among processors must be established in advance. The correlation turns out to be extremely simple: each processor should know row numbers involved in any other processor. As a more realistic example, Equation (11a) is parallelized as follows:

$$Lp = - \sum_{j \rightarrow i} \left[\hat{S}_p - \hat{D}^* \sum_{k \rightarrow j} R(\hat{S}_u) \right] - \sum_{j \rightarrow i} (\hat{D}^{**}u) + \sum_{j \rightarrow i} (\hat{D}u),$$

where indexes for local matrices or vectors are omitted for simplicity.

3.3. Fixed-geometry verifications and performance test

To verify the generic formulation, an axisymmetric pipe flow is calculated, with the configuration shown in Figure 4. The numerical results generated by the present quadratic finite element method are compared with those generated by the MAC finite volume method (FVM) code [15], where the MAC staggered grid is employed [16], as well as with commercial FVM-based software Fluent. Figure 5(a) shows verification and mesh resolution study, which is easily resolved for the present method but takes more effort for the FVM on MAC staggered grid to achieve the same convergence. This is because in second-order FVM, all domain and boundary integrals are evaluated at centers, equivalent to one-point quadrature. The two-point or three-point quadrature used in the present method is more efficient for axisymmetric situation where the velocity profile is not linear. Figure 5(b) shows the comparison of several results on a pipe flow and the inaccuracy of the numerical result based on the boundary layer equations [17]. Both the present method and MAC FVM are non-dimensional whereas Fluent is a dimensional code, and a comparison between them is tricky. One might take for granted that as long as the Reynolds number is set the same, a dimensional code should agree with a non-dimensional code. This is not true. The rule of thumb is to further set the density in the dimensional code as unity, in addition to the same numbers for geometry and inlet velocity magnitude. Table II demonstrates a reasonably good scale up of logistic parallelization. Owing to the *logistic* nature of the parallelization, it is sufficient to run only few time steps for scale up study.

A 2D pulsatile channel flow is calculated for error analysis, with the configuration shown in Figure 6. A transient incompressible channel flow under a non-conservative body force is considered

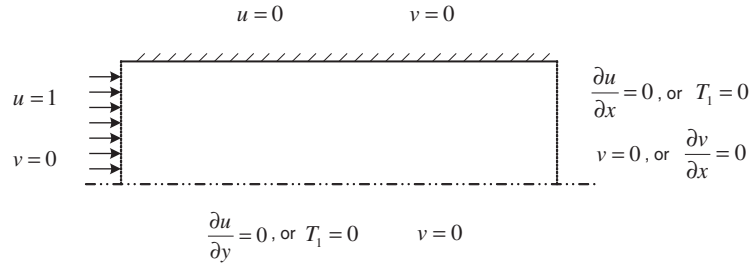


Figure 4. Configuration for the axisymmetric pipe flow. The radius of pipe is set to unity. The pressure is free from boundary conditions, or vanished as implied in free traction boundary condition (if used) at exit.

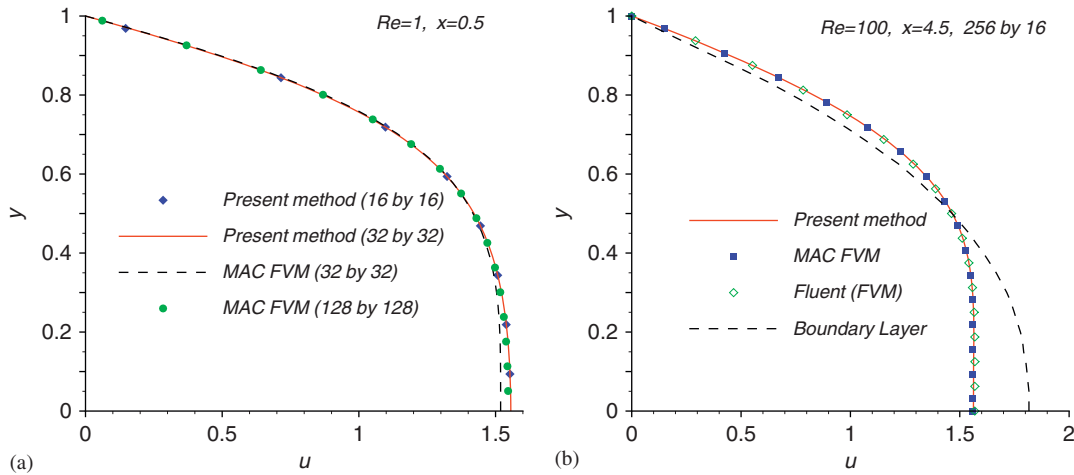


Figure 5. Comparisons of streamwise velocity profiles for the axisymmetric pipe flow. (a) Comparison for $Re=1$ at $x=0.5$ with truncation length of $x=2.0$ and (b) comparison for $Re=100$ at $x=4.5$ with truncation length of $x=32.0$.

Table I. Time scheme study for pulsatile flow at $Re=1000$.

Δt	AB1 ($\times 10^{-4}$)	AB2 ($\times 10^{-4}$)	Si2 ($\times 10^{-4}$)	Li2 ($\times 10^{-4}$)
0.001	4.496	4.405	4.405	4.406
0.002	6.063	5.712	5.712	5.715
0.004	10.46	9.533	9.532	9.540
0.008	20.33	18.29	18.29	18.31
0.016	Diverge	Diverge	Diverge	36.51
0.032				73.28
0.064				147.2
0.128				Diverge

The table shows L_2 norms (root mean square) of numerical results against the exact solution. All numerical results are based on a 16×16 mesh with quadratic elements and at time point 10.24, which requires sufficient number of time steps.

to test spatial implementation, body force effect, time schemes, and local mass conservation. The geometry and boundary conditions are illustrated in Figure 6. The exact solution is prescribed as $u = \sin t \cos \pi x \sin 2\pi y$, $v = \sin t \sin \pi x \sin^2 \pi y$, and $p = 0$. The non-conservative transient body force can be derived as

$$f_x = \cos t \cos \pi x \sin 2\pi y - \pi \sin^2 t \sin 2\pi x \sin^2 \pi y + \frac{5\pi^2}{Re} \sin t \cos \pi x \sin 2\pi y,$$

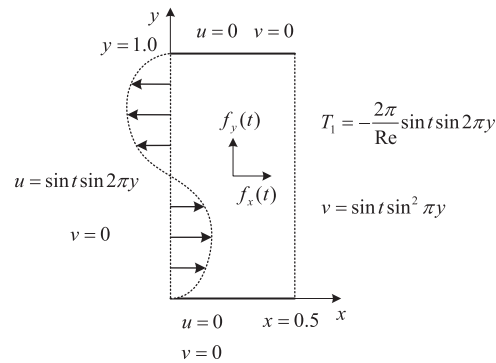


Figure 6. Configuration for 2D pulsatile channel flow. Expressions for body forces can be found in [18]. Except the traction boundary condition where the pressure plays a role, no boundary conditions are imposed on the pressure.

Table II. Scale-up study of the logistic parallelization for pipe (with axial length 4.0) flow at $Re=10$. The experiments are performed on: a 256×64 quadratic mesh, two time steps and with step size 0.001, two source-term iterations, 32 linear iterations for the pressure and 16 for the velocity.

Number of processors	1	2	4
cpu time (s)	66	26	13
Relative speed (versus serial)	1	2.54	5.08

$$f_y = \cos t \sin \pi x \sin^2 \pi y + \pi \sin^2 t \sin^2 \pi y \sin 2\pi y + \frac{\pi^2}{Re} \sin t \sin \pi x (\sin^2 \pi y - 2 \cos 2\pi y).$$

Excellent agreement of numerical results with the exact solutions is observed. Table I shows the stability and time convergence rate of four different time schemes, forward Euler (AB1), second-order Adams–Bashforth (AB2), second-order semi-implicit (Si2), and second-order linear implicit (Li2). The table demonstrates a perceptible advantage of the linear implicit time scheme introduced in this paper.

4. NUMERICAL METHOD FOR CAPILLARY JET

4.1. Configuration

Figure 7 shows the configuration of the problem and the schematic of the mesh. The flow domain represents a spatially periodic part of a jet, after deduction of constant mainstream velocity. Driven by sinusoidal disturbance and with symmetric boundary conditions imposed on two lateral ends, the initially quiescent flow remains contained inside the domain. While in a 2D case the capillary disturbance causes an oscillation, in an axisymmetric situation it typically results in breakup. Laboratory experiments, such as [19], can be simplified to this flow model. More details regarding flow physics can be found in [10, 20]. For several reasons, this benchmark problem is selected for demonstration of improvements in numerical techniques. In contrast to the elaborated benchmark results on gravitational waves, such as in [8], the capillary case is less simulated. The flow geometry is simple, yet it possesses all basic features of free-surface flows so that numerical techniques can be well elucidated, and yet it is complex enough to manifest advantages of one numerical method over another. Owing to large curvature and swift physical process right before the breakup, this numerical case is very sensitive. Hence, it demands accurate and robust numerical methods. Techniques developed to cope with the sensitivity of capillary breakup may be useful for other moving boundary problems.

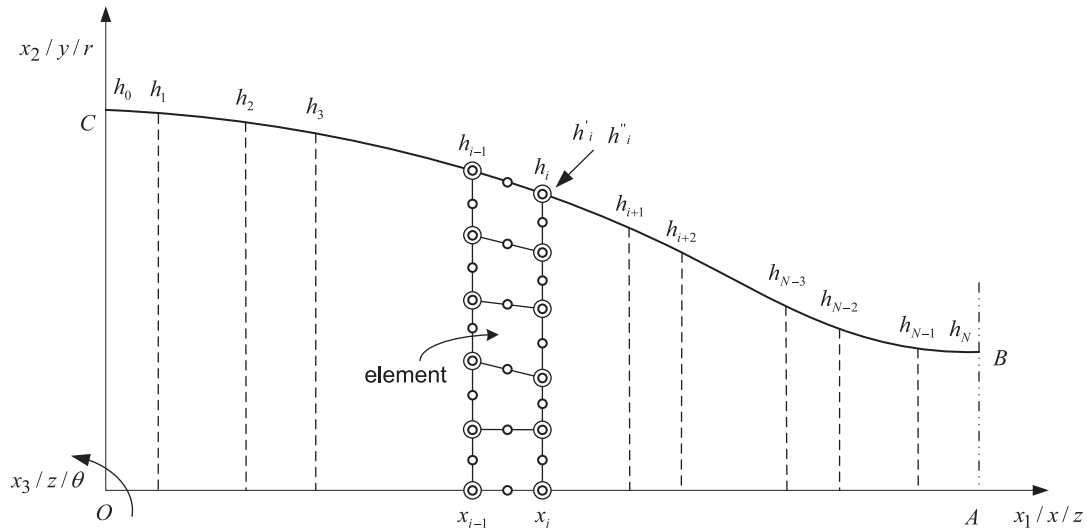


Figure 7. Configuration for capillary free-surface flows with symmetric (along line O-C and line A-B) boundary conditions, and the schematic of the mesh and height functions. $\Delta x_i \equiv x_{i+1} - x_i$, and \forall_i is used to denote the subvolume between h_i and h_{i+1} .

Figure 7 can be regarded as a snapshot of the free-surface flow on time level n characterized by h^n (and more generally by \vec{x}^n) which describes the geometry and node locations, by \vec{v}^n which describes node velocity, and by \vec{u}^n which describes field velocity. Three symmetric boundary conditions are imposed on the bottom and two lateral ends, and the kinematic and dynamic boundary conditions are imposed on the top free-surface. The objective here is to evolve the flow to a snapshot on time level $n+1$. As mentioned before, this evolution can be divided into two stages, a mesh-frozen stage and a field-frozen stage, as shown in Figure 3. During the mesh-frozen stage from time level n to time level $n+1$, h^n (and more generally \vec{x}^n) and \vec{v}^n remain the same but \vec{u}^n evolves to \vec{u}^{n+1} , which requires calculation of traction T^n . During the field frozen stage, \vec{u}^n and \vec{u}^{n+1} remain the same whereas h^n (and possibly \vec{x}^n) and \vec{v}^n evolve.

4.2. Dynamic boundary condition

The solution field is updated during the mesh-frozen stage. However, in the free-surface case the traction in Equation (7) must be substituted in, that is, dynamic boundary condition must be imposed. With the Taylor expansion of h_{i-1} and h_{i+1} at x_i , we obtain second-order explicit scheme

$$h'_i = \frac{(x_i - x_{i-1})^2(h_{i+1} - h_i) + (x_{i+1} - x_i)^2(h_i - h_{i-1})}{(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} + \mathcal{O}[(\Delta x)^2], \tag{12a}$$

$$h''_i = 2 \frac{(x_i - x_{i-1})(h_{i+1} - h_i) - (x_{i+1} - x_i)(h_i - h_{i-1})}{(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} + \mathcal{O}[(\Delta x)^2]. \tag{12b}$$

First and second derivatives at the two ends can be extrapolated with second-order accuracy. Normal vectors at mid nodes (between two geometric nodes) on the free-surface are taken as the averages of the normal vectors at neighboring geometric nodes. Curvatures at mid nodes can be also taken as the averages of neighboring values. For problems with symmetric boundary conditions, first and second derivatives at the two ends can be calculated more accurately. As a matter of fact, symmetric lateral boundary conditions have entered the whole equation system through imposition on velocities. However, a further and direct imposition of symmetry on derivatives related to geometry is consistent with the existing system and improves the accuracy. For this purpose, a ghost mirror segment is introduced right outside the left end, and it is set at $x_{-1} = 2x_0 - x_1$ and

$h_{-1} = h_1$. Similarly, at the right end $x_{N+1} = 2x_N - x_{N-1}$ and $h_{N+1} = h_{N-1}$. Then, Equations (12a) and (12b) can be applied at the two ends.

Once the first and the second derivatives of the height function at discrete points are known, the normal vector, curvature, and traction can be calculated. The normal vector on the free-surface is

$$\hat{n} = \left(-\frac{h'}{(1+h'^2)^{1/2}}, \frac{1}{(1+h'^2)^{1/2}} \right).$$

The curvature differs in 2D and axisymmetric cases

$$\kappa = \nabla \cdot \hat{n} = -\frac{h''}{(1+h'^2)^{3/2}} + \frac{c_a}{h(1+h'^2)^{1/2}}.$$

For both 2D and axisymmetric flows, surface traction on the free surface is

$$\begin{aligned} \sigma_{ij}n_j &= T_i = -\frac{\kappa}{We}n_i, \\ We &\equiv \frac{\rho U'^2 H'}{\gamma}, \end{aligned} \quad (13)$$

where γ is the surface tension coefficient and U' and H' represent characteristic velocity and length. For numerical examples of this paper $H' = 1.0$, the undisturbed height of the free-surface. Lack of the characteristic velocity in the capillary flow renders a comparison between numerical results and experimental data very inconvenient, but this does not pose any problem for a comparison between two computational results.

For problems with symmetric boundary conditions, the first and second derivatives on the free-surface can be calculated with a fourth-order compact scheme. For simplicity, only the evenly spaced situation is considered here. Using a five-point difference scheme, all discrete second-order derivatives can be expressed in terms of discrete heights

$$h_i'' = \frac{-h_{i-2} + 16h_{i-1} - 30h_i + 16h_{i+1} - h_{i+2}}{12(\Delta x)^2} + \mathcal{O}[(\Delta x)^4].$$

The above explicit interpolation of the second derivative extends $4\Delta x$ in space. Excessive span deteriorates accuracy and undermines simplicity and flexibility. Thus, geometric manipulations are better restricted to the host element or to its immediate neighbors. Following the idea of implicit interpolation used in compact finite difference [21], for evenly spaced heights we can derive the relation

$$\frac{1}{10}h_{i-1}'' + h_i'' + \frac{1}{10}h_{i+1}'' = \frac{6}{5} \frac{h_{i-1} - 2h_i + h_{i+1}}{\Delta x^2} + \mathcal{O}[(\Delta x)^4].$$

Then, a tridiagonal system of linear algebraic equations is formed after imposition of symmetric boundary conditions. This approach seem to feature a smaller stencil. As can be shown, first and second derivatives at mid points can be accurately calculated based on the values at geometric nodes on the same segment. Unfortunately, the zero derivative at mid points is difficult to obtain on a single stencil. Therefore, the following three steps are implemented instead:

- First, we may work out the heights at mid points. By expanding $h_{i-1/2}$, h_i , h_{i+1} , and $h_{i+3/2}$ at $h_{i+1/2}$, we deduce implicit interpolation

$$h_{i-1/2} + 6h_{i+1/2} + h_{i+3/2} = 4h_i + 4h_{i+1} + \mathcal{O}[(\Delta x)^4],$$

which only spans $2\Delta x$ (though it involves immediate neighboring segments). With symmetric boundary condition imposed, at two ends

$$7h_{1/2} + h_{3/2} = 4h_0 + 4h_1 + \mathcal{O}[(\Delta x)^4],$$

$$h_{N-3/2} + 7h_{N-1/2} = 4h_{N-1} + 4h_N + \mathcal{O}[(\Delta x)^4]$$

- Next, using five-point explicit difference, we obtain

$$h_i'' = \frac{-h_{i-1} + 16h_{i-1/2} - 30h_i + 16h_{i+1/2} - h_{i+1}}{3(\Delta x)^2} + \mathcal{O}[(\Delta x)^4],$$

$$h_i' = \frac{h_{i-1} - 8h_{i-1/2} + 8h_{i+1/2} - h_{i+1}}{6\Delta x} + \mathcal{O}[(\Delta x)^4],$$

which only span $2\Delta x$ and involve host segments only.

- Finally, we may expand h_i and h_{i+1} at $h_{i+1/2}$, h_i' and h_{i+1}' at $h_{i+1/2}'$, and h_i'' and h_{i+1}'' at $h_{i+1/2}''$. We then acquire

$$h_{i+1/2}'' = -\frac{1}{4}h_i'' + \frac{3}{2}\frac{h_{i+1}' - h_i'}{\Delta x} - \frac{1}{4}h_{i+1}'' + \mathcal{O}[(\Delta x)^4],$$

$$h_{i+1/2}' = -\frac{1}{4}h_i' + \frac{3}{2}\frac{h_{i+1} - h_i}{\Delta x} - \frac{1}{4}h_{i+1}' + \mathcal{O}[(\Delta x)^4],$$

which only span Δx and involve one segment.

4.3. IBT method for kinematic boundary condition

During the field-frozen stage there are two tasks, determination of new positions and new velocities of massless nodes. The new position of nodes on free-surface is essentially about the new geometry of the free-surface. Here we elaborate the first approach, IBT of flux method. The new geometry of the free-surface relates to the new subvolumes underneath (\forall_i in Figure 7), which relates to velocities of massless nodes and field velocities. The overall procedure includes three stages. Subvolumes have to be updated first. Subvolumes relate to motions of both fluid particles and nodes

$$\frac{d\forall_i}{dt} = - \sum_{j=0}^{N_y-1} \oint_{\Gamma_{ij}^e} (\vec{u} - \vec{v}) \cdot \hat{n} d\Gamma, \tag{15}$$

for $i=0, 1, 2, \dots$. In Equation (15), Γ_{ij}^e stands for the boundary of an element and N_y is the number of elements in y direction. With forward Euler

$$\forall_i^{n+1} = \forall_i^n - \Delta t \left[\sum_{j=0}^{N_y-1} \oint_{\Gamma_{ij}^e} (\vec{u} - \vec{v}) \cdot \hat{n} d\Gamma \right]^n.$$

To achieve higher-order accuracy, it is necessary to know both \vec{v}^{n+1} and the new positions of nodes in Equation (15). Hence, this necessitates an iterative procedure.

Surface reconstruction is conducted in the second stage. The case for 2D is rather straightforward, but a squeeze technique is introduced in this paper for the axisymmetric situation with the following nonlinear relation:

$$\forall_i = \frac{\pi}{3}(x_{i+1} - x_i)(h_i^2 + h_i h_{i+1} + h_{i+1}^2).$$

This relation involves N subvolumes and $N + 1$ discrete heights, so that while heights can uniquely determine subvolumes the reverse is not true. The supplemental equation can be obtained by the Taylor expansion of h_1 and h_2 at x_0 , then imposition of $h_0' = 0$. Hence, a self-contained small nonlinear system of equations can be obtained as

$$3h_0 - 4h_1 + h_2 = 0,$$

$$h_0^2 + h_0 h_1 + h_1^2 = \frac{3\forall_0}{\pi(x_1 - x_0)} \equiv c_0,$$

$$h_1^2 + h_1 h_2 + h_2^2 = \frac{3V_1}{\pi(x_2 - x_1)} \equiv c_1,$$

which contains two quadratic algebraic equations. After reduction of variable, the above equation system produces a complicated single nonlinear equation, which anyway has to resort to an iterative procedure to solve. Instead, we may directly iterate the original equation system with a squeeze technique, whose steps are presented in the following:

- *Step 1:* Calculate h_0 and h_2 based on predicted h_1^* :

$$h_0 = \frac{-h_1^* + \sqrt{4c_0 - 3h_1^{*2}}}{2},$$

$$h_2 = \frac{-h_1^* + \sqrt{4c_1 - 3h_1^{*2}}}{2}.$$

- *Step 2:* Calculate modified h_1^{**} :

$$h_1^{**} = \frac{3h_0 + h_2}{4}.$$

- *Step 3:* If the change of h_1 is within prescribed error bound err ,

$$\frac{|h_1^{**} - h_1^*|}{1.0 + |h_1^*|} < \text{err},$$

then the iteration ends.

- *Step 4:* Otherwise, let the weighted average be the new predicted value

$$h_1^* \leftarrow (1 - \lambda)h_1^* + \lambda h_1^{**}$$

and go back to step 1. The parameter λ is typically taken as 0.5.

After the iteration the remaining heights can be progressively calculated through

$$h_i = \frac{-h_{i-1} + \sqrt{4c_{i-1} - 3h_{i-1}^2}}{2},$$

for $i=3, 4, \dots$. The accuracy of the iteration and the overall simulation can be monitored by checking the symmetry property on the right end

$$h_{N-2} - 4h_{N-1} + 3h_N = 0.$$

One may take a left-to-right sweep, in which the zero tangent on the left end is used, and take a right-to-left sweep then take an average to obtain the final discrete heights.

Third and finally, the mesh evolution stage includes update of new node positions and new node velocities. Once new discrete heights are calculated, the new positions of nodes can be determined. In this paper, nodes are evenly distributed in both horizontal and vertical directions, and concomitant with the new free-surface. Finally, the motions of nodes must be updated. Since nodes move in the y -direction only

$$v_{ij} = \frac{dy_{ij}}{dt}, \quad i=0, 1, \dots, N, \quad j=0, 1, \dots, N_y.$$

With second-order approximation,

$$v_{ij}^{n+1} = -v_{ij}^n + 2 \frac{y_{ij}^{n+1} - y_{ij}^n}{\Delta t}. \quad (16)$$

Table III. Convergence test of squeeze technique where N is the number of segments.

Error tolerance	$N=32, \lambda=0.5$	$N=128, \lambda=0.5$	$N=32, \lambda=0.45$
10^{-2}	2	2	3
10^{-4}	3	3	5
10^{-6}	3	3	7
10^{-8}	3	3	9
10^{-10}	4	4	11
10^{-12}	4	4	13
10^{-14}	4	4	15

The first column shows error tolerance and the remaining three columns show number of iterations.

Note that in this indirect approach, new positions of nodes are determined first and new node velocities are determined afterwards, through differentiation in Equation (16). Differentiation demands accurate calculation of node positions.

The curvature calculation and the surface reconstruction are tested for the axisymmetric case. The function under test is $1.0+0.9\cos((\pi/2)z)$. First, a sampling of the height function is made at a set of points and V_i are calculated based on the trapezoidal rule. Then, the squeeze technique is used to recover h_i , based on which the numerical curvatures at sampling points are calculated. Numerical curvatures at mid points of segments are taken as averages of neighbors, and alternatively one may first obtain average derivatives then calculate curvatures. Good agreement between numerical results and exact profile is observed. It is also tested that when the mesh size reduces to half the curvature error drops to one fourth, which indicates second-order convergence rate. Table III shows the rapid convergence of squeeze technique. The table demonstrates that the selection of λ matters, nevertheless even for $\lambda=0.45$ the convergence rate is rather high. The table also shows that mesh size has little influence on convergence.

4.4. Direct boundary tracking method for kinematic boundary condition

The centerpiece of the field-frozen stage is to find new positions and velocities of nodes. Here we elaborate the second approach, direct boundary tracking. The new positions of nodes are arbitrary except that *on the free surface* the kinematic boundary condition must be satisfied:

$$(\vec{u}^{n+1} - \vec{v}^{n+1}) \cdot \hat{n}^{(n+1)} = 0, \tag{17}$$

where \vec{u}^{n+1} is known after the mesh-frozen stage. Throughout the whole computational domain, node velocities and node positions are related via

$$\vec{v} = \frac{d\vec{x}}{dt}. \tag{18}$$

If nodes are restricted to move vertically only, Equation (17) reduces to

$$v_2^{n+1} = \frac{\vec{u}^{n+1} \cdot \hat{n}^{(n+1)}}{n_2^{(n+1)}}, \tag{19}$$

and *on the free-surface* Equation (18) reduces to

$$v_2 = \frac{dh}{dt}. \tag{20}$$

Because $\hat{n}^{(n+1)}$ is a function of height h^{n+1} , Equations (19) and (20) form a coupled system, which cannot be decoupled exactly but can be decoupled approximately, or handled by iteration

as follows:

- Step 1, calculate node velocities based on the best known h

$$v_2 = \frac{\vec{u}^{n+1} \cdot \hat{n}}{n_2}.$$

- Step 2, bookkeep $h^* = h$.
- Step 3, calculate height function via

$$h = h^n + \frac{v_2^n + v_2}{2} \Delta t.$$

- Step 4, check convergence

$$\max \left(\frac{|h - h^*|}{1.0 + |h|} \right) < \text{err},$$

where err is an error tolerance typically taken as 10^{-10} . If the convergence is not met, go back to step 1.

5. NUMERICAL RESULTS FOR CAPILLARY JET

5.1. Capillary wave and capillary jet breakup

With a variety of parameters and the symmetric boundary conditions on the two lateral ends and on the bottom, the 2D capillary jet oscillates and diminishes gradually. Figure 8 shows a parameter study for 2D capillary wave, initiated by a disturbance in the form of $-A \cos((\pi/L)x)$. Figure 8(a) shows the effect of Weber number on the frequency of decaying water wave. It indicates that a smaller Weber number results in a faster oscillation. Figure 8(b) shows that the viscosity not only damps but also slows down oscillations, consistent with a simple oscillator in vibration theory. Figure 8(c) shows that larger wave number speeds up the oscillation and Figure 8(d) shows that amplitude of disturbance slows down frequency slightly. This qualitative study provides some guidance for selection of parameters and serves as a good free-surface benchmark problem. Table IV shows the reasonable efficiency of the logistic parallelization for the capillary wave.

With the same boundary conditions and with typical parameters as 2D capillary wave, the axisymmetric capillary jet usually breaks up. This type of breakup has been extensively studied in [10], where the flow physics was the focus. Here the focus is shifted to numerical aspects. Figure 9 shows the initial disturbed surface and the profile close to breakup. The flow field is initially at rest and triggered by the cosine disturbance, then the flow develops slowly. As the time goes on, the flow field changes faster and faster. Close to breakup, the flow field and the free-surface display an accelerated process, with a time scale several orders of magnitude smaller than earlier stages. What is shown in Figure 9 is treated as the standard breakup case, and flow parameters and numerical techniques used are specified in the caption of the figure. The criteria for variable time steps include: (a) when the solution is changing faster, a smaller time step should be used; (b) the new time step should be on the same order of magnitude as the old one; (c) the new time step should be within some range. For simplicity, the variable time step is prescribed in the computer code as: compared with initial time step, the time step size is cut down to $\frac{1}{2}$ at $t = 10.50$, to $\frac{1}{4}$ at $t = 11.00$, to $\frac{1}{8}$ at $t = 11.25$, to $\frac{1}{16}$ at $t = 11.50$, and to $\frac{1}{32}$ at $t = 11.75$. Table V shows the breakup study on the standard capillary jet breakup. To avoid excessive points and computation, with the present method the breakup is defined at the time point that the radius of the jet neck (that is, at the breakup location) is 5% of the undisturbed height. Table V shows good agreement of the current results with those from [10].

5.2. Numerical accuracy and sensitivity

Capillary jet is rather a sensitive physical process that requires robust and accurate numerical treatments. For example, the symmetric lateral boundary conditions have been imposed onto flow

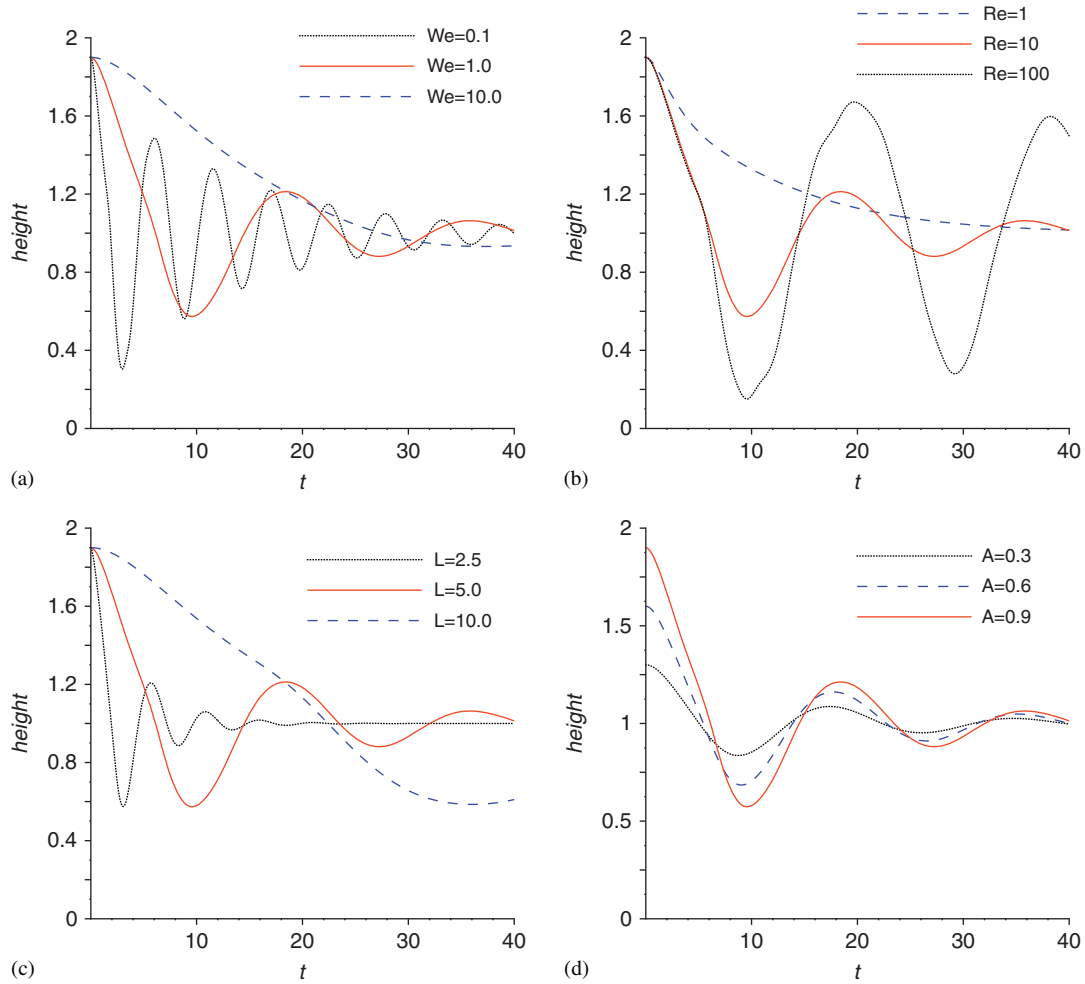


Figure 8. Parameter study for 2D capillary wave: time history of the height at the right-most point (the peak point at $t=0$). The solid lines on all four figures are for the default case: $We=1$, $Re=10$, half wavelength $L=5.0$, and initial perturbation amplitude $A=0.9$. (a) Effect of Weber number on the frequency of decaying capillary wave; (b) effect of viscosity on the frequency; (c) effect of wave number on the frequency; and (d) effect of amplitude of disturbance on frequency. All cases are carried out on a 80×8 mesh with 0.001 time step size except that the case with twice wave length is carried out on a 160×8 mesh.

Table IV. Scale-up study of the logistic parallelization for the 2D capillary wave (with the same parameters for the standard case of capillary breakup). The experiment is based on a 1024×64 mesh.

Number of processors	1	2	4
cpu time (s)	2142	930	597
Relative speed (versus serial)	1	2.30	3.59

equations during the mesh-frozen stage. However, a further incorporation of the symmetry into derivative calculations makes the field-solving and free-surface tracking much more accurate and robust. Note that such a repeat application of symmetric boundary conditions does not impair the solvability, because no excessive equations are created. The main reason behind the numerical sensitivity is fast time scale right before the breakup. In the following, a numerical analysis is made from several different perspectives.

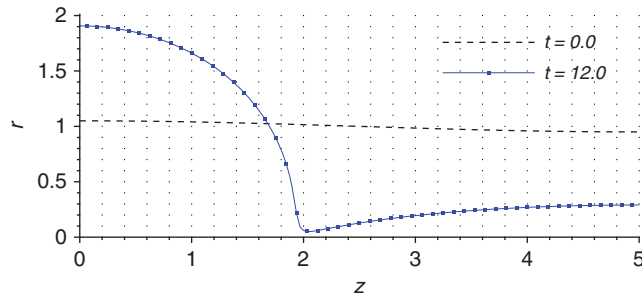


Figure 9. The initial surface shape of the capillary jet and the profile close to the breakup. The flow parameters for this *standard case* of jet breakup are $Re=10.0$, $We=1.0$, half wavelength $L=5.0$, and initial disturbance amplitude $A=0.05$. The calculations are based on: 160×16 mesh with 8-1 elements (quadratic for velocity and constant discontinuous for pressure), linear implicit time scheme with variable step size initially equal to 0.001, fourth-order compact scheme for the dynamic boundary condition, and iterative Direct Boundary Tracking for the kinematic boundary condition.

Table V. Breakup study of the base jet breakup case on various meshes and time step sizes.

	Time for breakup	Neck location	Satellite radius	Swell radius
Reference [10] (20×4)	11.85	2.12	0.28	1.91
80×8	11.95	2.13	0.29	1.89
160×16	12.00	2.03	0.29	1.91

The parameters of the flow are $Re=10$, $We=1$, $L=5.0$, and $A=0.05$. Present calculations are based on variable time step initially 0.001. The reference results were based on a 40×4 mesh for a full wavelength.

While the quadratic 8-1 element demonstrates superb robustness over linear 4-1 element in numerous situations, Figure 10 shows the advantage of the 4-1 element in the jet breakup. Under the same selected parameters, the simulation on 4-1 element can proceed up to breakup, whereas the 8-1 element fails. Note that the number of elements in two cases are the same, so that virtually there are twice numbers of free-surface points in the 8-1 case. This phenomena is the consequence of the averaging of curvature (or derivatives) at mid points along the free-surface, in the case of 8-1 element. In 4-1 element, the curved free-surface segment is approximated by a straight line segment. Then the two end nodes, which are both the velocity nodes and geometric nodes, explicitly follow some rules and evolve to next locations. In the 8-1 element case, the midpoints are slightly displaced from the free-surface and lack explicit rules to follow. To improve accuracy, the quadrilateral elements should be modified, that is, the straight line segment on the free-surface should be replaced by second-order polynomial to match the interpolation polynomial (and to match the number of nodes on element edge). Since Figure 10 is carried out on a rather coarse mesh, it is reasonable to question the continued advantage of the 4-1 element on refined mesh. Numerical experiments show that the more expensive 8-1 element never shows better performance, with various time step sizes, mesh sizes, and tracking techniques. Nevertheless, due to robustness of domain calculations, throughout this paper the 8-1 element is used by default. This numerical case somewhat casts cloud on higher-order numerical methods, such as spectral methods, in solving problems where the more stringent precision is required for boundaries. Since under such a circumstance, the much improved accuracy in domain calculation is not helpful, unless high accuracy in boundary treatment is achieved.

Figure 11 shows an evaluation of time schemes, and it is clear that the linear implicit scheme performs much better. This is a slightly anti-intuitive phenomenon. The field-frozen stage is the same for both cases, and the time scheme matters for the mesh-frozen stage only. During the mesh-frozen stage, the field velocity and pressure are calculated, based on the old mesh (on the time level n). But the actual scenario is rather real-time: the field is being updated on a mesh that is also under updating. The approximation of the real-time dynamic mesh by a static frozen mesh

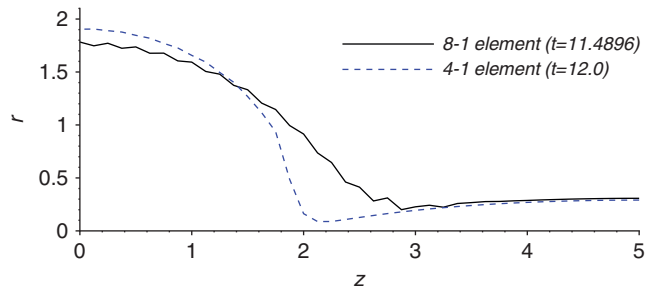


Figure 10. Element evaluation for the standard case of jet breakup. The calculation is based on the indirect boundary tracking of flux method with forward Euler, on 40×4 mesh and with time step size 0.0008.

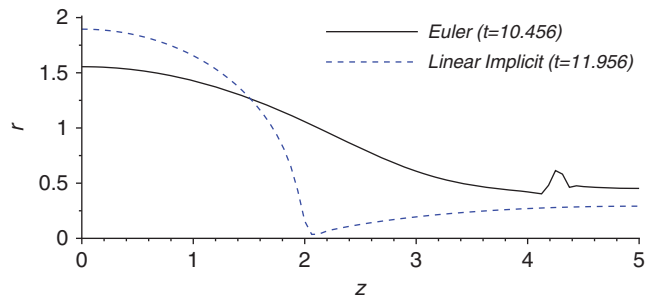


Figure 11. Time scheme evaluation for the standard case of jet breakup. The calculations are carried out with iterative Direct Boundary Tracking, on an 80×8 mesh with variable time step size initially equal to 0.0005. The symmetric fourth-order compact scheme is used for dynamic boundary conditions.

on time level n reminds us of the first-order forward or backward Euler. Hence, intuitively both cases in Figure 11 have been ruined by some low-order numerical errors in geometric treatment, so that a second-order time scheme is no longer expected to display the advantage. However, the numerical evidence does not uphold this expectation.

The compact scheme for calculations of first and second derivatives features high-order accuracy and smaller stencil. Usually, an implicit scheme may cost more arithmetic operations. However, here the implicitness is for operations on surfaces only; thus the overall speed is not hampered. A closer look at the shape of the free-surface is shown in Figure 12. The accuracy of the compact scheme in the vicinity of breakup allows the further growth of jet necking. In contrast, the less accurate treatment causes the simulation to break down prematurely.

Figure 13 shows the advantage of direct boundary tracking over IBT. The surface tracking of iterative DBT at one spatial point is not directly dependent on its neighbor (though ultimately they are all related to each other). In contrast, the trapezoidal rule (or some more accurate rules) during the progressive procedure of IBT relates one height directly with another, so that numerical error passes from one height to another without damping. To be more specific, the under estimate (as an example) of height immediately causes an over estimate of the next height, and subsequently causes an under estimate of the following height. Thus, the numerical error propagates globally.

Further, after surface reconstruction in the IBT of flux method, the free-surface node velocities are calculated through a differentiation equation (16). The numerical error during height calculation is first amplified by the time step through the node position and node velocity relation Equation (16). Then the node velocity \vec{v} enters the governing equation and the flux calculation, and its value is abated by the multiplication with time step. At the end, the numerical error originated from height remains in the same order of magnitude, and a smaller time step offers little help on this issue. In contrast, for DBT the node position and node velocity relation on the free-surface is used in the opposite direction. That is, the free-surface is identified through node velocities on free-surface,

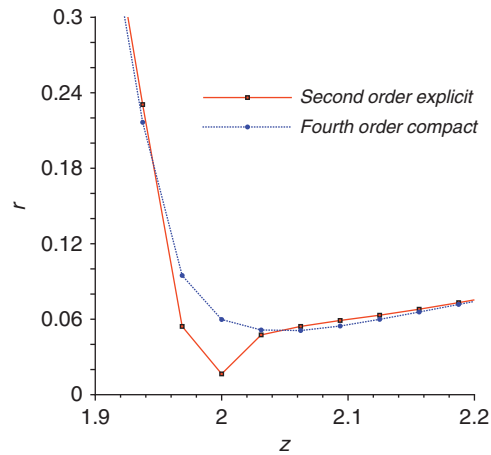


Figure 12. Evaluation of derivative calculations for dynamic boundary conditions (at $t=12.0$ for the standard case: 160×16 mesh, iterative Direct Boundary Tracking, and linear implicit with variable time step size initially equal to 0.001).

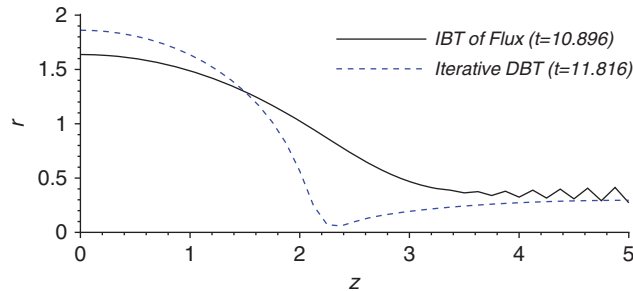


Figure 13. Evaluation of boundary tracking techniques for the standard breakup benchmark problem, on a 40×4 mesh with variable time step size initially equal to 0.0005.

which is a time integration. The numerical error during node velocity calculation is difficult to pass on, because of the multiplication by time step.

6. CONCLUSIONS

In this paper, the capillary jet flow is calculated with a special ALE formulation and some boundary tracking techniques. The node velocity, field velocity, and fluid velocity are clarified and the special ALE is rigorously derived with the notion of mesh-frozen and field-frozen and with the notion of tentatively inertial coordinates. The continuous equation system is discretized in space with finite element method and in time with linear implicit scheme. The resulting discrete system is solved with a discrete operator splitting technique and superposition-based logistic parallelization. The correctness of the formulation and the implementation is confirmed by all numerical results. Fixed-geometry flows are calculated to demonstrate stability of linear implicit scheme, which is fully linear and implicit, and the reasonable efficiency of logistic parallelization. For capillary jet flow, both an indirect and an iterative direct boundary tracking techniques are implemented. A parameter study of 2D capillary wave is demonstrated as well as parallel performance test, then it is shown that under the same parameters the axisymmetric jet breaks up. Good numerical results of breakup parameters are obtained and a match with reference data is observed. A compact scheme is introduced to improve the accuracy of derivative calculations for the dynamic boundary condition, which features high accuracy and less involvement of segments/elements. The squeeze

technique used in surface reconstruction demonstrates rapid convergence. Numerical analysis shows the sensitivity of the capillary breakup, in that the imposition of boundary conditions, calculations of derivatives and curvature, selection of spatial element, and selection of time scheme must be judiciously made. Numerical results also demonstrate the importance of avoiding global error propagation and the preference of integration over differentiation. Relevant future work could include an extension of the current formulation and computer code to the breakup of a dielectric coaxial jet in an electric field [13], which is driven by both capillary (surface) and electric (body) forces.

REFERENCES

1. Chan RKC, Street RL. A computer study of finite-amplitude water waves. *Journal of Computational Physics* 1970; **6**:68–94.
2. Peskin CS. Numerical analysis of blood flow in the heart. *Journal of Computational Physics* 1977; **25**:220.
3. Hirt CW, Nichols BD. Volume of fluids (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics* 1981; **39**:201–225.
4. Osher S, Sethian JA. Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics* 1988; **79**:12–49.
5. Hirt CW, Amsden AA, Cook JL. An arbitrary Lagrangian–Eulerian computing method for all flow speeds. *Journal of Computational Physics* 1974; **14**:227–253.
6. Hughes TJR, Liu WK, Zimmermann TK. Lagrangian–Eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering* 1981; **29**:329–349.
7. Huerta A, Liu WK. Viscous flow with large free surface motion. *Computer Methods in Applied Mechanics and Engineering* 1988; **69**:277–324.
8. Ramaswamy B. Numerical simulation of unsteady viscous free surface flow. *Journal of Computational Physics* 1990; **90**:396–430.
9. Behr M. Stabilized finite element methods for incompressible flows with emphasis on moving boundaries and interfaces. *Ph.D. Thesis*, University of Minnesota, 1992.
10. Mashayek F, Ashgriz N. A height-flux method for simulating free surface flows and interfaces. *International Journal for Numerical Methods in Fluids* 1993; **17**:1035–1054.
11. Chippada S, Ramaswamy B, Wheeler MF. Numerical simulation of hydraulic jump. *International Journal for Numerical Methods in Engineering* 1994; **37**:1381–1397.
12. Donea J, Huerta A, Ponthot J-Ph, Rodriguez-Ferran A. *Arbitrary Lagrangian–Eulerian Methods*. Encyclopedia of Computational Mechanics. Wiley: New York, 2004.
13. Loscertales IG, Barrero A, Guerrero I, Cortijo R, Marquez M, Ganan-Calvo AM. Micro/nano encapsulation via electrified coaxial liquid jets. *Science* 2002; **295**:1695–1698.
14. Fischer PF. Spectral element solution of Navier–Stokes equations on high performance distributed-memory parallel processors. *Ph.D. Thesis*, Massachusetts Institute of Technology, 1989.
15. Zhang KKQ, Minkowycz WJ, Mashayek F. Exact factorization technique for numerical simulations of incompressible Navier–Stokes flows. *International Journal of Heat and Mass Transfer* 2006; **49**:535–545.
16. Harlow FH, Welch JF. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids* 1965; **8**:2182–2189.
17. Kays WM, Crawford ME, Weigand B. *Convective Heat and Mass Transfer* (4th edn), Chapter 7. McGraw-Hill Professional: New York, 2004.
18. Zhang KKQ, Rovagnati B, Gao Z, Minkowycz WJ, Mashayek F. An introduction to lattice grid. *Numerical Heat Transfer Part B—Fundamentals* 2007; **51**:415–431.
19. Tjahjadi M, Stone HA, Ottino JM. Satellite and subsatellite formation in capillary breakup. *Journal of Fluid Mechanics* 1992; **243**:297–317.
20. Ashgriz N, Mashayek F. Temporal analysis of capillary jet breakup. *Journal of Fluid Mechanics* 1995; **291**:163–190.
21. Zhang KKQ, Shotorban B, Minkowycz WJ, Mashayek F. A compact finite difference method on staggered grid for Navier–Stokes flows. *International Journal for Numerical Methods in Fluids* 2006; **52**:867–881.