High-order finite-element framework for the efficient simulation of multifluid flows

Thibaut Métivet, Vincent Chabannes, Mourad Ismail, Christophe Prud'Homme

MDPI

Article

High-order finite-element framework for the efficient simulation of multifluid flows.

Thibaut Metivet [1,2,†,*], Vincent Chabannes [1], Mourad Ismail [2] and Christophe Prud'homme [1]

[1] IRMA - UMR 7501, Cemosis - Center of Modeling and Simulation; Université de Strasbourg/CNRS; Strasbourg, F-67000, France
[2] LIPhy - UMR 5588 Laboratoire Interdisciplinaire de Physique; Université Grenoble Alpes/CNRS; Grenoble, F-38041, France
* metivet@math.unistra.fr
† Corresponding author

Version August 31, 2018 submitted to Mathematics

Abstract: We present in this paper a comprehensive framework for the simulation of multifluid flows based on the implicit level-set representation of interfaces and on an efficient solving strategy of the Navier-Stokes equations. The mathematical framework relies on a modular coupling approach between the level-set advection and the fluid equations. The space discretization is performed with possibly high-order stable finite elements while the time discretization features implicit Backward Differentation Formulae of arbitrary order. This framework has been implemented within the Feel++ library, and features seamless distributed parallelism with fast assembly procedures for the algebraic systems and efficient preconditioning strategies for their resolution. We also present simulation results for a three-dimensional multifluid benchmark, and highlight the importance of using high-order finite elements for the level-set discretization for problems involving the geometry of the interfaces, such as the curvature or its derivatives.

Keywords: multifluid flows; level-set method; high-order finite elements; Navier-Stokes equations; finite-element toolbox; parallel computing

1. Introduction

Understanding and predicting the dynamics of systems consisting of multiple immiscible fluids in contact is a great challenge for numerical computations, as they involve bulk coupling of the fluids – related to the long-range features of the Navier-Stokes equations – and possible strong surface effects.

Such systems are ubiquitous in the physics world, ranging from simple drops immersed in or impacting another fluid, to fluid or gaz mixing in climate or engineering simulations. A lot of efforts has been thus recently put in the development of efficient numerical methods to solve these strongly coupled fluid problems while tracking the interfaces to accurately account for the surface effects.

One of the main difficulties of these multifluid simulations is to keep track of the interfaces between the fluids as they evolve in time, as well as to characterize accurately their geometry. Two main approaches are used to account for these interface changes. In the first one, the interface is tracked explicitely with a moving mesh. This large class of "front-tracking" methods – which embrace for example the Arbitrary Lagrangian-Eulerian (ALE [1]), the Fat Boundary Method (FBM, [2]) – feature a very accurate description of the contact surface, but at high computational costs. The second approach uses only a fixed mesh, and represents the interface implicitely with some additional field. This is the approach used by *e.g.* the Volume Of Fluid (VOF [3]), the Phase-Field method ([4,5]) and the Level-Set

method (LS [6]). These methods in general provide easier coupling formulations and smaller algebraic problems but with less accuracy regarding the description of the interface.

In this work, we present a level-set based framework for the simulation of generic multifluid flows. It features efficient solving strategies for the fluid – incompressible Navier-Stokes – equations and the level-set advection. This framework is implemented using the open-source FEEL++ library – Finite Element Embedded Library in C++ – [7–9] within the **`FluidMechanics`**, **`AdvectionDiffusionReaction`**, **`LevelSet`** and **`MultiFluid`** toolboxes.

These toolboxes expose user-friendly interfaces and allow versatile parametrization of the problems and solving strategies while managing the parallel assembly of the finite-element algebraic systems and their resolution seamlessly.

We also present a 3D benchmark for our toolboxes. This numerical experiment is an extension of the classic rise of a drop in a viscous fluid in 2D. We use two different setups and compare our results with other approaches to validate our framework.

We stress that all the implementations and testcases presented below are freely available [10,11] and can be used for a large class of fluid-structure interaction and suspension problems.

2. Simulating multifluid flows

We consider a system consisiting of several non-miscible fluids with different physical properties occupying some domain Ω. We denote $\Omega_i \subset \Omega$ ($\Omega = \bigcup_i \Omega_i$) the domain occupied by the i^{th} fluid. We want to study the dynamics of such a system, which means both solving for the fluids motion and tracking each fluid subdomain as it evolves in time.

2.1. Fluid equations

In this work, we assume that all the considered fluids are incompressible, and thus obey the incompressible Navier-Stokes equations

$$\rho\left[\partial_t \vec{u} + (\vec{u}\cdot\nabla)\vec{u}\right] - \nabla\cdot\sigma(\vec{u},p) = \vec{f_b} + \vec{f_s} \tag{1}$$

$$\nabla\cdot\vec{u} = 0 \tag{2}$$

where $\vec{u}$ and p are respectively the velocity and pressure fields, ρ is the fluid density, σ is the fluid stress tensor and $\vec{f_b}$, $\vec{f_s}$ are the bulk and surface forces respectively. We consider in the following Newtonian fluids, so that

$$\sigma(\vec{u},p) = -p\,\mathbf{I} + 2\mu\,\boldsymbol{D}(\vec{u}) \tag{3}$$

with μ the fluid viscosity and $\boldsymbol{D}(\vec{u}) = \frac{1}{2}\left(\nabla\vec{u} + \nabla\vec{u}^T\right)$ is the fluid strain tensor.

Note that these equations are satisfied for each fluid independently. In a multifluid system, it must be supplemented with boundary conditions for the velocity and pressure at both the domain boundary and at the interface between two fluids. In this work, we assume that there is no slip at the interfaces, so that the velocity and pressure fields are continuous at all the interfaces. The boundary conditions at the domain boundaries can be Dirichlet or Neumann:

$$\begin{aligned} \vec{u} &= \vec{g}_D \quad \text{on } \partial\Omega_D \\ \sigma(\vec{u},p)\,\vec{n} &= \vec{g}_N \quad \text{on } \partial\Omega_N \end{aligned} \tag{4}$$

2.2. Interface tracking: the level-set method

Let us consider two disjoint – not necessarily connected – fluid domains Ω_i, $\Omega_j \subset \Omega$ such that $\partial\Omega_i \cap \partial\Omega_j \neq \emptyset$, and denote Γ_{ij} the interface between them. In order to efficiently track this interface $\Gamma_{ij}(t)$ as it evolves in time, we use the level set method [6,12,13] which provides a natural way to compute geometrical properties of the interface and to handle possible topological changes in a completely Eulerian framework.

It features a Lipschitz continuous scalar function ϕ (the *level set* function) defined on the whole domain. This function is arbitrarily chosen to be positive in Ω_i, negative in Ω_j and zero on Γ_{ij}, so that the interface is implicitely represented by the 0-level of ϕ.

As Ω_i and Ω_j evolve following the Navier-Stokes eq. (2) dynamics, Γ_{ij} gets transported by the velocity field $\boldsymbol{u}$ (uniquely defined at the interface by assumption and construction) and therefore obeys the advection equation:

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u} \cdot \nabla \phi = 0. \tag{5}$$

We choose as level set function the signed distance to the interface

$$\phi(\boldsymbol{x}) = \begin{cases} \operatorname{dist}(\boldsymbol{x}, \Gamma_{ij}) & \boldsymbol{x} \in \Omega_i, \\ 0 & \boldsymbol{x} \in \Gamma_{ij}, \\ -\operatorname{dist}(\boldsymbol{x}, \Gamma_{ij}) & \boldsymbol{x} \in \Omega_j, \end{cases} \tag{6}$$

as the intrinsic property $|\nabla \phi| = 1$ eases the numerical resolution of the advection equation and the regularity of the distance function allows us to use ϕ as a support for the regularized interface Dirac and Heaviside functions (see below).

However, the advection equation (5) does not preserve the property $|\nabla \phi| = 1$ and it is necessary to reset $\phi(t)$ to a distance function without moving the interface, [14–16]. To redistantiate $\phi(t)$ and enforce $|\nabla \phi(t)| = 1$, we can either solve a Hamilton-Jacobi equation, which "transports" the isolines of ϕ to their proper positions, or use the fast marching method, which resets the values of ϕ to the distance to the interface from one degree of freedom to the next, starting from the interface. We provide further details about our numerical implementation below (c.f. section 3.5.3).

As mentioned above, the signed distance to the interface ϕ allows us to easily define the regularized interface-related Dirac and Heaviside functions which can be used to compute integrals on the interface [17,18] or on a fluid subdomain

$$\delta_\varepsilon(\phi) = \begin{cases} 0, & \phi \leq -\varepsilon, \\ \frac{1}{2\varepsilon}\left[1 + \cos\left(\frac{\pi\phi}{\varepsilon}\right)\right], & -\varepsilon \leq \phi \leq \varepsilon, \\ 0, & \phi \geq \varepsilon. \end{cases} \tag{7}$$

$$H_\varepsilon(\phi) = \begin{cases} 0, & \phi \leq -\varepsilon, \\ \frac{1}{2}\left[1 + \frac{\phi}{\varepsilon} + \frac{\sin(\frac{\pi\phi}{\varepsilon})}{\pi}\right], & -\varepsilon \leq \phi \leq \varepsilon, \\ 1, & \phi \geq \varepsilon. \end{cases} \tag{8}$$

where ε is a parameter controlling the "numerical thickness" of the interface. We note in these definitions that enforcing $|\nabla \phi(t)| = 1$ is critical to ensure that the interfacial support of δ_ε and H_ε is kept constant and larger than the mesh size. Typically, we choose $\varepsilon \sim 1.5\, h$, with h the average mesh size of the elements crossed by the 0 iso-value of ϕ.

The Heaviside function is used to define physical quantities which have different values on each subdomain. For example, we define the density and viscosity of two-fluid flows as $\rho = \rho_j + (\rho_i - \rho_j) H_\varepsilon(\phi)$ and $\mu = \mu_j + (\mu_i - \mu_j) H_\varepsilon(\phi)$. The delta function allows to define quantities on the interface, in particular in the variational formulations, where we replace integrals over the interface Γ with integrals over the entire domain Ω using the smoothed delta function: if ϕ is a signed distance function (i.e. $|\nabla \phi| = 1$), we have $\int_\Gamma 1 \simeq \int_\Omega \delta_\varepsilon(\phi)$.

In the following, we consider without loss of generality only the two-fluid case, with only one level set function tracking the interface between fluids 1 and 2.

2.3. Finite element formulation

We use finite elements methods to solve eqs. (2) and (5), and work with a continous Galerkin formulation. As mentioned above, this can be done by smoothing the discontinuities of the fluid parameters (e.g. the fluid density and viscosity) at the interfaces using the regularized Dirac and Heaviside functions eqs. (7) and (8). The continuity of the velocity and pressure fields is then imposed strongly in the formulation, and we can work with function spaces defined on the whole domain Ω.

We introduce the usual $L^2(\Omega)$ function space of square integrable functions, $H^1(\Omega)$ function space of square integrable functions as well as their gradients and the vectorial Sobolev spaces $\mathbf{H}^1(\Omega) = \left\{\vec{v} \in \left[H^1(\Omega)\right]^d\right\}$, $\mathbf{H}^1_{g_D}(\Omega) = \left\{\vec{v} \in \mathbf{H}^1(\Omega), \vec{v}|_{\partial\Omega_D} = \vec{g}_D\right\}$ and $\mathbf{H}^1_{0_D}(\Omega) = \left\{\vec{v} \in \mathbf{H}^1(\Omega), \vec{v}|_{\partial\Omega_D} = \vec{0}\right\}$.

The variational formulation of the two-fluid coupling problem eqs. (2) and (5) then reads

$$\textit{Find } (\vec{u}, p, \phi) \in \mathbf{H}^1_{g_D}(\Omega) \times L^2(\Omega) \times H^1(\Omega) \textit{ s.t. } \forall (\vec{v}, q, \psi) \in \mathbf{H}^1_{0_D}(\Omega) \times L^2(\Omega) \times H^1(\Omega),$$

$$\int_\Omega \rho(\phi)\left[\partial_t \vec{u} + (\vec{u}\cdot\nabla)\vec{u}\right]\cdot\vec{v} + \int_\Omega \sigma(\vec{u}, p; \mu(\phi)) : \nabla\vec{v} = \int_\Omega \vec{f}_b \cdot \vec{v} + \int_\Omega \vec{f}_s(\phi)\cdot\vec{v} + \int_{\partial\Omega_N} \vec{g}_N \cdot \vec{v} \tag{9}$$

$$\int_\Omega (\nabla\cdot\vec{u})\ q = 0 \tag{10}$$

$$\int_\Omega (\partial_t\phi + \vec{u}\cdot\nabla\phi)\ \psi = 0 \tag{11}$$

where we have integrated by parts the stress tensor term and used the Neumann boundary condition in eq. (4). The surface force term $\int_\Omega \vec{f}_s \cdot \vec{v}$ is here generically written as a bulk integral, the surfacic aspect being hidden in the force expression $\vec{f}_s(\phi)$ which in general contains some Dirac distribution $\delta(\phi)$. In our case, the surface forces actually account for the interfacial forces between the two fluids (i.e. the surface tension), so that the surface integral can be evaluated as a bulk integral using the regularized level set delta function.

3. Numerical setup

3.1. The Feel++ *toolboxes*

The numerical implementation is performed using the Feel++ — finite element C++ library — [7–9]. Feel++ allows to use a very wide range of Galerkin methods and advanced numerical methods such as domain decomposition methods including mortar and three fields methods, fictitious domain methods or certified reduced basis. The ingredients include a very expressive embedded language, seamless interpolation, mesh adaption and seamless parallelization. It has been used in various contexts including the development and/or numerical verification of (new) mathematical methods or the development of large multi-physics applications [19–21]. The range of users span from mechanical engineers in industry, physicists in complex fluids, computer scientists in biomedical applications to applied mathematicians thanks to the shared common mathematical embedded language hiding linear algebra and computer science complexities.

Feel++ provides a mathematical kernel for solving partial differential equation using arbitrary order Galerkin methods (FEM, SEM, CG, DG, HDG, CRB) in 1D, 2D, 3D and manifolds using simplices and hypercubes meshes [7–9,22] :

i. a polynomial library allowing for a wide range polynomial expansions including H_{div} and H_{curl} elements,
ii. a lightweight interface to Boost.UBlas, Eigen3 and PETSc/SLEPc as well as a scalable in-house solution strategy
iii. a language for Galerkin methods starting with fundamental concepts such as function spaces, (bi)linear forms, operators, functionals and integrals,

iv. a framework that allows user codes to scale seamlessly from single core computation to thousands of cores and enables hybrid computing.

Feel++ provides also an environment for modeling and solving various kinds of scientific and engineering problems. The framework, implemented in several Feel++ toolboxes, provides a language to describe these models. Based on JSON and INI(we use the `.cfg` extension) file formats, it is possible to configure and simulate a large class of models by defining the relevant physical quantities – such as the material properties, the boundary conditions, the sources, the couplings and the solvers. In this paper, we have used the **`MultiFluid`** toolbox which allows to setup all the necessary ingredients for a multifluid flow simulation. This toolbox, which mainly manages the coupling between the fluids present in the system, is built on others toolboxes inclined toward monophysics problem resolution, namely the **`FluidMechanics`**, **`LevelSet`** and **`AdvectionDiffusionReaction`** toolboxes.

3.2. Fluid-interface coupling

The coupling between the fluid and the level-set in eqs. (9) to (11) is highly non-linear and solving these equations monolithically would require specific non-linear solvers adapted to each particular coupling force $\vec{f}_s(\phi)$. In order to ease the implementation and development processes, and to benefit from efficient solving strategies, we choose an explicit – non-monolithic – coupling in our numerical approach.

At each time step, the fluid equations are first solved with the physical parameters and the surface forces computed using the last-step level-set function. We then use the obtained fluid velocity to advect the level-set and get the new interface position. We can then apply some redistantiation procedure depending on the chosen strategy before proceeding to the next iteration.

The fluid-level-set coupling algorithm then writes

for $n = 0$ **to** N_t **do**
  update $\delta^{n+1} \leftarrow \delta(\phi^n)$, $H^{n+1} \leftarrow H(\phi^n)$;
  update $\rho^{n+1} \leftarrow \rho(\phi^n)$, $\mu^{n+1} \leftarrow \mu(\phi^n)$;
  update $\vec{f}_s^{n+1} \leftarrow f(\phi^n)$;
  $\vec{u}^{n+1}$, $p^{n+1} \leftarrow$ solve fluid;
  $\phi^{n+1} \leftarrow$ solve level-set;
  possibly redistantiate ϕ^{n+1};
end

Note that the successive resolution of the fluid and level-set equations can also be iterated within one time step, until a fix point of the system of equations is reached. In practice however, for reasonably small time steps, the fix-point solution is already obtained after the first iteration.

3.3. Space/Time discretization

We introduce $\mathcal{T}_h \equiv \{K_e, 1 \leq e \leq N_{elt}\}$ a compatible tessellation of the domain Ω, and denote $\Omega_h = \bigcup_{e=1}^{N_{elt}} K_e$ the discrete – unstructured – mesh associated with average mesh size h. We work within the continuous Galerkin variational formulation framework, and use Lagrange finite elements to spatially discretize and solve the equations governing the evolutions of the fluid and the level-set. We thus introduce $\mathcal{P}_h^k \equiv \mathcal{P}_h^k(\Omega_h)$ the discrete (h-dependent) finite element space spanned by Lagrange polynomials of order k.

Then, from the time interval $[0, T]$, we select $M + 1$ equidistributed times: $\{t_i = i\ \delta t, 0 \leq i \leq M\}$, where δt is the time step. In the Navier-Stokes and level-set advection equations, we apply a fully implicit time discretization by using a backward differentiation formula of arbitrary order N named

BDF_N. Let ϕ be a function and denote $\phi^{(i)}$ the function at time t_i. The time derivative of this function is then discretized as

$$\partial_t \phi = \frac{1}{\delta t}\left[\alpha_0 \phi^{(n+1)} - \sum_{i=1}^{N} \alpha_i \phi^{(n+1-i)}\right] + \mathcal{O}(\delta t^{N+1}) \tag{12}$$

where the α_i are BDF_N coefficients. In the numerical benchmarks reported below, we used BDF_1 and BDF_2, which write respectively

$$\text{BDF}_1 : \quad \partial_t \phi \approx \frac{1}{\delta t}\left[\phi^{(n+1)} - \phi^{(n)}\right] \tag{13}$$

$$\text{BDF}_2 : \quad \partial_t \phi \approx \frac{1}{\delta t}\left[\frac{3}{2}\phi^{(n+1)} - 2\phi^{(n)} + \frac{1}{2}\phi^{(n-1)}\right] \tag{14}$$

3.4. Solving the incompressible Navier-Stokes equations

The spatial discretization of the Navier-Stokes equations is handled via a inf-sup stable finite element (Taylor-Hood) $[\mathcal{P}_h^{k+1}]^d$-$\mathcal{P}_h^k$, see *e.g.* [23]. We define V_h and Q_h the discrete function spaces where we search the velocity and the pressure solutions respectively. There are given by :

- $V_h = \left\{\vec{v} \in \mathbf{H}^1_{g_D} \cap [\mathcal{P}_h^{k+1}]^d\right\}$
- $Q_h = \left\{q \in \mathcal{P}_h^k\right\}$

We need also to introduce the test function space on the velocity with $W_h = \left\{\vec{v} \in \mathbf{H}^1_{0_D}(\Omega_h) \cap [\mathcal{P}_h^{k+1}]^d\right\}$. Hereafter we consider the case $k = 1$. The discrete weak formulation associated to (9)-(10) reads:

Find $(\vec{u}_h^{(n+1)}, p_h) \in V_h \times Q_h$ *such that* $\forall (\vec{v}_h, p_h) \in W_h \times Q_h$:

$$\begin{aligned}\int_{\Omega_h} \rho(\phi)\left[\frac{\alpha_0}{\delta t}\vec{u}_h^{(n+1)} + \left(\vec{u}_h^{(n+1)} \cdot \nabla\right)\vec{u}_h^{(n+1)}\right] \cdot \vec{v}_h + \int_{\Omega_h} \sigma(\vec{u}_h^{n+1}, p_h^{n+1}; \mu(\phi)) : \nabla \vec{v}_h + \int_{\Omega_h}\left[\nabla \cdot \vec{u}_h^{(n+1)}\right] q_h \\ = \int_{\Omega_h} \sum_{i=1}^{N} \frac{\alpha_i}{\delta t}\vec{u}^{(n+1-k)} \cdot \vec{v}_h + \int_{\Omega_h} \vec{f}_b \cdot \vec{v}_h + \int_{\Omega_h} \vec{f}_s(\phi) \cdot \vec{v}_h + \int_{\partial\Omega_N} \vec{g}_N \cdot \vec{v}\end{aligned} \tag{15}$$

The non-linear problem represented by (15) is solved monolithically with Newton's method. From an algebraic point of view, at each non-linear iteration, the following classical saddle-point system is inverted

$$\begin{pmatrix} A & B \\ C & 0 \end{pmatrix}\begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{16}$$

where A corresponds to the velocity block and B, C to the velocity/pressure coupling. This system is solved with GMRES using the SIMPLE preconditioner, see [23].

In the benchmark presented at next section, we have only Dirichlet boundary condition type defined on the whole boundary of the domain (i.e. $\partial\Omega_N \equiv \emptyset$). In this case, the problem represented by the weak formulation (15) is not well-posed, the pressure is defined up to a constant. For solve this issue, our strategy consist to

1. add the information to the Krylov subspace method (GMRES) that the system has a null space, i.e. the pressure constant.
2. rescale the pressure solution after each iteration of the Newton algorithm by imposing a mean pressure equal to 0.

Another solution available in **`FluidMechanics`** toolbox is to add a Lagrange multiplier in order to impose the mean of the pressure. The disadvantages of this approach are to increase the stencil of the matrix and to complexify its block structure. This implies a reduction in the efficiency of the solver.

3.5. The level-set framework

3.5.1. Level-set advection

As mentioned above, the transport of the level-set by the fluid is accounted by the advection equation eq. (11) in variational form. This equation is discretized in time using a backward differentiation formula (BDF_N) and space discretization is performed using the $\mathcal{P}_h^k$ discrete spaces introduced above. However, as is well-known for central differencing schemes in hyperbolic partial differential equations, the *naive* Galerkin discretization of eq. (11) on $\mathcal{P}_h^k$ can lead to spurious oscillatory instabilities. To circumvent this well-known problem, we stabilize the discrete advection equation. Four different stabilization methods are supported in our framework: the Streamline Upwind Petrov-Galerkin (SUPG, [24]), the Galerkin Least Squares (GLS, [25,26]), the Sub-Grid Scale (SGS, [27]) and the Continuous Interior Penalty (CIP, [28]) methods. Detailed description of these methods can be found in [29–31]. It should be noted that the CIP stabilization is much more costly than the three others, as it densifies the corresponding algebraic system, and requires larger stencils and thus larger connectivity tables.

In the benchmarks, we used the GLS stabilization method, introducing the bilinear form

$$S_{GLS} = \sum_{K\in\mathcal{T}_h} \int_K \tau\, \mathcal{L}[\psi] \left(\mathcal{L}[\phi] - \sum_{i=1}^{N} \alpha_i \phi^{(n+1-i)} \right) \tag{17}$$

with $\mathcal{L}[\phi] = \frac{\alpha_0}{\delta t}\phi + \vec{u}\cdot\nabla\phi$, τ a coefficient chosen to adjust the stabilization to the local advection strength. The choice for the parameter τ was extensively discussed, in particular for the case of advection-diffusion-reaction (c.f. [27,32,33]), and we provide several of them in our `AdvectionDiffusionReaction` toolbox. However, for the specific case of pure transient advection, which can also be seen as an advection-reaction equation after time discretization, we use the simple expression

$$\tau = \frac{1}{\frac{2|\vec{u}|}{h} + \frac{2\alpha_0}{\delta t}} \tag{18}$$

In summary, the discrete FEM we solve for the level-set at time $t+\delta t$ given the values at previous times is

Find $\phi^{(n+1)} \in \mathcal{P}_h^k$ *s.t.* $\forall \psi \in \mathcal{P}_h^k$,

$$\int_{\Omega_h} \left[\frac{\alpha_0}{\delta t}\phi^{(n+1)} + \vec{u}^{(n+1)}\cdot\nabla\phi^{(n+1)} \right] \psi + S\left(\phi^{(n+1)}, \psi; \vec{u}^{(n+1)}, \{\phi^{(i)}\}_{i\le n}\right) = \int_{\Omega_h} \sum_{i=1}^{N} \frac{\alpha_i}{\delta t}\phi^{(n+1-i)}\,\psi \tag{19}$$

where $S\left(\phi^{(n+1)}, \psi; \vec{u}^{(n+1)}, \{\phi^{(i)}\}_{i\le n}\right)$ is the SUPG, GLS, SGS or CIP stabilization bilinear, which vanishes as $h\to 0$ and in the three first cases involves the previous time steps to ensure consistency of the stabilized equation with its continuous version.

The assembly is performed in parallel with automatic choice of the appropriate quadrature order and seamless inter-process communication management.

The linear system is then solved using the PETSc library [34] with a GMRES solver preconditioned with an additive Schwartz Method (GASM [35]).

194 3.5.2. Geometrical quantities

The level-set description of interfaces is very convenient when it comes to compute geometrical parameters required in the interface forces. As an example, the normal and curvature of the surface can naturally be obtained from ϕ as

$$\vec{n} = \frac{\nabla\phi}{|\nabla\phi|} \tag{20}$$

$$\kappa = \nabla\cdot\vec{n} \tag{21}$$

195 These quantities are defined on the whole domain in an Eulerian way, and can therefore naturally be
196 used in surface integrals as introduced in eq. (9).

To compute such derivatives of fields, we use projection operators in order to work with fields still in $\mathcal{P}_h^k$. To this end, we introduce the L^2 projection operator $\Pi^k_{L^2}$ and the smoothing projection operator Π^k_{sm}, defined respectively as

$$\begin{aligned}\Pi^k_{L^2} &: V_h \to \mathcal{P}_h^k \\ &u \mapsto u_{L^2} \equiv \Pi^k_{L^2}(u) \quad \text{s.t.} \quad \forall v \in \mathcal{P}_h^k \\ &\int_{\Omega_h} u_{L^2}\, v = \int_{\Omega_h} u\, v\end{aligned} \tag{22}$$

197 and

$$\begin{aligned}\Pi^k_{sm} &: V_h \to \mathcal{P}_h^k \\ &u \mapsto u_{sm} \equiv \Pi^k_{sm}(u) \quad \text{s.t.} \quad \forall v \in \mathcal{P}_h^k \\ &\int_{\Omega_h} u_{sm}\, v + \varepsilon \int_{\Omega_h} \nabla u_{sm} \colon \nabla v - \varepsilon \int_{\partial\Omega_h} \left(\vec{N}\cdot\nabla u_{sm}\right) v = \int_{\Omega_h} u\, v\end{aligned} \tag{23}$$

198 with ε a – small – smoothing parameter typically chosen as $\varepsilon \approx 0.03h/k$. These projection operators
199 can be defined for both scalar and vector fields in the same way by using the appropriate contractions.
200 Note that the smoothing projection operator introduces some artificial diffusion which is controlled
201 by ε. This diffusion is reponsible for the smoothing of the projected field, but can also introduce
202 artefacts in the computation, so that ε needs to be carefully chosen depending on the simulation. In
203 practice, the smoothing operator Π^k_{sm} is used to compute derivatives of order $\geq k+1$ of fields in $\mathcal{P}_h^k$,
204 as such order of derivation are subject to noise.
205 The projection operators are implemented by the `Projector` class, which optimizes the assembly
206 of the algebraic systems corresponding to eqs. (22) and (23) by storing the constant terms – such as
207 the mass matrix for example – to prevent unnecessary computations at each projection. We usually
208 also store the preconditioner of the system for reuse.
209 The projection linear systems are solved using PETSc's GMRES solver with a GASM
210 preconditioning method.
211 In the benchmarks below, we used $\vec{n} = \Pi^1_{L^2}\left(\frac{\nabla\phi}{|\nabla\phi|}\right)$ and $\kappa = \Pi^1_{sm}\left(\nabla\cdot\vec{n}\right)$ for $\mathcal{P}_h^1$ simulations, and
212 $\vec{n} = \Pi^1_{L^2}\left(\frac{\nabla\phi}{|\nabla\phi|}\right)$ and $\kappa = \Pi^1_{L^2}\left(\nabla\cdot\vec{n}\right)$ for higher order ones.

213 3.5.3. Redistantiation

214 As mentioned above, as the level-set is advected by the fluid, it loses its "distance" property, i.e.
215 $|\nabla\phi| = 1$ is not in general preserved by the advection equation. Therefore, to ensure numerical stability
216 and prevent accumulation or rarefaction of the level-set iso-lines which support the regularized Dirac
217 and Heaviside functions, one needs to redistantiate the level-set, i.e. recover a signed distance function
218 to $\{\phi = 0\}$. Too main approaches can be used: the first one relies on a efficient direct computation of

the distance using the well-known "fast-marching algorithm" [16], while the second consists in solving an Hamilton-Jacobi equation [14,15] which steady state enforces $|\nabla\phi| = 1$. Both methods are implemented in our framework; we provide some details in the following.

The fast-marching method is an efficient algorithm to solve the Eikonal equation in general, with an algorithmic complexity of $\mathcal{O}\left(N_{dof}\log(N_{dof})\right)$. It uses the upwind nature of this equation to "march" away from the 0-iso-level to the rest of the domain, hence the Dijkstra-like complexity. Our implementation of the fast-marching algorithm is based on [31], and therefore strongly relies on ϕ being a $\mathcal{P}_h^1$ field.

If the level-set is of order $k > 1$, we can still perform the fast-marching redistantiation using a $\mathcal{P}_{iso}^1$ space as proposed in [36]. This order-1 Lagrange function space is constructed on a mesh obtained by replacing all the $\mathcal{P}_h^k$ degrees of freedom by nodes. The $\mathcal{P}_{iso}^1$ hence has the same number of degrees of freedom than the original $\mathcal{P}_h^k$ one. We can then project $\phi^k \in \mathcal{P}_h^k$ onto $\phi_{iso}^1 \in \mathcal{P}_{iso}^1$, perform the fast-marching on ϕ_{iso}^1, and project back.

The Hamilton-Jacobi method follows a different approach, and works directly within the finite-element framework. It consists in solving an Hamilton-Jacobi-like equation which steady states is a signed distance function, namely

$$\partial_\tau\phi + sign(\phi)\left(1 - |\nabla\phi|\right) = 0 \tag{24}$$

The sign(ϕ) term anchors the position of $\{\phi = 0\}$, and ensures that the redistantiation front gets transported from this 0-iso-level, inward or outward depending on the initial sign of ϕ. In practice, this equation is discretized using the advection framework introduced above as

Find $\phi^{(n+1)} \in \mathcal{P}_h^k$ *s.t.* $\forall\psi \in \mathcal{P}_h^k$

$$\int_{\Omega_h}\left[\frac{\alpha_0}{\delta\tau}\phi^{(n+1)} + sign(\phi^{(0)})\frac{\nabla\phi^{(n)}}{|\nabla\phi^{(n)}|}\cdot\nabla\phi^{(n+1)}\right]\psi + S\left(\phi^{(n+1)}\right) = \int_{\Omega_h}\left[sign(\phi^{(0)}) + \sum_{i=1}^{N}\frac{\alpha_i}{\delta\tau}\,\phi^{(n+1-i)}\right]\psi \tag{25}$$

with $sign(\phi^{(0)}) \equiv 2\,H_\epsilon(\phi^{(0)}) - \frac{1}{2}$ and $S\left(\phi^{(n+1)}\right)$ a stabilization bilinear form. This equation is usually solved for a few iterations with $\delta\tau \sim h$. The main advantage of the Hamilton-Jacobi approach is that it can be used straigthforwardly for high-order level-set functions. However, it is in general slower than the fast-marching method, and numerical errors can quickly accumulate and lead to spurious motion of the interface resulting in strong loss of mass of the $\{\phi \leq 0\}$ domain. To avoid these pitfalls, the number of iterations and the pseudo time-step $\delta\tau$ must be carefully chosen, which is not an easy task in general.

4. 3D rising drops benchmark

We now present a 3D benchmark of our numerical approach, using the Navier-Stokes solver developed with the `Feel++` library described in [37]. This benchmark is a three-dimensional extension of the 2D benchmark introduced in [38] and realised using `Feel++` in [39]. The setup for this benchmark was also used in [40] to compare several flow solvers.

4.1. Benchmark problem

The benchmark consists in simulating the rise of a 3D drop in a Newtonian fluid. The equations solved are the aforementioned coupled incompressible Navier Stokes equations for the fluid and advection equation for the level set eqs. (9) to (11) with $\vec{f_b}$ and $\vec{f_s}$ respectively the gravitational and surface tension forces, defined as:

$$\vec{f_b} = \rho_\phi\,\vec{g} \tag{26}$$

$$\vec{f_s} = \sigma\kappa\vec{n}\Big|_\Gamma \simeq \sigma\kappa\vec{n}\,\delta_\varepsilon(\phi) \tag{27}$$

with $\vec{g} \equiv -0.98\,\vec{e}_z$ the gravity acceleration and σ the surface tension.

We consider Ω a cylinder with radius $R = 0.5$ and height $H = 2$, filled with a fluid and containing a droplet of another imiscible fluid. We denote $\Omega_1 = \{\vec{x}, \quad s.t.\, \phi(\vec{x}) > 0\}$ the domain outside the droplet, $\Omega_2 = \{\vec{x}, \quad s.t.\, \phi(\vec{x}) < 0\}$ the domain inside the drop and $\Gamma = \{\vec{x}, \quad s.t.\, \phi(\vec{x}) = 0\}$ the interface. We impose no-slip boundary conditions $\vec{u}\big|_{\partial\Omega} = 0$ on Ω walls. The simulation is run from $t = 0$ to 3.

Initially, the drop is spherical with radius $r_0 = 0.25$ and is centered on the point $(0.5, 0.5, 0.5)$ assuming that the bottom disk of the Ω cylinder is centered at the origin. Figure 1 shows this initial setup.

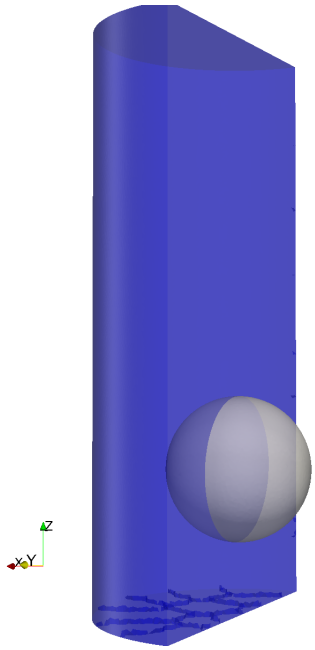


Figure 1. Initial setup for the benchmark.

We denote with indices 1 and 2 the quantities relative to the fluid in respectively Ω_1 and Ω_2. The parameters of the benchmark are then ρ_1, ρ_2, μ_1, μ_2 and σ. We also define two dimensionless numbers to characterize the flow: the Reynolds number which is the ratio between inertial and viscous terms and is defined as

$$\mathrm{Re} = \frac{\rho_1 \sqrt{|\vec{g}|(2r_0)^3}}{\nu_1},$$

and the Eötvös number represents the ratio between the gravity force and the surface tension

$$\mathrm{E}_0 = \frac{4\rho_1 |\vec{g}| r_0^2}{\sigma}.$$

Table 1 reports the values of the parameters used for two different test cases proposed in [40]. At $t = 3$, the first one leads to an ellipsoidal-shaped drop while the second one gives a skirted shape due to the larger density and viscosity contrasts between the inner and outer fluids.

Table 1. Numerical parameters taken for the benchmarks.

Tests	ρ_1	ρ_2	ν_1	ν_2	σ	Re	E_0
Case 1 (ellipsoidal drop)	1000	100	10	1	24.5	35	10
Case 2 (skirted drop)	1000	1	10	0.1	1.96	35	125

To quantify our simulation results, we use three quantities characterizing the shape of the drop at each time-step: the center-of-mass

$$\vec{x}_c = \frac{1}{|\Omega_2|}\int_{\Omega_2} \vec{x},$$

the rising velocity – focusing on the vertical component

$$\vec{u}_c = \frac{1}{|\Omega_2|}\int_{\Omega_2} \vec{u},$$

and the sphericity – defined as the ratio between the area of a sphere with same volume and the area of the drop –

$$\Psi = \frac{4\pi\left(\frac{3}{4\pi}|\Omega_2|\right)^{\frac{2}{3}}}{|\Gamma|}.$$

Note that in the previous formulae, we have used the usual "mass" and area of the drop, respectively defined as $|\Omega_2| = \int_{\Omega_2} 1$ and $|\Gamma| = \int_{\Gamma} 1$.

4.2. Simulation setup

The simulations have been performed on the supercomputer of the Grenoble CIMENT HPC center up to 192 processors. To control the convergence of our numerical schemes, the simulations have been run with several unstructured meshes, which characteristics are summarized in table 2.

We run the simulations looking for solutions in finite element spaces spanned by Lagrange polynomials of order $(2, 1, k)$ for respectively the velocity, the pressure and the level set. The corresponding numbers of degrees of freedom for each mesh size are also reported in table 2.

Table 2. Mesh properties and degrees of freedom: mesh characteristic size, number of tetrahedra, number of points, number of order 1 degrees of freedom, number of order 2 degrees of freedom and total number of degrees of freedom of the simulation.

Mesh properties			Finite-element DOF		
h	Tetrahedra	Points	Order 1 DOF	Order 2 DOF	#DOF
0.025	380 125	62 546	62 546	490 300	1 595 992
0.02	842 865	136 932	136 932	1 092 644	3 551 796
0.0175	1 148 581	186 136	186 136	1 489 729	4 841 459
0.015	1 858 603	299 595	299 595	2 415 170	7 844 700
0.0125	2 983 291	479 167	479 167	3 881 639	12 603 251

Table 3. Numerical parameters used for simulations and resulting simulation times for each test case.

Numerical parameters			Total time (h)	
h	#proc	Δt	Case 1	Case 2
0.025	64	1×10^{-2}	3.5	3.6
0.02	128	9×10^{-3}	4.8	5.1
0.0175	128	8×10^{-3}	8.9	9.5
0.015	192	7×10^{-3}	12.3	13.5
0.0125	192	6×10^{-3}	33.8	39.6

The Navier-Stokes equations are solved using Newton's method and the resulting linear system is solved with a preconditioned flexible Krylov GMRES method using the SIMPLE preconditioner introduced in [41]. The inner "inversions" of the velocity block matrix are performed using a block Jacobi preconditioner, with an algebraic multigrid (GAMG) preconditioner for each velocity component block.

Note that these velocity block inversions are only preconditioned – without any KSP iteration. The nested Schur complement "inversion" required for SIMPLE is solved with a few iterations of GMRES preconditioned with an algebraic multigrid (GAMG).

The linear advection equation is solved with a Krylov GMRES method, preconditioned with an Additive Schwarz Method (GASM) using a direct *LU* method as sub-preconditionner.

5. Results

In this section, we analyze the simulations of the rising drop for the two cases with low-order – $k = 1$ – sections 5.1 and 5.2 and high-order – $k = 2$ – section 5.3 level set discretization space. Except for the comparison between the fast-marching and the Hamilton-Jacobi methods section 5.1.1, the level set redistantiations were performed with the fast-marching method every 10 time-steps.

5.1. Case 1: the ellipsoidal drop

Figure 2a shows the shape of the drop in the $x - z$ plane at the final $t = 3$ time step for the different aforementioned mesh sizes. The shapes are similar and seem to converge when the mesh size is decreasing. The drop reaches a stationary circularity as shown in fig. 2d, and its topology does not change. The velocity increases until it attains a constant value. Figure 2c shows the results obtained for the different mesh sizes. The evolution of the mass of the drop versus time is shown in fig. 2e. It highlights the rather good mass conservation property of our simulation setup, as about 3% of the mass is at most lost for the coarsest mesh, while the finest one succeeds in keeping the loss in mass below 0.7%.

We also note that our simulation perfectly respects the symmetry of the problem and results in a axially symmetric final shape of the drop, as shown in fig. 3.

5.1.1. Comparison between Hamilton-Jacobi and fast-marching reinitialization

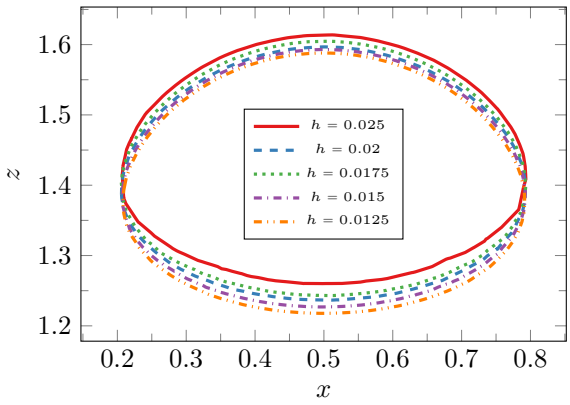
As mentioned in section 3.5.3, two reinitialization procedures can be used to overcome the "deformation" of the level set which becomes more and more different from the distance to the interface function as it is advected with the fluid velocity. The fast-marching method resets the values of ϕ on the degrees of freedom away from the interface to match the corresponding distance. The Hamilton-Jacobi method consists in solving an advection equation which steady solution is the wanted distance function.

We have run the $h = 0.0175$ simulation with both reinitialization methods to evaluate the properties of each one, and compare them using the monitored quantities. Figure 5 gives the obtained results.

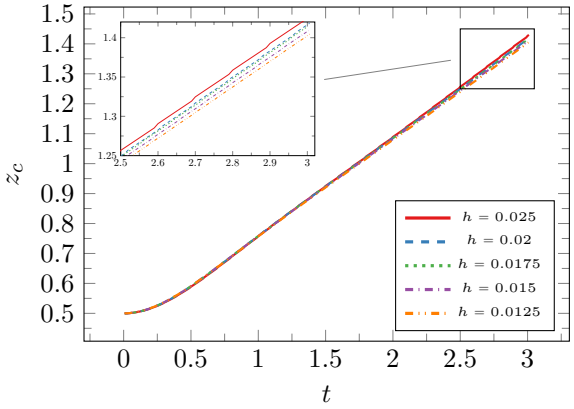
The first observation is that the mass loss (see fig. 5e) is considerably reduced when using the FM method. It goes from about 18% mass lost between $t = 0$ and $t = 3$ for the Hamilton-Jacobi method to less than 2% for the fast-marching method. This resulting difference of size can be noticed in fig. 4. The other main difference is the sphericity of the drop. Figure 5d shows that when using the fast-marching method, the sphericity decreases really quickly and stabilises to a much lower value than the one obtained with the Hamilton-Jacobi method. This difference can be explained by the fact that the fast-marching method does not smooth the interface. The shape can then contain some small irregularities leading to a bad sphericity. Even so, with both methods the sphericity stays quite constant after the first second of the simulation. The rising velocity and the vertical position do not show any significant difference between the two reinitialization methods.

5.1.2. Comparison with previous results

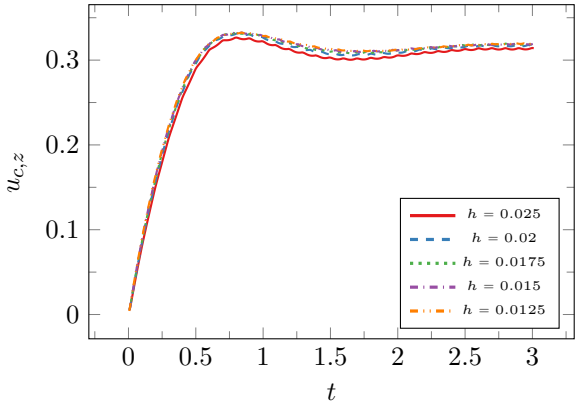
Figure 6 shows a plot of our results compared to the ones presented in [40]. In this paper, the authors perform simulations on the same setup and with the same test cases as considered here. To ensure consistency of their results, they use three different flow solvers (hence three different space discretization methods) coupled with two different interface capturing methods: the DROPS and NaSt3D solvers coupled to a level set approach, and the OpenFOAM solver which uses a volume-of-fluid method.



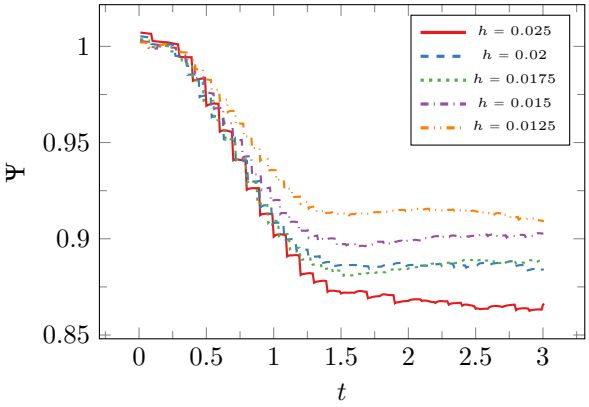
(a) Shape at final time ($t = 3$) in the vertical $x - z$ plane.



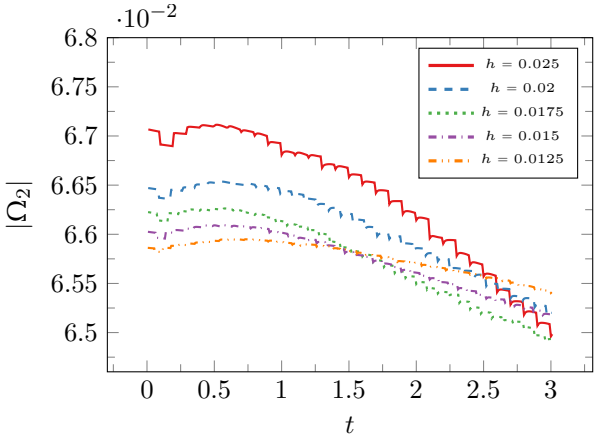
(b) z_c center-of-mass vertical component.



(c) Vertical velocity.



(d) Sphericity.



(e) Mass.

Figure 2. Results for the ellipsoidal test case (case 1).

To evaluate the effect of the characteristic mesh size, we plot the results we obtained for the simulations run with both $h = 0.025$ and $h = 0.0125$ along with the results from [40].

We can observe an overall good agreement between our results and the benchmark performed in [40].

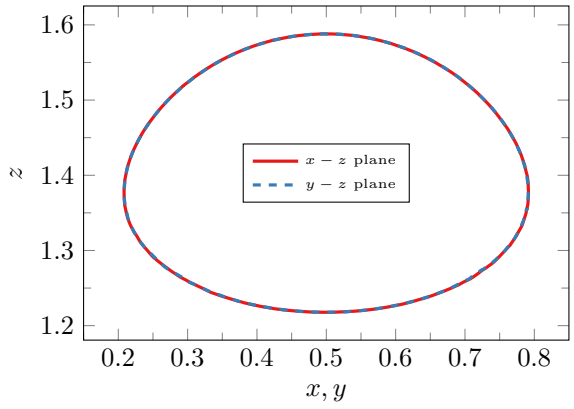
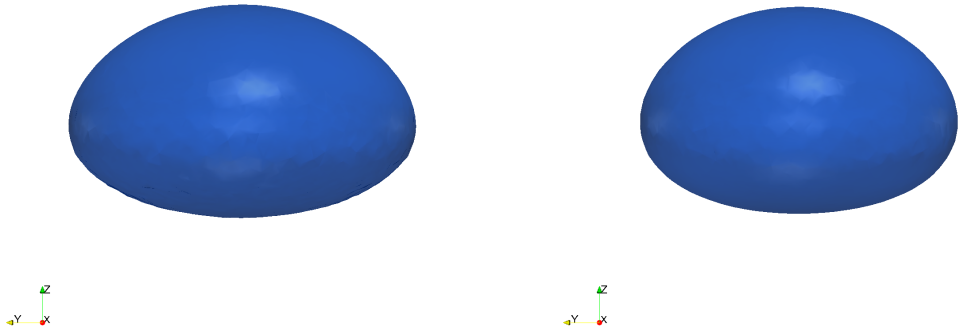


Figure 3. Shape at final time in the $x-z$ and $y-z$ planes for test case 1 ($h = 0.0125$).



(a) Fast Marching method **(b)** Hamilton-Jacobi method

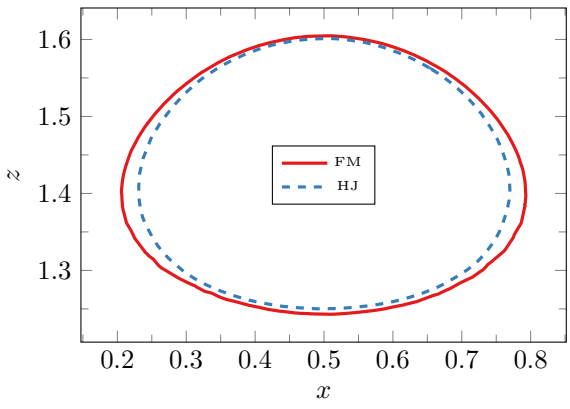
Figure 4. 3D shape at final time ($t = 3$) in the $x-y$ plane for test case 1 ($h = 0.0175$).

5.2. Case 2: the skirted drop

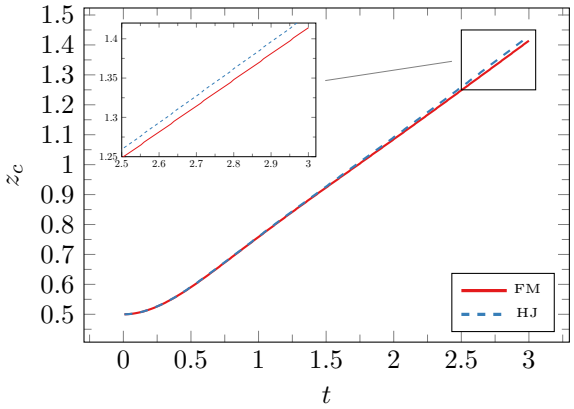
In the second test case, the drop gets more deformed because of the lower surface tension and the higher viscosity and density contrasts. Figure 7 displays the monitored quantities for this test case. We observe that the shape of the "skirt" of the drop at $t = 3$ is quite strongly mesh dependent, but converges as the mesh is refined. The other characteristics of the drop are not so dependent on the mesh refinement, even for the geometrically related ones, such as the drop mass, which shows a really small estimation error (only 2% difference between the coarsest and finest meshes), and displays the really good conservation properties of our simulations. We again also note in fig. 8 the symmetry of the final shape of the drop, which highlights the really good symmetry conservation properties of our approach.

5.2.1. Comparison between Hamilton-Jacobi and fast-marching reinitialization

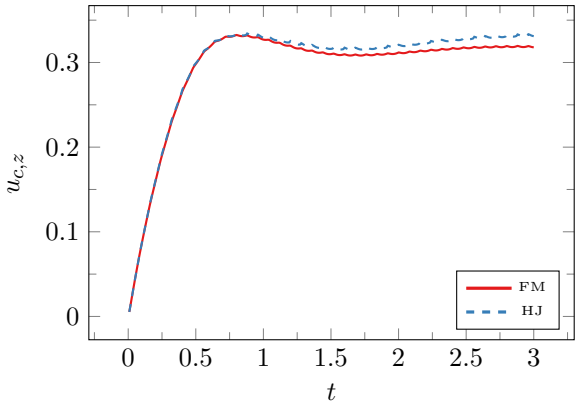
As for the test case 1, we provide a comparison of the results for the test case 2 obtained using either the fast-marching or the Hamilton-Jacobi reinitialization method. These results, obtained for an average mesh size ($h = 0.0175$) are shown in fig. 10. As before, they highlight noticeable differences between the two methods for geometrically related quantities such as mass loss, sphericity and final shape. We can even observe a non-negligible difference for the latter in the region of the "skirt". This difference, mainly related to the diffusive properties of the Hamilton-Jacobi method, can also be



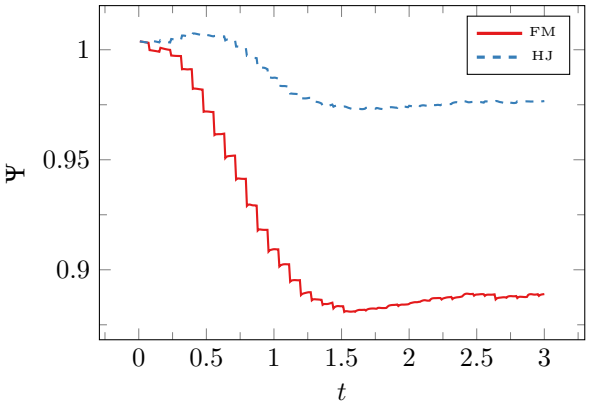
(a) Shape at final time ($t = 3$) in the vertical $x - z$ plane.



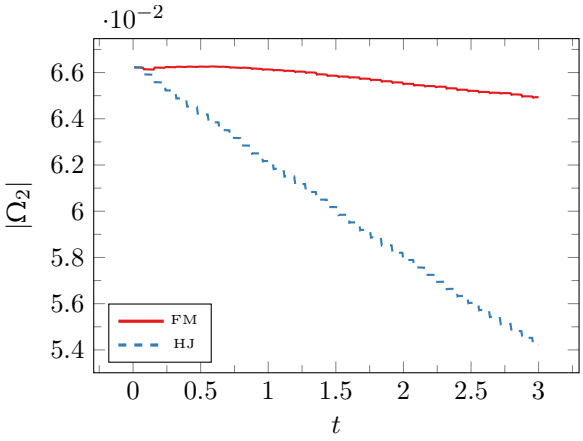
(b) z_c center-of-mass vertical component.



(c) Vertical velocity.

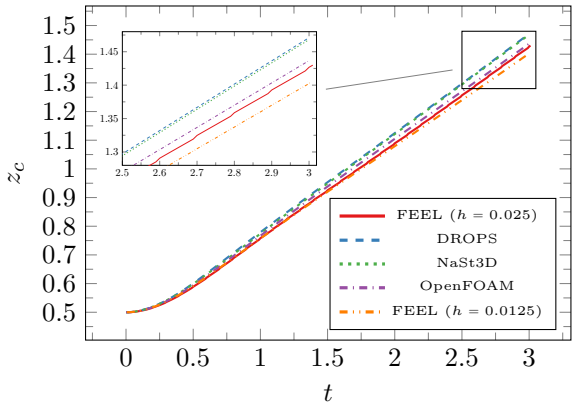


(d) Sphericity.

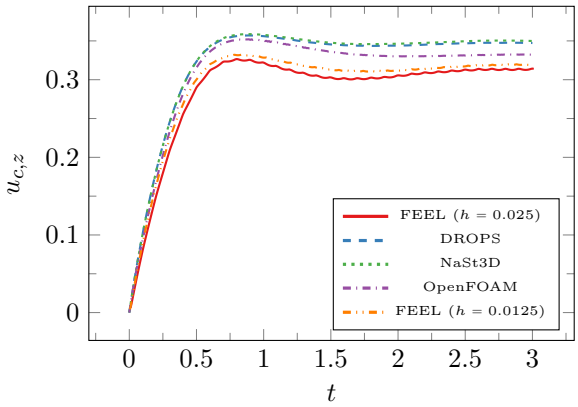


(e) Mass.

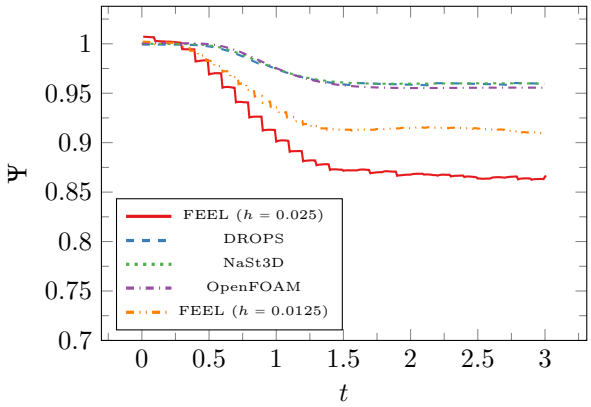
Figure 5. Comparison between the Fast Marching method (FM) and the Hamilton-Jacobi (HJ) method for test case 1 (ellipsoidal drop). The characteristic mesh size is $h = 0.0175$.



(a) z_c center-of-mass vertical component.



(b) Vertical velocity.



(c) Sphericity.

Figure 6. Comparison between our results (denoted FEEL) and the ones from [40] for the test case 2 (the ellipsoidal drop).

observed on the 3D shapes in fig. 9. The good agreement of the results obtained using the fast-marching method tends to suggest that the Hamilton-Jacobi method is not accurate enough – or would require more careful and costly adjustment of its parameters – for this kind of three-dimensional simulation.

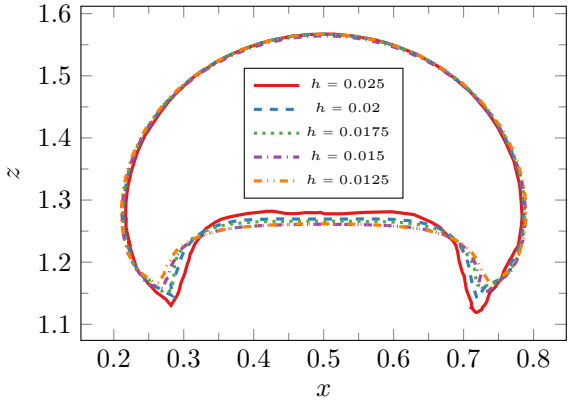
5.2.2. Comparison with previous results

As in section 5.1.2, we compare our results to the benchmark [40], and show the relevant quantities in fig. 11.

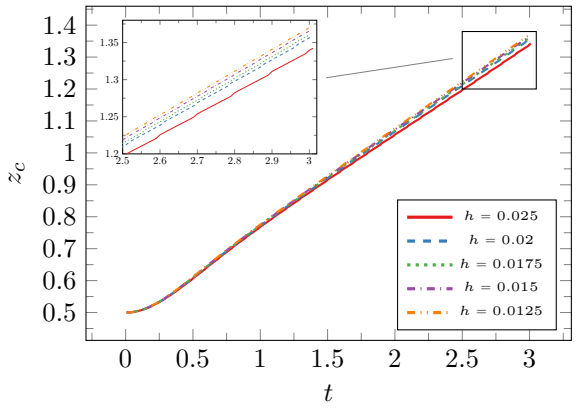
We also observe a good agreement between our simulations and the ones from the benchmark. We however note that the final shape of the skirted drop is very sensitive to the mesh and none of the groups agree on the exact shape which can explain the differences that we see on the parameters in fig. 11 at time $t > 2$.

5.3. High-order simulations

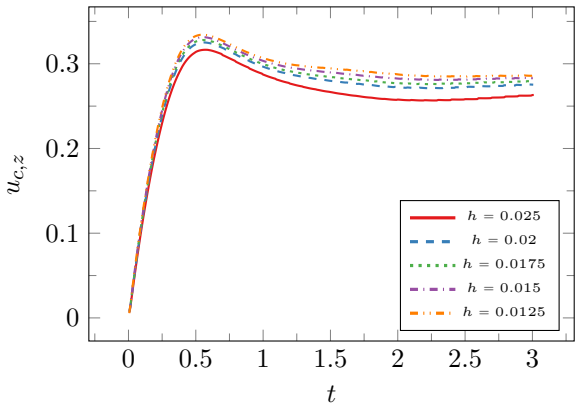
As already mentioned, our framework naturally allows the use of high-order Galerkin discretization spaces. As an illustration, we present here benchmark simulation results performed using finite element spaces spanned by Lagrange polynomials of order $(2, 1, 2)$ and $(2, 1, 3)$ for each test case. The mesh size considered here is $h = 0.02$, and the results are shown in fig. 12 and fig. 13 for test cases 1 and 2 respectively. We expect the increase in order of the level-set field to improve the overall accuracy.



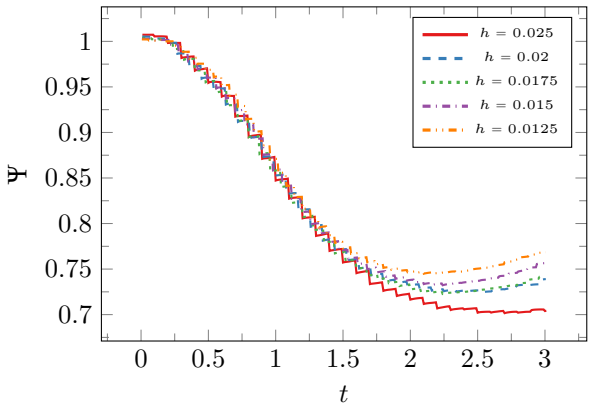
(a) Shape at final time ($t = 3$) in the vertical $x - z$ plane.



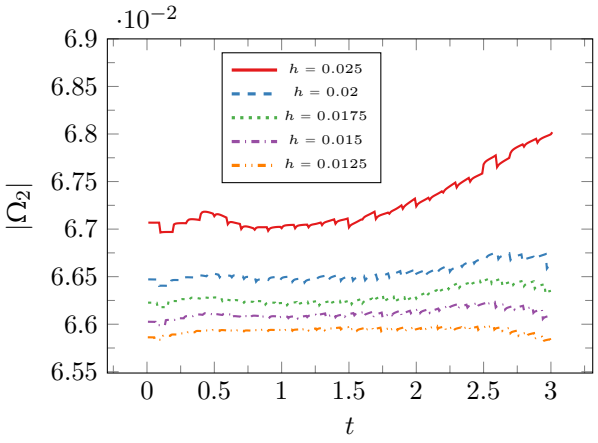
(b) z_c center-of-mass vertical component.



(c) Vertical velocity.



(d) Sphericity.



(e) Mass.

Figure 7. Results for the skirted test case (case 2).

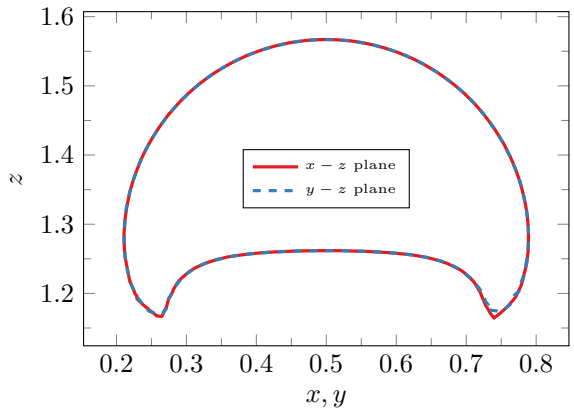
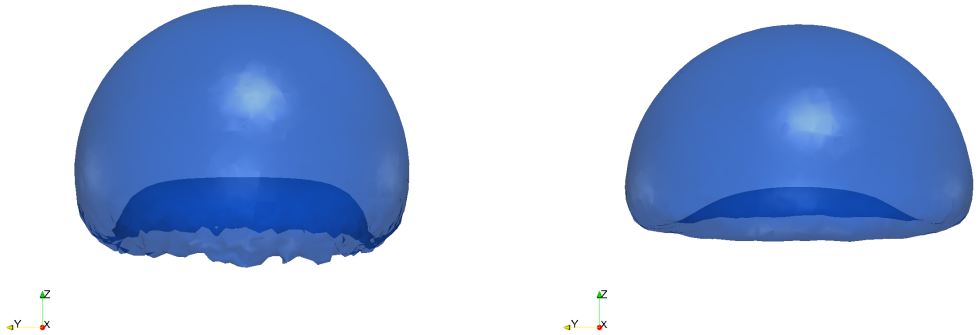


Figure 8. Shape at final time in the $x-z$ and $y-z$ planes for test case 2 ($h = 0.0125$).



(a) Fast Marching method **(b)** Hamilton-Jacobi method

Figure 9. 3D shape at final time ($t = 3$) in the $x-y$ plane for test case 2 ($h = 0.0175$).

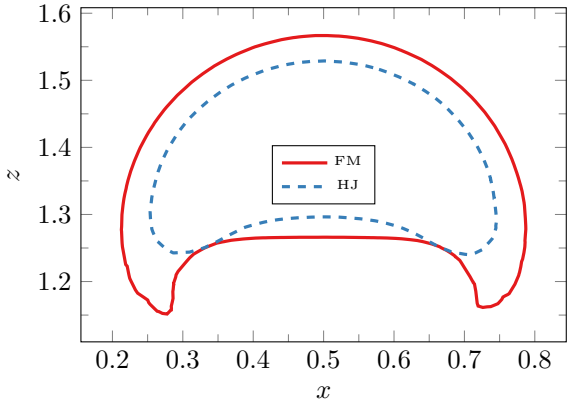
We can indeed observe that the final shapes of high-order simulations look smoother in both cases, as confirmed by the sphericity plots. The effect is highly noticeable on the "skirt" which appears for the second test-case, which looks even smoother than the one obtained with the finest ($h = 0.0125$) $(2, 1, 1)$ simulation.

Let us also highlight that the small differences observed with the $(2, 1, 3)$ simulations, in particular for the final shapes, are most likely related to the absence of articifial diffusion error in the computation of geometrical quantities (especially for the curvature), which suggest more robust and realistic results for these simulations.

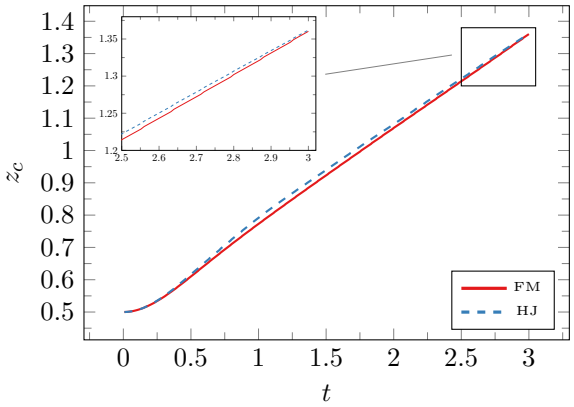
We can also notice that more "physically" controlled quantities, such as the position of the center-of-mass and the vertical velocity are less impacted by the polynomial order of the level-set component, which is not so surprising, as these quantities are mainly determined by the (level-set-dependent) fluid equations, which discretization orders where kept constant for this analysis.

6. Conclusion and outlooks

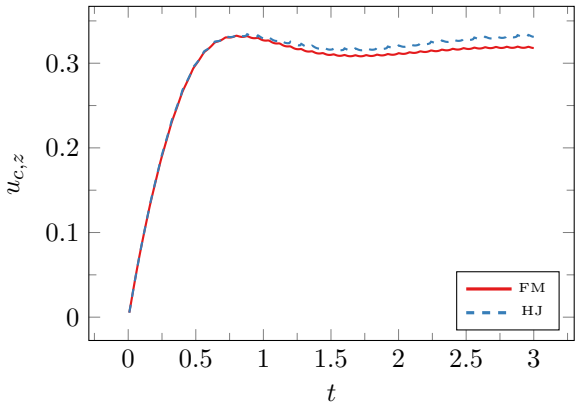
We have presented in this paper a comprehensive numerical framework for the simulation of multifluid flows. This framework is based on level-set methods solved by a (possibly high-order) finite element method. The explicit coupling between the level-set and the fluid has proven to be efficient



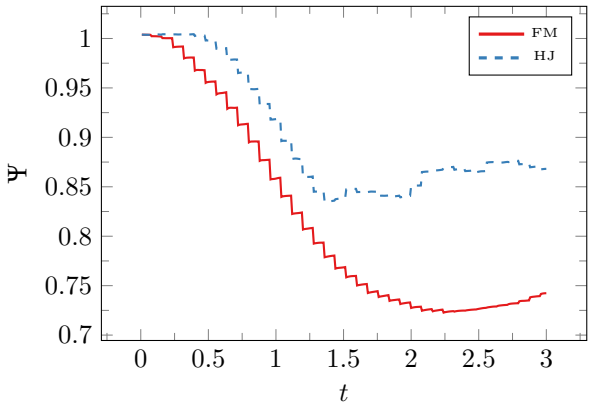
(a) Shape at final time ($t = 3$) in the vertical $x - z$ plane.



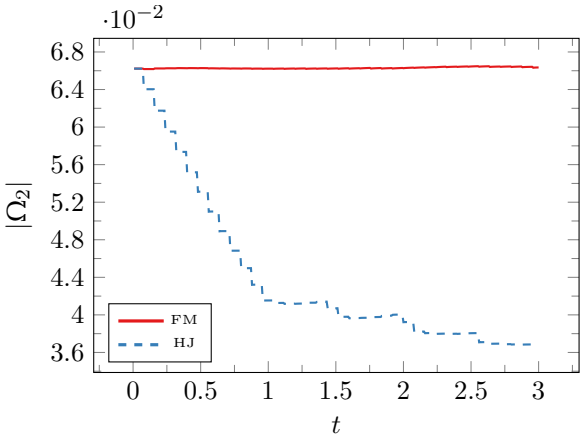
(b) z_c center-of-mass vertical component.



(c) Vertical velocity.

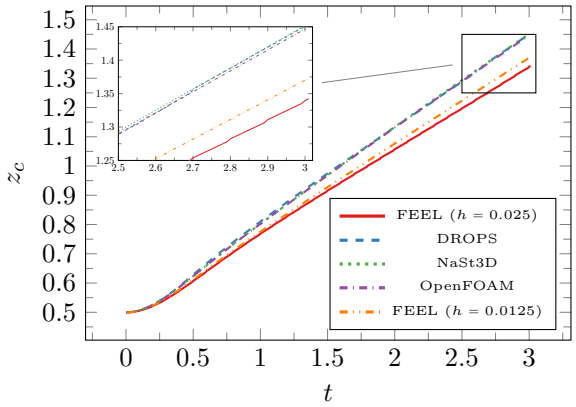


(d) Sphericity.

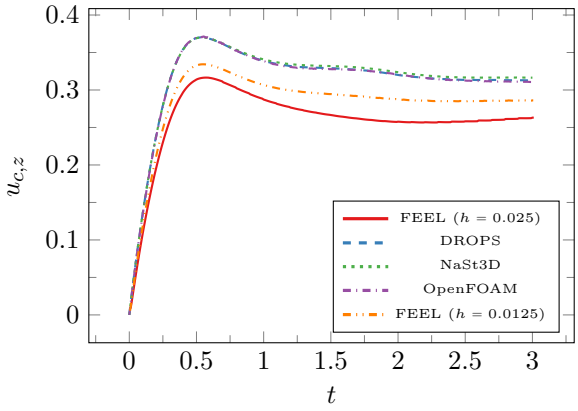


(e) Mass.

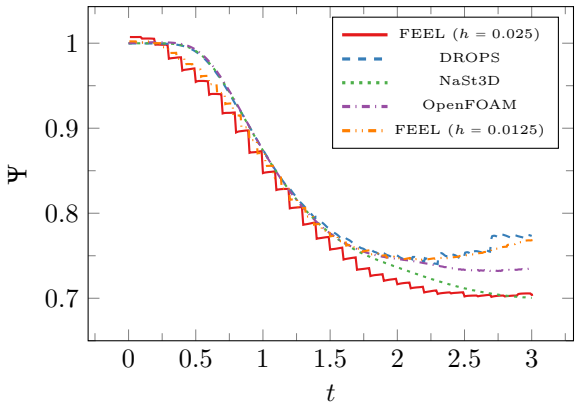
Figure 10. Comparison between the Fast Marching method (FM) and the Hamilton-Jacobi (HJ) method for test case 2 (skirted drop). The characteristic mesh size is $h = 0.0175$.



(a) z_c center-of-mass vertical component.

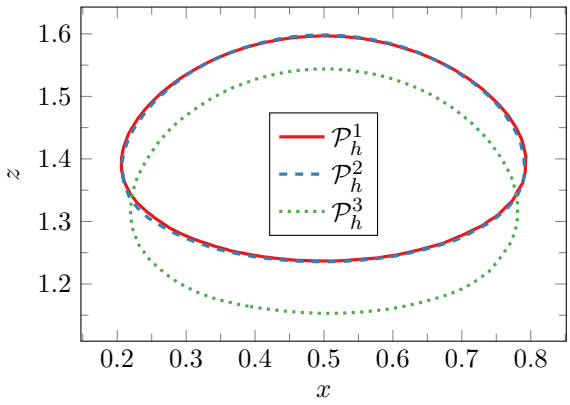


(b) Vertical velocity.

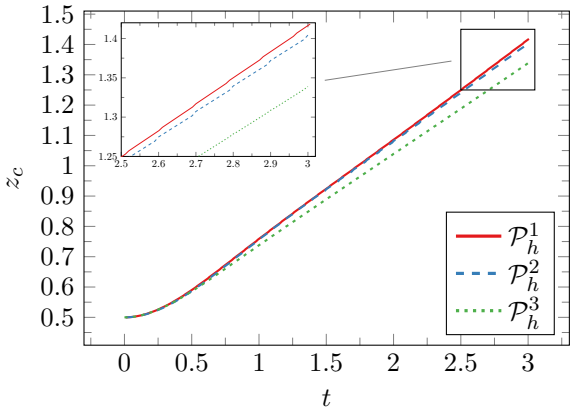


(c) Sphericity.

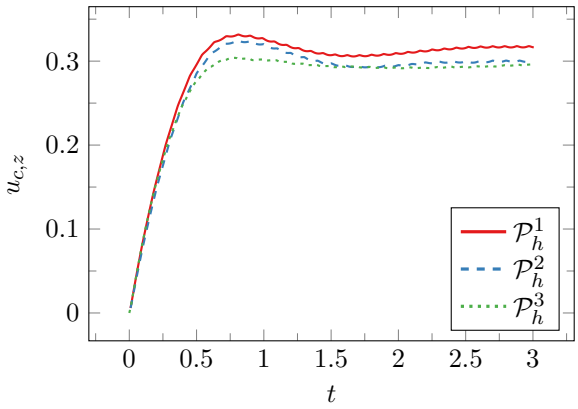
Figure 11. Comparison between our results (denoted FEEL) and the ones from [40] for the test case 2 (the skirted drop).



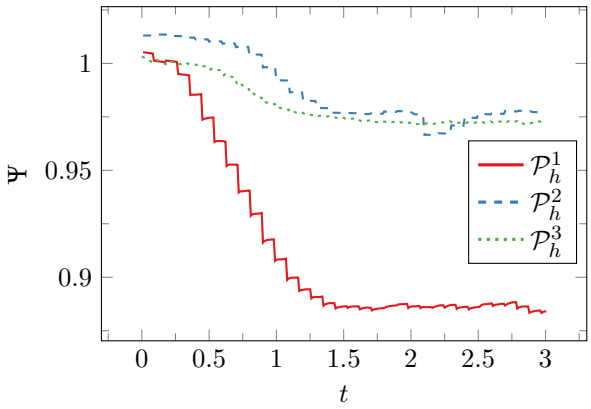
(a) Shape at final time ($t = 3$) in the vertical $x - z$ plane.



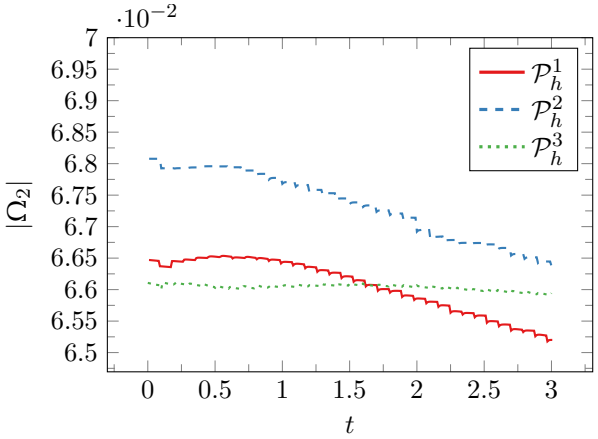
(b) z_c center-of-mass vertical component.



(c) Vertical velocity.

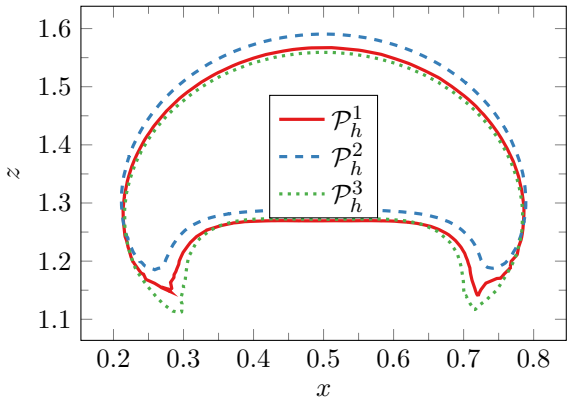


(d) Sphericity.

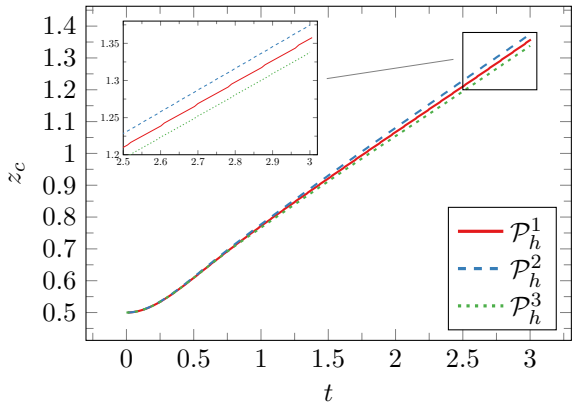


(e) Mass.

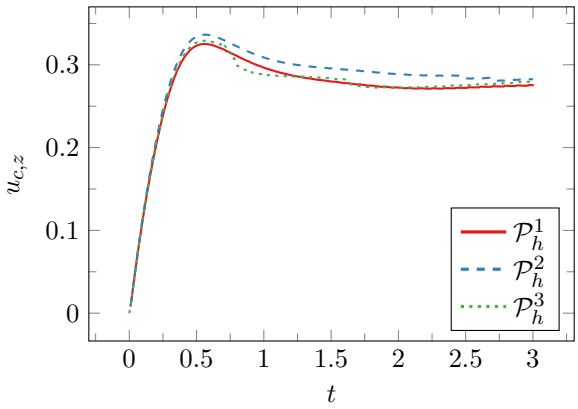
Figure 12. Comparison between $\mathcal{P}_h^1$, $\mathcal{P}_h^2$ and $\mathcal{P}_h^3$ simulations for the ellipsoidal test case ($h = 0.02$).



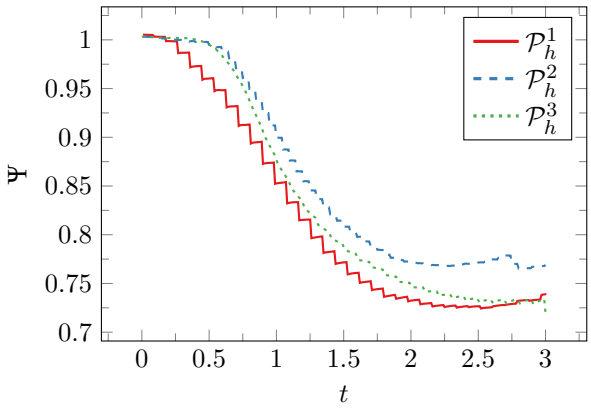
(a) Shape at final time ($t = 3$) in the vertical $x - z$ plane.



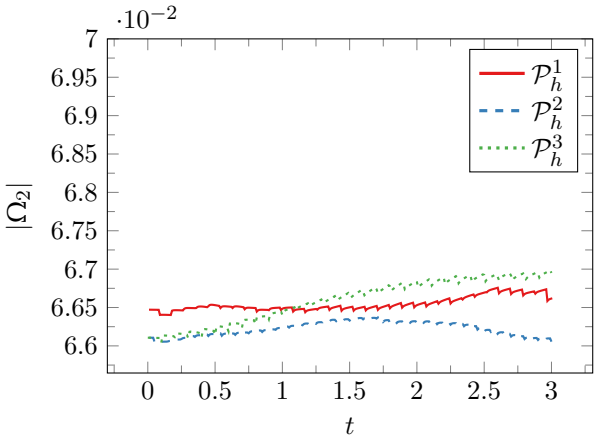
(b) z_c center-of-mass vertical component.



(c) Vertical velocity.



(d) Sphericity.



(e) Mass.

Figure 13. Comparison between $\mathcal{P}_h^1$ and $\mathcal{P}_h^2$ simulations for the skirted test case ($h = 0.02$).

and has allowed us to take advantage of reliable and efficient preconditioning strategies to solve the fluid equations.

The framework has been implemented within the Feel++ toolboxes and leverages the efficiency of the library to run on large numbers of processors in parallel. It also features user-friendly interfaces, and allows for easy model setup and parametrization using the JSON and CFG standard formats. The use of state-of-the-art metaprogramming techniques allows to seamlessly perform simulations in two or three dimension, and to increase the polynomial order of the finite elements. We highlight again that both the implementation and the benchmarks configuration files are open-source and available online [10,11].

The presented `MultiFluid` framework has been validated using a three-dimensional two-fluid numerical benchmark and achieved results in agreement with the simulations performed with other methods.

We have also compared two different level-set reinitialization procedures (the fast-marching and the Hamilton-Jacobi methods) and observed significantlty different behaviors, in particular the former is much better at mass conservation than the latter.

High-order simulations were performed to highlight the increased smoothness of the computed interfaces. High-order discretizations of the level-set function also greatly helps for the computation of geometrical quantities, such as the curvature of the interface. It avoids the need for artificial diffusion in the computation of such derivatives, which can be prove essential for accurate accounts of surface effects. In particular the $(2, 1, 3)$ simulations which feature complete diffusion-free geometrical quantities suggest that increasing the order of the level-set discretization can be of great interest when seeking highly accurate results regarding shapes, or when physical forces involving high derivatives of the level-set field are present.

Further improving the accuracy of the level-set and related quantities using higher order and/or hybrid methods is still ongoing.

The framework presented and validated here provides the building blocks for the simulation of complex fluids in complex geometries. Its versatility shall be used in a near future to better understand the flow of blood cells in realistic vascular systems, or the dynamics of swimming droplets interacting with external surfactants.

Acknowledgments: The authors would like to thank the French National Research Agency (MN-VIVABRAIN projects ANR-12-MONU-0010) for their financial support. Thibaut Metivet and Christophe Prud'homme would like to acknowledge the support of the Labex IRMIA. Most of the computations presented in this paper were performed using the Froggy platform of the CIMENT infrastructure https://ciment.ujf-grenoble.fr, which is supported by the Rhône-Alpes region (GRANT CPER07 13 CIRA) and the EquipMeso project (ANR-10-EQPX-29-01).

Author Contributions: T.Metivet conceived and designed the multifluid framework and the benchmark; T.Metivet, V.Chabannes and C.Prud'homme implemented the frameworks and the Feel++ library; T.Metivet performed the simulations; M.Ismail provided insight about the level-set method; T.Metivet and V.Chabannes wrote the paper.

1. Hirt, C.; Amsden, A.A.; Cook, J. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of computational physics* **1974**, *14*, 227–253.
2. Maury, B. A fat boundary method for the Poisson problem in a domain with holes. *Journal of scientific computing* **2001**, *16*, 319–339.
3. Hirt, C.W.; Nichols, B.D. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of computational physics* **1981**, *39*, 201–225.
4. Fix, G.J. Phase field methods for free boundary problems **1982**.
5. Langer, J. Models of pattern formation in first-order phase transitions. In *Directions in Condensed Matter Physics: Memorial Volume in Honor of Shang-Keng Ma*; World Scientific, 1986; pp. 165–186.
6. Osher, S.; Sethian, J.A. Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of computational physics* **1988**, *79*, 12–49.

7. Prud'Homme, C.; Chabannes, V.; Doyeux, V.; Ismail, M.; Samake, A.; Pena, G. Feel++: A Computational Framework for Galerkin Methods and Advanced Numerical Methods.
8. Prud'homme, C.; Chabannes, V.; Pena, G. Feel++: Finite Element Embedded Language in C++. Free Software available at http://www.feelpp.org. Contributions from A. Samake, V. Doyeux, M. Ismail and S. Veys.
9. Prud'homme, C. A domain specific embedded language in C++ for automatic differentiation, projection, integration and variational formulations. *Scientific Programming* **2006**, *14*.
10. Feel++ Consortium. Multifluid Flow Benchmarks. http://docs.feelpp.org/cases/0.105/multifluid/. Accessed: 2018-08-31.
11. Feel++ Consortium. Feel++ Github. https://github.com/feelpp/feelpp. Accessed: 2018-08-31.
12. Sethian, J. *Level Set Methods and Fast Marching Methods*; Cambridge University Press, 1996.
13. Stanley Osher, R.F. *Level Set Methods and Dynamic Implicit Surfaces*; Springer, S.S. Antman, J.E. Marsden, L. Sirovich.
14. Rouy, E.; Tourin, A. A viscosity solutions approach to shape-from-shading. *SIAM Journal on Numerical Analysis* **1992**, *29*, 867–884.
15. Sussman, M.; Smereka, P.; Osher, S. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics* **1994**, *114*, 146 – 159. doi:http://dx.doi.org/10.1006/jcph.1994.1155.
16. Sethian, J.A. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*; Vol. 3, Cambridge university press, 1999.
17. Chang, Y.C.; Hou, T.; Merriman, B.; Osher, S. A level set formulation of Eulerian interface capturing methods for incompressible fluid flows. *Journal of computational Physics* **1996**, *124*, 449–464.
18. Cottet, G.H.; Maitre, E. A level set method for fluid-structure interactions with immersed surfaces. *Mathematical models and methods in applied sciences* **2006**, *16*, 415–438.
19. Caldini Queiros, C.; Chabannes, V.; Ismail, M.; Pena, G.; Prud'Homme, C.; Szopos, M.; Tarabay, R. Towards large-scale three-dimensional blood flow simulations in realistic geometries **2013-06**. pp. 17, Accepted in ESAIM: Proc.
20. Chabannes, V.; Ismail, M.; Prud'Homme, C.; SZOPOS, M. Hemodynamic simulations in the cerebral venous network: A study on the influence of different modeling assumptions. *Journal of Coupled Systems and Multiscale Dynamics* **2015**, *3*, 23–37. doi:10.1166/jcsmd.2015.1062.
21. Daversin, C.; Prudhomme, C.; Trophime, C. Full 3D MultiPhysics Model of High Field PolyHelices Magnets. *IEEE Transactions on Applied Superconductivity* **2016**, *26*, 1–4. doi:10.1109/TASC.2016.2516241.
22. Daversin, C.; Prud'Homme, C. Simultaneous Empirical Interpolation and Reduced Basis Method: Application to Non-linear Multi-Physics Problem. In *Model Reduction of Parametrized Systems*; Springer, 2017; pp. 17–35.
23. Quarteroni, A.; Quarteroni, S. *Numerical models for differential problems*; Vol. 2, Springer, 2009.
24. Brooks, A.N.; Hughes, T.J. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer methods in applied mechanics and engineering* **1982**, *32*, 199–259.
25. Franca, L.P.; Hughes, T.J. Two classes of mixed finite element methods. *Computer Methods in Applied Mechanics and Engineering* **1988**, *69*, 89 – 129. doi:http://dx.doi.org/10.1016/0045-7825(88)90168-5.
26. Hughes, T.J.; Franca, L.P.; Hulbert, G.M. A new finite element formulation for computational fluid dynamics: VIII. The galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering* **1989**, *73*, 173 – 189. doi:http://dx.doi.org/10.1016/0045-7825(89)90111-4.
27. Hauke, G. A simple subgrid scale stabilized method for the advection–diffusion-reaction equation. *Computer Methods in Applied Mechanics and Engineering* **2002**, *191*, 2925–2947.
28. Douglas, J.; Dupont, T. Interior penalty procedures for elliptic and parabolic Galerkin methods. In *Computing methods in applied sciences*; Springer, 1976; pp. 207–216.
29. Chaple, R.P.B. *Numerical stabilization of convection-diffusion-reaction problems*; Delft University of Technology, 2006.

30. Codina, R. Comparison of some finite element methods for solving the diffusion-convection-reaction equation. *Computer Methods in Applied Mechanics and Engineering* **1998**, *156*, 185–210.
31. Winkelmann, C. Interior penalty finite element approximation of Navier-Stokes equations and application to free surface flows. PhD thesis, 2007.
32. Tezduyar, T.E.; Osawa, Y. Finite element stabilization parameters computed from element matrices and vectors. *Computer Methods in Applied Mechanics and Engineering* **2000**, *190*, 411–430.
33. Franca, L.P.; Valentin, F. On an improved unusual stabilized finite element method for the advective–reactive–diffusive equation. *Computer Methods in Applied Mechanics and Engineering* **2000**, *190*, 1785–1800.
34. Balay, S.; Gropp, W.D.; McInnes, L.C.; Smith, B.F. Efficient Management of Parallelism in Object-Oriented Numerical Software Libraries. Modern Software Tools in Scientific Computing. Birkhauser Press, 1997, pp. 163–202.
35. Balay, S.; Abhyankar, S.; Adams, M.F.; Brown, J.; Brune, P.; Buschelman, K.; Dalcin, L.; Eijkhout, V.; Gropp, W.D.; Kaushik, D.; Knepley, M.G.; May, D.A.; McInnes, L.C.; Mills, R.T.; Munson, T.; Rupp, K.; Sanan, P.; Smith, B.F.; Zampini, S.; Zhang, H.; Zhang, H. PETSc Users Manual. Technical Report ANL-95/11 - Revision 3.9, Argonne National Laboratory, 2018.
36. Doyeux, V. Modélisation et simulation de systèmes multi-fluides. Applications aux écoulements sanguins. PhD thesis, 2014. Thèse de doctorat dirigée par Peyla, Philippe et Ismail, Mourad Physique Grenoble 2014.
37. Chabannes, V.; Pena, G.; C.Prud'homme. High order fluid structure interaction in 2D and 3D. Application to blood flow in arteries. Fifth International Conference on Advanced COmputational Methods in ENgineering (ACOMEN 2011), 2011.
38. Hysing, S.; Turek, S.; Kuzmin, D.; Parolini, N.; Burman, E.; Ganesan, S.; Tobiska, L. Quantitative benchmark computations of two-dimensional bubble dynamics. *International Journal for Numerical Methods in Fluids* **2009**, *60*, 1259–1288. doi:10.1002/fld.1934.
39. Doyeux, V.; Guyot, Y.; Chabannes, V.; Prud'Homme, C.; Ismail, M. Simulation of two-fluid flows using a finite element/level set method. Application to bubbles and vesicle dynamics. *Journal of Computational and Applied Mathematics* **2013**, *246*, 251–259.
40. Adelsberger, J.; Esser, P.; Griebel, M.; Groß, S.; Klitz, M.; Rüttgers, A. 3D incompressible two-phase flow benchmark computations for rising droplets. Proceedings of the 11th World Congress on Computational Mechanics (WCCM XI), Barcelona, Spain, 2014, Vol. 179.
41. Patankar, S.V.; Spalding, D.B. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International journal of heat and mass transfer* **1972**, *15*, 1787–1806.