Fully-coupled parallel solver for the simulation of two-phase incompressible flows

Simon El Ouafa, Stéphane Vincent, Vincent Le Chenadec, Benoît Trouette

Fully-coupled parallel solver for the simulation of two-phase incompressible flows

Simon El Ouafa*, Stéphane Vincent, Vincent Le Chenadec, Benoît Trouette

[a]*Laboratoire MSME, CNRS UMR 8208, Université Gustave Eiffel, 5 Boulevard Descartes Marne-la-Vallée, 77454, France*

Abstract

In the framework of the in-house code Fugu, a fully-coupled solver is developed for massively parallel simulations of three-dimensional incompressible multiphase flows. The linearized momentum and continuity equations arising from the implicit solution of the fluid velocities and pressure are solved simultaneously. The method uses a BiCGStab(2) [1] iterative solver with an original preconditioner for the velocity block and an approximation of the inverse of the Schur complement. This is achieved by using PFMG or SMG from HYPRE and an efficient sparse matrix-vector multiplication using the CSR storage format. The construction and the tracking of the interface separating the different involved phases is based on a conservative VOF method. Test cases, such as a spherical bubble rising in quiescent liquid and the free fall of a dense sphere, are performed to validate the models, especially in the presence of strong density and viscosity ratios between fluids. Other cases, such as the phase inversion, demonstrate the ability of the new fully-coupled solver to solve two-phase problems with more than 1 billion degrees of freedom with excellent scalability.

Keywords: Fully coupled solver, linear system preconditioning, two-phase flows, large density and viscosity ratios, HPC

*Corresponding author

Email addresses: simon.elouafa@univ-eiffel.fr (Simon El Ouafa), stephane.vincent@univ-eiffel.fr (Stéphane Vincent), vincent.le-chenadec@univ-eiffel.fr (Vincent Le Chenadec), benoit.trouette@univ-eiffel.fr (Benoît Trouette)

[1]MSME, Univ Gustave Eiffel, CNRS UMR 8208, Univ Paris Est Creteil, F-77454 Marne-la-Vallée, France

1. Introduction

Multiphase flows with separated phases are ubiquitous in nature and in industrial applications. Energy [2, 3], environment [4], chemical engineering [5, 6], hydrology, material processes [7, 8], petroleum engineering [9], vehicle design [10] or civil engineering, are examples of applications where the conditions and two-phase flow regimes encountered spread wide and far in terms of density and viscosity ratios, inertial, viscous, gravity or capillary effects. Because of the complexity of the underlying physics, the understanding of these flows is of high interest in order to control or prevent malfunctions such as for boiling in nuclear power plants, overflow in dams or cavitation in turbo-machinery, to mention just a few.

Experimental investigations, although widely developed in many research works, sometimes suffers from certain limitations or implementation difficulties. It is even more difficult to apply to some two-phase problems. Indeed, in addition to the complexity of these flows, i.e. the topology of the interfaces (separated or dispersed phases), the nature of the interactions (friction, capillary effects, etc ...), the flow regimes (in terms of Reynolds or Weber numbers, for example) or the presence of different scales in space and time (from the separated phase to the scale of small interfaces), it is sometimes impossible to simultaneously reproduce the real conditions at the laboratory scale and to measure quantities of interest such as droplet size distributions, interfacial area density or turbulence intensity. Consequently, numerical simulation has become an indispensable tool to better understand and therefore control the bearing of complex two-phase flows. Over the past few decades, this approach has become a central tool for investigation, adopted by the fluid mechanics community to complement experiments.

However, numerical simulation faces several challenges that need to be overcome, not only in terms of the sensitivity of the numerical results to the choice of model and mesh, but also in terms of the robustness and accuracy of the solver and solution algorithms, allowing the simulation of complex two-phase problems with high density or viscosity ratios without causing numerical oscillations or divergence and with acceptable time-to-solution. The simulation of these flows at a small scale requires taking into account the turbulence far from the interfaces and the coupling between the fluids present at the interfaces on the other hand where the vorticity is generated as a result of the high shear caused by large density or viscosity ratios. The mathematical structure of the Navier-Stokes incompressible equations considered in the

present work (algebraic non-linear differential system) make their solution challenging, especially in the presence of high density and viscosity ratios which deteriorate the conditioning of the system and whose spatial distributions vary in space and time. Thus, it is necessary to make numerical solvers and related algorithms run on massively parallel machines, and to optimise them in order to take advantage of intensive computing distributed over thousands of processors.

Two aspects are responsible for the fidelity of the solution: the first is related to the choice of the model and the second to the efficiency of the solver and resolution algorithms on HPC architectures. Concerning the former, there exists two approaches to numerical modeling of two-phase flows. The so-called two-fluid models describe multiphase mixing at different levels [11, 12]. In their most complete version, these models take the form of two Navier-Stokes systems, coupled by additional source terms that represent the exchanges (mass, momentum and energy) between the two phases. An important aspect is that both sets of equations hold over the entirety of the domain, which challenges the model in the void regions (where only one phase is present). In contrast, the one-fluid approach consists in modelling the two-phase flow as a single fluid, possessing sharply varying thermodynamic properties as well as source terms representing the capillary effects [13]. This results in a single set of Navier-Stokes equations, supplemented by a scalar advection equation to characterize the interface position. The one-fluid model is devoted to fully resolved interfaces without any further closure requirements whereas two-fluid models are generally adapted to two-phase dispersed flows laying on a scale separation assumption. In the present work, the one-fluid model will be considered.

The latter iss related to the treatment of velocity-pressure coupling which, in the incompressible limit, has been the subject of numerous documented studies and is relatively well understood as far as single-phase flows are concerned. On the contrary, in the context of two-phase flows with high density and viscosity ratios, numerical difficulties appear, linked to the presence of large interface deformations and associated material fluid property jumps. They lead to ill-conditioned linear systems. At the discrete level, the problem takes the form of a saddle point system which is costly to solve. Upon discretization and linearization, it leads to a system of the form

$$\begin{bmatrix} F_{\mathbf{u}} & B_p^T \\ B_u & 0 \end{bmatrix} \begin{pmatrix} \mathbf{u}^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} \tag{1}$$

where $F_{\mathbf{u}}$ includes the convective and viscous transport operators, B_p^T the pressure gradient operator, B_u the velocity divergence and $\mathbf{u}^{n+1}$ and p^{n+1} denote the velocity and the pressure variables. There are two approaches to solve this large system of

equations: either the use of specialized solvers (e.g. iterative with preconditioning), in which case the method is referred to as exact, monolithic or coupled method, or the approximation by a simpler problem whose solution is more affordable (Chorin-Temam type projection [14] and its variants [15, 16], methods also referred to as approximate or segregated).

Segregated methods do not solve for all of the unknowns at the same time. Instead, they approximate the original system via operator splitting, resulting in two decoupled equations: one to update the velocity field and the other the pressure field. Instead, coupled methods solve both fields (velocity and pressure) simultaneously, thus preserving the consistency of the discretized system with the continuous equations. In coupled methods, the original saddle point system is inverted, thus keeping the velocity-pressure coupling at each time step. This saddle point system can be solved either directly by a fully-coupled solver [17, 18, 19, 20, 21] or with an augmented Lagrangian (AL) method [22, 23, 10]. The interest of these methods is to allow a very accurate and robust resolution, without splitting errors.

The present work develops a coupled method, in particular block preconditioners dedicated to the resolution of the saddle point systems from the discretization of the one-fluid model. They rely on recent advances in the field of iterative solvers and preconditioners, and their implementation with open source parallel libraries (HYPRE [24], MUMPS [25, 26] and LIS [27]). The objective is to include the massively parallel computing component in the 3D numerical modeling of a physical problem in order to reduce the execution time of computationally expensive applications.

The rest of the manuscript describes the models, the numerical algorithms and their performances. Different test cases are presented in order to verify the accuracy and robustness of the proposed fully-coupled solver: a three-dimensional rising bubble, the free fall of a dense sphere and a phase inversion between two incompressible liquids.

2. Models and numerical methods

All the developments described and validated in the present work are part of the Fugu code developed by the Heat and Mass Transfer team of MSME laboratory at Gustave Eiffel University.

2.1. Governing equations of two-phase flows

In this work, the dynamics of the two immiscible phases is described by the one-fluid model (OFM) [28]. The incompressible Navier-Stokes equations are solved for an equivalent single fluid with variable material properties, with an additional source term in the momentum equation that introduces the capillary force. An additional transport equation describes the evolution of the phase function C, an indicator field transported by the incompressible fluid velocity $\mathbf{u}$. One defines $C(t, \mathbf{x}) = 1$ ($C(t, \mathbf{x}) = 0$) if $\mathbf{x}$ belongs to the domain occupied by fluid 1 (fluid 2) at time t. The OFM is flexible in the sense that it can be solved on a fixed mesh, and can be used to simulate various geometries or interface conditions. With mixing rules for the effective density ρ and viscosity μ, the governing equations read

$$\begin{cases} \rho\dfrac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \bar{\bar{\mathbf{B}}} \cdot (\mathbf{f}(\mathbf{u}) - \mathbf{u}_\infty) = -\nabla p + \nabla \cdot \bar{\bar{\mathbf{T}}} + \rho \mathbf{g} + \mathbf{F}_s & \text{(2a)} \\ \nabla \cdot \mathbf{u} = 0 & \text{(2b)} \\ \dfrac{\partial C}{\partial t} + \mathbf{u} \nabla C = 0 & \text{(2c)} \\ \rho(C) = \rho_1 C + (1 - C)\rho_2 & \text{(2d)} \\ \mu(C) = \mu_1 C + (1 - C)\mu_2 & \text{(2e)} \end{cases}$$

where $\mathbf{u} = (u, v, w)^T$ is the fluid velocity, p the pressure field, t the time, ρ and μ are the density and viscosity of the equivalent fluid given by an arithmetic average, with ρ_1, ρ_2, μ_1 and μ_2 the densities and the viscosities of fluids 1 and 2, respectively. In addition, the viscous stress tensor for a Newtonian fluid is $\bar{\bar{\mathbf{T}}} = \left[\mu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right)\right]$, whereas $\mathbf{F}_s = \sigma \kappa \mathbf{n} \delta_i$ is the capillary term acting on the interface, modelled in this study by the Continuum Surface tension Force model (CSF) [29]. The normal vector to the interface is $\mathbf{n}$, σ is the coefficient of surface tension, δ_i is the surface Dirac function and κ is the local interfacial curvature. Finally, $\bar{\bar{\mathbf{B}}} \cdot (\mathbf{f}(\mathbf{u}) - \mathbf{u}_\infty)$ is a Darcy-like penalty term used for specifying domain boundary conditions or immersed boundaries [30]. For example, along a boundary Γ, the tensor $\bar{\bar{\mathbf{B}}}$ has diagonal components that tend to infinity while they are identically zero inside the fluid domain Ω. The user-specified function $\mathbf{f}(\mathbf{u})$ can vary over space and specify problem-dependent boundary conditions (Dirichlet, Neumann, Robin).

2.2. Discretization of mass and momentum equations

Without loss of generality, a conservative VOF approach from [31] is used to solve Eq. (2c) beforehand rewritten in its conservative form. As the interface tracking

technique is not the main topic of this paper, the focus is on the discretization of mass and momentum equations and the resulting linear system. The system (2) is then split. Phase function evolution (Eq. (2c)) is first solved. From the updated volume fraction field C, mixing rules (Eqs. (2d) and (2e)) are applied in order to obtain the fluid effective density ρ and viscosity μ. Finally, a sub system made up of the mass and the momentum conservation equations (Eqs. (2a)-(2b)) remains to be solved.

2.2.1. Time integration

For unsteady applications, the time step Δt is an important numerical parameter for the stability and the quality of the numerical solution. The solution is obtained over different discrete times, given by $t^n = n\Delta t$, with n the iteration number. For sake of simplicity, the time integration is performed in the following with a first order backward Euler. Note that the use of high order time discretization schemes, such as the Gear scheme, does not introduce additional difficulty. In such case, the convective terms are linearized by extrapolating the velocity at time $(n+1)$. With the first order Euler scheme, the resulting semi-discrete system is written as:

$$\begin{cases} \dfrac{\rho}{\Delta t}\left(\mathbf{u}^{n+1}-\mathbf{u}^{n}\right)+\nabla\cdot\left(\rho\mathbf{u}^{n}\otimes\mathbf{u}^{n+1}\right)+\bar{\bar{\mathbf{B}}}\cdot\left(\mathbf{f}(\mathbf{u}^{n+1})-\mathbf{u}_{\infty}\right)=-\nabla p^{n+1} \\ \qquad\qquad +\nabla\cdot\bar{\bar{\mathbf{T}}}^{n+1}+\rho\mathbf{g}+\mathbf{F}_{s} \end{cases} \tag{3a}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0 \tag{3b}$$

where $\bar{\bar{\mathbf{T}}}^{n+1}=\left[\mu\left(\nabla\mathbf{u}^{n+1}+(\nabla\mathbf{u}^{n+1})^{T}\right)\right]$.

2.2.2. Spatial discretization

The OFM formulation (Eqs. (2)) is solved on Cartesian uniform staggered meshes, as originally proposed by [32]. The discrete velocity and pressure variables are then arranged in a staggered way, each one having its appropriate control volume and connectivities with other coupled variables as illustrated in Fig 1. Staggered meshes guarantee the consistency of the differential operators such as the divergence and the gradient and it also avoids spurious oscillations on the pressure field. Furthermore, the spatial operators acting in the one-fluid formulation are discretized using a finite volume approximation [33, 34]. This method consists of integrating conservation equations on each control volume $\Omega_{(i,j,k)}$ ($\Omega_{\mathbf{u}(i,j,k)}$ for $\mathbf{u}$ and $\Omega_{p(i,j,k)}$ for p) that are delimited by the surface $\Sigma_{(i,j,k)}$ ($\Sigma_{\mathbf{u}(i,j,k)}$ for $\mathbf{u}$ and $\Sigma_{p(i,j,k)}$ for p). Here, i, j and k are control volume indices in x-, y- and z-directions, respectively. Using the divergence

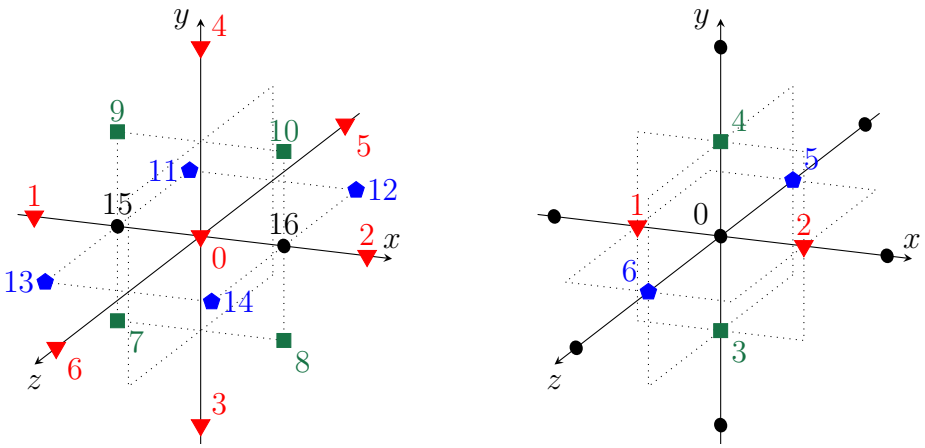


Figure 1: Staggered u-velocity (left) and pressure (right) unknowns and their respective local indices. Velocity control volume is centered on u-velocity ($\triangledown$ symbols). Local index 0 stands for global indices i, j and k. Directional full (± 1) or half shifts ($\pm 1/2$) allow to reach neighbours used in the 3 points discretization stencil. The horizontal velocity is linked through divergence and diffusive terms to v- and w- components ($\square$ and $\pentagon$) and also to the pressure ($\circ$). Similar notations and connectivities are adopted for the treatment of v- and w-components. All the three velocity components then involve 17 coupled variables at the discrete level.

and gradient theorems, the spatial integration of sub-system (3) leads to

$$\begin{cases} \left[M_u^{(\rho)} + N_u^{(\rho)} + \ L_u^{(\mu)}\right] \mathbf{u}^{n+1} + B_p^T p^{n+1} = \mathbf{f} & \text{(4a)} \\ B_u \mathbf{u}^{n+1} = 0 & \text{(4b)} \end{cases}$$

where

$$
\begin{cases}
B_u \mathbf{u}^{n+1} = \displaystyle\int_{\Sigma_{p(i,j,k)}} \mathbf{u}^{n+1} \cdot \mathbf{n}\, dS & \text{(5a)} \\
M_u^{(\rho)} \mathbf{u}^{n+1} = \displaystyle\int_{\Omega_{\mathbf{u}(i,j,k)}} \left[\frac{\rho \mathbf{u}^{n+1}}{\Delta t} + \bar{\bar{\mathbf{B}}} \cdot \mathbf{f}(\mathbf{u}^{n+1}) \right] dV & \text{(5b)} \\
N_u^{(\rho)} \mathbf{u}^{n+1} = \displaystyle\int_{\Sigma_{\mathbf{u}(i,j,k)}} (\rho \mathbf{u}^{n+1} \otimes \mathbf{u}^{n}) \cdot \mathbf{n}\, dS & \text{(5c)} \\
L_u^{(\mu)} \mathbf{u}^{n+1} = - \displaystyle\int_{\Sigma_{\mathbf{u}(i,j,k)}} \bar{\bar{\mathbf{T}}}^{n+1} \cdot \mathbf{n}\, dS & \text{(5d)} \\
B_p^T p^{n+1} = \displaystyle\int_{\Sigma_{\mathbf{u}(i,j,k)}} p^{n+1} \mathbf{n}\, dS & \text{(5e)} \\
\mathbf{f} = \displaystyle\int_{\Omega_{\mathbf{u}(i,j,k)}} \left[\frac{\rho}{\Delta t} \mathbf{u}^{n} + \bar{\bar{\mathbf{B}}} \cdot \mathbf{u}_\infty + \rho \mathbf{g} + \mathbf{F}_s \right] dV & \text{(5f)}
\end{cases}
$$

with $\mathbf{n}$ the outward pointing normal with respect to the control volume $\Sigma_{i,j,k}$. To characterize the fluxes across the surfaces $\Sigma_{\mathbf{u}(i,j,k)}$ or $\Sigma_{p(i,j,k)}$ of the control volume $\Omega_{\mathbf{u}(i,j,k)}$ or $\Omega_{p(i,j,k)}$, the mesh spacings Δx, Δy and Δz are defined, respectively in the x-, y- and z-directions. For the sake of simplicity, only constant mesh spacing are presented here. Moreover, the integrals are explicitly written for the velocity component u in the x-direction (the extension to the other components is carried out in the same way) and for each control volume. Details are given in appendix Appendix A with particular focus on the treatment of the viscous stress tensor. The penalty terms used for handling boundary conditions are discussed in appendix Appendix B.

2.2.3. Solution of the linear system

Finite volumes and penalty methods on a staggered mesh, together with an implicit temporal discretization of Eqs. (2a) and (2b), result in large and non-symmetric linear saddle point systems. The unknowns of the problem, the discrete velocity $\mathbf{u}$ and pressure p fields, are coupled by the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$. As presented in the introduction, this saddle point system can be reformulated as follows:

$$
\begin{bmatrix} F_{\mathbf{u}} & B_p^T \\ B_u & 0 \end{bmatrix} \begin{pmatrix} \mathbf{u}^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} \tag{6}
$$

with $F_{\mathbf{u}} = M_u^{(\rho)} + N_u^{(\rho)} + L_u^{(\mu)}$. The reduced system can be rewritten as $A\mathbf{x} = \mathbf{b}$ with A the matrix associated with the discretization coefficients in time and space,

$\mathbf{b}$ the right-hand side associated with each component of the solution vector and $\mathbf{x} = (\mathbf{u}^{n+1}, p^{n+1})^T$ the updated flow variables.

In order to compute $\mathbf{x}$, it is necessary to choose a method for solving this linear system Eq. (6). Many techniques exist to do so, such as direct methods that include Gaussian elimination, factorisation techniques (LU, QR, Cholesky, ...) and multi-frontal methods. These techniques are very robust with respect to various problems but typically do not scale well with problem size and becomes prohibitively expensive for three-dimensional configurations. Alternative approaches are iterative methods, such as fixed-point (Jacobi, Gauss-Seidel, relaxation, ...), Krylov or multigrid methods. They are however usually effective for specific kind of matrices (symmetric, dense or sparse matrices for example). It is therefore necessary to estimate the structure of the linear system. According to the discretization schemes proposed in the previous paragraph, with the evaluations of fluxes on faces of each control volumes, each equation of the linear system couples only a few components of the solution. With the choice of 3 point wide stencil per direction, if u-component equation is for example considered, 7 degrees of freedom are coupled by the diffusive or inertial terms, 8 more are used for the coupling between u and v as well as between u and w and 2 more are needed for the pressure gradient (see Fig. 1). On the whole, 17 degree of freedom are coupled for each control volume. In addition, 6 additional degree of freedom are used for the velocity divergence. The same reads for equations v- and w-velocity components. With the example of a mesh composed of $N_x \times N_y \times N_z$ control volumes, and $N = N_x = N_z = N_z$, the solution vector has $4N^3 + 3N^2$ unknowns, the additional $3N^2$ coming from the choice of staggered meshes. The matrix A will be a square sparse matrix of $(4N^3 + 3N^2)^2$ coefficients. Thus, only 17 non-zero diagonals (see Fig. 2) acting on each velocity component and 6 non-zero diagonals coming from the discretization of the velocity divergence will be involved in the matrix. In order to save memory space, only the non-zero coefficients arising from the discretization of the incompressible Navier-Stokes equations (Eqs. (2a)-(2b)) are stored using the Compressed Storage Raw (CSR) format.

It is thus clear that the matrix generated by the discretization of the Navier-Stokes equations will be very sparse. There is therefore no interest in using direct methods. A survey of the litterature reveals that only two recent studies were documented to have tackled two-phase problems with large density and viscosity ratios by a fully-coupled approach. In the first one, Bootland et al [17] discretize the conservation equations with 2D finite elements using an implicit scheme for the inertial term and apply the sparse direct solver SuperLU [35] to the velocity block. However, an approximation of the inverse of the pressure Schur complement matrix is performed using the PCD operator technique (Pressure Convection Diffusion) [17]. In

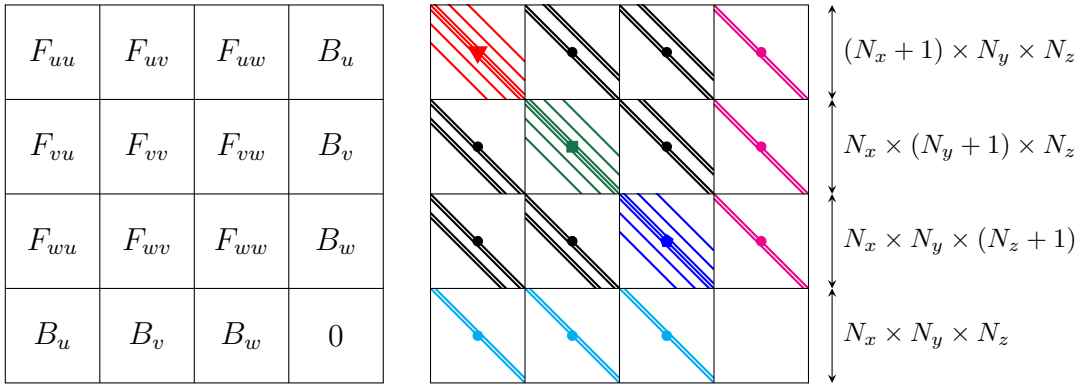


Figure 2: Structure of the fully-coupled matrix, 7 diagonal form the 3 directional points stencil (in red for u-, green for v- and blue for w-component, respectively). Black lines for velocity coupling, magenta for pressure gradient and cyan for divergence.

the second study, Nangia et al [18] use 3D finite volumes with an explicit scheme for the inertial term and a flexible GMRES Krylov solver preconditioned by a variable-coefficient projection method solver. Our proposed strategy to solve the sparse linear system resulting from the discretization of the motion equations is to employ a Krylov solver, here the BiCGStab(2) [1] solver, on the entire system. This involves building an efficient preconditioner $\mathcal{P} \approx A$ to stabilize and accelerate the iterative solver convergence. Indeed, for difficult problems corresponding to ill-conditioned matrices, it is essential to combine the BiCGStab(2) algorithm with a suitable preconditioner to avoid possible numerical instabilities and to speed-up convergence. Then, instead of solving $A\mathbf{x} = \mathbf{b}$, the left-preconditioned system $\mathcal{P}^{-1}A\mathbf{x} = \mathcal{P}^{-1}\mathbf{b}$ is preferred, for which $\text{cond}(\mathcal{P}^{-1}A) < \text{cond}(A)$ is expected. Therefore, in the BiCGStab(2) algorithm, every matrix-vector product $\mathbf{z} = A\mathbf{x}$ is followed by the computation of $\mathcal{P}^{-1}\mathbf{z}$. The same transformation must be applied to the right-hand side $\mathbf{b} \leftarrow \mathcal{P}^{-1}\mathbf{b}$. The pseudo code given in Alg. 3 in appendix Appendix C describes the overall algorithm to solve the system up to a chosen threshold ε starting from an initial guess $\mathbf{x}^{(0)}$. The following section focuses on the construction of the preconditioner $\mathcal{P}$.

2.3. Block preconditioning techniques

To build the preconditioning matrix $\mathcal{P}$ required in subsection 2.2.3, a large variety of preconditioners exist. Among them, ILU type preconditioning, also called incomplete Gauss factorisation, allows to reach low residuals when several millions of unknowns are involved. However, these preconditioners are not easily scalable as they induce a global dependence on the unknowns, and therefore require numerous

and repeated exchanges in the domain decomposition framework. It is possible to use ILU techniques and their derived versions in a block implementation, i.e. the preconditioning is performed in a decomposed fashion where the linear operators, decomposed along rows, neglect the contributions from the out-of-core degrees-of-freedoms. Nevertheless, in most of the representative cases, the classical preconditioners of the literature such as ILU fail to provide an efficient solution of the fully-coupled Navier-Stokes equations. This was highlighted in a number of documented studies [36], in the context of the augmented Lagrangian method for example with applications to both unsteady laminar and turbulent two-phase flows, but it was limited to 3D cases around 100 million cells. The strategy implemented in this work is to build the preconditioner of the problem by taking a LU block decomposition of the original matrix A and to introduce a second spatial discretization of the momentum equation in the discrete pressure space. Applied on the linear system (6), the LU block factorization reads

$$\begin{bmatrix} F_{\mathbf{u}} & B_p^T \\ B_u & 0 \end{bmatrix} = \begin{bmatrix} I_u & 0 \\ B_u F_{\mathbf{u}}^{-1} & I_p \end{bmatrix} \begin{bmatrix} F_{\mathbf{u}} & B_p^T \\ 0 & S_p \end{bmatrix} \tag{7}$$

where $S_p = -B_u F_{\mathbf{u}}^{-1} B_p^T$ is the Schur complement of the pressure block. Thus, if the upper-triangular block U is considered as preconditioner, the preconditioning operator $\mathcal{P}$ directly reads:

$$\mathcal{P} = \begin{bmatrix} F_{\mathbf{u}} & B_p^T \\ 0 & S_p \end{bmatrix} \tag{8}$$

According to this choice, the iterative solver would need exactly two iterations to compute the solution [37]. However, it is not feasible to use the exact Schur complement S_p and the velocity block $F_{\mathbf{u}}$ as a part of the preconditioning operator, as they require the knowledge of their inverses, respectively S_p^{-1} and $F_{\mathbf{u}}^{-1}$, two dense matrices that are time-dependent and are indeed more expensive than solving the saddle point (6) by direct methods. Due to this difficulty, an approximation of the action of the inverse of the velocity block $F_{\mathbf{u}}$ and the Schur complement S_p on any vector has to be considered. The way to achieve this is described in the next sections.

2.3.1. The velocity block

It is important to note here that an appropriate resolution of the Navier-Stokes equations entails the use of an efficient preconditioning for the velocity blocks. This is especially true when an augmented Lagrangian or a fully-coupled solver is used. These velocity blocks are sparse, large and ill-conditioned in the presence of large viscosity ratios. They therefore require resolution by an efficient and robust preconditioning, which must address the challenge raised by the coupling between velocity

components due to the large viscosity ratios typically encountered in multiphase flows applications. In this context, this section describes some techniques for the preconditioning of the coupling between velocity components, essentially based on an approximate algebraic preconditioner for the Schur complement.

Let us consider the sparse and non-symmetric matrix $F_{\mathbf{u}}$, arising from the discretization of the linearized momentum equations (Eqs. (5)). This matrix can be split into a 3×3 block matrix, rewritten in the following compact form

$$F_{\mathbf{u}} = \begin{bmatrix} A_{uv} & A_{uvw} \\ A_{wuv} & F_{ww} \end{bmatrix} \tag{9}$$

with

$$A_{uv} = \begin{bmatrix} F_{uu} & F_{uv} \\ F_{vu} & F_{vv} \end{bmatrix}, \quad A_{uvw} = \begin{bmatrix} F_{uw} \\ F_{vw} \end{bmatrix} \quad \text{and} \quad A_{wvu} = \begin{bmatrix} F_{wu} & F_{wv} \end{bmatrix} \tag{10}$$

The preconditioning studied in our case takes advantage of the block LU decomposition of Eq. (9), by carefully considering the specific block upper triangular preconditioner given by:

$$\tilde{F}_{\mathbf{u}} = \begin{bmatrix} A_{uv} & A_{uvw} \\ 0 & F_{ww} - A_{wvu} A_{uv}^{-1} A_{uvw} \end{bmatrix} \tag{11}$$

where $A_{wvu} A_{uv}^{-1} A_{uvw}$ is the exact Schur complement of $F_{\mathbf{u}}$, which can be rewritten as below by introducing a new matrix G:

$$\begin{aligned} F_{ww} - A_{wvu} A_{uv}^{-1} A_{uvw} &= F_{ww} \left(I - F_{ww}^{-1} A_{wvu} A_{uv}^{-1} A_{uvw} \right) \\ &= F_{ww} \left(I - G \right) \end{aligned} \tag{12}$$

As a result, the direct action of the inverse of $\tilde{F}_{\mathbf{u}}$ on a standard vector $\mathbf{z_u} = (z_u, z_v, z_w)^T$ should be achieved by a backward elementary substitution. This can be achieved by solving two linear systems and performing a single matrix-vector product. In the corresponding algorithm 1, approximations of the inverses of $F_{ww}\left(I - G\right)$ and A_{uv} are required.

Algorithm 1 Block triangular preconditioner for the velocity block. Approximation of the inverse of the velocity block $\tilde{F}_u$ on a vector $\mathbf{z_u} = (z_u, z_v, z_w)^T$, the solution vector is $\mathbf{r_u} = (r_u, r_v, r_w)^T$, with $\mathbf{r_u}$ =$\mathbf{b_u}$, $\mathbf{q_u}$, $\mathbf{v_u}$, $\mathbf{s_u}$, $\mathbf{w_u}$ and $\mathbf{t}$ are a sequence of vectors generated at each BiCGStab(2) (for further details, the reader is referred to algorithm 3 in appendix Appendix C).

1: $F_{ww}\left(I - G\right)\mathbf{r}_w = \mathbf{z}_w$ ▷ Solve

2: Update $\mathbf{r}_{uv} \longleftarrow \mathbf{r}_{uv} - A_{uvw}\mathbf{z}_w$ with $\mathbf{r}_{uv} = (r_u, r_v)^T$ ▷ Update

3: $A_{uv}\mathbf{r}_{uv} = \mathbf{z}_{uv}$ with $\mathbf{z}_{uv} = (z_u, z_v)^T$ ▷ Solve

---

In terms of application of the inverse of the block $F_{ww}(I-G)$ to the vector $\mathbf{z}$, it is assumed that the matrices $(I-G)$ and F_{ww} are invertible, and that the inverse $(I-G)^{-1}$ can be developed as an infinite series of expansions of the different terms that are present in the operator $(I-G)$. Using the following Neumann series expansion [38] and the definition of G (Eq. (12)), it follows that

$$\begin{aligned}(I-G)^{-1} &= \sum_{k=0}^{\infty} G^k \\ &= \sum_{k=0}^{\infty} \left(F_{ww}^{-1} A_{wvu} A_{uv}^{-1} A_{uvw}\right)^k\end{aligned} \tag{13}$$

Thus, the inverse of the block $F_{ww}(I-G)$ reads:

$$[F_{ww}(I-G)]^{-1} = \left[\sum_{k=0}^{\infty} (F_{ww}^{-1} A_{wvu} A_{uv}^{-1} A_{uvw})^k\right] F_{ww}^{-1}. \tag{14}$$

When a single term (order 0) of the infinite series is retained, $\tilde{F}_u$ from Eq. (11) becomes the block Gauss-Seidel preconditioner. In the case where $A_{uvw} = 0$ is assumed , $\tilde{F}_{\mathbf{u}}$ is a block Jacobi preconditioner.

In the problem at hand, the matrix from Eq. (14) is obtained by truncating the expansion at order 0, which leads the following:

$$[F_{ww}(I-G)]^{-1} \simeq F_{ww}^{-1} \tag{15}$$

When two terms are retained (order 1), a block triangular preconditioner is built:

$$[F_{ww}(I-G)]^{-1} \simeq \left(I + F_{ww}^{-1} A_{wvu} A_{uv}^{-1} A_{uvw}\right) F_{ww}^{-1} \tag{16}$$

Regardless of the truncation order, the inverse of the block A_{uv} is not known explicitly in Eq. (11). As a consequence, an approximation of this block has to be found. This is done using the same reasoning as above from Eq. (11) but considering the A_{uv} block defined in Eq. (10) instead of $F_{\mathbf{u}}$. To order 0, this reads

$$\tilde{A}_{uv}^{(0)} = \begin{bmatrix} F_{uu} & F_{uv} \\ 0 & F_{vv} \end{bmatrix} \tag{17}$$

and to order 1

$$\tilde{A}_{uv}^{(1)} = \begin{bmatrix} F_{uu} & F_{uv} \\ 0 & F_{vv} - F_{vu} F_{uu}^{-1} F_{uv} \end{bmatrix} \tag{18}$$

Eventually, the form of our velocity preconditioner at orders 0 and 1 are:

$$\tilde{F}_{\mathbf{u}}^{(0)} \approx \begin{bmatrix} F_{uu} & F_{uv} & F_{uw} \\ 0 & F_{vv} & F_{vw} \\ 0 & 0 & F_{ww} \end{bmatrix}, \quad \tilde{F}_{\mathbf{u}}^{(1)} \approx \begin{bmatrix} F_{uu} & F_{uv} & F_{uw} \\ 0 & F_{vv} - F_{vu}F_{uu}^{-1}F_{uv} & F_{vw} \\ 0 & 0 & F_{ww} - A_{wvu}A_{uv}^{-1}A_{uvw} \end{bmatrix} \tag{19}$$

The Gauss-Seidel block was found to perform slightly better than the Jacobi block and triangular block preconditioners when compared in 2D simulations of a rising bubble at different CFL numbers. It was therefore preferred. Finally, no improvement in time-to-solution were obtained when increasing the order of the approximation.

2.3.2. The Schur complement

In the literature, a number of preconditioning techniques for the Schur complements S_p have been proposed. In the context of Stokes flow, [39] proposed an approach for homogeneous flows. Later a more sophisticated preconditioner for heterogeneous flows was developed by [40]. In the context of the incompressible Navier-Stokes equations with constant coefficients, numerous issues werer met in the construction a good approximation of the Schur complement S_p. [41] suggested a new technique referred to as LSC (Least Squares Commutator). More recently, [42] have studied a preconditioner coined PCD (Pressure Convection Diffusion). A few studies have also been dedicated to two-phase flows at large density and viscosity ratios. The projection preconditioner, which has been developed by [43] for variable coefficient problems, and adapted by [18], is one example. In addition, [17] have proposed extensions of the PCD and LSC techniques to two-phase flows. To improve performances, the PCD preconditioning with suitable scalability are retained in this work. This preconditioner is given by $S_p^{-1} = \hat{S}_{PCD}$ with

$$\hat{S}_{PCD}^{-1} = (M_p^{(1/\mu)})^{-1} + (A_p^{(1/\rho)})^{-1}(N_p^{(1)} + \Delta t^{-1} M_p^{(1)})(M_p^{(1)})^{-1} \tag{20}$$

with the following operators defined for the pressure space discretization

$$\begin{cases} A_p^{(1/\rho)}\phi = \displaystyle\int_{\Omega_{p(i,j,k)}} \nabla \cdot \left(\rho^{-1}\nabla\phi\right) dV & \text{(21a)} \\ M_p^{(1/\mu)}\phi = \displaystyle\int_{\Omega_{p(i,j,k)}} \frac{1}{\mu} dV & \text{(21b)} \\ N_p^{(1)}\phi = \displaystyle\int_{\Omega_{p(i,j,k)}} \nabla \cdot (\mathbf{u}\phi)\, dV & \text{(21c)} \\ & \text{(21d)} \end{cases}$$

where $M_p^{(1/\mu)}$ is the diagonal pressure mass matrix scaled by the inverse of the viscosity $1/\mu$, $M_p^{(1)}$ is the diagonal standard pressure mass matrix, $N_p^{(1)}$ represents the convective term in the pressure space, and $A_p^{(1/\rho)}$ is the scaled Laplacian that corresponds to the discretization of the term $\nabla \cdot (\rho^{-1} \nabla \phi)$. In order to approximate the inverse of the Schur complement $\hat{S}_{PCD}^{-1}$, the multigrid solver of HYPRE library [24] is used, for which the action of $A_p^{(1/\rho)}$ is required. This solver was indeed designed specifically for elliptic equations with variable coefficients. The two diagonal mass matrices are then inverted, namely $(M_p^{(1/\mu)})^{-1}$ and $(M_p^{(1)})^{-1}$, by applying a rescaling to suitable vectors followed by a matrix-vector product for the operator $F_p^{(1)} = N_p^{(1)} + \Delta t^{-1} M_p^{(1)}$. Furthermore, the pressure convection-diffusion preconditioner requires the construction of a Laplacian $A_p^{(1/\rho)}$ and a convection-diffusion operator $N_p^{(1)} + \Delta t^{-1} M_p^{(1)}$, together with appropriate choices of boundary conditions (the reader is referred to [44] for more information). In relation to the velocity block, it requires the resolution of three linear systems, involving velocity blocks F_{ww}, F_{vv} and F_{uu}, which are all performed using the aforementioned multigrid solver. These resolutions are completed by two matrix-vector products and two updates of the right-hand side. The overall algorithm is summarized in the algorithm 2 presented in appendix Appendix C.

Algorithm 2 Representation of the application of the block triangular preconditioner. $M_p^{(1/\mu)}$ is the diagonal pressure mass matrix scaled by the inverse of the viscosity $1/\mu$, $M_p^{(1)}$ is the diagonal standard pressure mass matrix, $N_p^{(1)}$ represents the standard convective matrix in the pressure space, $A_p^{(1/\rho)}$ is the scaled Laplacian, which corresponds to the discretization of the term $\nabla \cdot (\rho^{-1} \nabla \phi)$, and F_p is the convection-diffusion-reaction operator for the momentum equation.

SOLVE $\begin{bmatrix} F_{uu} & F_{uv} & F_{uw} & B_{pu}^T \\ 0 & F_{vv} & F_{vw} & B_{pv}^T \\ 0 & 0 & F_{ww} & B_{pw}^T \\ 0 & 0 & 0 & \hat{S}_{PCD} \end{bmatrix} \begin{pmatrix} \mathbf{r}_u \\ \mathbf{r}_v \\ \mathbf{r}_w \\ \mathbf{r}_p \end{pmatrix} = \begin{pmatrix} \mathbf{z}_u \\ \mathbf{z}_v \\ \mathbf{z}_w \\ \mathbf{z}_p \end{pmatrix}$

Approximate the inverse of Schur $\hat{S}_{PCD}$

$\mathbf{r}_{p_1} = (M_p^{(1)})^{-1} \mathbf{z}_p$

$\mathbf{r}_{p_2} = (M_p^{(1/\mu)})^{-1} \mathbf{z}_p$ ▷ Solve

$\mathbf{r}_{p_3} = F_P^{(1)} \mathbf{r}_{p_2}$ ▷ Matrix-vector product for the operator $F_P^{(1)}$

$A_P^{(1/\rho)} \ \mathbf{r}_{p_4} = \mathbf{r}_{p_3}$ ▷ Solve Laplace operator

$\mathbf{r}_p = \mathbf{r}_{p_4} + \mathbf{r}_{p_1}$ ▷ Update $\mathbf{r}_p$

$\mathbf{z}_u = \mathbf{z}_u - B_{pu}^T \mathbf{r}_p$ ▷ Update $\mathbf{z}_u$

$\mathbf{z}_v = \mathbf{z}_v - B_{pv}^T \mathbf{r}_p$ ▷ Update $\mathbf{z}_v$

$\mathbf{z}_w = \mathbf{z}_w - B_{pw}^T \mathbf{r}_p$ ▷ Update $\mathbf{z}_w$

Approximate the velocity block $\hat{F}_u$

$\mathbf{r}_w = F_{ww}^{-1} \mathbf{z}_w$ ▷ Solve

$\hat{\mathbf{r}}_{uv} = F_{uw} \mathbf{r}_w + F_{vw} \mathbf{r}_w$ ▷ Matrix-vector product for the operator F_{uw} and F_{vw}

$\mathbf{z}_{uv} = \mathbf{z}_{uv} - \hat{\mathbf{r}}_{uv}$ ▷ Update

$\mathbf{r}_v = F_{vv}^{-1} \mathbf{z}_v$ ▷ Solve

$\hat{\mathbf{r}}_u = F_{uv} \mathbf{r}_v$ ▷ Matrix-vector product for the operator F_{uv}

$\mathbf{z}_u = \mathbf{z}_u - \hat{\mathbf{r}}_u$ ▷ Update

$\mathbf{r}_u = F_{uu}^{-1} \mathbf{z}_u$ ▷ Solve

3. Fully coupled solver parallelization

The numerical simulation of an unsteady two-phase flow, possibly in turbulent regime, requires very important resources in terms of mesh size, computational time and memory storage. The decomposition domain based parallelization of the different steps, resulting from the equations discretization, is essential to achieve calculations on grids with more than a billion points. Indeed, parallelization has multiple objectives. On the one hand, to obtain reasonable computational times, ideally inversely

proportional to the computational resources. And on the other hand, to be able to deal with large meshes corresponding to realistic simulations in terms of dimensionless numbers and of resolution of small-scale phenomena.

Numerical tests have revealed that about 90% of the total CPU time is dedicated to the linear solver for serial applications. Therefore, the focus for optimization concerns the parallel algorithmic processing of this specific part of the code, by leveraging the combined use of iterative solver BiCGStab(2), block and multigrid preconditioners. The solver requires a large number of matrix-vector multiplications, dot products, linear combinations and several applications of the multigrid preconditioners. These operations are the most important in terms of execution times. A domain decomposition approach is used to share the spatial data across processes, using a Cartesian decomposition of the global mesh, with MPI exchanges across subdomain boundaries. This parallelization is adapted to massively parallel computers with distributed memory.

3.1. Parallel operations

In the BiCGStab(2) algorithm (see. Algorithm 3), for each iteration, are performed:

- 4 matrix-vector multiplications

  The strategy used involves partitioning the rows of the matrix, as well as the vectors between the processors. Each processor P_i performs locally the product of a block A_i and a vector X_i. The resulting vector $(A \times X)_i$ has the dimension of the number of rows of the matrix A_i. Therefore, no communication is needed in the local computation of the matrix-vector product, instead, this distribution implements short communication messages before the beginning of each calculation.

- 9 dot products.

  The 9 dot products of this algorithm represent global values using distributed components of local vectors. Thus, each processor calculates a partial dot product of two corresponding local vectors by multiplying these two vectors and summing the components of the partial result. Then each processor sends its partial result to all the others and receives the products of the other processors that it sums to its own result. Therefore, a global communication is necessary in this scalar product.

- 10 linear combinations.

Each processor contains a part of each global vector included in the operation $X_i \leftarrow X_i + \alpha Y_i$, called local vector. The result vector of the linear combination obtained is also a local vector. Therefore, no communication is necessary in this operation.

- 16 multigrid preconditioner applications, corresponding to 4 multigrid per component (u, v, w and p).

  To apply the preconditioner $\mathcal{P}$ which is involved in the BiCGStab(2) algorithm, 20 matrix subsystems per iteration are needed to be solved. This is satisfied by using the HYPRE parallel computation library, which offers a large choice of interfaces (structured, semi-structured or algebraic) and multigrid preconditioners (algebraic as BoomerAMG and geometric like PFMG or SMG). In order to take advantage of the higher efficiency of PFMG, this solution strategy has been implemented in the code as it reveals to be the most efficient multigrid solver for our multiphase problems and fully-coupled approach.

3.2. Performance evaluation

The classical approach to qualifying a solver in the high-performance computing domain is to check its scalability. It allows to evaluate the performance of a parallel solver when the number of cores is increased. Two metrics, the speed-up S and the efficiency E are defined and two kinds of scalability achieved within two different contexts are distinguished:

- Strong scalability: for a fixed problem size, the number of processes is increased. Ideally, one hopes for linear scalability. In the case of an application that performs a large number of computations, the aim is to find the point at which a reasonable computational time is achieved but still limits the overhead induced by a parallel process.

  The speed-up $S(p)$ is defined as the gain of a parallel computation with p processes compared to the same algorithm over a number of reference processes p_{ref}.

  $$S(p) = \frac{t_{ref}}{t_p} p_{ref} \tag{22}$$

  where t_{ref} is the time cost for p_{ref} processes (note that t_{ref} not necessary equals to t_{seq} as the reference might not be the sequential case), and t_p the amount of time to complete the same unit of work with p processes.

- Weak scalability: both number of processes and problem size are increased proportionately, such that the quantity of data to be processed per process holds constant. In the case of ideal weak scalability, a constant execution time of the program should be achieved. This corresponds to a constant efficiency which is therefore independent of the number of processes.

  In this case the speed-up is defined by

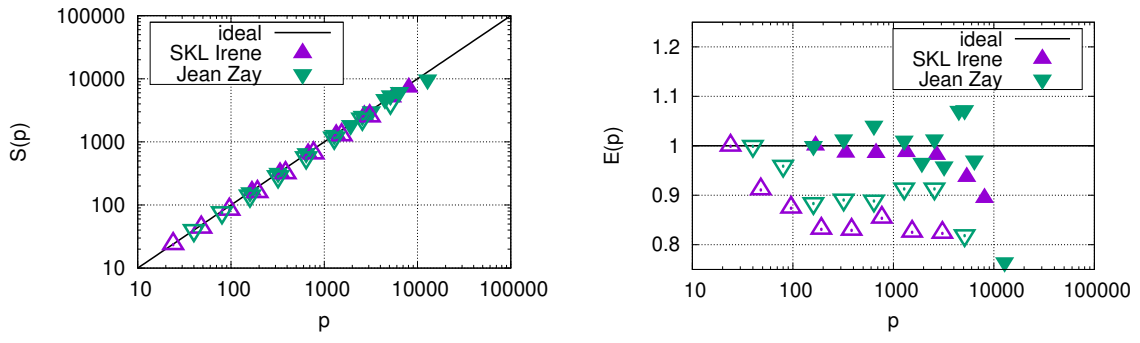
$$S(p) = \frac{t_{ref}}{t_p} p \tag{23}$$

For both cases, the efficiency provides an assessment of the "performance" of the parallel computation. In theory it is less than 1, but for superscalar algorithms the efficiency can exceed this conventional value. The following expression is used to define the efficiency:

$$E(p) = \frac{S(p)}{p} \tag{24}$$

In this section, the parallel performance of the in-house code Fugu and specifically the new fully-coupled solver are evaluated. For this purpose, we will consider the strong and weak scalability of the solver as defined above, by checking the evolution of two criteria: the speed-up and the efficiency.

The weak and strong scalability tests are performed on the simulation of the liquid-liquid phase inversion [45, 46], on two French supercomputers: SKL Irene, from CEA's Very Large Computing Centre (TGCC) in Bruyères-le-Châtel and for the Jean Zay: HPE SGI 8600 computer from the Institute for Development and Resources in Intensive Scientific Computing (IDRISS-CNRS). The number of processors is increased until 12800.



(a) $S(p)$ (b) $E(p)$

Figure 3: Strong (filled symbols) scalability on a 138 million scalar cells mesh and weak scalability (empty ones) on meshes corresponding to $60^3 \times p$ (from 5 millions up to 1.1 billion scalar cells).

Figure 3 presents the speed-up and efficiency for strong and weak approaches. The previous definitions (Eqs. (22)-(24)) allow a direct comparison of metrics. Both scalability analysis have been performed over 30 time iterations of the phase inversion problem tackled in Sec. 4.1 at fixed residual $\varepsilon = 10^{-4}$ of the iterative solver. Performances have been evaluated using complete computing nodes, i.e. the number of cores/processors is a multiple of 24 (40) for the SKL Irene (Jean Zay) supercalculator.

First, a good scalability S associated with the fully-coupled solver is obtained (Fig. 3a) with a speed-up close to the ideal one. No significant differences are visible between the two supercomputers.

Thereafter, it can be seen that the code is able to exploit the available computing resources on these supercomputers. The parallel efficiency E is 96% for a calculation on 6400 cores with 27^3 cells/core on Jean Zay and 89% for a calculation on 8064 cores with 25^3 cells/core on Irene. In the worst case, the weak scalability study (Fig. 3b) shows that the parallel efficiency obtained with 60^3 cells/core loading does not deteriorate that much when increasing the problem size. Indeed, the computational overhead is less than 20% up to a 5120 cores on Jean Zay and 3072 cores on SKL Irene.

4. Applications to physical cases

This part is addressed to the validation of the fully-coupled solver, and more specifically the numerical methodologies developed, specifically on two-phase flow test cases. Single phase validations on standard flow problems of the literature have

previously been considered in [20] for the 2D fully coupled method. The contribution of the fully-coupled method when tackling complex problems with high density and viscosity ratio will also be studied.

4.1. Phase inversion between two incompressible liquids

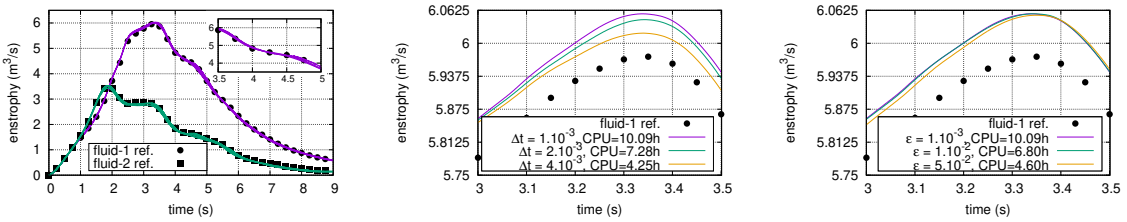
A two-phase flow, water-oil for instance, where one phase is dispersed in another, can correspond to a phase inversion, i.e., the dispersed phase becoming the continuous phase. This unsteady phenomenon is then accompanied by several small and large scale interfacial processes, such as deformation, ligament formation, interface breakdown and coalescence [45].

In this work, a three-dimensional phase inversion case recently studied by [46], is investigated. In the following, the geometry and the physical parameters of the problem are briefly presented before a discussion about the numerical results as well as their comparisons with the reference [46]. An initial cubic blob of light liquid, referred to as fluid 1, is placed at a bottom corner of a cubic box filled with a heavier liquid, called fluid 2. The size of the box is (H, H, H), while the size of the blob of light fluid is $(H/2, H/2, H/2)$. All box walls are considered as slip walls, so that the normal components as well as normal derivatives of tangential components are zero. Gravitational acceleration g is applied in the vertical direction. The dimensionless numbers governing the dynamics of this problem are therefore the Reynolds number $Re = \rho_1 H U_g/\mu_1$ and the Weber number $We = \rho_1 H {U_g}^2/\sigma$. The velocity scale used in previous studies [45] is $U_g = \sqrt{gH/2}(\rho_2 - \rho_1)/\rho_1$. The liquids properties are $\rho_1 = 1000$ kg/m^3, $\rho_2 = 900$ kg/m^3, $\mu_1 = \mu_2 = 1$ Pa·s, $g = -9.81$ m/s^2 and $\sigma = 0.45$ kg/s^2. The height $H = 1$ m, leads to $Re = 211$ and $We = 121$.

The macroscopic quantities of interest are defined and widely discussed in [45]. In this paper, the focus is on the enstrophy time evolution. Concerning the numerical parameters, the simulations are carried out on four grid ranging from 128^3 to 1024^3 cells. The residual of the iterative BiCGStab(2) solver is initially fixed to $\varepsilon = 10^{-5}$. As far as the time derivatives, a constant time step $\Delta t = 10^{-3}$ s is chosen, and a time corresponding to $10s$ of the flow motion is crossed. For the smaller mesh 1024^3, it should be noted that the simulations were performed on 6696 processors of the HPC Occigen cluster of CINES and have taken about 125h of computing time.

Before discussions about the numerical properties of the simulations, it should be noted that all quantities of interest (kinetic and potential energy, enstrophy, volume of light fluid in the top of the cavity) defined in [45] well converge, for all times, through the references solution from a mesh size of 256^3 (see Appendix D).

Figure 4 presents the time evolution of the enstrophy integrated for both fluids



(a) (b) (c)

Figure 4: 4a Enstrophy in fluid 1 and 2. 4b Enstrophy in fluid 1 for various time steps. 4c Enstrophy in fluid 1 for various residuals.

over the whole domain. Results are compared with those of [46].

Figure 4b shows effect of the time step on the enstrophy solution near the first peak. With a fixed residual, $\varepsilon = 10^{-3}$, and a reference time step $\Delta t = 10^{-3}$ s, the computational cost is reduced by almost 30% with a time step $\Delta t/2$ and about 60% with $\Delta t/4$. On this range of time steps, the obtained peak intensity varies less than 1%.

A similar study is presented in Fig. 4c for the residual values. The time step is fixed, $\Delta t = 10^{-3}$ s. The initial residual is $\varepsilon_0 = 10^{-3}$. With a 10 times larger value of residual, $\varepsilon = 10\varepsilon_0$, the computational cost is reduced by 30% and by 50% with a residual threshold $\varepsilon = 50\varepsilon_0$. On the plotted interval, the difference between simulations are about 2 while a robust 1% difference is again observed from the reference solution [46].

4.2. 3D rising bubble

This first case is based on a numerical benchmark [47] investigating rising bubbles in a liquid column. The studied cases allow to evaluate the robustness of the complete algorithm when the interface is in motion and undergoes relatively large deformations. The authors [47] compare different solutions of two configurations, achieved using numerical codes developed by different teams. Three groups participated in these two test cases: group 1 (IGPM, RWTH Aachen) with the DROPS code Finite element/Level-set, group 2 (INS, University of Bonn) with the code NaSt3D finite difference/Level set, and group 3 with the code OpenFOAM finite volume/VOF. A qualitative analysis of the characteristic parameters of the rising bubble, such as the time evolution of the centre of mass or the rising velocity of the bubble, allowed the authors to establish reference solutions for chosen parameters. Here are compared

the simulation results with the reference solutions from [47][2].

A brief description of the problem set-up and characteristics is presented. Initially, a bubble of radius $R_0 = 0.25$ m and centre (0.5 m, 0.5 m, 0.5 m) is set up inside a rectangular tank of dimensions $[0, 1] \times [0, 1] \times [0.2]$ m^3. Under the effect of buoyancy force, the bubble rises in the z-direction. The bubble density, ρ_2, is lower than that of the surrounding liquid (ρ_1). No slip boundary conditions at the walls is assumed, i.e., $\mathbf{u} = \mathbf{0}$ on all boundaries. The dimensionless numbers of this flow are the Reynolds number $Re = \rho_1 U_g D/\mu_1$, and the Eötvös number $Eo = \rho_1 {U_g}^2 D/\sigma$ yielding the ratio between gravitational forces and surface tension effects. The terms ρ_1 and μ_1 are the density and viscosity of the carrier fluid, D the bubble diameter, $U_g = \sqrt{gD}$ the gravitational velocity and σ the surface tension coefficient. In both tackled configurations, the evolution of the bubble is studied over $3s$. Table 1 gives the physical properties and dimensionless parameters associated with the two configurations.

Case	ρ_1 (kg/m^3)	ρ_1/ρ_2	μ_1 (Pa·s)	μ_1/μ_2	g (m/s^2)	σ (N/m^2)	Re	Eo
1	1000	10	10	10	0.981	24.5	35	10
2	1000	1000	10	100	0.981	1.96	35	125

Table 1: Physical parameters describing the test cases. Here, ρ_2 and μ_2 refers to the fluid density and viscosity respectively inside the bubble whereas ρ_1 and μ_1 are related to the density and viscosity of the surrounding fluid. Also, g is the magnitude of the gravity acceleration and σ is the surface tension coefficient.

For the qualitative analysis of the results, the following characteristics are studied: the barycenter $\mathbf{x}_c$ of the bubble and the rise velocity $\mathbf{u}_c$. Note that these quantity definitions can be found in [47]. to validate our solver, we have defined a reference solution on a fine mesh according to literature reference [47].

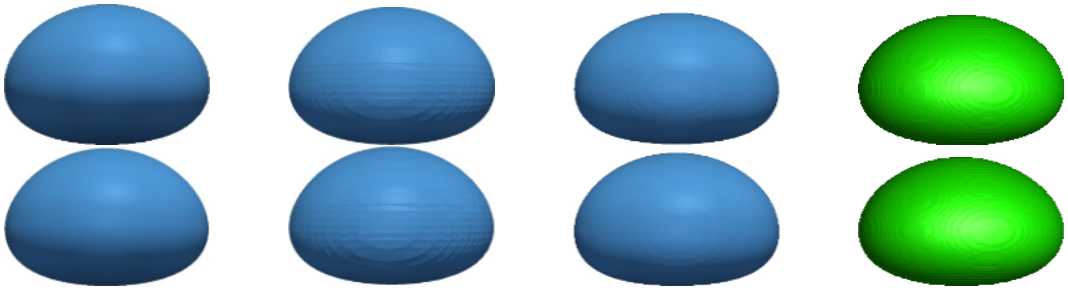
For both test cases, the simulations were carried out on three grids with a mesh size $h = 1/32$, $1/64$ and $1/128$ and time step size $\Delta t = 10^{-3}$ s. Our reference solution is defined as the one achieved on the finest grid $128 \times 128 \times 256$, i.e. $h = 1/128$ with the fully-coupled solver. It will be further compared with those of the three groups from the benchmark. It should be noted that the reference solution of the first group (with DROPS code) is provided on a regular initial grid containing $4 \times 8 \times 4$ cells, each being subdivided into 6 tetrahedra. This grid is then dynamically refined near the interface, leading to a mesh size of $h = 1/32$ within the refinement zone. The

[2]The references data of all quantities of interest are available on the website: `http://wissrech.ins.uni-bonn.de/research/projects/risingbubblebenchmark`.

time step size $\Delta t = 2.5 \times 10^{-4}$ s. The results of the second group (NaSt3DGPF) is achieved with $121 \times 121 \times 241$ grid cells and time step size $\Delta t = 10^{-4}$ s. Lastly, group 3 (OpenFOAM) has constructed a reference solution on a regular grid with $128 \times 128 \times 256$ cells and a time step $\Delta t = 10^{-4}$ s.

Test case 1

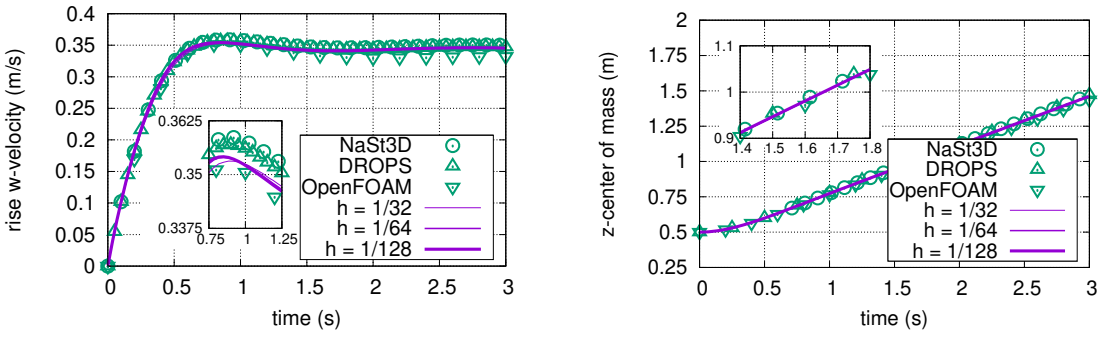
The simulations of the rising bubble obtained at instant $t = 2$ s and $t = 3$ s are shown in Fig. 5. It can be observed that the overall shape of the bubble is similar with the fully-coupled approach to that of others groups. Indeed, as for the references, the bubble reaches a stable ellipsoidal shape by extending itself in directions perpendicular to the flow. Time evolution of the rising velocity are shown



(a) DROPS (b) NaSt3D (c) OpenFOAM (d) present work

Figure 5: Bubble rise for test case 1. The iso value $C = 0.5$ of colour function is plotted at times $t = 3$ s (first row) and $t = 2$ s (second row) from different codes. Snapshots a, b and c are extracted from [47].

in figure 6a



(a) rise velocity

(b) center of mass position

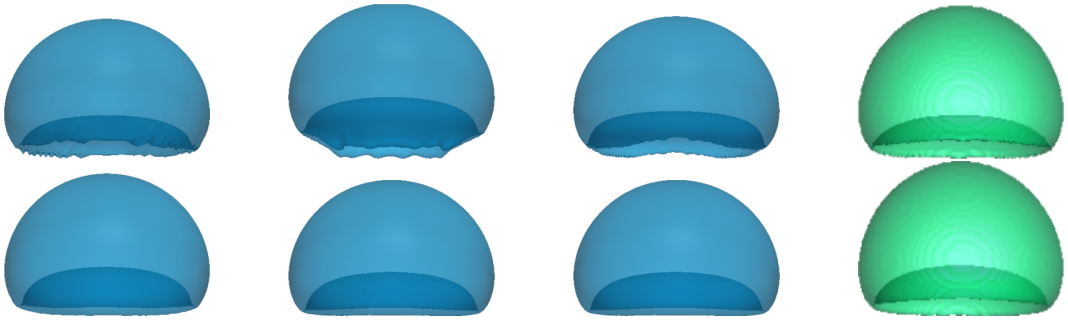
Figure 6: Test case 1. a Rise velocity w-component as un function of time. b Center of mass position. Plain lines stand for the present work while symbols refer to benchmark results [47] from different codes.

for different grid refinements. Also, the state of the simulation between 0.75 s and 1.25 s is visualized in a zoomed extracted in Fig. 6a. It can be concluded that a very nice convergence of the solutions is obtained for the fully-coupled solver from grid $h = 1/32$. Furthermore, the solutions provide results in good agreement with those of the DROPS and NaSt3D codes. Furthermore, in the final part of the simulation, significant differences are noted between the fully-coupled solver and the OpenFOAM code. With respect to the evolution of the position of the centre of mass, we can clearly see in Figs. 6b that the solutions converge from the grid $h = 1/32$, they are in accordance with the references. It has to be noticed that with the fully coupled solver, solutions equivalent to other teams have been obtained by using larger time steps, which is a nice feature of the fully coupled approach.

Test case 2

This test was also carried out on grids with a mesh size $h = 1/32$, $1/64$ and $1/128$ and time step size $\Delta t = 10^{-3}$ s. As for the reference solutions, the same numerical parameters are used as in the previous case. According to the diagram of [48], the bubble in test 2 is in a dimpled ellipsoidal-cap regime where its shape can be strongly deformed with the interface dug in its wake. This is in particular due to the value of the surface tension that is lower than in test case 1. In Figure 7, the evolution of the interface of test 2 is reported at $t = 2$ s and $t = 3$ s. The numerical shapes corresponding to $C = 0.5$ iso-surface correlates well with experimental result predicted by the diagram of [48]. As illustrated in this figure, the fully-coupled solver provides results comparable to the one of the benchmark. In fact, in the resolved

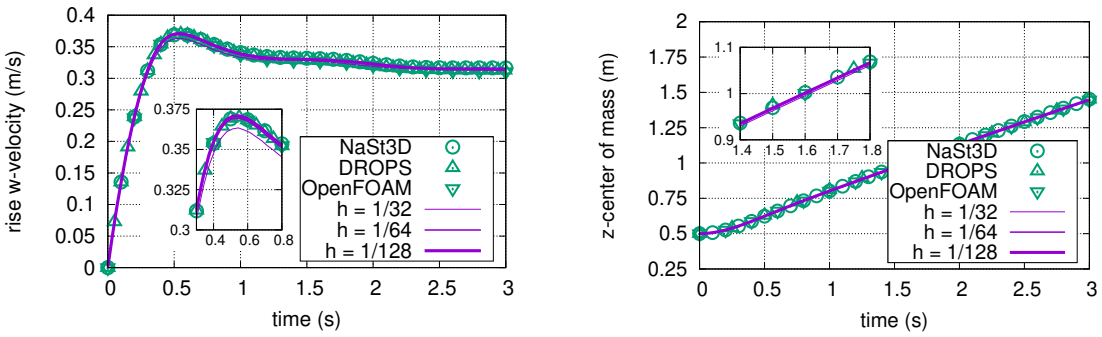
regions of the bubble, the overall shape of the bubble is converged as for the other groups in the benchmark. However, the bubble shapes strongly differ at the bottom edge between the codes. The rising velocity over time is visualized in Figures 8b



(a) DROPS (b) NaSt3D (c) OpenFOAM (d) present work

Figure 7: Same as Fig. 5 for test case 2.

and 8a. We notice that all the codes have a velocity peak which is of the order of 0.37 at $t = 0.5$. After this time, the bubble velocity decreases to reach a steady state. Also, our simulations are in better agreement with the results of DROPS and OpenFOAM codes than the simulations of NaSt3D code. Finally, Fig. 8a shows the evolution of the barycenter of the bubble. Here, the numerical results show good agreement between all codes with less sensitivity of the results regarding the numerical approaches. On a global point of view, the Fugu solver has been favourably compared to the results of benchmark [47], by suing larger time steps, which validates the accuracy and the physical relevancy of the fully-coupled solver.



(a) rise velocity (b) center of mass position

Figure 8: Same as Fig. 6 for test case 2.

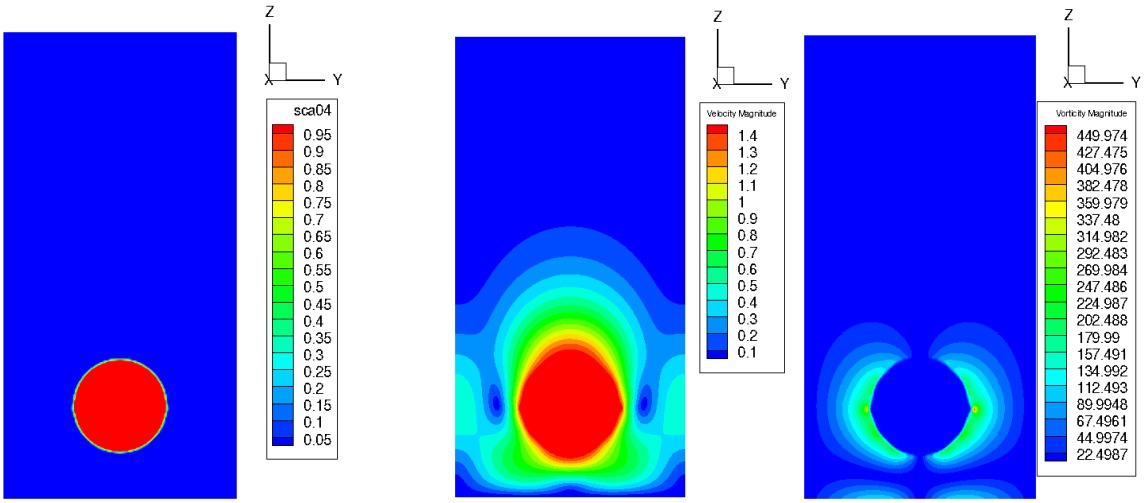
4.3. Free fall of a dense sphere

The free fall of a dense sphere in air may seem like a simple process at first sight. However, predicting the behaviour of this two-phase system perfectly is not easy on a numerical point of view. Indeed, the high density and viscosity ratios lead to ill-conditioned linear systems and difficulties in solving issues. Consequently, this test case has a great interest to us for two reasons. On the one hand, it allows checking whether the resolution is robust, when complex unsteady two-phases problems are undertaken, in which density and viscosity ratio may exceed 10^6. On the other hand, since the exact solution of the falling velocity ($w_c = -gt$ m/s in a void medium) and center of mass ($z_c = -gt^2/2 + z_0$ m) are known, thereafter we can evaluate the accuracy of the fully-coupled solver.

The initial configuration consists of highly viscous liquid droplet of radius $r = 0.0125$ m, density $\rho = \rho_2$ and dynamic viscosity $\mu = \mu_2$ which is released without initial velocity in air. Gravity is set at $g = -9.81$ m/s^{-2} in the z-direction, while the surface tension coefficient σ is set to zero. The sphere is centered at $(x_0, y_0, z_0) = (0.05$ m, 0.05 m, 0.15 m) in a parallelepipedic cavity full of air whose density and viscosity are $\rho = \rho_1$ and $\mu = \mu_1$ respectively. The cavity is 0.2 m high, 0.1 m long and 0.1 m wide.

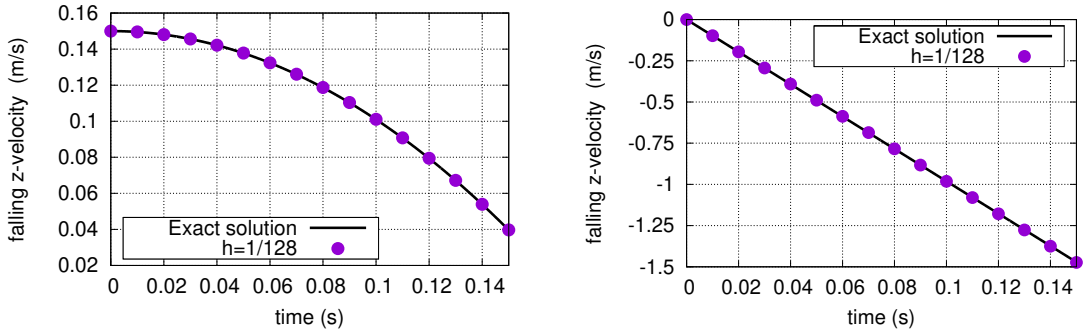
Concerning the numerical parameters, the simulations are carried out on a Cartesian grid containing $128 \times 128 \times 256$ cells, with a residual of $\varepsilon = 10^{-4}$ for the BiCGStab(2). As far as the time derivatives, a constant time step $\Delta t = 1 \times 10^{-4}$ s is chosen, and 1500 time steps are computed, corresponding to 0.15 second of the flow motion. It has to be noted that the simulation were performed on 144 processors of the HPC cluster Occigen of CINES and required about 5h of computing time. The physical parameters for the test case are $\rho_1 = 1$ kg/m^3, $\mu_1 = 1$ Pa·s, $\rho_2/\rho_1 = \mu_2/\mu_1 = 10^6$ and $g = -9.81$ m/s^2.

As in the case of the rising bubble, the qualitative analysis of the results is focused on the barycenter $\mathbf{x}_c$ of the bubble and the rise velocity $\mathbf{u}_c$. The simulation results of the fully-coupled solver after 1500 time iterations are illustrated in Fig. 9. The corresponding colour function at the interface, vertical velocity and vorticity magnitude are presented. It can be seen that the fully coupled solver accurately predicts the fall of the sphere {by conserving the spherical shape of the solid when neglecting the drag force. Unquestionably, the computation shows that the strain rate tensor inside the ball vanishes as $\|\nabla \cdot \bar{\bar{\mathbf{T}}}\| = \mathcal{O}(\mu_1/\mu_2)$. The Figs. (10a and 10b) reports the evolution over time of the position of the falling velocity and the centre of mass of the sphere, obtained by the fully-coupled solver when the reference grid $128 \times 128 \times 256$ is considered as well as the corresponding analytical solutions. The numerical solution is in good agreement with the reference solution.



(a) phase function C (b) $\|\mathbf{u}\|$ (c) $\|\nabla \times \mathbf{u}\|$

Figure 9: Numerical simulation of a free fall of dense sphere at $t = 0.144$ s, obtained on a fine mesh $128 \times 128 \times 256$, by the fully coupled solver. The different fields presented in each row are: left color function, middle: velocity magnitude and right: vorticity magnitude.



(a) (b)

Figure 10: Vertical velocity and center of mass for test case 1 $\left(\rho_1/\rho_2 = \mu_1/\mu_2 = 10^6\right)$ obtained by the fully coupled method with several meshes. (a) Vertical velocity and (b) center of mass.

5. Conclusions

An original fully-coupled solver has been presented for computing three-dimensional unsteady and incompressible two phase flows, in which the velocity-pressure coupling is kept at each time step. The tracking of the interface has been modelled by solving an advection equation of phase function by means of conservative VOF scheme. The

Navier-Stokes equations have been successfully solved using our new fully-coupled solver thanks to the development of new block preconditioning techniques for the velocity block. The latter has been done by means of an infinite series development of the different terms of the velocity block. Further, we have demonstrated the ability of the solver to deal with complex cases, for high density and viscosity ratios, such as the case of the free fall of dense sphere or the rising bubble in a liquid column. In addition, we have shown that the fully-coupled solver is able to resolve two-phase problems on more than 1 billions cells, with excellent scalability.

Declaration of competing interest

The authors have no competing interest to declare rgarding the publication of this article.

Acknowledgements

We are grateful for access to the computational facilities of the French CINES (National computing center for higher education) and TGCC granted by GENCI under project numbers A0092B06115. We thank the technical and administrative teams of these supercomputer centers and agencies for their kind and efficient help.

Appendix A. Discretization with finite volumes in Fugu code

Appendix A.1. Convective term

One recalls the convective term in equation 3a, discretized to first order in time, and integrated over the control volume Ω, bounded by the surface Σ:

$$\int_\Omega \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u})\, dV = \int_\Sigma \rho u_l u_m n_m \, dS \approx \int_\Sigma \rho^{n+1} u_m^n u_l^{n+1} n_m \, dS \tag{A.1}$$

The discretization on the 3D staggered mesh of the convective term is performed with a centered scheme in a semi-implicit way. It is written for the velocity component u:

$$\begin{aligned}
\text{(A.1)} \Rightarrow & \rho_{i,j,k}^{n+1} \left(\frac{u_{i,j,k}^n + u_{i+1,j,k}^n}{2} \frac{u_{i,j,k}^{n+1} + u_{i+1,j,k}^{n+1}}{2} - \frac{u_{i-1,j,k}^n + u_{i,j,k}^n}{2} \frac{u_{i-1,j,k}^{n+1} + u_{i,j,k}^{n+1}}{2} \right) \Delta y \Delta z \\
& + \rho_{i,j,k}^{n+1} \left(\frac{v_{i,j,k}^n + v_{i,j+1,k}^n}{2} \frac{u_{i,j,k}^{n+1} + u_{i,j+1,k}^{n+1}}{2} - \frac{v_{i,j-1,k}^n + v_{i,j,k}^n}{2} \frac{u_{i,j-1,k}^{n+1} + u_{i,j,k}^{n+1}}{2} \right) \Delta x \Delta z \\
& + \rho_{i,j,k}^{n+1} \left(\frac{w_{i,j,k}^n + w_{i,j,k+1}^n}{2} \frac{u_{i,j,k}^{n+1} + u_{i,j,k+1}^{n+1}}{2} - \frac{w_{i,j,k-1}^n + w_{i,j,k}^n}{2} \frac{u_{i,j,k-1}^{n+1} + u_{i,j,k}^{n+1}}{2} \right) \Delta x \Delta y
\end{aligned} \tag{A.2}$$

with Δx, Δy and Δz the sizes of the parallelepipedic control volume in the x-, y- and z-direction, respectively.

Appendix A.2. Viscous terms

The viscous stress tensor is decomposed as $T_{lm} = \kappa \Lambda_{lm} + \zeta \Xi_{lm} - \eta \Gamma_{lm}$ where Λ, Ξ and Γ tensors are defined in by [49]. The coefficient κ, ζ and η are elongational, shear and rotational viscosities. They are linked to the dynamical viscosity by $\kappa = \zeta = 2\mu$ and $\eta = \mu$. Tensors Λ, Ξ and Γ respectively read

$$\Lambda = \begin{pmatrix} \frac{\partial u}{\partial x} & 0 & 0 \\ 0 & \frac{\partial v}{\partial y} & 0 \\ 0 & 0 & \frac{\partial w}{\partial z} \end{pmatrix}, \quad \Xi = \begin{pmatrix} 0 & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & 0 & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & 0 \end{pmatrix}, \quad \Gamma = \begin{pmatrix} 0 & \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} & \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \\ \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} & 0 & \frac{\partial v}{\partial z} - \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial x} - \frac{\partial u}{\partial z} & \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} & 0 \end{pmatrix} \tag{A.3}$$

The viscous operator is approximated implicitly using standard second-order, centered finite-differences. It is written explicitly for the u-velocity component:

Appendix A.2.1. Elongation viscosity $(\kappa\Lambda_{lm})_{,m}$, $u$component

$$\int \frac{\partial}{\partial x}\left(\kappa\frac{\partial u}{\partial x}\right)dV = \left(\kappa_{i+\frac{1}{2},j,k}\frac{u^{n+1}_{i+1,j,k}-u^{n+1}_{i,j,k}}{\Delta x} - \kappa_{i-\frac{1}{2},j,k}\frac{u^{n+1}_{i,j,k}-u^{n+1}_{i-1,j,k}}{\Delta x}\right)\Delta y\Delta z \tag{A.4}$$

Appendix A.2.2. Shear viscosity $(\zeta\Xi_{lm})_{,m}$, $u$component

$$\begin{aligned}\int \frac{\partial}{\partial y}\left(\zeta\frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial z}\left(\zeta\frac{\partial u}{\partial z}\right)dV = &\left(\zeta_{i,j+\frac{1}{2},k}\frac{u^{n+1}_{i,j+1,k}-u^{n+1}_{i,j,k}}{\Delta y} - \zeta_{i,j-\frac{1}{2},k}\frac{u^{n+1}_{i,j,k}-u^{n+1}_{i,j-1,k}}{\Delta y}\right)\Delta x\Delta z\\ &+\left(\zeta_{i,j,k+\frac{1}{2}}\frac{u^{n+1}_{i,j,k+1}-u^{n+1}_{i,j,k}}{\Delta z} - \zeta_{i,j,k-\frac{1}{2}}\frac{u^{n+1}_{i,j,k}-u^{n+1}_{i,j,k-1}}{\Delta z}\right)\Delta x\Delta y\end{aligned} \tag{A.5}$$

Appendix A.2.3. Rotational viscosity $(-\eta\Gamma_{lm})_{,m}$, $u$component

$$\begin{aligned}&\int \frac{\partial}{\partial y}\left(-\eta\frac{\partial u}{\partial y}+\eta\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial z}\left(-\eta\frac{\partial u}{\partial z}+\eta\frac{\partial w}{\partial x}\right)dV\\ &= \left(-\eta_{i,j+\frac{1}{2},k}\frac{u^{n+1}_{i,j+1,k}-u^{n+1}_{i,j,k}}{\Delta y} + \eta_{i-\frac{1}{2},j+1,k}\frac{v^{n+1}_{i,j+1,k}-v^{n+1}_{i-1,j+1,k}}{\Delta x}\right)\Delta x\Delta z\\ &+\left(\eta_{i,j-\frac{1}{2},k}\frac{u^{n+1}_{i,j,k}-u^{n+1}_{i,j-1,k}}{\Delta y} - \eta_{i-\frac{1}{2},j,k}\frac{v^{n+1}_{i,j,k}-v^{n+1}_{i-1,j,k}}{\Delta x}\right)\Delta x\Delta z\\ &+\left(-\eta_{i,j,k+\frac{1}{2}}\frac{u^{n+1}_{i,j,k+1}-u^{n+1}_{i,j,k}}{\Delta z} + \eta_{i-\frac{1}{2},j,k+1}\frac{w^{n+1}_{i,j,k+1}-w^{n+1}_{i-1,j,k+1}}{\Delta x}\right)\Delta x\Delta y\\ &+\left(\eta_{i,j,k-\frac{1}{2}}\frac{u^{n+1}_{i,j,k}-u^{n+1}_{i,j,k-1}}{\Delta z} - \eta_{i-\frac{1}{2},j,k}\frac{w^{n+1}_{i,j,k}-w^{n+1}_{i-1,j,k}}{\Delta x}\right)\Delta x\Delta y\end{aligned} \tag{A.6}$$

Appendix A.3. Pressure Gradient ∇p, $u$component

For the horizontal velocity component u, the pressure gradient is discretized as follows:

$$\int \frac{\partial p}{\partial x} dV = (p_{i+\frac{1}{2},j,k} - p_{i-\frac{1}{2},j,k})\Delta x \Delta y \Delta z \tag{A.7}$$

Appendix A.4. Divergence $\nabla \cdot \mathbf{u}^{n+1}$f the velocity field

The divergence $\nabla \cdot \mathbf{u}$ of the velocity field $\mathbf{u}^{n+1} = (u^{n+1}, v^{n+1}, w^{n+1})^T$ is approximated at cell centers by

$$\int \nabla \cdot \mathbf{u}^{n+1} dV = \left(\frac{u^{n+1}_{i+\frac{1}{2},j,k} - u^{n+1}_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{v^{n+1}_{i,j+\frac{1}{2},k} - v^{n+1}_{i,j-\frac{1}{2},k}}{\Delta y} + \frac{w^{n+1}_{i,j,k+\frac{1}{2}} - w^{n+1}_{i,j,k-\frac{1}{2}}}{\Delta z} \right) \Delta x \Delta y \Delta z \tag{A.8}$$

Appendix B. Physical boundary conditions

To complete the one-fluid formulation, the boundary conditions of the fluid domain are imposed via the penalty term $\bar{\bar{\mathbf{B}}} \cdot (\mathbf{f}(\mathbf{u}) - \mathbf{u}_\infty)$ on $\Gamma = \partial\Omega$. Boundary conditions such as Dirichlet or Neumann can be enforced on the domain borders. Since the physical boundaries of the fluid domain coincide with the nodes of the scalar meshes, it is no longer the variable $\mathbf{u}^{n+1}$ that has to be penalised, but a function $\mathbf{f}(\mathbf{u}^{n+1})$ of this variable. where

$$\bar{\bar{\mathbf{B}}} = \begin{bmatrix} \alpha_u & 0 & 0 \\ 0 & \alpha_v & 0 \\ 0 & 0 & \alpha_w \end{bmatrix} \tag{B.1}$$

is a tensor field whose diagonal components tend to infinity along the boundary Γ and are identically zero inside the fluid domain Ω. Here, $\mathbf{f}(\mathbf{u}^{n+1})$ is a discrete function of u^{n+1}, v^{n+1} and w^{n+1} , which is written as a linear combination of resolved velocities $u^{n+1}_{i,j,k}$, $v^{n+1}_{i,j,k}$, $w^{n+1}_{i,j,k}$ and their neighbors:

$$f(u) = a_0 u^{n+1}_{i,j,k} + a_1 u^{n+1}_{i-1,j,k} + a_2 u^{n+1}_{i+1,j,k} + a_3 u^{n+1}_{i,j-1,k} + a_4 u^{n+1}_{i,j+1,k} + a_5 u^{n+1}_{i,j,k-1} + a_6 u^{n+1}_{i,j,k+1} \tag{B.2}$$

$$f(v) = a_0 v^{n+1}_{i,j,k} + a_1 v^{n+1}_{i-1,j,k} + a_2 v^{n+1}_{i+1,j,k} + a_3 v^{n+1}_{i,j-1,k} + a_4 v^{n+1}_{i,j+1,k} + a_5 v^{n+1}_{i,j,k-1} + a_6 v^{n+1}_{i,j,k+1}, \tag{B.3}$$

$$f(w) = a_0 w^{n+1}_{i,j,k} + a_1 w^{n+1}_{i-1,j,k} + a_2 w^{n+1}_{i+1,j,k} + a_3 w^{n+1}_{i,j-1,k} + a_4 w^{n+1}_{i,j+1,k} + a_5 w^{n+1}_{i,j,k-1} + a_6 w^{n+1}_{i,j,k+1}. \tag{B.4}$$

The treatment of the Dirichlet boundary conditions, applied to the left boundary for example is controlled by the coefficients a_i which are then set to the following values: $a_0 = a_2 = \frac{1}{2}$ and $a_1 = a_3 = a_4 = a_5 = a_6 = 0$. Thus, the discrete momentum equation on the left boundary becomes:

$$NS(u_{i,j,k}^{n+1}) + \alpha_u \left(\frac{1}{2} u_{i,j,k}^{n+1} + \frac{1}{2} u_{i+1,j,k}^{n+1} - u_\infty \right) = 0. \tag{B.5}$$

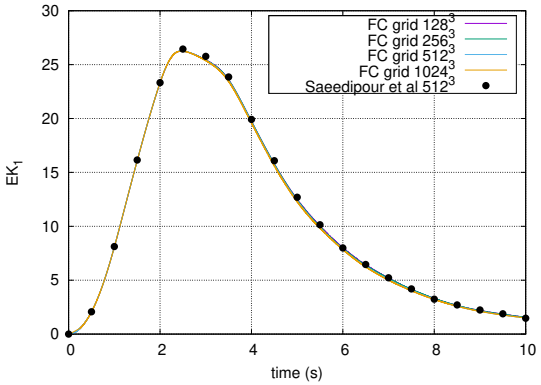
Appendix C. Algorithms

Appendix C.1. Iterative solver general structure

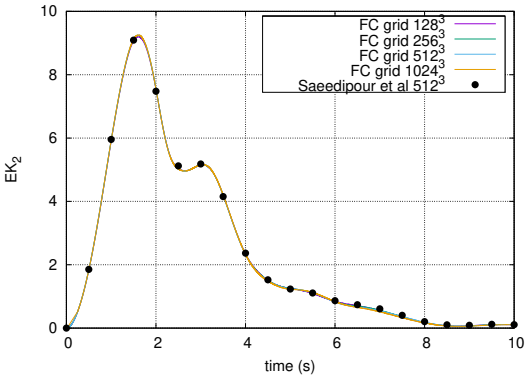
Algorithm 3 Solving strategy with BiCGStab(2) solver assuming preconditioner $\mathcal{P}$ is known.

procedure INITIALISATION
  $k = 0$ $k \in \mathbb{N}^+$, ▷ iteration number
  $\mathbf{b} = \mathcal{P}^{-1}\mathbf{b}$ ▷ Precond is applied on r.h.s.
  $\mathbf{z} = A\mathbf{x}^{(0)}$ ▷ $\mathbf{x}^{(0)} \in \mathbb{R}$ is an initial guess
  $\mathbf{q} = \mathcal{P}^{-1}\mathbf{z}$
  $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{q}$ ▷ Initial residual of the preconditioned system
  $\tilde{\mathbf{r}} = \mathbf{r}^{(0)}$ ▷ $\tilde{\mathbf{r}}$ is an arbitrary vector, such that $\tilde{\mathbf{r}} \cdot \mathbf{r}^{(0)} \neq 0$
  $\xi_0 = \omega_2 = 1$; $\alpha = 0$; $\mathbf{u} = \mathbf{0}$
end procedure
while $\frac{\|\mathbf{r}^{(k)}\|_2}{\|\mathbf{r}^{(0)}\|_2} > \varepsilon$ **do**
  $k = k + 1$
  $\xi_0 = -\omega_2 \xi_0$
  EVEN STEP
  $\xi_1 = \tilde{\mathbf{r}} \cdot \mathbf{r}^{(k-1)}$; $\beta = \alpha\xi_1/\xi_0$; $\xi_0 = \xi_1$ ▷ dot product #1
  $\mathbf{u} = \mathbf{r}^{(k-1)} - \beta\mathbf{u}$ ▷ linear combination #1
  $\mathbf{z} = A\mathbf{u}$ ▷ matrix-vector #1
  $\mathbf{v} = \mathcal{P}^{-1}\mathbf{z}$ ▷ preconditionner #1
  $\gamma = \tilde{\mathbf{r}} \cdot \mathbf{v}$; $\alpha = \xi/\gamma$ ▷ dot product #2
  $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \alpha\mathbf{v}$ ▷ linear combination #2
  $\mathbf{z} = A\mathbf{r}^{(k)}$ ▷ matrix-vector #2
  $\mathbf{s} = \mathcal{P}^{-1}\mathbf{z}$ ▷ preconditionner #2
  $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \alpha\mathbf{u}$ ▷ linear combination #3
  ODD STEP
  $\xi_1 = \tilde{\mathbf{r}} \cdot \mathbf{s}$; $\beta = \alpha\xi_1/\xi_0$ ▷ dot product #3
  $\mathbf{v} = \mathbf{s} - \beta\mathbf{v}$ ▷ linear combination #4
  $\mathbf{z} = A\mathbf{v}$ ▷ matrix-vector #3
  $\mathbf{w} = \mathcal{P}^{-1}\mathbf{z}$ ▷ preconditionner #3
  $\gamma = \tilde{\mathbf{r}} \cdot \mathbf{w}$; $\alpha = \xi_0/\gamma$ ▷ dot product #4
  $\mathbf{u} = \mathbf{r}^{(k)} - \beta\mathbf{u}$ ▷ linear combination #5
  $\mathbf{r}^{(k)} = \mathbf{r}^{(k)} - \alpha\mathbf{v}$ ▷ linear combination#6
  $\mathbf{s} = \mathbf{s} - \alpha\mathbf{w}$ ▷ linear combination #7
  $\mathbf{z} = A\mathbf{s}$ ▷ matrix-vector #4
  $\mathbf{t} = \mathcal{P}^{-1}\mathbf{z}$ ▷ preconditionner #4
  GENERALIZED CONJUGATE GRADIENT PART
  $\omega_1 = \mathbf{r}^{(k)} \cdot \mathbf{s}$ ▷ dot product #5
  $\lambda = \mathbf{s} \cdot \mathbf{s}$ ▷ dot product#6
  $\nu = \mathbf{s} \cdot \mathbf{t}$ ▷ dot product#7
  $\tau = \mathbf{t} \cdot \mathbf{t}$ ▷ dot product#8
  $\omega_2 = \mathbf{r}^{(k)} \cdot \mathbf{t}$ ▷ dot product #9
  $\tau = \tau - \nu^2/\lambda$; $\omega_2 = (\omega_2 - \nu\omega_1/\lambda)/\tau$; $\omega_1 = (\omega_1 - \nu\omega_2)/\lambda$
  $\mathbf{x}^{(k)} = \mathbf{x}^{(k)} + \omega_1\mathbf{r}^{(k)} + \omega_2\mathbf{s} + \alpha\mathbf{u}$ ▷ linear combination #8
  $\mathbf{r}^{(k)} = \mathbf{r}^{(k)} - \omega_1\mathbf{s} - \omega_2\mathbf{t}$ ▷ linear combination #9
  $\mathbf{u} = \mathbf{u} - \omega_1\mathbf{v} - \omega_2\mathbf{w}$ ▷ linear combination #10
end while

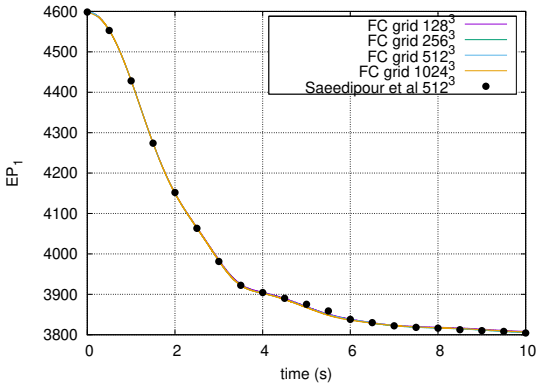
Appendix D. Phase inversion results



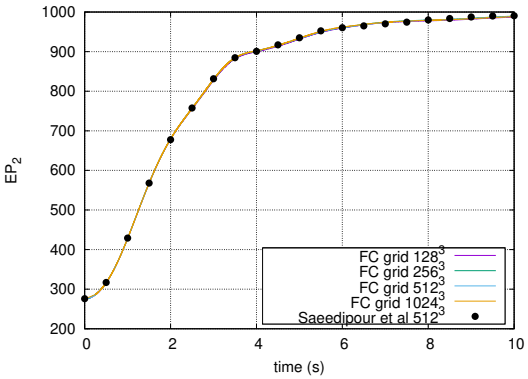
(a) Kinetic energy in fluid 1



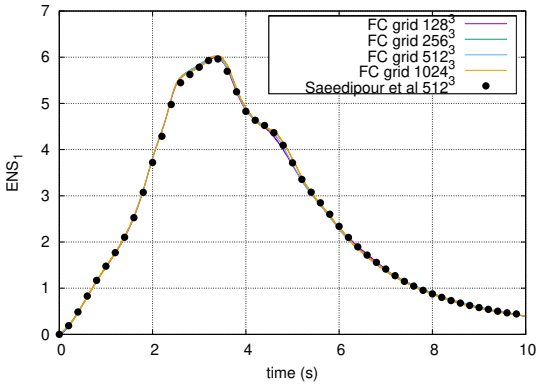
(b) Kinetic energy in fluid 2



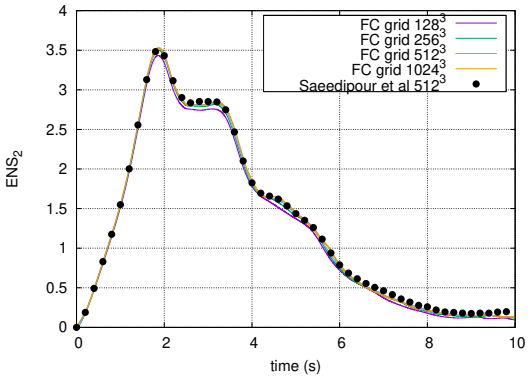
(c) Potential energy in fluid 1



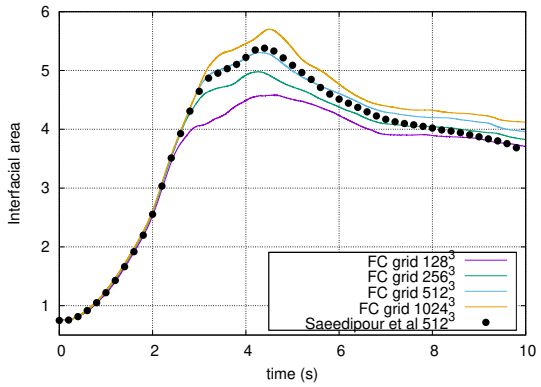
(d) Potential energy in fluid 2



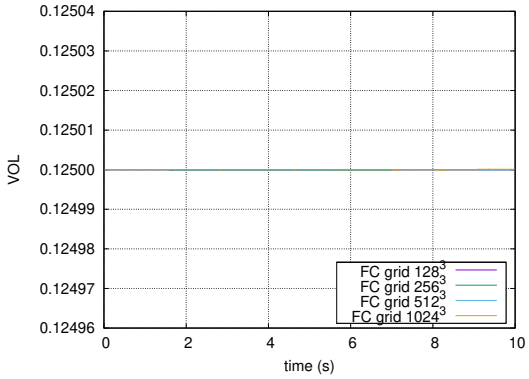
(a) Enstrophy in fluid 1



(b) Enstrophy in fluid 2



(c) Interfacial area



(d) Mass conservation

References

[1] J. J. Dongarra, I. S. Duff, D. C. Sorensen, H. A. Van der Vorst, Numerical linear algebra for high-performance computers, SIAM, 1998.

[2] S. Fleau, S. Mimouni, N. Mérigoux, S. Vincent, Validation of a multifield approach for the simulations of two-phase flows, Computational Thermal Sciences: An International Journal 7 (5-6) (2015).

[3] G. Davy, E. Reyssat, S. Vincent, S. Mimouni, Euler–euler simulations of condensing two-phase flows in mini-channel: Combination of a sub-grid approach and an interface capturing approach, International Journal of Multiphase Flow 149 (2022) 103964.

[4] P. Lubin, S. Vincent, S. Abadie, J.-P. Caltagirone, Three-dimensional large eddy simulation of air entrainment under plunging breaking waves, Coastal engineering 53 (8) (2006) 631–655.

[5] G. C. Agbangla, P. Bacchin, E. Climent, Collective dynamics of flowing colloids during pore clogging, Soft Matter 10 (33) (2014) 6303–6315.

[6] A. Ozel, J. B. de Motta, M. Abbas, P. Fede, O. Masbernat, S. Vincent, J.-L. Estivalezes, O. Simonin, Particle resolved direct numerical simulation of a liquid–solid fluidized bed: comparison with experimental data, International Journal of Multiphase Flow 89 (2017) 228–240.

[7] D. Lacanette, S. Vincent, E. Arquis, P. Gardin, Numerical simulation of gas-jet wiping in steel strip galvanizing process, ISIJ international 45 (2) (2005) 214–220.

[8] S. Vincent, G. Balmigere, C. Caruyer, E. Meillot, J.-P. Caltagirone, Contribution to the modeling of the interaction between a plasma flow and a liquid jet, Surface and Coatings Technology 203 (15) (2009) 2162–2171.

[9] R. F. Cerqueira, E. E. Paladino, F. Evrard, F. Denner, B. van Wachem, Multiscale modeling and validation of the flow around taylor bubbles surrounded with small dispersed bubbles using a coupled vof-dbm approach, International Journal of Multiphase Flow 141 (2021) 103673.

[10] S. Vincent, A. Sarthou, J.-P. Caltagirone, F. Sonilhac, P. Février, C. Mignot, G. Pianet, Augmented lagrangian and penalty methods for the simulation of

two-phase flows interacting with moving solids. application to hydroplaning flows interacting with real tire tread patterns, Journal of computational physics 230 (4) (2011) 956–983.

[11] M. Ishii, T. Hibiki, Thermo-fluid dynamics of two-phase flow, Springer Science & Business Media, 2010.

[12] D. A. Drew, S. L. Passman, Theory of multicomponent fluids, Vol. 135, Springer Science & Business Media, 2006.

[13] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct numerical simulations of gas–liquid multiphase flows, Cambridge university press, 2011.

[14] A. J. Chorin, Numerical solution of the navier-stokes equations, Mathematics of computation 22 (104) (1968) 745–762.

[15] J.-L. Guermond, Remarques sur les méthodes de projection pour l'approximation des équations de navier–stokes, Numerische Mathematik 67 (4) (1994) 465–473.

[16] J.-L. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Computer methods in applied mechanics and engineering 195 (44-47) (2006) 6011–6045.

[17] N. Bootland, A. Bentley, C. Kees, A. Wathen, Preconditioners for two-phase incompressible navier–stokes flow, SIAM Journal on Scientific Computing 41 (4) (2019) B843–B869.

[18] N. Nangia, B. E. Griffith, N. A. Patankar, A. P. S. Bhalla, A robust incompressible navier-stokes solver for high density ratio multiphase flows, Journal of Computational Physics 390 (2019) 548–594.

[19] M. El Ouafa, S. Vincent, V. Le Chenadec, Monolithic solvers for incompressible two-phase flows at large density and viscosity ratios, Fluids 6 (1) (2021) 23.

[20] M. El Ouafa, S. Vincent, V. Le Chenadec, Navier-stokes solvers for incompressible single- and two-phase flows, Communications in Computational Physics 29 (4) (2021) 1213–1245.

[21] M. El Ouafa, Développement d'un solveur tout-couplé parallèle 3d pour la simulation des écoulements diphasiques incompressibles à forts rapports de viscosités et de masses volumiques, Ph.D. thesis, Université Gustave Eiffel (12 2022). doi:10.13140/RG.2.2.29047.91040.

[22] M. Fortin, R. Glowinski, Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems, Elsevier, 2000.

[23] S. Vincent, J.-P. Caltagirone, P. Lubin, T. N. Randrianarivelo, An adaptative augmented lagrangian method for three-dimensional multimaterial flows, Computers & fluids 33 (10) (2004) 1273–1289.

[24] R. D. Falgout, U. M. Yang, hypre: A library of high performance preconditioners, in: International Conference on computational science, Springer, 2002, pp. 632–641.

[25] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, J. Koster, A fully asynchronous multifrontal solver using distributed dynamic scheduling, SIAM Journal on Matrix Analysis and Applications 23 (1) (2001) 15–41.

[26] P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, T. Mary, Performance and scalability of the block low-rank multifrontal factorization on multicore architectures, ACM Transactions on Mathematical Software (TOMS) 45 (1) (2019) 1–26.

[27] H. Kotakemori, H. Hasegawa, A. Nishida, Performance evaluation of a parallel iterative method library using openmp, in: Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05), IEEE, 2005, pp. 5–pp.

[28] I. Kataoka, Local instant formulation of two-phase flow, International Journal of Multiphase Flow 12 (5) (1986) 745–758.

[29] J. U. Brackbill, D. B. Kothe, C. Zemach, A continuum method for modeling surface tension, Journal of computational physics 100 (2) (1992) 335–354.

[30] P. Angot, C.-H. Bruneau, P. Fabrie, A penalization method to take into account obstacles in incompressible viscous flows, Numerische Mathematik 81 (4) (1999) 497–520.

[31] G. D. Weymouth, D. K.-P. Yue, Conservative volume-of-fluid method for free-surface simulations on cartesian-grids, Journal of Computational Physics 229 (8) (2010) 2853–2865.

[32] F. H. Harlow, J. E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, The physics of fluids 8 (12) (1965) 2182–2189.

[33] S. V. Patankar, Numerical heat transfer and fluid flow, CRC press, 2018.

[34] C. Hirsch, Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics, Elsevier, 2007.

[35] X. S. Li, J. W. Demmel, Super lu-dist: A scalable distributed-memory sparse direct solver for unsymmetric linear systems, ACM Trans. Math. Softw. 29 (2) (2003) 110–140. doi:10.1145/779359.779361.
URL https://doi.org/10.1145/779359.779361

[36] S. Vincent, Contribution à la modélisation et à la simulation numérique d'écoulements diphasiques de fluides non miscibles, Ph.D. thesis, Habilitation à Diriger des Recherches de l'Université Bordeaux 1 (2010).

[37] H. Elman, D. Silvester, A. Wathen, Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics, Oxford University Press, 2014. doi:10.1093/acprof:oso/9780199678792.001.0001.
URL https://doi.org/10.1093/acprof:oso/9780199678792.001.0001

[38] K. S. Miller, On the inverse of the sum of matrices, Mathematics magazine 54 (2) (1981) 67–72.

[39] H. Elman, D. Silvester, Fast nonsymmetric iterations and preconditioning for navier–stokes equations, SIAM Journal on Scientific Computing 17 (1) (1996) 33–46.

[40] J. Cahouet, J.-P. Chabard, Some fast 3d finite element solvers for the generalized stokes problem, International Journal for Numerical Methods in Fluids 8 (8) (1988) 869–895.

[41] H. C. Elman, Preconditioning for the steady-state navier–stokes equations with low viscosity, SIAM Journal on Scientific Computing 20 (4) (1999) 1299–1316.

[42] D. Kay, D. Loghin, A. Wathen, A preconditioner for the steady-state navier–stokes equations, SIAM Journal on Scientific Computing 24 (1) (2002) 237–256.

[43] M. Cai, A. Nonaka, J. B. Bell, B. E. Griffith, A. Donev, Efficient variable-coefficient finite-volume stokes solvers, Communications in Computational Physics 16 (5) (2014) 1263–1297.

[44] H. C. Elman, R. S. Tuminaro, Boundary conditions in approximate commutator preconditioners for the navier-stokes equations, Electronic Transactions on Numerical Analysis 35 (2009) 257–280.

[45] J.-L. Estivalezes, W. Aniszewski, F. Auguste, Y. Ling, L. Osmar, J.-P. Caltagirone, L. Chirco, A. Pedrono, S. Popinet, A. Berlemont, et al., A phase inversion benchmark for multiscale multiphase flows, Journal of Computational Physics 450 (2022) 110810.

[46] M. Saeedipour, S. Vincent, J.-L. Estivalezes, Toward a fully resolved volume of fluid simulation of the phase inversion problem, Acta Mechanica 232 (7) (2021) 2695–2714.

[47] J. Adelsberger, P. Esser, M. Griebel, S. Groß, M. Klitz, A. Rüttgers, 3d incompressible two-phase flow benchmark computations for rising droplets, in: Proceedings of the 11th world congress on computational mechanics (WCCM XI), Barcelona, Spain, Vol. 179, 2014, p. 274–5285.

[48] R. Clift, J. R. Grace, M. E. Weber, Bubbles, drops, and particles, Academic Press (2005).

[49] J.-P. Caltagirone, S. Vincent, Sur une méthode de pénalisation tensorielle pour la résolution des équations de navier–stokes, Comptes Rendus de l'Académie des Sciences-Series IIB-Mechanics 329 (8) (2001) 607–613.