Journal of Computational Physics 273 (2014) 416–434

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A fast pressure-correction method for incompressible two-fluid flows

Michael S. Dodd [1], Antonino Ferrante [*,2]

William E. Boeing Department of Aeronautics & Astronautics, University of Washington, Seattle, WA 98195, USA

ARTICLE INFO

Article history:
Received 17 October 2013
Received in revised form 7 March 2014
Accepted 19 May 2014
Available online 28 May 2014

Keywords:
Variable density Navier–Stokes equations
Projection method
Volume-of-fluid method
Multiphase flow
Interfacial flow
Turbulent flow
Direct numerical simulation

ABSTRACT

We have developed a new pressure-correction method for simulating incompressible two-fluid flows with large density and viscosity ratios. The method's main advantage is that the variable coefficient Poisson equation that arises in solving the incompressible Navier–Stokes equations for two-fluid flows is reduced to a constant coefficient equation, which can be solved with an FFT-based, fast Poisson solver. This reduction is achieved by splitting the variable density pressure gradient term in the governing equations. The validity of this splitting is demonstrated from our numerical tests, and it is explained from a physical viewpoint.

In this paper, the new pressure-correction method is coupled with a mass-conserving volume-of-fluid method to capture the motion of the interface between the two fluids but, in general, it could be coupled with other interface advection methods such as level-set, phase-field, or front-tracking. First, we verified the new pressure-correction method using the capillary wave test-case up to density and viscosity ratios of 10,000. Then, we validated the method by simulating the motion of a falling water droplet in air and comparing the droplet terminal velocity with an experimental value. Next, the method is shown to be second-order accurate in space and time independent of the VoF method, and it conserves mass, momentum, and kinetic energy in the inviscid limit. Also, we show that for solving the two-fluid Navier–Stokes equations, the method is 10–40 times faster than the standard pressure-correction method, which uses multigrid to solve the variable coefficient Poisson equation. Finally, we show that the method is capable of performing fully-resolved direct numerical simulation (DNS) of droplet-laden isotropic turbulence with thousands of droplets using a computational mesh of 1024^3 points.

1. Introduction

In the incompressible flow of two immiscible fluids, the density is discontinuous at the interface, and the density ratio can be of order one thousand (e.g., air–water). Thus, simulating two-fluid flows using a constant density approximation, e.g., the Boussinesq approximation, is not justified. Therefore, the density change between the two fluids must be accounted for directly when solving the governing equations of fluid motion. From a numerical point of view, this introduces additional

* Corresponding author.
E-mail addresses: doddms@uw.edu (M.S. Dodd), ferrante@aa.washington.edu (A. Ferrante).
[1] Ph.D. candidate, William E. Boeing Department of Aeronautics & Astronautics, University of Washington, Seattle.
[2] Assistant Professor, William E. Boeing Department of Aeronautics & Astronautics, University of Washington, Seattle.

http://dx.doi.org/10.1016/j.jcp.2014.05.024
0021-9991/

challenges which are not present when solving incompressible single-fluid flows. These challenges are to obtain a numerical method that is computationally efficient, second-order accurate, and stable for large density ratios.

A common solution technique for the incompressible Navier–Stokes equations is the projection method [1]. In the projection method, a Poisson equation for pressure must be solved numerically at each time step. This operation takes most of the computational time in the projection method. Consequently, much work has gone into developing so-called "fast Poisson solvers", which use a combination of fast Fourier transforms (FFT) and Gauss elimination to solve the Poisson equation directly in Fourier space (e.g., [2–5]). However, fast Poisson solvers cannot solve the Poisson equation for pressure that arises in two-fluid flows when advancing the solution from time t^n to t^{n+1} ($\Delta t = t^{n+1} - t^n$):

$$\nabla \cdot \left(\frac{1}{\rho^{n+1}} \nabla p^{n+1} \right) = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^*, \; Al \tag{1}$$

where ρ is the density, p is the pressure, and $\mathbf{u}^*$ is the approximate velocity field at t^{n+1}. Fast Poisson solvers require a constant coefficient on the left-hand side of the Poisson equation, whereas the coefficient $(1/\rho^{n+1})$ on the left-hand side of Eq. (1) varies in space and time. To solve the variable coefficient equation (1), a standard practice has been to use iterative methods, e.g., multigrid methods [6–8], Krylov methods [9,10] or Krylov methods preconditioned with multigrid [11]. The downside is that iterative methods are often slower than fast Poisson solvers. In fact, they can be up to ten times slower [2]. Additionally, an iterative method's operation count depends on problem parameters (e.g., density ratio) and convergence tolerance, whereas fast Poisson solvers have the advantage of fixed operation counts.

To reduce the computational time in solving variable density incompressible flows, Guermond and Salgado [12] adopted a penalty formulation, whereby only a constant coefficient Poisson equation must be solved at each time step. More recently, for solving the two-fluid coupled Navier–Stokes Cahn–Hilliard (phase-field) equations, Dong and Shen [13] developed a velocity-correction method that transforms the Poisson equation (1) from a variable to a constant coefficient equation. The underlying idea is to split the variable-coefficient pressure-gradient term into a constant term and a variable term, and then treat the constant term implicitly and the variable term explicitly as

$$\frac{1}{\rho^{n+1}} \nabla p^{n+1} \rightarrow \frac{1}{\rho_0} \nabla p^{n+1} + \left(\frac{1}{\rho^{n+1}} - \frac{1}{\rho_0} \right) \nabla \hat{p}, \tag{2}$$

where ρ_0 is a constant to be defined in Section 2 and $\hat{p}$ is an explicit approximation to the pressure at time level $n+1$, specifically,

$$\hat{p} = \begin{cases} p^n & \text{if } J = 1, \\ p^* = 2p^n - p^{n-1} & \text{if } J = 2, \end{cases} \tag{3}$$

where the choice of $J = 1$ or 2 indicates, respectively, constant or linear extrapolation of p^{n+1}.[3] Then, applying the substitution (Eq. (2)) to the left-hand side of Eq. (1), leads to a constant coefficient Poisson equation for pressure,

$$\nabla^2 p^{n+1} = \nabla \cdot \left[\left(1 - \frac{\rho_0}{\rho^{n+1}} \right) \nabla \hat{p} \right] + \frac{\rho_0}{\Delta t} \nabla \cdot \mathbf{u}^* \tag{4}$$

that can be solved directly using a fast Poisson solver. Also, in contrast to solving Eq. (1), there is no need to assemble variable coefficient matrices at each time step when solving Eq. (4). Therefore, by using the splitting technique in Eq. (2), there is potential for significant speedup in solving the Navier–Stokes equations for incompressible two-fluid flows.

In this paper, we aim to answer the following questions on applying the splitting technique (Eq. (2)) to the two-fluid Navier–Stokes equations:

- Is the split method (Eq. (4)) physically less accurate than the unsplit method (Eq. (1)) and does the accuracy of the split method depend on the type of extrapolation chosen in Eq. (3)?
- What are the computational savings of using the split method? Specifically, how much faster is it to solve Eq. (4) directly versus Eq. (1) iteratively?

To answer these questions, we first develop a two-fluid pressure-correction method with a constant coefficient Poisson equation, which can be solved directly with a fast Poisson solver. Next, we apply the method to several two-fluid flows to verify and validate the new method, quantify its performance, and evaluate its spatial and temporal accuracy.

The paper is organized as follows: in Section 2 we describe the new pressure-correction method for uniform Cartesian grids and periodic boundary conditions and its coupling to the volume-of-fluid (VoF) method. Section 3 compares the numerical solutions obtained when using the standard, unsplit method (i.e., solving Eq. (1), called Unsplit), the split method, Eq. (2), using $J = 1$ (called FastP^n), and the split method using $J = 2$ (called FastP*). Next, we assess the performance of the unsplit and split formulations by comparing the CPU time when solving Eq. (1) using a multigrid solver to the CPU time

[3] A similar strategy for handling variable coefficients in the advection–diffusion equation was proposed by Gottlieb and Orszag [14, Section 9, p. 114].

when solving Eq. (4) using a fast Poisson solver. Section 4 verifies and validates the new pressure-correction/volume-of-fluid flow solver. This includes two canonical flows and DNS of droplet-laden decaying isotropic turbulence.

The potential computational savings suggest that our method is well-suited for fully-resolved DNS of bubble- and droplet-laden turbulent flows in simple geometries. In addition, because Eq. (1) is present regardless of the method used to advect the fluid interface, the pressure-correction method presented is useful in other interface-capturing (e.g., level-set or phase-field) and front-tracking methods. Furthermore, the splitting technique of the new pressure-correction method could be extended to simulate low-Mach number flows that use the pressure-correction method, e.g., [15].

2. Mathematical description

2.1. Governing equations

The governing equations for the incompressible flow of two immiscible fluids are

$$\nabla \cdot \mathbf{u} = 0, \tag{5a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = \frac{1}{\rho}\left[-\nabla p + \frac{1}{\mathrm{Re}}\nabla \cdot \left(\mu\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right) + \frac{1}{\mathrm{We}}\mathbf{f}_\sigma\right] + \frac{1}{\mathrm{Fr}}\mathbf{g} \tag{5b}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is the fluid velocity, $p = p(\mathbf{x}, t)$ is the pressure, $\rho = \rho(\mathbf{x}, t)$ is the density, $\mu = \mu(\mathbf{x}, t)$ is the dynamic viscosity, $\mathbf{g}$ is the gravitational acceleration, and $\mathbf{f}_\sigma = \mathbf{f}_\sigma(\mathbf{x}, t)$ is the force per unit volume due to surface tension,

$$\mathbf{f}_\sigma = \sigma\kappa\delta(\mathbf{x} - \mathbf{x}_s)\mathbf{n}, \tag{6}$$

where σ is the surface tension coefficient, κ is the curvature of the interface between the two fluids (denoted fluid 1 and fluid 2), $\mathbf{n}$ is the unit normal pointing into fluid 2 at the interface, and δ is the Dirac δ-function that is needed to impose the force only at the interface position $\mathbf{x}_s$. In this paper, the surface tension coefficient is constant in space and time, and the symbol σ is replaced by its non-dimensional value 1/We. In Eq. (5b), Re, We, and Fr are, respectively, the Reynolds, Weber, and Froude numbers defined as:

$$\mathrm{Re} = \frac{\tilde{U}\tilde{L}\tilde{\rho}_1}{\tilde{\mu}_1}, \qquad \mathrm{We} = \frac{\tilde{\rho}_1\tilde{U}^2\tilde{L}}{\tilde{\sigma}}, \qquad \mathrm{Fr} = \frac{\tilde{U}^2}{\tilde{g}\tilde{L}}, \tag{7}$$

where $\tilde{U}$, $\tilde{L}$, $\tilde{\rho}_1$, $\tilde{\mu}_1$, $\tilde{\sigma}$, and $\tilde{g}$ denote, in order, the reference dimensional velocity, length, density, dynamic viscosity, surface tension coefficient, and gravitational acceleration used to non-dimensionalize the governing equations (5a) and (5b). The subscripts 1 and 2 indicate fluid 1 and fluid 2, respectively. We have chosen to non-dimensionalize the density and dynamic viscosity in Eq. (5b) by choosing fluid 1 as the reference phase, which makes $\rho_1 = 1$ and $\mu_1 = 1$. Throughout the paper, all variables are dimensionless unless they are accented with $\sim$. Also, we impose periodic boundary conditions in all spatial directions unless noted otherwise.

2.2. Pressure-correction method

Our two-fluid pressure-correction method is developed starting from that of Chorin [1] and Temam [16] for single-phase flows. The solution algorithm proceeds by advecting the volume fraction of fluid 2, $C(\mathbf{x}, t)$, based on the known velocity field $\mathbf{u}^n$. The VoF advection algorithm [17] that computes $C^{n+1}(\mathbf{x})$ is briefly described in the next section. The volume fraction has value $C = 0$ in fluid 1, $C = 1$ in fluid 2, and $0 < C < 1$ in cells containing the interface separating the two fluids. After computing C^{n+1}, the density and viscosity are computed at time level $n+1$ as

$$\begin{aligned}\rho^{n+1}(\mathbf{x}) &= \rho_2 C^{n+1}(\mathbf{x}) + \rho_1\left[1 - C^{n+1}(\mathbf{x})\right]\\ \mu^{n+1}(\mathbf{x}) &= \mu_2 C^{n+1}(\mathbf{x}) + \mu_1\left[1 - C^{n+1}(\mathbf{x})\right].\end{aligned} \tag{8}$$

Next, an approximate velocity is computed by first defining $\mathbf{RU}^n$ as

$$\mathbf{RU}^n = -\nabla \cdot \left(\mathbf{u}^n\mathbf{u}^n\right) + \frac{1}{\mathrm{Re}}\left[\frac{1}{\rho^{n+1}}\nabla \cdot \left(\mu^{n+1}\left(\nabla \mathbf{u}^n + \left(\nabla \mathbf{u}^n\right)^T\right)\right)\right] + \frac{1}{\mathrm{We}}\left[\frac{1}{\rho^{n+1}}\kappa^{n+1}\nabla C^{n+1}\right] + \frac{1}{\mathrm{Fr}}\mathbf{g}, \tag{9}$$

which is the right-hand side of the momentum equation (5b) without the pressure gradient term. The surface tension force, $\mathbf{f}_\sigma$, is computed as $\mathbf{f}_\sigma = \sigma\kappa\nabla C$ by adopting Brackbill's continuum surface force approach [18]. Next, as recommended in [18], we multiply $\mathbf{f}_\sigma$ in Eq. (5b) by $\rho(\mathbf{x}, t)/\langle\rho\rangle$, where $\langle\rho\rangle \equiv \frac{1}{2}(\rho_1 + \rho_2)$, such that $\mathbf{f}_\sigma$ is constant across the interface, which reduces the magnitude of the spurious currents at high density ratios (not shown). Thus, Eq. (9) becomes

$$\mathbf{RU}^n = -\nabla \cdot \left(\mathbf{u}^n\mathbf{u}^n\right) + \frac{1}{\mathrm{Re}}\left[\frac{1}{\rho^{n+1}}\nabla \cdot \left(\mu^{n+1}\left(\nabla \mathbf{u}^n + \left(\nabla \mathbf{u}^n\right)^T\right)\right)\right] + \frac{1}{\mathrm{We}}\left[\frac{\kappa^{n+1}\nabla C^{n+1}}{\langle\rho\rangle}\right] + \frac{1}{\mathrm{Fr}}\mathbf{g}. \tag{10}$$

The interface curvature κ^{n+1} is computed using the height-function method [19] with improvements by López et al. [20]. The equations are integrated in time using the Adams–Bashforth scheme:

Table 1
Summary of two-fluid pressure-correction methods.

Method	Poisson equation	$\hat{p}$	Poisson solver
FastP*	(16)	$2p^n - p^{n-1}$	Fast Poisson
FastP^n	(16)	p^n	Fast Poisson
Unsplit	(13)	–	Multigrid

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = \frac{3}{2}\mathbf{RU}^n - \frac{1}{2}\mathbf{RU}^{n-1}. \tag{11}$$

In the pressure-correction method, to satisfy the divergence-free condition (5a), the velocity field is advanced in time by applying the pressure-correction to the approximate velocity, $\mathbf{u}^*$, as

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\rho^{n+1}}\nabla p^{n+1}. \tag{12}$$

Then, by taking the divergence of Eq. (12) and imposing the divergence-free condition on $\mathbf{u}^{n+1}$, we get a Poisson equation for pressure with variable coefficient $(1/\rho^{n+1})$:

$$\nabla \cdot \left(\frac{1}{\rho^{n+1}}\nabla p^{n+1}\right) = \frac{1}{\Delta t}\nabla \cdot \mathbf{u}^*. \tag{13}$$

Eqs. (11), (12), and (13) are solved in a standard, two-fluid pressure-correction method (e.g., [21,22]), which we call the Unsplit method. We solve Eq. (13) iteratively using the *hypre* library [23,24]. Specifically, we use the semicoarsening multigrid (SMG) method for structured grids, which was first introduced in [25].

In our new pressure-correction method, the goal is to obtain a constant coefficient Poisson equation in place of the variable coefficient Poisson equation (13). To this end, we split the pressure gradient term into a constant part $(1/\rho_0)$ and variable part $(1/\rho^{n+1})$, and then treat the constant part implicitly and the variable part explicitly as

$$\frac{1}{\rho^{n+1}}\nabla p^{n+1} \rightarrow \frac{1}{\rho_0}\nabla p^{n+1} + \left(\frac{1}{\rho^{n+1}} - \frac{1}{\rho_0}\right)\nabla \hat{p}, \tag{14}$$

where $\rho_0 = \min(\rho_1, \rho_2)$ for numerical stability [13] and $\hat{p}$ is given by Eq. (3). We show in Section 3 that the choice $J = 1$ (called FastP^n because $\hat{p} = p^n$) reduces the physical accuracy of the solution for flows with finite surface tension. We will show in Section 3 that, for our pressure-correction method, the choice $J = 2$ (called FastP* because $\hat{p} = p^* = 2p^n - p^{n-1}$), is required to accurately simulate flows with finite surface tension. Substituting (14) in Eq. (12) gives:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\left[\frac{1}{\rho_0}\nabla p^{n+1} + \left(\frac{1}{\rho^{n+1}} - \frac{1}{\rho_0}\right)\nabla \hat{p}\right]. \tag{15}$$

Then, by taking the divergence of (15) while imposing $\nabla \cdot \mathbf{u}^{n+1} = 0$ yields a constant coefficient Poisson equation,

$$\nabla^2 p^{n+1} = \nabla \cdot \left[\left(1 - \frac{\rho_0}{\rho^{n+1}}\right)\nabla \hat{p}\right] + \frac{\rho_0}{\Delta t}\nabla \cdot \mathbf{u}^*, \tag{16}$$

which can be solved directly using a fast Poisson solver (e.g., [4,5]). Finally, using (15), we update the velocity field by applying the pressure-correction to $\mathbf{u}^*$ as

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t\left[\frac{1}{\rho_0}\nabla p^{n+1} + \left(\frac{1}{\rho^{n+1}} - \frac{1}{\rho_0}\right)\nabla \hat{p}\right]. \tag{17}$$

In summary, our new pressure-correction algorithm advances the numerical solution in time using the following four steps:

1. Compute C^{n+1} by advecting the C field using the VoF advection algorithm described in Section 2.3 and [17], or compute C^{n+1} using other methods, e.g., level-set or front-tracking. Then, compute cell-centered values for density and viscosity at time level $n+1$ using Eq. (8).
2. Compute $\mathbf{RU}^n$ using Eq. (10), and then compute $\mathbf{u}^*$ using Eq. (11).
3. Compute p^{n+1} by solving the constant-coefficient Poisson equation (16) using a fast Poisson solver, e.g., [5].
4. Compute $\mathbf{u}^{n+1}$ using Eq. (17) in which the pressure correction p^{n+1} is applied to the approximate velocity $\mathbf{u}^*$.

In step 3, we solve the Poisson equation using a combination of one-dimensional FFTs in the x- and y-directions, and Gauss elimination in the z-direction [5]. The three methods are summarized in Table 1.

The time step (Δt) must be restricted to ensure numerical stability due to our choice of explicit treatment of the convective, viscous, and surface tension terms. Δt is calculated as

$$\Delta t \leq \frac{1}{2}\min(\Delta t_c, \Delta t_\nu, \Delta t_\sigma), \tag{18}$$

where

$$\Delta t_c = \frac{\Delta x}{|U|_{\max}} \tag{19}$$

$$\Delta t_\nu = \frac{\mathrm{Re}\,\Delta x^2}{6} \tag{20}$$

$$\Delta t_\sigma = \sqrt{\frac{\mathrm{We}(\rho_1+\rho_2)\Delta x^3}{4\pi}}. \tag{21}$$

For cases of low Re, the viscous stability restriction (20) can be eliminated by treating the viscous terms implicitly (see Appendix A), however, in all the simulations presented in this paper, the viscous terms were treated explicitly as in Eq. (10) either because: (i) $\Delta t_\nu > \Delta t_\sigma$, (ii) the stability restriction imposed by large density and viscosity gradients was greater than the restriction imposed by Δt_ν, or (iii) switching to implicit time integration would not reduce the number of time steps enough to offset the additional computational cost of solving three Helmholtz equations at each time step. For cases of low We, the capillary time step restriction (21) can be lessened by a method outlined in [26] or eliminated by treating the surface tension implicitly [27], however these options were not pursued for this paper.

Finally, the numerical methods described in this section have been implemented using Fortran 90, the message passing interface (MPI), OpenMP, and FFTW [28]. To perform parallel multi-core simulations, the domain is decomposed such that the data is distributed in memory in the y-direction and contiguous in memory in the x- and z-directions. The following algorithm is used to solve the Poisson equation for p^{n+1} in parallel.

Algorithm of parallel fast Poisson solver (PFPS).

1. Perform real-to-complex FFT in the x-direction to the right-hand side of Eq. (16), which is a three-dimensional (3-D) scalar field;
2. Transpose the 3-D complex data among the computing cores such that the resulting data is distributed in the x-direction and contiguous in the y-direction;
3. Perform complex-to-complex FFT in the y-direction of the 3-D data;
4. Perform Gauss elimination in the z-direction to obtain p^{n+1} in Fourier space [5];
5. Perform complex-to-complex, inverse FFT in the y-direction of the 3-D data;
6. Transpose the 3-D complex data among the computing cores such that it is distributed in the y-direction and contiguous in the x-direction;
7. Perform complex-to-real, inverse FFT in the x-direction of the resulting 3-D data to obtain p^{n+1}.

2.2.1. Spatial discretization

We discretize Eqs. (10), (16), and (17) in 3-D on a uniform staggered Cartesian mesh using central differences. For brevity, we report here the details of the spatial discretization in 2-D. The extension to 3-D follows analogously from the 2-D discretization. The u-component of velocity is located at $x_{i+1/2,j}$, the v-component at $x_{i,j+1/2}$, and all other variables are centered at $x_{i,j}$. Our discretization has the advantage over colocated schemes of not requiring an auxiliary cell-centered velocity field.

We begin discretizing Eq. (10) by categorizing the terms as convective fluxes **RCU**, diffusive fluxes **RDU**, and body forces **RBU** (surface tension and gravity):

$$\mathbf{RU} = \mathbf{RCU} + \mathbf{RDU} + \mathbf{RBU} \tag{22}$$

with

$$\begin{aligned}
\mathbf{RCU} &= -\nabla\cdot\left(\mathbf{u}^n\mathbf{u}^n\right)\\
\mathbf{RDU} &= \frac{1}{\mathrm{Re}}\left[\frac{1}{\rho^{n+1}}\nabla\cdot\left(\mu^{n+1}\left(\nabla\mathbf{u}^n+\left(\nabla\mathbf{u}^n\right)^T\right)\right)\right]\\
\mathbf{RBU} &= \frac{1}{\mathrm{We}}\left[\frac{\kappa^{n+1}\nabla C^{n+1}}{\langle\rho\rangle}\right]+\frac{1}{\mathrm{Fr}}\mathbf{g}.
\end{aligned} \tag{23}$$

The convective fluxes **RCU** are equivalent to those found in one-fluid flow so we do not include their discretization here (see, e.g., [29]). We include the discretization of **RDU** and **RBU** for completeness and because there is no well-established second-order discretization of these terms. The components of the viscous flux vector **RDU** are discretized and computed as follows:

$$RDU_{i+1/2,j} = \frac{1}{\mathrm{Re}}\frac{1}{(\rho_{i+1/2,j})}\left(\frac{DUX_{i+1,j}-DUX_{i,j}}{\Delta x}+\frac{DUY_{i+1/2,j+1/2}-DUY_{i+1/2,j-1/2}}{\Delta y}\right) \tag{24}$$

and

$$RDV_{i,j+1/2} = \frac{1}{\mathrm{Re}}\frac{1}{(\rho_{i,j+1/2})}\left(\frac{DUY_{i+1/2,j+1/2} - DUY_{i+1/2,j-1/2}}{\Delta x} + \frac{DVY_{i,j+1} - DVY_{i,j}}{\Delta y}\right), \tag{25}$$

where Δx and Δy are the grid sizes in the x- and y-directions, respectively. In Eqs. (24) and (25), the density at the staggered locations is computed using the arithmetic mean as

$$\rho_{i+1/2,j} = \frac{1}{2}(\rho_{i+1,j} + \rho_{i,j}) \quad \text{and} \quad \rho_{i,j+1/2} = \frac{1}{2}(\rho_{i,j+1} + \rho_{i,j}). \tag{26}$$

The strain rates in Eqs. (24) and (25) are calculated as

$$DUX_{i,j} = 2\mu_{i,j}\left(\frac{u_{i+1/2,j} - u_{i-1/2,j}}{\Delta x}\right) \tag{27}$$

$$DUY_{i+1/2,j+1/2} = \mu_{i+1/2,j+1/2}\left(\frac{u_{i+1/2,j+1} - u_{i+1/2,j}}{\Delta y} + \frac{v_{i+1,j+1/2} - v_{i,j+1/2}}{\Delta x}\right) \tag{28}$$

$$DVY_{i,j} = 2\mu_{i,j}\left(\frac{v_{i,j+1/2} - v_{i,j-1/2}}{\Delta y}\right), \tag{29}$$

where the staggered viscosity in Eq. (28) is computed using the arithmetic mean

$$\mu_{i+1/2,j+1/2} = \frac{1}{4}(\mu_{i+1,j+1} + \mu_{i+1,j} + \mu_{i,j+1} + \mu_{i,j}). \tag{30}$$

Note that in Eq. (24), we substituted *DUY* for *DVX* because of the symmetry of the rate-of-strain tensor.

Next, we compute the components of the body force vector **RBU** where the non-dimensional gravitational acceleration in Eq. (5b) is $g = 1$ and in the negative y-direction ($\mathbf{g} = -\hat{\mathbf{j}}$). Special treatment is required when gravity is oriented in the direction of periodic boundaries to prevent uniform acceleration of both fluids [7,30]. We explicitly account for the hydrostatic pressure term in the momentum equation (5b) by adding $\mathbf{f}_\mathrm{h}/\rho$ to the right-hand side of Eq. (5b), where

$$\mathbf{f}_\mathrm{h} = -\frac{\rho_1 V_1 + \rho_2 V_2}{V_1 + V_2}\mathbf{g}, \tag{31}$$

and V_1 and V_2 are the total volumes of fluid 1 and 2. The discretized components of **RBU** are

$$RBU_{i+1/2,j} = \frac{1}{\mathrm{We}}\frac{\kappa_{i+1/2,j}}{\langle\rho\rangle}\left(\frac{C_{i+1,j} - C_{i,j}}{\Delta x}\right) \tag{32}$$

$$RBV_{i,j+1/2} = \frac{1}{\mathrm{We}}\frac{\kappa_{i,j+1/2}}{\langle\rho\rangle}\left(\frac{C_{i,j+1} - C_{i,j}}{\Delta y}\right) + \frac{1}{\mathrm{Fr}}\left(\frac{f_\mathrm{h}}{\rho_{i,j+1/2}} - 1\right), \tag{33}$$

where, as a reminder, $\langle\rho\rangle \equiv \frac{1}{2}(\rho_1 + \rho_2)$.

The surface tension force $\mathbf{f}_\sigma$ must be discretized at the staggered grid locations to be consistent with the location of the pressure gradient term [9]. This discretization produces an exact balance of the pressure gradient and the surface tension if the curvature is exact [17]. The curvature is computed at the staggered grid point $x_{i+1/2,j}$ as

$$\kappa_{i+1/2,j} = \begin{cases} \kappa_{i+1,j} & \text{if } \kappa_{i,j} = 0 \\ \kappa_{i,j} & \text{if } \kappa_{i+1,j} = 0 \\ \frac{1}{2}(\kappa_{i+1,j} + \kappa_{i,j}) & \text{otherwise.} \end{cases} \tag{34}$$

Next, we discretize the right-hand side of Eq. (16), which serves as input to the fast Poisson solver:

$$\nabla^2 p^{n+1}_{i,j} = \frac{DPX_{i+1/2,j} - DPX_{i-1/2,j}}{\Delta x} + \frac{DPY_{i,j+1/2} - DPY_{i,j-1/2}}{\Delta y} + DIVU_{i,j} \tag{35}$$

where

$$DPX_{i+1/2,j} = \left(1 - \frac{\rho_0}{\rho_{i+1/2,j}}\right)\left(\frac{\hat{p}_{i+1,j} - \hat{p}_{i,j}}{\Delta x}\right), \tag{36}$$

$$DPY_{i,j+1/2} = \left(1 - \frac{\rho_0}{\rho_{i,j+1/2}}\right)\left(\frac{\hat{p}_{i,j+1} - \hat{p}_{i,j}}{\Delta y}\right), \tag{37}$$

and

$$DIVU_{i,j} = \frac{\rho_0}{\Delta t}\left(\frac{u^*_{i+1/2,j} - u^*_{i-1/2,j}}{\Delta x} + \frac{v^*_{i,j+1/2} - v^*_{i,j-1/2}}{\Delta y}\right). \tag{38}$$

Finally, the components of Eq. (17) are discretized as

$$u_{i+1/2,j}^{n+1} = u_{i+1/2,j}^{*} - \Delta t\left[\frac{1}{\rho_0}\left(\frac{p_{i+1,j}^{n+1} - p_{i,j}^{n+1}}{\Delta x}\right) + \left(\frac{1}{\rho_{i+1/2,j}} - \frac{1}{\rho_0}\right)\left(\frac{\hat{p}_{i+1,j} - \hat{p}_{i,j}}{\Delta x}\right)\right] \tag{39}$$

and

$$v_{i,j+1/2}^{n+1} = v_{i+1/2,j}^{*} - \Delta t\left[\frac{1}{\rho_0}\left(\frac{p_{i,j+1}^{n+1} - p_{i,j}^{n+1}}{\Delta y}\right) + \left(\frac{1}{\rho_{i,j+1/2}} - \frac{1}{\rho_0}\right)\left(\frac{\hat{p}_{i,j+1} - \hat{p}_{i,j}}{\Delta y}\right)\right]. \tag{40}$$

2.3. Volume-of-fluid method

We are using a recently developed VoF method that is mass-conserving, consistent ($0 \leq C \leq 1$), and wisps-free [17]. A brief description is reported here.

In the VoF method, the sharp interface between the two immiscible fluids is determined using the VoF color function, C, which represents the volume fraction of fluid 2 in each computational cell. In our VoF method, the interface between the two phases is reconstructed using a piecewise linear interface calculation (PLIC) [31]. The interface reconstruction in each computational cell consists of two steps: the computation of the interface normal $\mathbf{n} = (n_x, n_y, n_z)$ and the computation of the interface location. The algorithm that we use to evaluate the interface normal is a combination of the centered-columns method [32] and Youngs' method [31] known as the mixed-Youngs-centered (MYC) method [33].

We define a characteristic function χ that has value $\chi = 1$ in fluid 2 and $\chi = 0$ in fluid 1. χ is governed by the following advection equation:

$$\frac{\partial \chi}{\partial t} + \mathbf{u} \cdot \nabla \chi = 0. \tag{41}$$

The volume fraction $C_{i,j,k}$ of grid cell i, j, k is related to the characteristic function χ by the integral relation

$$C_{i,j,k}(t) = \frac{1}{V_0}\int_{\Omega} \chi(\mathbf{x}, t)\mathrm{d}\mathbf{x}, \tag{42}$$

where V_0 is the volume of the i, j, k cell.

The volume fraction C is advanced in time using the Eulerian implicit – Eulerian algebraic – Lagrangian explicit (EI-EA-LE) algorithm originally proposed by Scardovelli et al. [34]. The original EI-EA-LE algorithm is globally mass-conserving but generates wisps and does not conserve the mass of the individual volumes tracked in the flow. We have improved this method by adding redistribution and wisp-suppression algorithms [17]. The redistribution algorithm corrects small inconsistencies in the volume fraction values (i.e., $C < 0$ and $C > 1$) that arise in the Eulerian algebraic (EA) step. The wisp suppression algorithm removes small errors arising from the finite machine precision. Because our method is consistent and wisps-free, it conserves mass both globally and locally within each fluid. The cell-centered volume fraction C is advected using the face-centered velocity field. The method displays second-order spatial accuracy for values of CFL number $\Delta t/\Delta x \leq 0.1$. Furthermore, the average geometrical error ($E_g = |C(\mathbf{x}) - C_{\text{exact}}(\mathbf{x})|$) is less than 1% for a moving droplet with 30 grid cells or more across the diameter. The complete description of the method and the results are reported by Baraldi et al. [17].

3. Comparison of FastP*, FastP^n, and Unsplit methods

To determine if the substitution (14) is a valid approximation in two-fluid flows for the choices $J = 1$ and $J = 2$, we investigated the agreement of numerical solutions obtained using the FastP^n and FastP* methods with the solution obtained using the Unsplit method, which serves as reference (Table 1).

3.1. Falling droplet

We simulate a water droplet falling in quiescent air. The dimensional constants for this case are:

$$\begin{aligned}
&\tilde{\rho}_{\mathrm{a}} = 1.204\ \mathrm{kg/m^3}, \qquad \tilde{\rho}_{\mathrm{w}} = 998.2\ \mathrm{kg/m^3},\\
&\tilde{\mu}_{\mathrm{a}} = 1.837 \times 10^{-5}\ \mathrm{kg/(m\,s)}, \qquad \tilde{\mu}_{\mathrm{w}} = 1.002 \times 10^{-3}\ \mathrm{kg/(m\,s)},\\
&\tilde{\sigma} = 7.28 \times 10^{-2}\ \mathrm{kg/s^2},\\
&\tilde{g} = 9.81\ \mathrm{m/s^2},
\end{aligned} \tag{43}$$

where the subscripts 'a' and 'w' denote air (fluid 1) and water (fluid 2), respectively. The initial dimensional droplet diameter is $\tilde{D}_0 = 1.4$ mm and the domain is a rectangular box with dimensions $(L_x \times L_y \times L_z) = (2D_0 \times 2D_0 \times 4D_0)$ where $D_0 = 0.5$. The domain is discretized with $64 \times 64 \times 128$ grid points, and the CFL number is $\Delta t/\Delta x = 0.025$ to satisfy Eq. (21). The box

Table 2
Non-dimensional parameters used for the rising bubble and falling droplet tests.

Case	Re	We	Fr	D_0	ρ_2/ρ_1	μ_2/μ_1
Falling droplet	304	0.127	100	0.5	829	54.5
Rising bubble	217	121	1.0	0.5	1.21e−3	2.45e−4

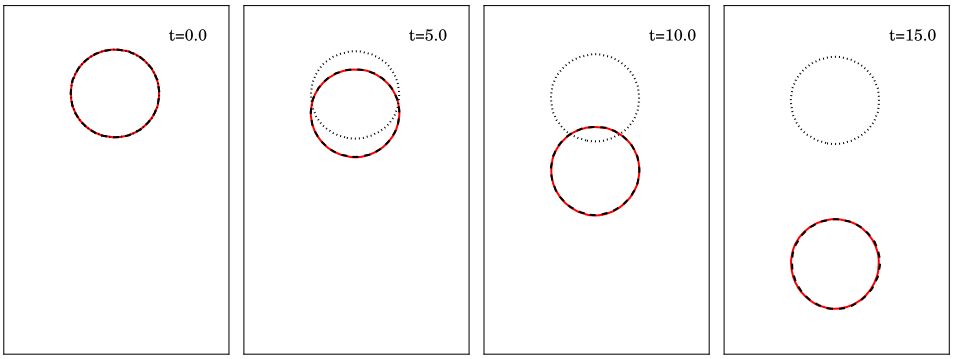


Fig. 1. Time development of a water droplet falling in air using the FastP* (dashed black line), FastP^n (dotted black line), and Unsplit (solid red line) methods. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

width $\tilde{L}_x$ and $\sqrt{100\tilde{g}\tilde{L}_x}$ are used as the reference length and velocity scales respectively in Eq. (7). The first row of Table 2 lists the non-dimensional parameters used in this study.

At time $t = 0$, we released the droplet from rest at $(x, y, z) = (0.5, 0.5, 1.5)$. The flow is integrated from $t = 0$ to $t = 3$ using one of the three different pressure-correction methods described in Section 2.2: FastP*, FastP^n, or Unsplit. Fig. 1 shows the time evolution of the cross-section of the droplet surface in the $y = 0.5$ plane. We find that the FastP* method is in excellent agreement with the Unsplit method, whereas the FastP^n method underpredicts the droplet displacement, in fact, the droplet remains nearly in its initial position. This verifies the FastP* method against the Unsplit method for the falling droplet case. In Section 4.4, we validate the FastP* method by comparing our numerical solution for the terminal velocity of a falling droplet to an experimental value.

3.2. Rising bubble

Next, we simulate a bubble with initial diameter $\tilde{D}_0 = 1.5$ cm rising in aqueous sugar water due to buoyancy. For this case, fluid 1 is water and fluid 2 is air. The second row of Table 2 lists the non-dimensional parameters used for this case. The computational domain and the dimensional parameters are the same as in the falling droplet case (Eq. (43)), except the dynamic viscosity of the sugar water is $\tilde{\mu}_w = 7.5 \times 10^{-2}$ kg/(m s). The domain is discretized with $64 \times 64 \times 128$ grid points, and the CFL number is $\Delta t/\Delta x = 0.005$ to maintain numerical stability due to the high viscosity ratio ($\mu_1/\mu_2 = 4080$). The box width $\tilde{L}_x$ and $\sqrt{\tilde{g}\tilde{L}_x}$ are used as the reference length and velocity scales, respectively, in Eq. (7). At time $t = 0$, we released the bubble from rest at $(x, y, z) = (0.5, 0.5, 0.5)$.

Again, we tested FastP*, FastP^n, and Unsplit for solving this flow. Fig. 2 shows the time evolution of the bubble surface cross-section in the $y = 0.5$ plane. The results show that the FastP* method is again in excellent agreement with the Unsplit method. In contrast, the FastP^n method is in poor agreement with the Unsplit method. Overall, FastP^n correctly predicts the bubble shape but overpredicts its displacement, whereas FastP* correctly predicts the bubble shape and position. This verifies the FastP* method against the Unsplit method for the rising bubble case as well.

3.3. Discussion on the FastP^n and FastP* methods

In this section, we examine the validity of splitting the variable coefficient pressure gradient term as in Eq. (2) from a physical viewpoint by considering three positions in a two-fluid flow: (i) in fluid 1, (ii) in fluid 2, and (iii) at the interface separating fluids 1 and 2.

For our example we assume $\rho_2 > \rho_1$, which gives $\rho_0 = \rho_1$. We consider the first position (in fluid 1) and write Eq. (2):

$$\frac{1}{\rho_1}\nabla p^{n+1} \rightarrow \frac{1}{\rho_1}\nabla p^{n+1} + \left(\frac{1}{\rho_1} - \frac{1}{\rho_1}\right)\nabla \hat{p} = \frac{1}{\rho_1}\nabla p^{n+1}. \tag{44}$$

In (44), we observe that the substitution (2) applied to fluid 1 is exact. Next, we consider (2) in fluid 2 for which the pressure substitution becomes

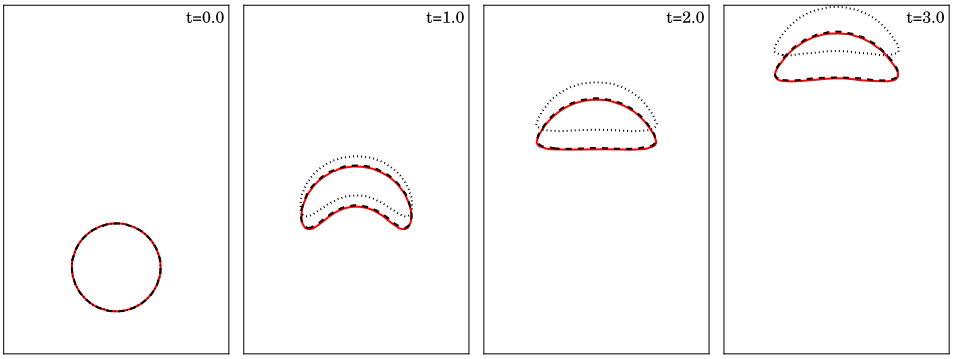


Fig. 2. Time development of an air bubble (initially spherical) in water using the FastP* (dashed black line), FastP^n (dotted black line), and Unsplit (solid red line) methods. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\frac{1}{\rho_2}\nabla p^{n+1} \rightarrow \frac{1}{\rho_1}\nabla p^{n+1} + \left(\frac{1}{\rho_2} - \frac{1}{\rho_1}\right)\nabla \hat{p}. \tag{45}$$

This substitution is a good approximation if $\nabla \hat{p} \approx \nabla p^{n+1}$ and exact if $\nabla \hat{p} = \nabla p^{n+1}$. In other words, (45) is valid if the local pressure is smooth in time (no shocks, i.e., $\nabla \hat{p} \approx \nabla p^{n+1}$), which holds for incompressible flows. Therefore, we conclude that (2) is valid in fluid 2. Last, we consider (2) at the interface, which gives

$$\frac{1}{\langle\rho\rangle}\nabla p^{n+1} \rightarrow \frac{1}{\rho_1}\nabla p^{n+1} + \left(\frac{1}{\langle\rho\rangle} - \frac{1}{\rho_1}\right)\nabla \hat{p}. \tag{46}$$

Again, this substitution is exact if $\nabla \hat{p} = \nabla p^{n+1}$ and valid if $\nabla \hat{p} \approx \nabla p^{n+1}$. However, at the interface, the local pressure field may not be smooth in time and the pressure jump $\Delta p = p_2 - p_1$ is governed by

$$\Delta p = \sigma\kappa + 2\mu_2 \mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{n} - 2\mu_1 \mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{n} \tag{47}$$

where $\mathbf{n}$ is the outward pointing unit normal [18]. We note that because the surface tension force is computed using ∇C, the discrete pressure jump (Δp) will be distributed over a region of thickness $\mathcal{O}(\Delta x)$. The challenge expressed in Eq. (46) is to accurately compute $\hat{p}$ using past pressure values (p^n, p^{n-1}, ...) in the sharp transition region such that $\nabla \hat{p} \approx \nabla p^{n+1}$. Our results suggest that a second-order approximation of $\hat{p}$ (FastP*) is significantly more accurate than a first-order approximation of $\hat{p}$ (FastP^n) for predicting the pressure gradient ∇p^{n+1} in the interfacial region. In fact, we found that the first-order approximation is only usable for $\Delta p \approx 0$, i.e., in the limit of zero surface tension ($\sigma = 0$) and inviscid flow ($\mu_2 = \mu_1 = 0$). We also found that using higher-order approximations beyond second-order gave inaccurate results.

While testing the FastP* method, we have found that only in extreme cases of combined high surface tension, curvature, and density ratio, e.g., micron size water droplets in air, the CFL number must be reduced by an amount proportional to the product of σ, κ, and $\max(\rho_2, \rho_1)/\min(\rho_2, \rho_1)$ to preserve numerical accuracy. However, in these extreme cases, the role of surface tension is only to maintain the droplet's spherical shape. Therefore, the surface tension coefficient σ can be reduced as long as the droplet remains spherical, and thus the CFL number can be increased without loss of accuracy. We note that for all the cases presented in this paper, the difference in droplet position and velocity of FastP* relative to Unsplit was less than 1%.

3.4. Performance

The solution of the Poisson equation for pressure is the computational bottleneck in pressure-correction methods, taking more than 50% of the computational time. In this section, we investigate the computational cost of solving the variable and constant coefficient Poisson equations for pressure in two-fluid flows with the iterative and direct (fast Poisson) solvers, respectively.

To solve the variable coefficient Poisson equation (13), we incorporated the *hypre* library [23] into our flow solver. The *hypre* library allows us to benchmark the latest parallel techniques for solving the variable coefficient Poisson equation (13) iteratively. We tested different combinations of preconditioners and solvers and found that *hypre* SMG outperformed all others for our test case on a Cartesian uniform grid. Also, we note that *hypre* SMG has good scaling up to 100,000 cores [35] and has been used in other massively parallel incompressible flow solvers, e.g., [36]. Thus, we used *hypre* SMG to solve Eq. (13) in our Unsplit method. Finally, to minimize the number of iterations, we used the previous pressure field (p^n) as an initial guess of the pressure solution (p^{n+1}).

For our test case we simulated a single droplet in decaying isotropic turbulence for 256 time steps. The parameters are:

$$\mathrm{Re} = 107{,}000, \qquad \mathrm{We} = 0.25, \qquad \mathrm{Fr} = \infty, \qquad D_0 = 0.25, \qquad \rho_2/\rho_1 = 1000, \qquad \mu_2/\mu_1 = 1. \tag{48}$$

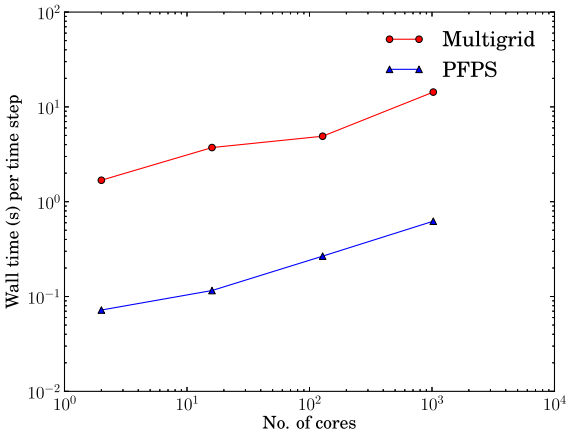


Fig. 3. Weak scaling of multigrid and fast Poisson solvers on Intel Xeon E5-2680.

Table 3
Wall clock time per time step (s) and speedup factor for the solution of the Poisson equation (middle column) and the two-fluid Navier–Stokes equations (right column) on a 1024^3 grid using 1024 computing cores on Intel Xeon E5-2680.

Multigrid tolerance	Poisson solver			Two-fluid Navier–Stokes solver		
	Multigrid (s)	PFPS (s)	Speedup	Unsplit (s)	FastP* (s)	Speedup
$\|\nabla \cdot \mathbf{u}^{n+1}\|_{\max} < 10^{-6}$	14.32	0.62	23.00	14.74	1.06	13.88
$\|\nabla \cdot \mathbf{u}^{n+1}\|_{\max} < 10^{-14}$	42.04	0.62	67.52	42.46	1.06	39.99

The initial Reynolds number based on the Taylor microscale of turbulence, λ, is $\mathrm{Re}_{\lambda 0} = 190$. The simulations were performed on an Intel Xeon E5-2680 system. We analyzed the weak scaling of the two Poisson solvers by keeping the problem size per core constant at $128^3/2$ grid points and increasing the number of cores from 2 to 1024. For example, when we used 1024 cores we used 1024^3 grid points. For all core counts, *hypre* SMG required 8 multigrid iterations when using a 10^{-6} tolerance. Fig. 3 shows the average CPU time required to solve the Poisson equations (13) and (16) using multigrid (*hypre* SMG) and our parallel implementation for multi-core computers of the fast Poisson solver by Schmidt et al. [37] (see Algorithm of PFPS in Section 2.2), respectively. PFPS is about twenty times faster than the multigrid method for all core counts tested. For all cases, the total computational time to solve the two-fluid Navier–Stokes equations ((11), (16), and (17)) for FastP* was ten times lower than that for the Unsplit method.

In addition, the FastP* method satisfies the divergence-free constraint on the velocity field to machine precision at every grid point (i.e., $|\nabla \cdot \mathbf{u}^{n+1}|_{\max} < 10^{-14}$). In contrast, the Unsplit method requires decreasing the convergence tolerance of the multigrid solver to achieve machine precision accuracy. Table 3 summarizes the wall clock time and speedup seen when using Multigrid versus PFPS and Unsplit versus FastP* for a tolerance of 10^{-6} and 10^{-14}. These results are for the 1024^3 grid (1024 compute cores). The table shows that for $|\nabla \cdot \mathbf{u}^{n+1}|_{\max} < 10^{-6}$ PFPS is 23 times faster than Multigrid for solving Eq. (13) and Eq. (16), respectively. If the tolerance is decreased to $|\nabla \cdot \mathbf{u}^{n+1}|_{\max} < 10^{-14}$, the speedup factor increases to 67.52. Table 3 shows that for $|\nabla \cdot \mathbf{u}^{n+1}|_{\max} < 10^{-6}$ FastP* is 13.88 times faster than Unsplit for solving Eqs. (11)–(13) and Eqs. (10), (16), and (17), respectively. For the decreased tolerance of $|\nabla \cdot \mathbf{u}^{n+1}|_{\max} < 10^{-14}$, the speedup factor increases to 39.99. As we mentioned, PFPS and FastP* satisfy $|\nabla \cdot \mathbf{u}^{n+1}|_{\max} < 10^{-14}$ by default, whereas for Unsplit, the Multigrid convergence tolerance must be decreased to 10^{-14}. The decreased tolerance increases the number of Multigrid iterations to 34 resulting in greater wall clock time. Table 3 also shows that for the Unsplit method the solution of Eq. (13) takes greater than 97% of the solution time, and for the FastP* method the solution of Eq. (16) takes greater than 58% of the solution time.

Finally, the pressure-splitting (14) is also advantageous for iterative methods like *hypre* SMG because it yields only constant coefficient matrices (see left-hand side of Eq. (16)), whereas standard Unsplit methods involve variable coefficient matrices (see left-hand side of Eq. (13)) that must be setup and computed at each timestep. We found that the setup of the variable coefficient matrix using *hypre* SMG for the Unsplit method costs 25% of the total Poisson solution time. Thus, using Eq. (14), saves computational time even when iterative methods are used.

4. Results

In this section, we test the new pressure-correction method (FastP*) coupled with our volume-of-fluid method [17] in two canonical flows as well as a more complex flow. The focus is on verification and validation of the new flow solver. In all the simulations, the total mass error for C is less than 10^{-8}. A prerequisite for a mass-conserving VoF advection scheme is

Table 4
Temporal errors and convergence rate for velocity (u) and pressure (p) in the Taylor–Green vortex flow with a fluid 2 cylinder centered in the domain.

	$E_{4\Delta t,2\Delta t}$	$E_{2\Delta t,\Delta t}$	Rate
u	5.94e−8	1.53e−8	1.96
p	3.67e−6	1.84e−6	0.99

Table 5
Spatial errors and convergence rate for velocity (u) and pressure (p) in the Taylor–Green vortex flow with a fluid 2 cylinder centered in the domain.

	$E_{4\Delta x,2\Delta x}$	$E_{2\Delta x,\Delta x}$	Rate
u	6.08e−3	1.51e−3	2.01
p	1.04e−1	2.63e−2	1.99

a divergence-free velocity field. Therefore, another advantage of the FastP* is that by solving Eq. (16) directly, the resulting discrete velocity field (Eq. (17)) is divergence free to machine precision.

4.1. Convergence rates

We assess the solver's numerical accuracy by computing its spatial and temporal convergence rates. Special consideration is needed when computing the temporal accuracy of the coupled pressure-correction/VoF flow solver because the VoF advection method uses first-order forward Euler to integrate Eq. (41) in time, which is the standard for geometrical VoF methods [30, Chapter 5, p. 108]. In Section 4.1.1, we compute the convergence rates independent of the method used to advect the interface by turning the VoF advection off such that $C = C(\mathbf{x})$ for all times. In Section 4.1.2 we present the convergence rates with the VoF advection turned on, i.e., $C = C(\mathbf{x}, t)$.

4.1.1. Convergence rates without VoF advection

We simulate a fluid cylinder (fluid 2) centered in a two-dimensional Taylor–Green vortex flow (fluid 1) [38]. The non-dimensional parameters for the flow are:

$$\mathrm{Re} = 200, \qquad \mathrm{We} = \infty, \qquad \mathrm{Fr} = \infty, \qquad D = 0.25, \qquad \rho_2/\rho_1 = 10, \qquad \mu_2/\mu_1 = 10. \tag{49}$$

The fluid velocity in the interior of the cylinder was set equal to the local Taylor–Green vortex velocity at time $t = 0$. In time, the vorticity of both fluids decays owing to viscous diffusion.

We simulated the flow on a computational mesh of 256^2 points. The smallest time step used was $\Delta t = 4.88\mathrm{e}{-5}$ and the number of time steps to reach $t_{\max} = 0.05$ was $N_t = 1024$. To evaluate the temporal convergence rate of the solver, the time step was increased to $2\Delta t$ and $4\Delta t$ ($N_t = 512$ and 256) and we computed the error between successive solutions. The L_2 error E for generic flow variable ξ is computed as:

$$E_{2\Delta t,\Delta t} = \frac{1}{N_x N_y}\sqrt{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left(\xi_{2\Delta t}^{(i,j)} - \xi_{\Delta t}^{(i,j)}\right)^2}. \tag{50}$$

Table 4 shows the error between successive time steps and the convergence rate. The convergence rate of the u-component of velocity is 1.96, therefore the velocity is second-order in time. The convergence rate of pressure is 0.99, thus it is first-order in time. Second-order pressure convergence in time can be achieved if $\Delta t \gg \mathrm{Re}\,\Delta x^2$ and periodic boundary conditions are applied [39]. The condition $\Delta t \gg \mathrm{Re}\,\Delta x^2$ is unlikely to be satisfied in high Reynolds number (Re) turbulent flows. Therefore, the accuracy of the pressure is expected to be first-order for high Re flows and cannot be improved by switching to semi-implicit or implicit time integration, e.g. Adams–Bashforth Crank–Nicolson scheme (see Appendix A) when using the pressure-correction method presented. There is a method in which the pressure is 3/2-order accurate in time, but it was developed for single-phase flow [40].

Next, we assess the spatial accuracy of the solver by successively refining the grid and computing the convergence rate of the solution. We again solve the above described Taylor–Green vortex case. We used grids with 32^2, 64^2, and 128^2 points, and we used CFL number $\Delta t/\Delta x = 0.001$ such that temporal errors are small. Table 5 shows the error between successive grid refinements and the convergence rate. Both velocity and pressure have a second-order convergence rate (2.01 and 1.99, respectively), which is consistent with our central difference discretization in space. In summary, FastP* is first-order in time for the pressure, second-order in time for the velocity, and second-order in space for both pressure and velocity.

4.1.2. Convergence rates with VoF advection

We assess the coupled FastP*/VoF flow solver's numerical accuracy by computing the spatial and temporal convergence rates. The test problem is a two-dimensional capillary wave in a periodic domain. The non-dimensional parameters for the

Table 6
Temporal errors and convergence rate for velocity component (u), pressure (p), and volume fraction (C) in the 2-D capillary wave flow.

	$E_{4\Delta t,2\Delta t}$	$E_{2\Delta t,\Delta t}$	Rate
u	7.32e−9	3.91e−9	0.90
p	4.67e−7	2.32e−7	1.01
C	7.38e−8	3.58e−8	1.04

Table 7
Spatial errors and convergence rate for velocity component (u), pressure (p), and volume fraction (C) in the 2-D capillary wave flow.

	$E_{4\Delta x,2\Delta x}$	$E_{2\Delta x,\Delta x}$	Rate
u	2.52e−5	6.33e−6	1.99
p	1.68e−3	4.52e−4	1.89
C	1.88e−3	5.14e−4	1.87

flow are:

$$\mathrm{Re} = 500, \qquad \mathrm{We} = 1.0, \qquad \mathrm{Fr} = \infty, \qquad H_0 = 0.05, \qquad \rho_2/\rho_1 = 20, \qquad \mu_2/\mu_1 = 20. \tag{51}$$

We do not use Prosperetti's solution [41] for computing the convergence rate of the error because it was derived for fluids of infinite depth and we are using a periodic domain. Instead, we compute the temporal convergence rate by comparing the error in the flow solution computed using successively smaller time steps Δt.

We simulated the flow on a 256^2 grid. The flow was integrated up to $t_{\max} = 0.0625$, and the smallest time step used was $\Delta t = 4.88\mathrm{e}{-5}$ ($N_t = 1280$). To evaluate the temporal convergence rate of the solver, the time step was increased to $2\Delta t$ and $4\Delta t$ ($N_t = 640$ and 320), and we computed the error between successive solutions. The L_2 error E for generic flow variable ξ (u, p, or C) is computed using Eq. (50).

Table 6 shows the error between successive time step sizes and the convergence rate for u, p, and C. The convergence rate of u is 0.90, which is slightly less than first-order. The convergence rate of pressure and the volume fraction are 1.01 and 1.04 respectively, thus it is first-order. Overall, the coupled flow solver is first-order accurate in time. The cause of the first-order temporal accuracy, despite using second-order Adams–Bashforth in Eq. (11), is that the volume fraction is advected by first integrating Eq. (41) in each cell and then integrating in time using forward Euler [21,17]. To overcome this limitation, higher order VoF advection methods have been employed, however they are inadequate for long time simulations [30, Chapter 4, p. 80]. The flow solver could be made fully second-order by using a second-order in time method to advect the interface.

Next, we assess the spatial accuracy of the solver by successively refining the grid and computing the convergence rate of the solution. We solve the above described capillary wave case on 32^2, 64^2, and 128^2 grids. The CFL number is held constant and set to $\Delta t/\Delta x = 0.0025$. Table 7 shows the error between successive grid refinements and the convergence rate. The velocity, pressure, and volume fraction all have nearly second-order convergence rates (1.99, 1.89, and 1.87, respectively), which is consistent with our central difference discretization in space (Section 2.2.1) and second-order VoF method [17]. We conclude that the coupled FastP*/VoF method is second-order accurate in space.

4.2. Conservation of momentum and kinetic energy

We have checked for discrete linear momentum and kinetic energy conservation in the inviscid limit by simulating a fluid cylinder centered in a two-dimensional Taylor–Green vortex using

$$\mathrm{Re} = \infty, \qquad \mathrm{We} = \infty, \qquad \mathrm{Fr} = \infty, \qquad D = 0.25, \qquad \rho_2/\rho_1 = 10, \qquad \mu_2/\mu_1 = 1. \tag{52}$$

such that the viscous dissipation of kinetic energy is zero and the interface cannot store energy. We used 128^2 grid points and, at $t = 0$, the fluid velocity in the interior of the cylinder was set equal to the local Taylor–Green vortex velocity. Fig. 4(a) shows the time development of the total linear momentum in the x_i direction,

$$\bar{p}_i = \int_V \rho u_i \mathrm{d}V \tag{53}$$

normalized by the magnitude of the largest initial momentum vector, $\|\mathbf{p}_0\|$. Fig. 4(b) shows the time development of the kinetic energy

$$K = \int_V \rho u_i u_i \mathrm{d}V \tag{54}$$

for the FastP* versus the Unsplit method. Fig. 4 shows that the FastP* method and the Unsplit method are in nearly exact agreement. Also, Fig. 4 shows that the FastP* method conserves momentum and kinetic energy. We associate the slight

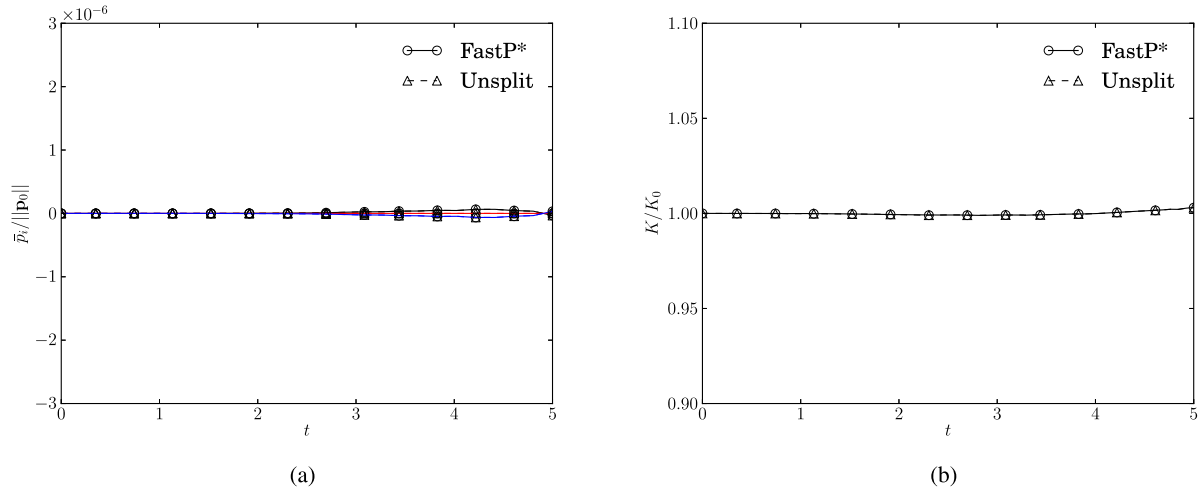


Fig. 4. Time development of (a) the x- (black), y- (red), and z-component (blue) of momentum using the FastP* and Unsplit method and (b) the kinetic energy using the FastP* and the Unsplit method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

increase in K seen for $t > 4$ to the increasing deformation of the cylinder interface, and thus decreasing cylinder resolution, $D/\Delta x$. For $t > 4$, parts of the cylinder interface are under resolved ($D/\Delta x < 6$) which decreases the accuracy of the VoF advection algorithm.

4.3. Verification: capillary wave

To verify the new pressure-correction method described in Section 2.2, we simulated the capillary wave test case for which there is an analytical solution that was derived by Prosperetti [41]. This test accounts for the effects of surface tension, density ratio, and viscosity ratio. This problem is defined as two immiscible fluids separated by an interface that is initialized with a sinusoidal shape with wavelength λ and small wave amplitude H_0. Prosperetti's analytical solution [41] that we use was derived for two fluids of infinite depth and with equal kinematic viscosities ν.

For our test, we simulated two fluids of equal depth in a periodic domain. The domain has dimensions $0 \leq x \leq 1$ and $-1.5 \leq y \leq 1.5$ (64×192 grid points). The volume fraction boundary conditions at the bottom and top boundaries are $C = 1$ for $y = -1.5$ and $C = 0$ for $y = 1.5$. The interface was initialized using a cosine wave with amplitude $H_0 = 0.01$ and wavelength $\lambda = 1.0$ centered at position $y = 0$. The non-dimensional parameters for this case are

$$\mathrm{Re} = 100, \qquad \mathrm{We} = 1, \qquad \mathrm{Fr} = \infty, \qquad \rho_2/\rho_1 = 10\text{–}10{,}000, \qquad \mu_2/\mu_1 = 10\text{–}10{,}000. \tag{55}$$

Fig. 5 shows the time development of the wave amplitude H for four cases corresponding to density and viscosity ratios of 10, 100, 1000, and 10,000. For all cases, the numerical simulation is in excellent agreement with Prosperetti's analytical solution, which verifies the pressure-correction method. For the density and viscosity ratio 10 and 100 cases, we used CFL number $\Delta t/\Delta x = 0.01$ so that temporal errors from the VoF advection scheme are negligible [17]. For the 1000 and 10,000 case, we were required to reduce $\Delta t/\Delta x$ to 0.0025 and 0.00025, respectively, to maintain numerical stability. This reduction was required for all the pressure-correction methods tested (i.e., Unsplit, FastP*, and FastP^n). Therefore, we conclude that this numerical stability restriction originates from large density and viscosity gradients produced by the VoF's sharp interface method and not the FastP* method.

4.4. Validation: falling droplet in quiescent air

To validate the new pressure-correction method described in Section 2.2, we simulated falling water droplets of varying initial diameter ($\tilde{D}_0 = 0.6$, 1.0, and 1.4 mm) in quiescent air. We determine if the solver can accurately predict the droplet's terminal velocity. We validate our numerical results against Beard's experimental results [42].

The flow is simulated in the droplet reference frame to avoid the extraordinary computational cost required for the fixed reference frame. For example, a 1 mm droplet released from rest will travel tens of meters before reaching its terminal velocity. The computational domain is a rectangular box with dimensions $(L_x \times L_y \times L_z) = (16D_0 \times 16D_0 \times 32D_0)$ and gravity directed in the negative z-direction. Periodic boundary conditions were imposed in the horizontal x- and y-directions. At the bottom boundary, we imposed a Neumann condition for pressure and a uniform inflow condition for velocity,

$$\frac{\partial p^{n+1}}{\partial z} = 0, \qquad u^* = u^{n+1} = 0, \qquad v^* = v^{n+1} = 0, \qquad w^* = w^{n+1} = V_{\mathrm{t,B}} \quad \text{at } z = 0, \tag{56}$$

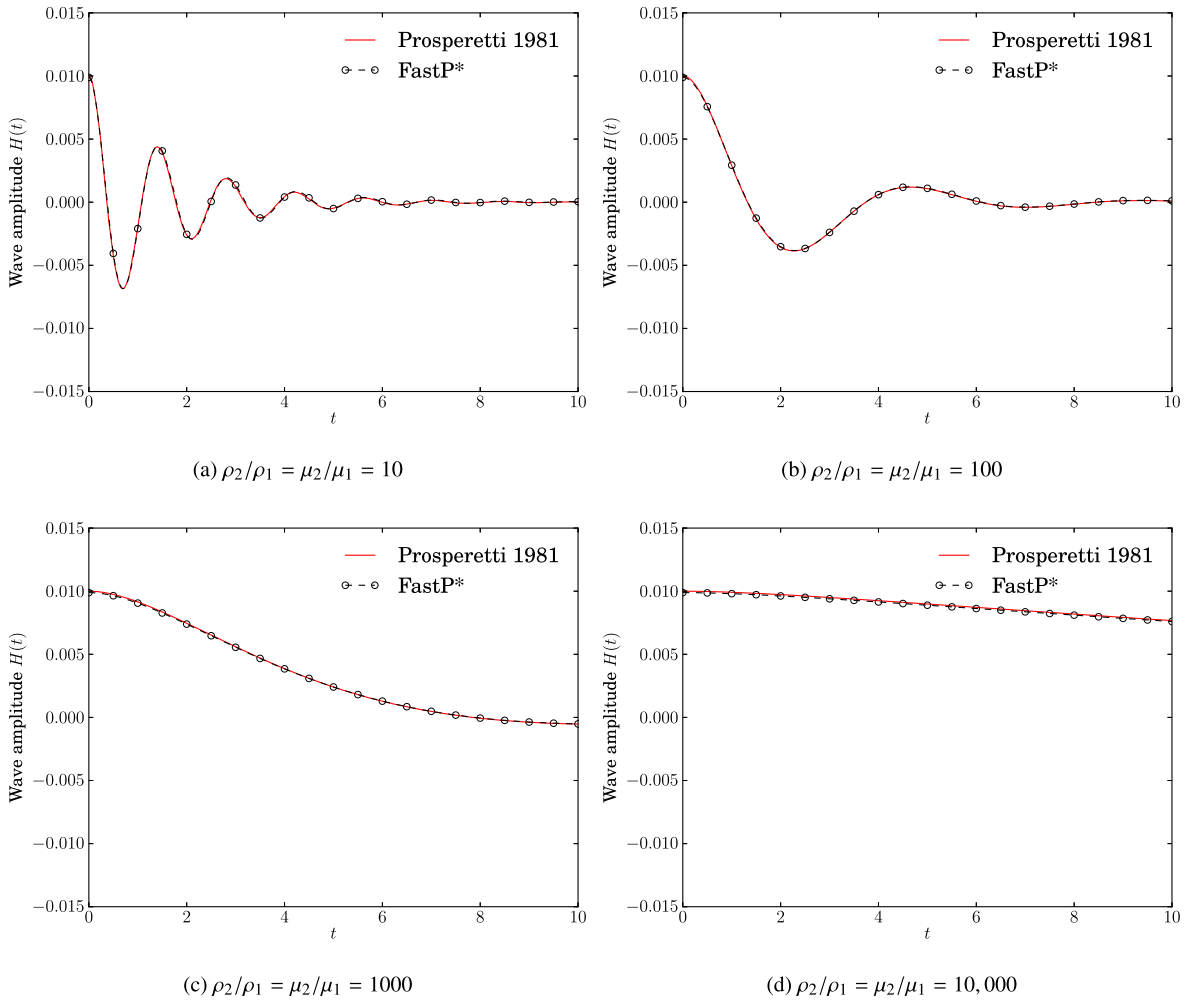


Fig. 5. Time development of the capillary wave amplitude for different density and viscosity ratios. Comparison of FastP* numerical solution versus Prosperetti's analytical solution [41].

where $V_{t,B}$ is the droplet's non-dimensional terminal velocity given by Beard's formula [42, Table 1]. At the top boundary, we imposed a Dirichlet condition for pressure and Neumann (stress-free) condition for u, v, and w

$$p^{n+1} = 0, \qquad \frac{\partial u^*}{\partial x} = \frac{\partial u^{n+1}}{\partial x} = 0, \qquad \frac{\partial v^*}{\partial y} = \frac{\partial v^{n+1}}{\partial y} = 0, \qquad \frac{\partial w^*}{\partial z} = \frac{\partial w^{n+1}}{\partial z} = 0 \quad \text{at } z = L_z. \tag{57}$$

The non-dimensional parameters given by Eq. (7) are computed using the box width $\tilde{L}_x$ as the reference length scale, Beard's terminal velocity $\tilde{V}_{t,B}$ [42] as the reference velocity scale, and the dimensional parameters given in Eq. (43). We choose the surrounding fluid (air) as fluid 1 and the droplet (water) as fluid 2. The density ratio $\rho_2/\rho_1 = 829.0$, viscosity ratio $\mu_2/\mu_1 = 54.5$, and initial non-dimensional droplet diameter $D_0 = 0.0625$, were the same for all cases. Table 8 shows the non-dimensional parameters for the three cases of different dimensional droplet diameter studied.

4.4.1. Terminal velocity

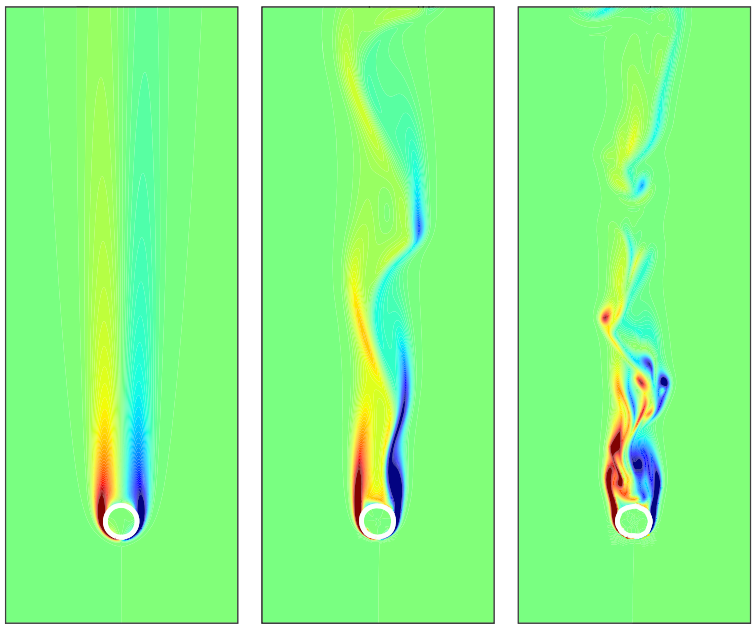
The flow is initialized as $(\tilde{u}, \tilde{v}, \tilde{w}) = (0, 0, \tilde{V}_{t,B})$ except in the (initially spherical) droplet interior, which is set to zero. We simulated the flow on a $512 \times 512 \times 1024$ grid, giving a droplet resolution of 32 grid points per diameter. The simulation is integrated in time from $t = 0$ to $t = 10$, allowing enough time for the droplet to reach equilibrium, i.e., a balance between the drag and gravitational force. This is marked by the droplet's vertical centroid velocity being nearly constant in time (i.e., $d\tilde{V}_d/dt \approx 0$). After this condition is met, the numerically determined terminal velocity $\tilde{V}_{t,DNS}$ can be found from $\tilde{V}_{t,DNS} = \tilde{V}_{t,B} + \tilde{V}_d$.

Table 8 displays the Reynolds, Weber, and Bond numbers defined, respectively, as

$$\mathrm{Re}_{D_0} = \frac{\tilde{\rho}_a \tilde{V}_{t,B} \tilde{D}_0}{\tilde{\mu}_a}, \qquad \mathrm{We}_{D_0} = \frac{\tilde{\rho}_a \tilde{V}_{t,B}^2 \tilde{D}_0}{\tilde{\sigma}}, \quad \text{and} \quad \mathrm{Bo}_{D_0} = \frac{(\tilde{\rho}_w - \tilde{\rho}_a)\tilde{g}\tilde{D}_0^2}{\tilde{\sigma}}. \tag{58}$$

Table 8
Falling droplet test parameters. Droplet terminal velocity from Beard's formula [42, Table 1], $\tilde{V}_{t,B}$, and from DNS result, $\tilde{V}_{t,DNS}$, and the % difference of the two velocities.

$\tilde{D}_0$ (mm)	Re_{D_0}	We_{D_0}	Bo_{D_0}	$\tilde{V}_{t,B}$ (m/s)	$\tilde{V}_{t,DNS}$ (m/s)	% difference
0.6	95.6	0.06	0.05	2.430	2.441	0.4
1.0	261.8	0.26	0.13	3.994	3.990	−0.1
1.4	472.5	0.61	0.26	5.149	4.957	−3.7



(a) $\tilde{D}_0 = 0.6$ mm (b) $\tilde{D}_0 = 1.0$ mm (c) $\tilde{D}_0 = 1.4$ mm

Fig. 6. Instantaneous vorticity contours ω_y in a vertical midplane of a water droplet falling in air (Table 8). The white line is the droplet interface ($C = 0.5$ isoline).

Also shown in Table 8 are the dimensional terminal velocities and % difference with the experimental value ($100 \times (\tilde{V}_{t,DNS} - \tilde{V}_{t,B})/\tilde{V}_{t,B}$). For all three cases, we show excellent agreement with Beard, which validates our VoF/pressure-correction method.

4.4.2. Droplet wake

For water droplets falling in air with $\mathrm{Bo}_{D_0} < 0.26$ (i.e., $D_0 = 0.6$, 1.0, and 1.4 mm), the droplet remains nearly spherical, specifically, there is less than 6% deviation in height to width ratio from that of a sphere [43, Chapter 7, p. 178]. Therefore, like rigid spheres, these droplets' wakes fall into different regimes depending only on the droplet Reynolds number Re_D. To test the ability of our solver to predict the flow characteristics of these regimes, we have chosen three droplet diameters such to produce wakes that fall into three different regimes. The first regime is for intermediate Reynolds number ($20 \lesssim \mathrm{Re}_D \lesssim 200$), which is characterized by laminar, steady axisymmetric flow, with an attached vortex ring [42]. Fig. 6(a) shows the instantaneous vorticity contours ω_y for an x–z midplane of the $\tilde{D}_0 = 0.6$ mm droplet with $\mathrm{Re}_{D_0} = 95.6$. The flow solver correctly predicts the laminar and axisymmetric stationary wake. The second regime is for moderate Reynolds number ($200 \lesssim \mathrm{Re}_D \lesssim 450$), which is characterized by asymmetric vorticity in the unsteady wake. Fig. 6(b) shows the vorticity ω_y contours for the $\tilde{D}_0 = 1.0$ mm droplet with $\mathrm{Re}_{D_0} = 261.8$. The third regime is for large Reynolds number ($\mathrm{Re}_D \gtrsim 450$), which is characterized by a chaotic wake. Fig. 6(b) shows the $\tilde{D}_0 = 1.4$ mm droplet with chaotic wake at $\mathrm{Re}_{D_0} = 472.5$. In conclusion, our flow solver predicts the main flow features of the droplet wake in the range $95.6 \leq \mathrm{Re}_D \leq 472.5$.

To summarize our results for the falling droplet, we found that the coupled pressure-correction/VoF flow solver accurately predicts the terminal velocity of falling water droplets with diameters $\tilde{D}_0 = 0.6$ to $\tilde{D}_0 = 1.4$ mm. Also, the flow solver accurately predicts the droplet wake when the droplet reaches its terminal velocity.

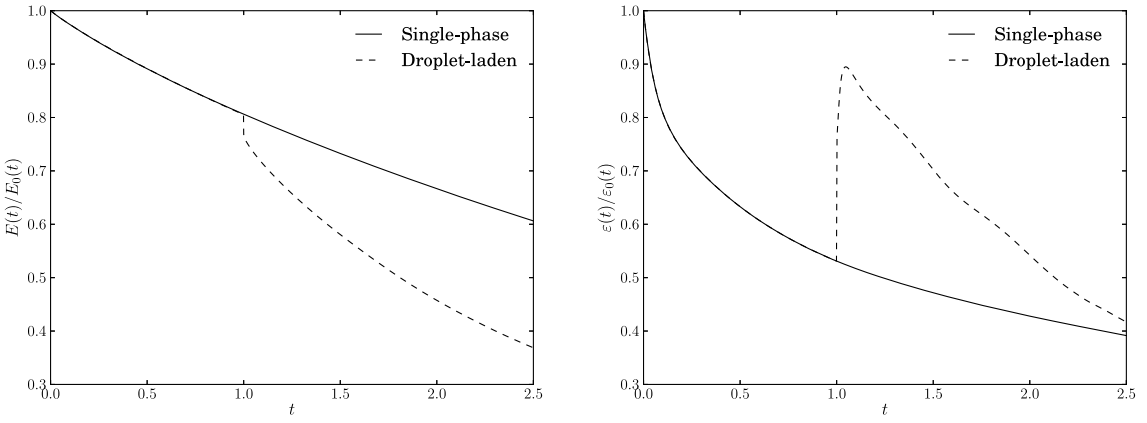


Fig. 7. Time development of turbulent statistics normalized by their initial value. Turbulent kinetic energy (left); dissipation rate (right).

4.5. DNS of droplet-laden isotropic turbulence

To show that the FastP* method is capable of solving two-fluid turbulent flows, we performed DNS of droplet-laden decaying isotropic turbulence. In the simulation we used the following parameters:

$$\mathrm{Re} = 64{,}200, \qquad \mathrm{We} = 1530, \qquad \mathrm{Fr} = \infty, \qquad D_0 = 0.03125, \qquad \rho_2/\rho_1 = 10, \qquad \mu_2/\mu_1 = 10. \tag{59}$$

The initial Taylor-scale Reynolds number is $\mathrm{Re}_{\lambda 0} = 75$. The computational domain is a cube with side length $L = 1$ and 1024^3 grid points. The carrier phase (fluid 1) is randomly seeded at $t = 1$ with 6260 spherical droplets (fluid 2) with diameter $D_0 = 0.03125$, corresponding to approximately one Taylor microscale, λ, and 20 Kolmogorov lengthscales, η. The droplet Weber number based on the carrier phase RMS velocity at release time is $\mathrm{We}_{\mathrm{rms}} = 0.1$ ($\mathrm{We}_{\mathrm{rms}} = \rho_1 U_{\mathrm{rms}}^2 D_0 \mathrm{We}$). The droplet resolution is 32 grid points per diameter and the droplet volume fraction is $\phi_v = 0.1$.

The droplets are released at $t = 1$ and their interior fluid velocity was initially set to zero. The simulation was integrated from $t = 0$ to $t = 2.5$ for 25,600 time steps.[4] Fig. 7(a) shows the time development of the turbulent kinetic energy (TKE, $\frac{1}{2}\langle \mathbf{u} \cdot \mathbf{u} \rangle$)[5] of the carrier phase for the droplet-laden flow and the single-phase flow. Note that the TKE was only computed for the carrier-phase. Fig. 7(a) shows that the presence of the droplets increases the TKE decay rate. Next, we computed the TKE dissipation rate ($\varepsilon = 2\nu\langle s_{ij} s_{ij} \rangle$, where s_{ij} is the rate-of-strain tensor of velocity fluctuations) for the single-phase and droplet-laden cases. Fig. 7(b) shows the time development of the dissipation rate for the carrier phase. When the droplets are injected ($t = 1$), the dissipation rate becomes larger than that for single-phase flow, thus increasing the TKE decay rate.

Fig. 8 shows instantaneous contours of the strain rate squared $s_{ij} s_{ij}$ in a cross-section of the full-domain and a close-up for an arbitrarily selected subdomain. There is increased $s_{ij} s_{ij}$ in the regions surrounding the droplet surface. Thus, $\varepsilon(t)$ increases in the droplet-laden case compared to the single-phase case. This in turn reduces the TKE relative to the single-phase case. A detailed description of the physical mechanisms of droplet-laden decaying isotropic turbulence will be the objective of a future paper.

5. Conclusion

We have developed an efficient and second-order accurate pressure-correction method for incompressible two-fluid flows. The method reduces the Poisson equation for pressure from variable to constant coefficient, starting from a splitting idea by Dong and Shen [13]. However, our numerical tests showed that applying this splitting to the pressure gradient term (i.e., using Eq. (2) with $\hat{p} = p^n$, called our FastP^n method) is only accurate for vanishingly small surface tension and viscosity. For viscous flows with finite surface tension, we showed that $\hat{p} = p^* = 2p^n - p^{n-1}$ should be used (FastP*). Additionally, we verified FastP* up to a density and viscosity ratio of 10,000, and validated the method for a water droplet falling in air. Finally, we showed that FastP* is capable of fully-resolved DNS of thousands of freely moving droplets in decaying isotropic turbulence on a 1024^3 grid. In summary, we have shown that FastP*:

- reduces the commonly encountered variable coefficient Poisson equation to a constant coefficient equation, which allows for the use of a direct Poisson solver,
- is in excellent agreement with the Unsplit method,

[4] This simulation required 30.7 hours on 1024 cores (Intel Xeon E5-2680 2.70 GHz processor), which gives a *grind time* of 4.11e−6 sec. The grind time is defined as the CPU time per grid point per time step.

[5] $\langle \ldots \rangle$ denotes the ensemble average of the enclosed quantity over the grid points occupied by fluid 1.

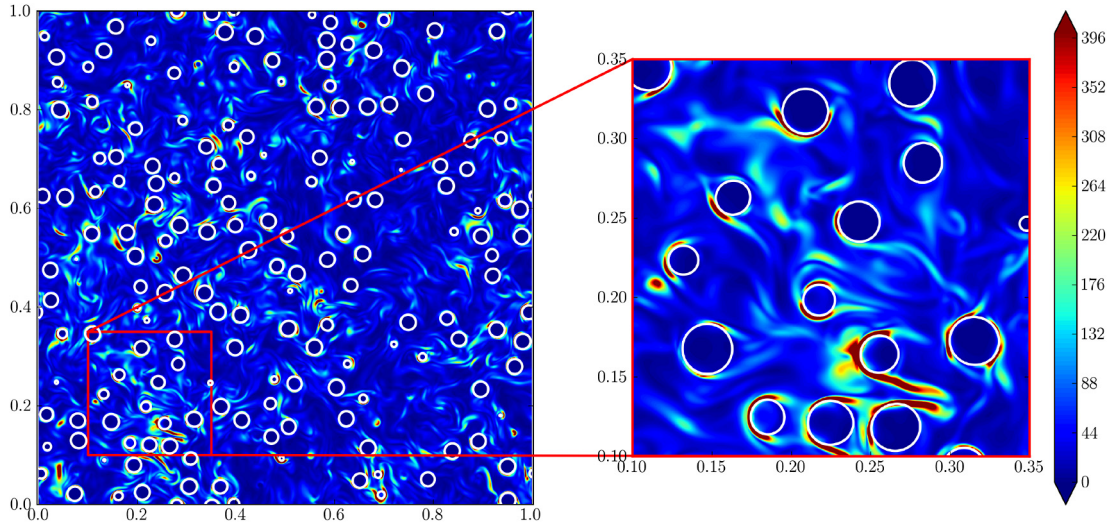


Fig. 8. Instantaneous contours in x–z plane of $s_{ij}s_{ij}$ at $t = 2.0$. Full domain (left); subdomain close-up (right). (For interpretation of the colors in this figure, the reader is referred to the web version of this figure.)

- is about 10–40 times faster than the Unsplit method and our fast Poisson solver is about 20–60 times faster than Multigrid, depending on the tolerance used for Multigrid,
- is second-order in space and time for velocity and second- and first-order in space and time, respectively, for pressure,
- is verified for density and viscosity ratios up to 10,000,
- is validated for a falling water droplet in quiescent air for $95.6 \leq \mathrm{Re} \leq 472.5$, $0.06 \leq \mathrm{We} \leq 0.61$, and $0.05 \leq \mathrm{Bo} \leq 0.26$,
- conserves mass, momentum, and kinetic energy (in the inviscid limit),
- is capable of fully-resolved DNS of thousands of freely moving droplets in decaying isotropic turbulence on a 1024^3 grid in $\mathcal{O}(10)$ hours using 1024 computing cores.

We conclude that FastP* is well-suited for simulating two-fluid flows requiring high spatial and temporal resolution throughout the domain, e.g., droplet- or bubble-laden turbulent flows. Also, because FastP* uses a direct solver, it can be easily adopted in existing incompressible flow codes that rely on a direct solver. Furthermore, FastP* is capable of simulating flows with Dirichlet and Neumann boundary conditions for pressure by using the appropriate modifications to the fast Poisson solver (see, e.g., [4,5]).

Acknowledgements

This work was supported by the National Science Foundation CAREER Award, Grant No. OCI-1054591. M.S.D. was also partially supported by the Dean's Fellowship from the College of Engineering at the University of Washington (UW), Seattle. The numerical simulations were performed in part on Hyak, high-performance computer cluster at UW, and in part on the Extreme Science and Engineering Discovery Environment (XSEDE) under XRAC Grant No. TG-CTS100024. XSEDE is supported by National Science Foundation Grant No. ACI-1053575. We specifically acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. Also, we specifically acknowledge the assistance of the XSEDE Extended Collaborative Support Service (ECSS) team members, Jay Alameda and Darren Adams, of the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana–Champaign. We also thank Dr. Uri Shumlak, Dr. Randall J. LeVeque, and Dr. James J. Riley for their constructive comments on a draft of this manuscript.

Appendix A. Implicit time integration of viscous terms

In solving the Navier–Stokes equations, the viscous numerical stability restriction ($\Delta t \leq (1/2)\Delta t_\nu$) can be eliminated by treating the viscous terms implicitly using Crank–Nicolson time integration. This allows for a greater time step in low Reynolds number simulations, which allows $\Delta t \gg \mathrm{Re}\,\Delta x^2$, a requirement for second-order temporal accuracy to be realized on the pressure [39]. The implicit treatment introduces a variable coefficient Helmholtz equation which has been most efficiently solved iteratively in other two-fluid pressure-correction methods (e.g., in [8]). Instead, we reduce the variable coefficient Helmholtz equations to constant coefficient equations (similarly to the pressure splitting in Eq. (2)) by using the basic splitting idea [14,13]. The constant coefficient Helmholtz equation can then be solved directly using a fast FFT-based elliptic solver (e.g., [4,5]). For time integration, we use second-order Adams–Bashforth for the convective and force terms and second-order Crank–Nicolson for the diffusive terms [44] which gives:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = \frac{1}{2}\left(3\mathbf{H}^n - \mathbf{H}^{n-1}\right) + \frac{1}{2}\frac{1}{\mathrm{Re}}\frac{1}{\rho^{n+1}}\left[\nabla\cdot\left(\mu^{n+1}\mathbf{D}(\mathbf{u}^n)\right) + \nabla\cdot\left(\mu^{n+1}\mathbf{D}(\mathbf{u}^*)\right)\right] \tag{A.1}$$

with

$$\mathbf{H}^k = -\nabla\cdot\left(\mathbf{u}^k\mathbf{u}^k\right) + \frac{1}{\rho^{k+1}}\frac{1}{\mathrm{We}}\mathbf{f}_\sigma^{k+1}$$
$$\mathbf{D}(\mathbf{u}^k) = \left(\nabla\mathbf{u}^k + \left(\nabla\mathbf{u}^k\right)^T\right). \tag{A.2}$$

The last viscous term in Eq. (A.1) with variable coefficients is split into a constant part and a variable part, and the constant part is treated implicitly and the variable part explicitly [13], i.e.,

$$\frac{1}{\rho^{n+1}}\nabla\cdot\left(\mu^{n+1}\mathbf{D}(\mathbf{u}^*)\right) \to \nu_0\nabla^2\mathbf{u}^* + \frac{1}{\rho^{n+1}}\nabla\cdot\left(\mu^{n+1}\mathbf{D}(\mathbf{u}^n)\right) - \nu_0\nabla^2\mathbf{u}^n \tag{A.3}$$

where $\nu_0 = \frac{1}{2}(\frac{\mu_1}{\rho_1} + \frac{\mu_2}{\rho_2})$. Unlike the pressure field, the velocity field is continuous across the interface, thus, we do not need to linearly extrapolate $\mathbf{u}^n$ in time as we did for the pressure in (14) (see Section 3.3). The substitution in (A.3) is consistent because we recover the constant-property Navier–Stokes equations if we set $\mu_1 = \rho_1 = \mu_2 = \rho_2 = 1$. Substituting Eq. (A.3) into Eq. (A.1) gives:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = \frac{1}{2}\left(3\mathbf{H}^n - \mathbf{H}^{n-1}\right) + \frac{1}{\mathrm{Re}}\frac{1}{\rho^{n+1}}\left[\nabla\cdot\left(\mu^{n+1}\mathbf{D}(\mathbf{u}^n)\right)\right] + \frac{1}{2}\frac{1}{\mathrm{Re}}\left(\nu_0\nabla^2\mathbf{u}^* - \nu_0\nabla^2\mathbf{u}^n\right). \tag{A.4}$$

Multiplying Eq. (A.4) by $(-2\,\mathrm{Re}/\nu_0)$ and rearranging terms gives the Helmholtz equation for $\mathbf{u}^*$:

$$\nabla^2\mathbf{u}^* - \frac{2\,\mathrm{Re}}{\nu_0\Delta t}\mathbf{u}^* = -\frac{2\,\mathrm{Re}}{\nu_0\Delta t}\mathbf{u}^n - \frac{\mathrm{Re}}{\nu_0}\left(3\mathbf{H}^n - \mathbf{H}^{n-1}\right) - \frac{2}{\nu_0}\frac{1}{\rho^{n+1}}\left[\nabla\cdot\left(\mu^{n+1}\mathbf{D}(\mathbf{u}^n)\right)\right] + \nabla^2\mathbf{u}^n. \tag{A.5}$$

Eq. (A.5) can be solved using a fast elliptic solver (e.g., [4,5]). Once $\mathbf{u}^*$ is known, the solution algorithm proceeds as described in Section 2.2 by next computing p^{n+1} from Eq. (16) and then $\mathbf{u}^{n+1}$ from Eq. (17). We have tested the method solving Eq. (A.5) for the capillary wave case and found excellent agreement with Prosperetti's analytical solution up to density and viscosity ratios 10,000.

References

[1] A.J. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comput. 22 (104) (1968) 745–762.
[2] R.W. Hockney, A fast direct solution of Poisson's equation using Fourier analysis, J. Assoc. Comput. Mach. 12 (1) (1965) 95–113.
[3] B.L. Buzbee, G.H. Golub, C.W. Nielson, On direct methods for solving Poisson's equations, SIAM J. Numer. Anal. 7 (4) (1970) 627–656.
[4] P.N. Swarztrauber, The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle, SIAM Rev. 19 (3) (1977) 490–501.
[5] H. Schmidt, U. Schumann, H. Volkert, Three dimensional, direct and vectorized elliptic solvers for various boundary conditions, Rep. 84-15, DFVLR-Mitt, 1984.
[6] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, J. Comput. Phys. 100 (1) (1992) 25–37.
[7] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, S. Zaleski, Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows, J. Comput. Phys. 152 (2) (1999) 423–456.
[8] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, J. Comput. Phys. 228 (16) (2009) 5838–5866.
[9] M. Francois, S. Cummins, E. Dendy, D. Kothe, J. Sicilian, M. Williams, A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework, J. Comput. Phys. 213 (1) (2006) 141–173.
[10] G.D. Weymouth, D.K.-P. Yue, Conservative volume-of-fluid method for free-surface simulations on Cartesian-grids, J. Comput. Phys. 229 (8) (2010) 2853–2865.
[11] M. Sussman, E.G. Puckett, A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2) (2000) 301–337.
[12] J.-L. Guermond, A. Salgado, A splitting method for incompressible flows with variable density based on a pressure Poisson equation, J. Comput. Phys. 228 (8) (2009) 2834–2846.
[13] S. Dong, J. Shen, A time-stepping scheme involving constant coefficient matrices for phase-field simulations of two-phase incompressible flows with large density ratios, J. Comput. Phys. 231 (17) (2012) 5788–5804.
[14] D. Gottlieb, S.A. Orszag, Numerical Analysis of Spectral Methods: Theory and Applications, SIAM, 1977.
[15] F. Nicoud, Conservative high-order finite-difference schemes for low-Mach number flows, J. Comput. Phys. 158 (1) (2000) 71–97.
[16] R. Temam, Sur l'approximation de la solution des équations de Navier–Stokes par la méthode des pas fractionnaires (II), Arch. Ration. Mech. Anal. 33 (5) (1969) 377–385.
[17] A. Baraldi, M.S. Dodd, A. Ferrante, A mass-conserving volume-of-fluid method: volume tracking and droplet surface-tension in incompressible isotropic turbulence, Comput. Fluids 96 (2014) 322–337.
[18] J. Brackbill, D. Kothe, C. Zemach, A continuum method for modeling surface tension, J. Comput. Phys. 100 (2) (1992) 335–354.
[19] S. Cummins, M. Francois, D. Kothe, Estimating curvature from volume fractions, Comput. Struct. 83 (6–7) (2005) 425–434.
[20] J. López, C. Zanzi, P. Gómez, R. Zamora, F. Faura, J. Hernández, An improved height function technique for computing interface curvature from volume fractions, Comput. Methods Appl. Mech. Eng. 198 (33) (2009) 2555–2564.
[21] E.G. Puckett, A.S. Almgren, J.B. Bell, D.L. Marcus, W.J. Rider, A high-order projection method for tracking fluid interfaces in variable density incompressible flows, J. Comput. Phys. 130 (2) (1997) 269–282.
[22] V. Le Chenadec, H. Pitsch, A monotonicity preserving conservative sharp interface flow solver for high density ratio two-phase flows, J. Comput. Phys. 249 (2013) 185–203.
[23] R.D. Falgout, U.M. Yang, Hypre: a library of high performance preconditioners, in: Computational Science – ICCS 2002, Springer, 2002, pp. 632–641.

[24] R.D. Falgout, J.E. Jones, U.M. Yang, The design and implementation of hypre, a library of parallel high performance preconditioners, in: Numerical Solution of Partial Differential Equations on Parallel Computers, Springer, 2006, pp. 267–294.
[25] S. Schaffer, A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients, SIAM J. Sci. Comput. 20 (1) (1998) 228–242.
[26] M. Sussman, M. Ohta, A stable and efficient method for treating surface tension in incompressible two-phase flow, SIAM J. Sci. Comput. 31 (4) (2009) 2447–2471.
[27] S. Hysing, A new implicit surface tension implementation for interfacial flows, Int. J. Numer. Methods Fluids 51 (6) (2006) 659–672.
[28] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, Proc. IEEE 93 (2) (2005) 216–231.
[29] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (1965) 2182–2189.
[30] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas–Liquid Multiphase Flows, Cambridge University Press, 2011.
[31] D. Youngs, Time-dependent multi-material flow with large fluid distortion, Numer. Methods Fluid Dyn. 1 (1) (1982) 41–51.
[32] G. Miller, P. Colella, A conservative three-dimensional Eulerian method for coupled solid–fluid shock capturing* 1, J. Comput. Phys. 183 (1) (2002) 26–82.
[33] E. Aulisa, S. Manservisi, R. Scardovelli, S. Zaleski, Interface reconstruction with least-squares fit and split advection in three-dimensional Cartesian geometry, J. Comput. Phys. 225 (2) (2007) 2301–2319.
[34] R. Scardovelli, E. Aulisa, S. Manservisi, V. Marra, A marker-VOF algorithm for incompressible flows with interfaces, in: Advances in Free Surface and Interface Fluid Dynamics, vol. 1, ASME Conf. Proc., Joint U.S.-European Fluids Engineering Division Conference, 2002, pp. 905–910.
[35] A.H. Baker, R.D. Falgout, T.V. Kolev, U.M. Yang, Scaling hypre's multigrid solvers to 100,000 cores, in: High-Performance Scientific Computing, Springer, 2012, pp. 261–279.
[36] J. Schmidt, J. Thornock, J. Sutherland, M. Berzins, Large scale parallel solution of incompressible flow problems using uintah and hypre, Technical report UUSCI-2012-002, Scientific Computing and Imaging Institute, 2012.
[37] H. Schmidt, U. Schumann, H. Volkert, Three dimensional, direct and vectorized elliptic solvers for various boundary conditions, Rep. 84-15, DFVLR-Mitt, 1984.
[38] G.I. Taylor, On the decay of vortices in a viscous fluid, Philos. Mag. XLVI (1923) 671–674.
[39] D. Brown, R. Cortez, M. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 168 (2) (2001) 464–499.
[40] L. Timmermans, P. Minev, F. Van De Vosse, An approximate projection scheme for incompressible flow using spectral elements, Int. J. Numer. Methods Fluids 22 (7) (1996) 673–688.
[41] A. Prosperetti, Motion of two superposed viscous fluids, Phys. Fluids 24 (1981) 1217.
[42] K.V. Beard, Terminal velocity and shape of cloud and precipitation drops aloft, J. Atmos. Sci. 33 (5) (1976) 851–864.
[43] R. Clift, J.R. Grace, M.E. Weber, Bubbles, Drops and Particles, Dover, 2005.
[44] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (1985) 308–323.