



HAL
open science

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

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

► **To cite this version:**

Thibaut Métivet, Vincent Chabannes, Mourad Ismail, Christophe Prud'Homme. High-order finite-element framework for the efficient simulation of multifluid flows. *Mathematics*, 2018, 6 (10), 10.3390/math6100203. hal-01865602

HAL Id: hal-01865602

<https://hal.science/hal-01865602>

Submitted on 31 Aug 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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 Differentiation 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 explicitly 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 implicitly 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 consisting 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

$$\rho [\partial_t \vec{u} + (\vec{u} \cdot \nabla) \vec{u}] - \nabla \cdot \sigma(\vec{u}, p) = \vec{f}_b + \vec{f}_s \quad (1)$$

$$\nabla \cdot \vec{u} = 0 \quad (2)$$

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

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

with μ the fluid viscosity and $\mathbf{D}(\vec{u}) = \frac{1}{2} (\nabla \vec{u} + \nabla \vec{u}^T)$ 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:

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

2.2. Interface tracking: the level-set method

Let us consider two disjoint – not necessarily connected – fluid domains $\Omega_i, \Omega_j \subset \Omega$ 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.

58 It features a Lipschitz continuous scalar function ϕ (the *level set* function) defined on the whole
 59 domain. This function is arbitrarily chosen to be positive in Ω_i , negative in Ω_j and zero on Γ_{ij} , so
 60 that the interface is implicitly 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 \mathbf{u} (uniquely defined at the interface by assumption and construction) and therefore obeys the advection equation:

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0. \quad (5)$$

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

$$\phi(\mathbf{x}) = \begin{cases} \text{dist}(\mathbf{x}, \Gamma_{ij}) & \mathbf{x} \in \Omega_i, \\ 0 & \mathbf{x} \in \Gamma_{ij}, \\ -\text{dist}(\mathbf{x}, \Gamma_{ij}) & \mathbf{x} \in \Omega_j, \end{cases} \quad (6)$$

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

64 However, the advection equation (5) does not preserve the property $|\nabla \phi| = 1$ and it is necessary
 65 to reset $\phi(t)$ to a distance function without moving the interface, [14–16]. To redifferentiate $\phi(t)$ and
 66 enforce $|\nabla \phi(t)| = 1$, we can either solve a Hamilton-Jacobi equation, which “transports” the isolines
 67 of ϕ to their proper positions, or use the fast marching method, which resets the values of ϕ to the
 68 distance to the interface from one degree of freedom to the next, starting from the interface. We provide
 69 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

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

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

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

74 The Heaviside function is used to define physical quantities which have different values on each
 75 subdomain. For example, we define the density and viscosity of two-fluid flows as $\rho = \rho_j + (\rho_i - \rho_j)H_\varepsilon(\phi)$
 76 and $\mu = \mu_j + (\mu_i - \mu_j)H_\varepsilon(\phi)$. The delta function allows to define quantities on the interface, in
 77 particular in the variational formulations, where we replace integrals over the interface Γ with integrals
 78 over the entire domain Ω using the smoothed delta function: if ϕ is a signed distance function (i.e.
 79 $|\nabla \phi| = 1$), we have $\int_\Gamma 1 \simeq \int_\Omega \delta_\varepsilon(\phi)$.

80 In the following, we consider without loss of generality only the two-fluid case, with only one level
 81 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 continuous 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 $\mathbf{H}^1(\Omega) = \{\vec{v} \in [H^1(\Omega)]^d\}$, $\mathbf{H}_{g_D}^1(\Omega) = \{\vec{v} \in \mathbf{H}^1(\Omega), \vec{v}|_{\partial\Omega_D} = \vec{g}_D\}$ and $\mathbf{H}_{0_D}^1(\Omega) = \{\vec{v} \in \mathbf{H}^1(\Omega), \vec{v}|_{\partial\Omega_D} = \vec{0}\}$.

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

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

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

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

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

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 adaptation 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,

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

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

130 3.2. Fluid-interface coupling

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

136 At each time step, the fluid equations are first solved with the physical parameters and the surface
137 forces computed using the last-step level-set function. We then use the obtained fluid velocity to advect
138 the level-set and get the new interface position. We can then apply some redistancing procedure
139 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 redistanciate  $\phi^{n+1}$ ;
end

```

140 Note that the successive resolution of the fluid and level-set equations can also be iterated within
141 one time step, until a fix point of the system of equations is reached. In practice however, for reasonably
142 small time steps, the fix-point solution is already obtained after the first iteration.
143

144 3.3. Space/Time discretization

145 We introduce $\mathcal{T}_h \equiv \{K_e, 1 \leq e \leq N_{elt}\}$ a compatible tessellation of the domain Ω , and denote
146 $\Omega_h = \bigcup_{e=1}^{N_{elt}} K_e$ the discrete – unstructured – mesh associated with average mesh size h . We work
147 within the continuous Galerkin variational formulation framework, and use Lagrange finite elements to
148 spatially discretize and solve the equations governing the evolutions of the fluid and the level-set. We
149 thus introduce $\mathcal{P}_h^k \equiv \mathcal{P}_h^k(\Omega_h)$ the discrete (h -dependent) finite element space spanned by Lagrange
150 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

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

where the α_i are BDF_N coefficients. In the numerical benchmarks reported below, we used BDF₁ and BDF₂, which write respectively

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

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

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 :

$$\begin{aligned} \bullet V_h &= \left\{ \vec{v} \in \mathbf{H}_{gD}^1 \cap [\mathcal{P}_h^{k+1}]^d \right\} \\ \bullet Q_h &= \left\{ q \in \mathcal{P}_h^k \right\} \end{aligned}$$

We need also to introduce the test function space on the velocity with $W_h = \left\{ \vec{v} \in \mathbf{H}_{0D}^1(\Omega_h) \cap [\mathcal{P}_h^{k+1}]^d \right\}$. 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 $\forall (\vec{v}_h, p_h) \in W_h \times Q_h$:

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

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

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

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.

172 *3.5. The level-set framework*173 *3.5.1. Level-set advection*

174 As mentioned above, the transport of the level-set by the fluid is accounted by the advection
 175 equation eq. (11) in variational form. This equation is discretized in time using a backward differentiation
 176 formula (BDF_N) and space discretization is performed using the \mathcal{P}_h^k discrete spaces introduced above.
 177 However, as is well-known for central differencing schemes in hyperbolic partial differential equations,
 178 the *naive* Galerkin discretization of eq. (11) on \mathcal{P}_h^k can lead to spurious oscillatory instabilities. To
 179 circumvent this well-known problem, we stabilize the discrete advection equation. Four different
 180 stabilization methods are supported in our framework: the Streamline Upwind Petrov-Galerkin (SUPG,
 181 [24]), the Galerkin Least Squares (GLS, [25,26]), the Sub-Grid Scale (SGS, [27]) and the Continuous
 182 Interior Penalty (CIP, [28]) methods. Detailed description of these methods can be found in [29–31]. It
 183 should be noted that the CIP stabilization is much more costly than the three others, as it densifies the
 184 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

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

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

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

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

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

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

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

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

192 The linear system is then solved using the PETSc library [34] with a GMRES solver preconditioned
 193 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

$$\vec{n} = \frac{\nabla\phi}{|\nabla\phi|} \quad (20)$$

$$\kappa = \nabla \cdot \vec{n} \quad (21)$$

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 $\Pi_{L^2}^k$ and the smoothing projection operator Π_{sm}^k , defined respectively as

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

197 and

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

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 responsible 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_{sm}^k 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 $\vec{n} = \Pi_{L^2}^1 \left(\frac{\nabla\phi}{|\nabla\phi|} \right)$ and $\kappa = \Pi_{sm}^1 (\nabla \cdot \vec{n})$ for \mathcal{P}_h^1 simulations, and
212 $\vec{n} = \Pi_{L^2}^1 \left(\frac{\nabla\phi}{|\nabla\phi|} \right)$ and $\kappa = \Pi_{L^2}^1 (\nabla \cdot \vec{n})$ for higher order ones.

213 3.5.3. Redifferentiation

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 redifferentiate the level-set, i.e. recover a signed distance function
218 to $\{\phi = 0\}$. Two main approaches can be used: the first one relies on an efficient direct computation of

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

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

227 If the level-set is of order $k > 1$, we can still perform the fast-marching redistantiation using a
 228 \mathcal{P}_{iso}^1 space as proposed in [36]. This order-1 Lagrange function space is constructed on a mesh obtained
 229 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
 230 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
 231 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

$$\partial_\tau \phi + \text{sign}(\phi) (1 - |\nabla\phi|) = 0 \quad (24)$$

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

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

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

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

241 4. 3D rising drops benchmark

242 We now present a 3D benchmark of our numerical approach, using the Navier-Stokes solver
 243 developed with the *Fee1++* library described in [37]. This benchmark is a three-dimensional extension
 244 of the 2D benchmark introduced in [38] and realised using *Fee1++* in [39]. The setup for this benchmark
 245 was also used in [40] to compare several flow solvers.

246 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:

$$\vec{f}_b = \rho_\phi \vec{g} \quad (26)$$

$$\vec{f}_s = \sigma \kappa \vec{n} \Big|_\Gamma \simeq \sigma \kappa \vec{n} \delta_\varepsilon(\phi) \quad (27)$$

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

248 We consider Ω a cylinder with radius $R = 0.5$ and height $H = 2$, filled with a fluid and containing a
 249 droplet of another immiscible fluid. We denote $\Omega_1 = \{\vec{x}, \text{ s.t. } \phi(\vec{x}) > 0\}$ the domain outside the droplet,
 250 $\Omega_2 = \{\vec{x}, \text{ s.t. } \phi(\vec{x}) < 0\}$ the domain inside the drop and $\Gamma = \{\vec{x}, \text{ s.t. } \phi(\vec{x}) = 0\}$ the interface. We
 251 impose no-slip boundary conditions $\vec{u}|_{\partial\Omega} = 0$ on Ω walls. The simulation is run from $t = 0$ to 3.

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

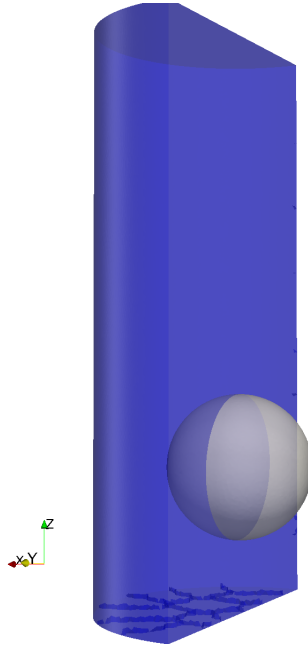


Figure 1. Initial setup for the benchmark.

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 $\rho_1, \rho_2, \mu_1, \mu_2$ 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

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

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

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

255 Table 1 reports the values of the parameters used for two different test cases proposed in [40]. At
 256 $t = 3$, the first one leads to an ellipsoidal-shaped drop while the second one gives a skirted shape due
 257 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 –

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

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

260 4.2. Simulation setup

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

264 We run the simulations looking for solutions in finite element spaces spanned by Lagrange
265 polynomials of order $(2, 1, k)$ for respectively the velocity, the pressure and the level set. The
266 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.

h	Mesh properties		Finite-element DOF		
	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

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

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

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

276 5. Results

277 In this section, we analyze the simulations of the rising drop for the two cases with low-order –
 278 $k = 1$ – sections 5.1 and 5.2 and high-order – $k = 2$ – section 5.3 level set discretization space. Except
 279 for the comparison between the fast-marching and the Hamilton-Jacobi methods section 5.1.1, the level
 280 set redistantsiations were performed with the fast-marching method every 10 time-steps.

281 5.1. Case 1: the ellipsoidal drop

282 Figure 2a shows the shape of the drop in the $x - z$ plane at the final $t = 3$ time step for the
 283 different aforementioned mesh sizes. The shapes are similar and seem to converge when the mesh size
 284 is decreasing. The drop reaches a stationary circularity as shown in fig. 2d, and its topology does not
 285 change. The velocity increases until it attains a constant value. Figure 2c shows the results obtained
 286 for the different mesh sizes. The evolution of the mass of the drop versus time is shown in fig. 2e. It
 287 highlights the rather good mass conservation property of our simulation setup, as about 3% of the mass
 288 is at most lost for the coarsest mesh, while the finest one succeeds in keeping the loss in mass below
 289 0.7%.

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

292 5.1.1. Comparison between Hamilton-Jacobi and fast-marching reinitialization

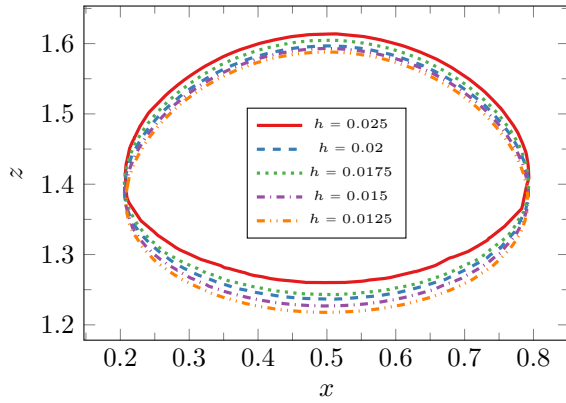
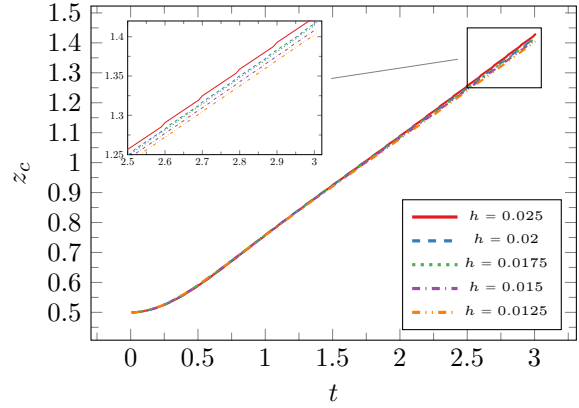
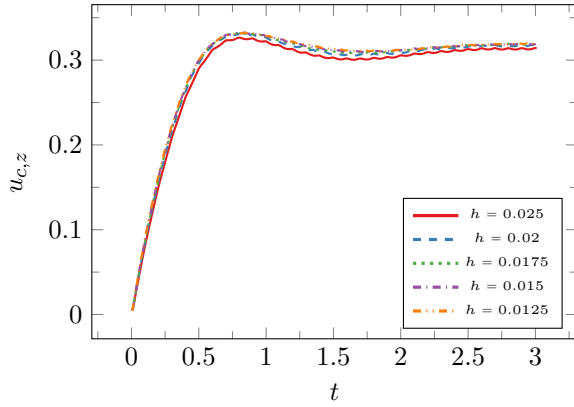
293 As mentioned in section 3.5.3, two reinitialization procedures can be used to overcome the
 294 “deformation” of the level set which becomes more and more different from the distance to the interface
 295 function as it is advected with the fluid velocity. The fast-marching method resets the values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

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

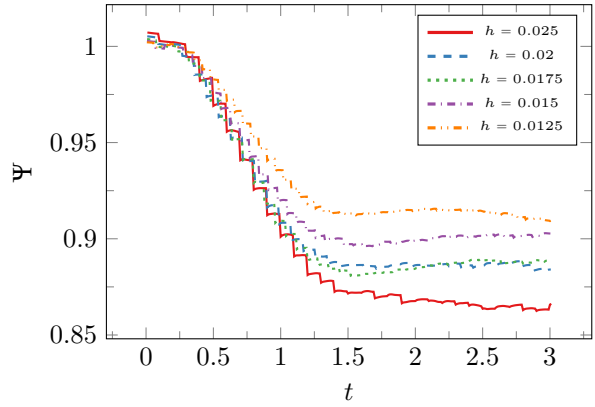
300 The first observation is that the mass loss (see fig. 5e) is considerably reduced when using the
 301 FM method. It goes from about 18% mass lost between $t = 0$ and $t = 3$ for the Hamilton-Jacobi
 302 method to less than 2% for the fast-marching method. This resulting difference of size can be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

310 5.1.2. Comparison with previous results

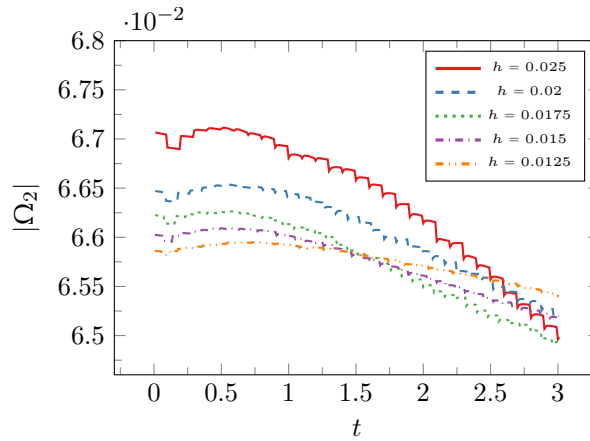
311 Figure 6 shows a plot of our results compared to the ones presented in [40]. In this paper, the
 312 authors perform simulations on the same setup and with the same test cases as considered here. To
 313 ensure consistency of their results, they use three different flow solvers (hence three different space
 314 discretization methods) coupled with two different interface capturing methods: the DROPS and
 315 NaSt3D solvers coupled to a level set approach, and the OpenFOAM solver which uses a volume-of-fluid
 316 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).

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

319 We can observe an overall good agreement between our results and the benchmark performed in
 320 [40].

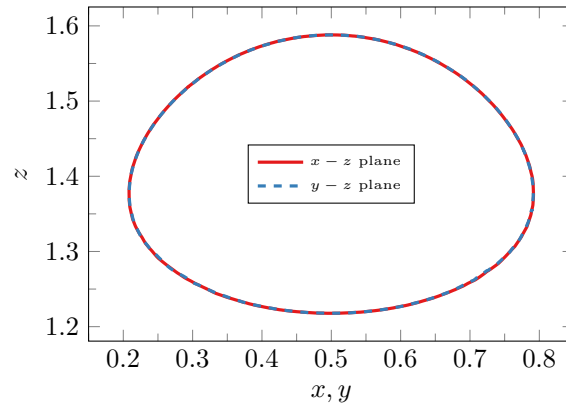


Figure 3. Shape at final time in the $x - z$ and $y - z$ planes for test case 1 ($h = 0.0125$).

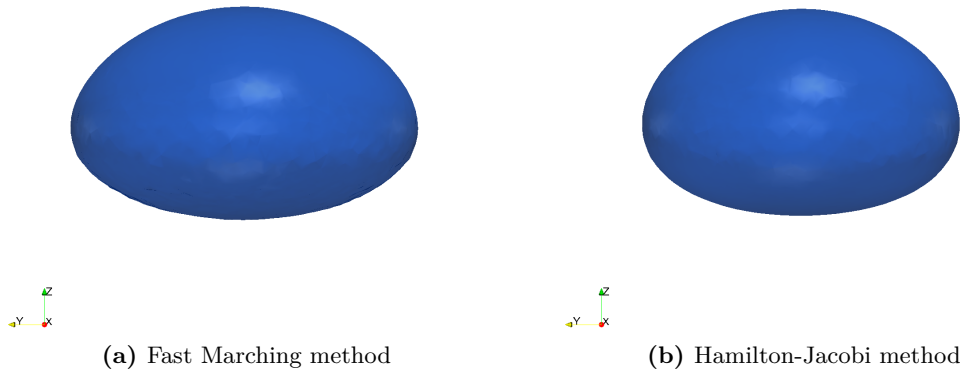


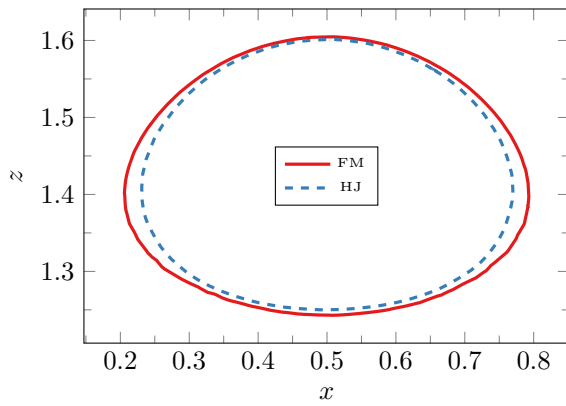
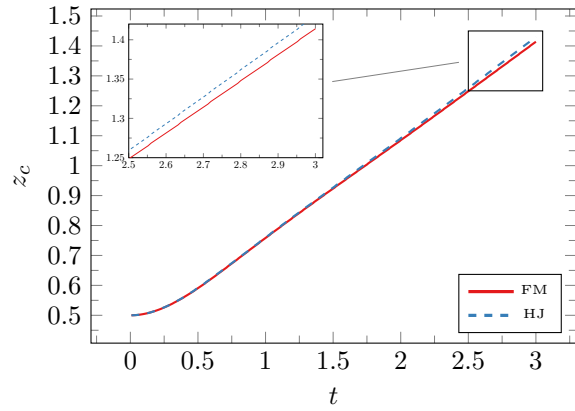
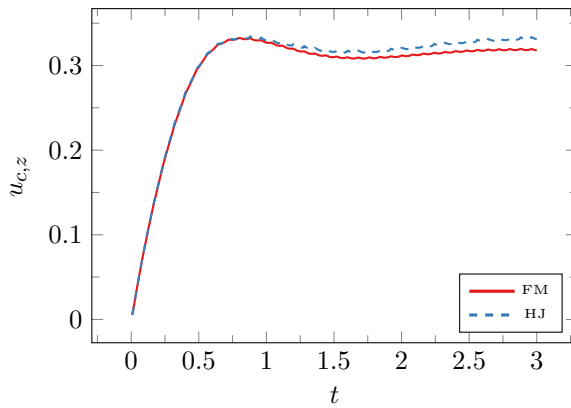
Figure 4. 3D shape at final time ($t = 3$) in the $x - y$ plane for test case 1 ($h = 0.0175$).

321 5.2. Case 2: the skirted drop

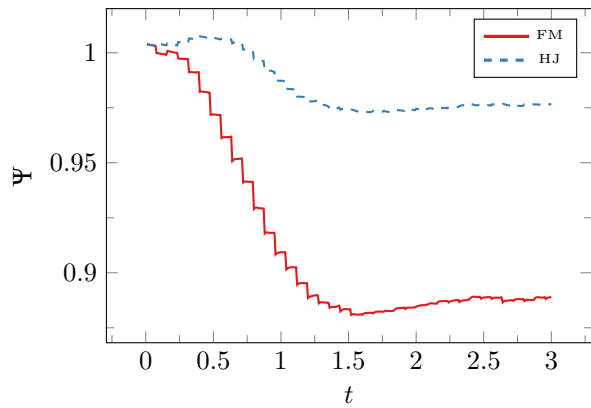
322 In the second test case, the drop gets more deformed because of the lower surface tension and the
 323 higher viscosity and density contrasts. Figure 7 displays the monitored quantities for this test case.
 324 We observe that the shape of the “skirt” of the drop at $t = 3$ is quite strongly mesh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

331 5.2.1. Comparison between Hamilton-Jacobi and fast-marching reinitialization

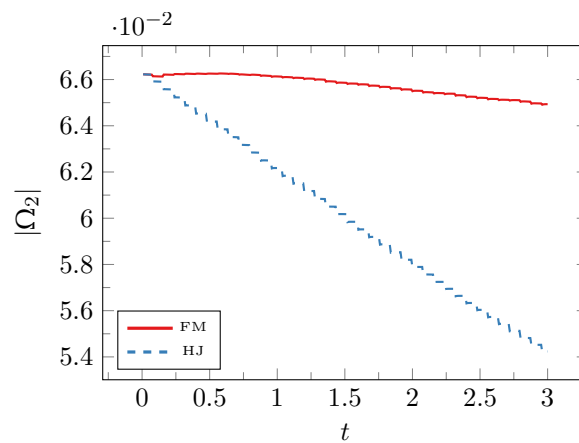
332 As for the test case 1, we provide a comparison of the results for the test case 2 obtained using
 333 either the fast-marching or the Hamilton-Jacobi reinitialization method. These results, obtained for an
 334 average mesh size ($h = 0.0175$) are shown in fig. 10. As before, they highlight noticeable differences
 335 between the two methods for geometrically related quantities such as mass loss, sphericity and final
 336 shape. We can even observe a non-negligible difference for the latter in the region of the “skirt”.
 337 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.

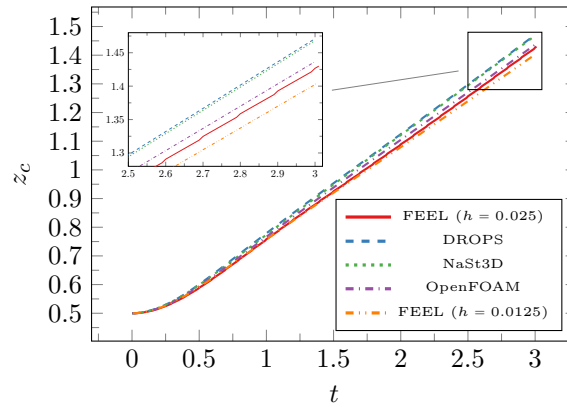
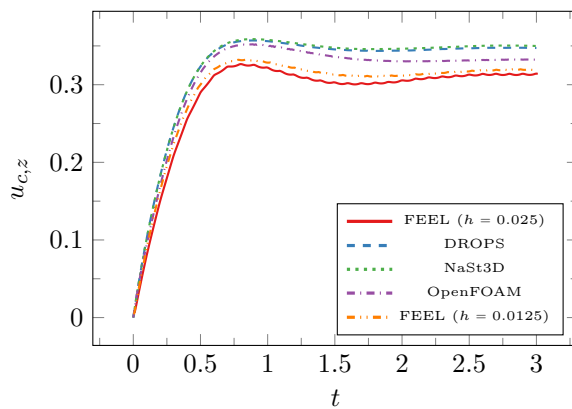


(d) Sphericity.

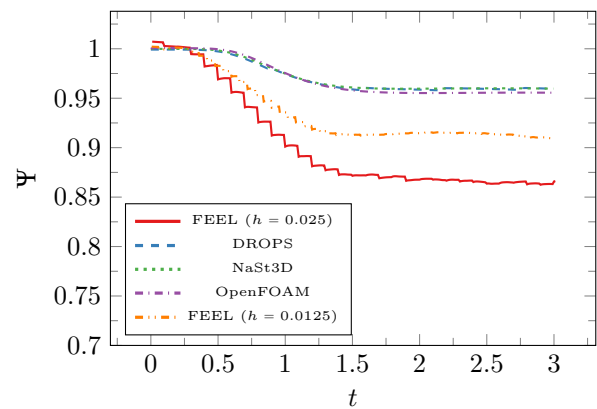


(e) Mass.

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).

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

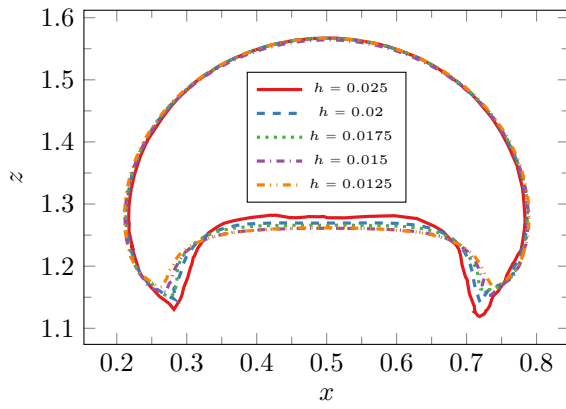
341 5.2.2. Comparison with previous results

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

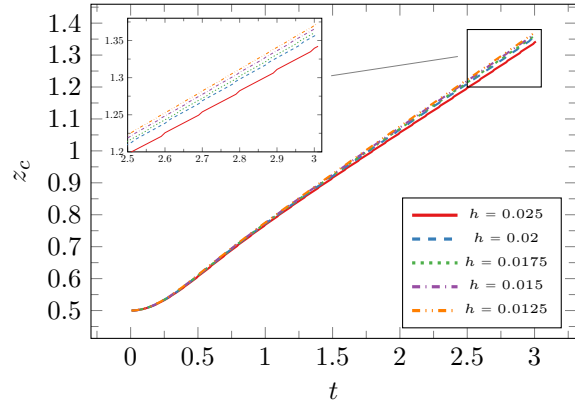
344 We also observe a good agreement between our simulations and the ones from the benchmark.
 345 We however note that the final shape of the skirted drop is very sensitive to the mesh and none of the
 346 groups agree on the exact shape which can explain the differences that we see on the parameters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

348 5.3. High-order simulations

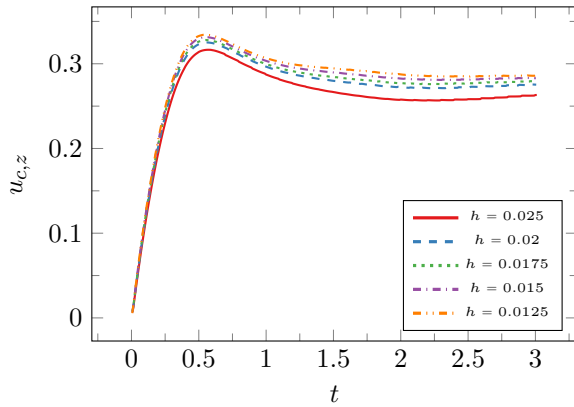
349 As already mentioned, our framework naturally allows the use of high-order Galerkin discretization
 350 spaces. As an illustration, we present here benchmark simulation results performed using finite element
 351 spaces spanned by Lagrange polynomials of order (2, 1, 2) and (2, 1, 3) for each test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



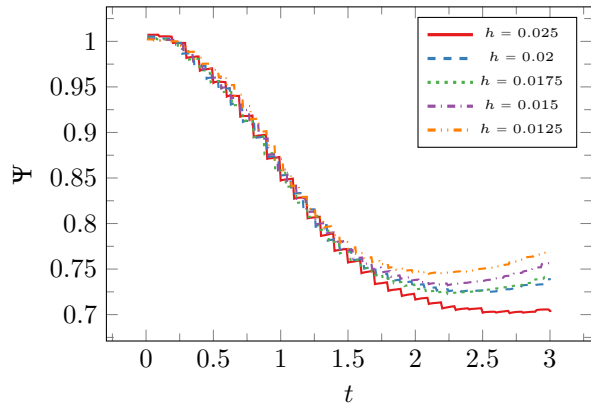
(a) Shape at final time ($t = 3$) in the vertical $x - z$ plane.



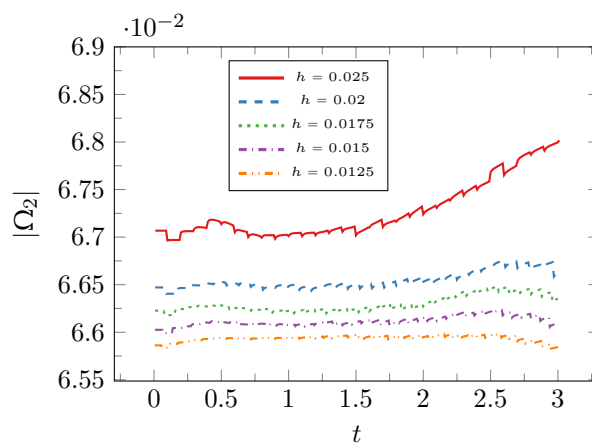
(b) z_c center-of-mass vertical component.



(c) Vertical velocity.



(d) Sphericity.



(e) Mass.

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

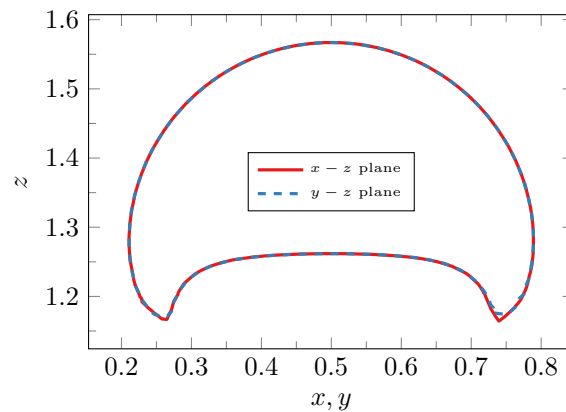


Figure 8. Shape at final time in the $x - z$ and $y - z$ planes for test case 2 ($h = 0.0125$).

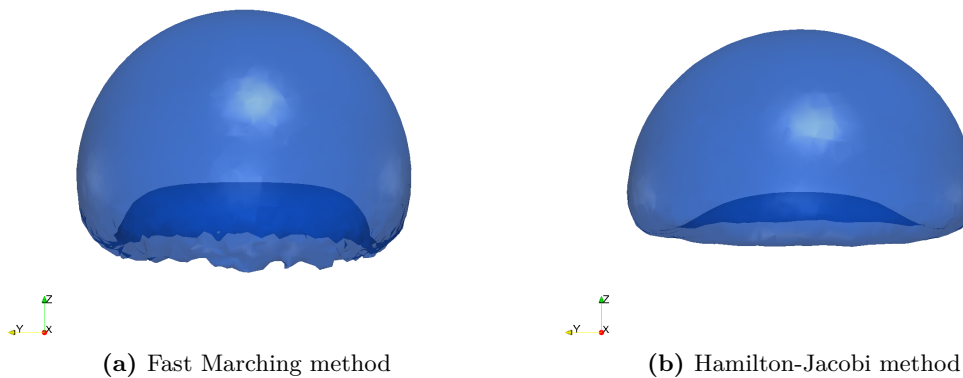


Figure 9. 3D shape at final time ($t = 3$) in the $x - y$ plane for test case 2 ($h = 0.0175$).

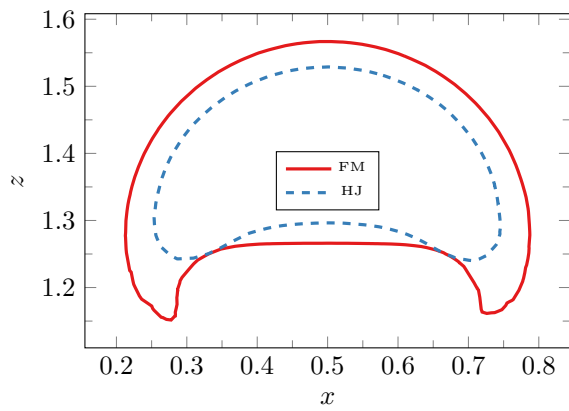
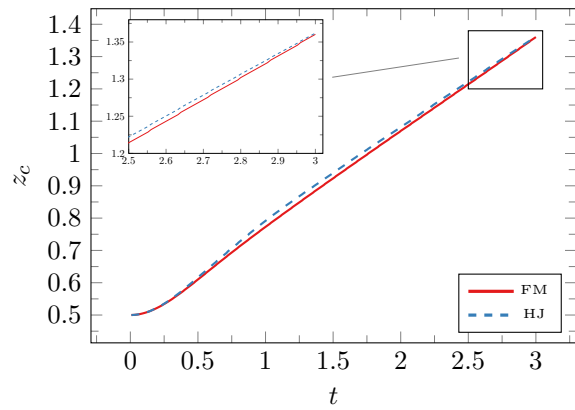
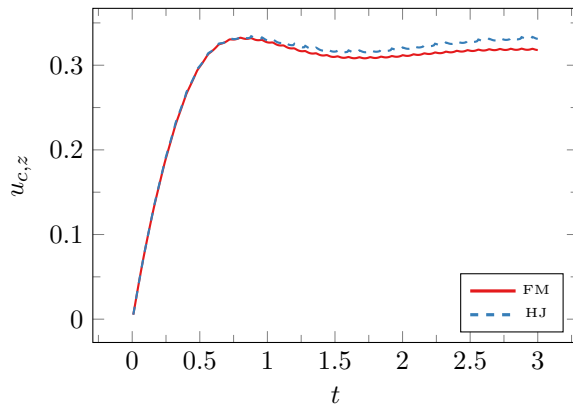
354 We can indeed observe that the final shapes of high-order simulations look smoother in both cases,
 355 as confirmed by the sphericity plots. The effect is highly noticeable on the “skirt” which appears for
 356 the second test-case, which looks even smoother than the one obtained with the finest ($h = 0.0125$)
 357 $(2, 1, 1)$ simulation.

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

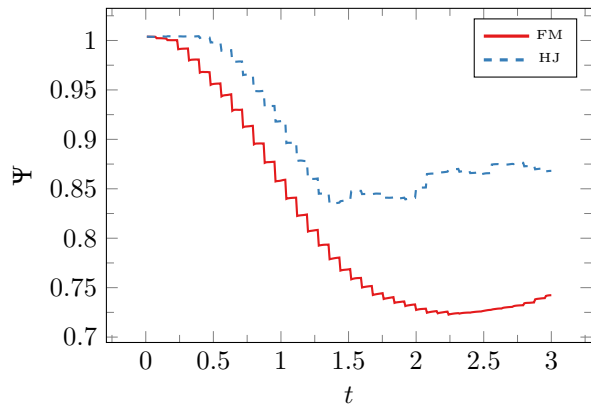
362 We can also notice that more “physically” controlled quantities, such as the position of
 363 the center-of-mass and the vertical velocity are less impacted by the polynomial order of the
 364 level-set component, which is not so surprising, as these quantities are mainly determined by the
 365 (level-set-dependent)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

366 6. Conclusion and outlooks

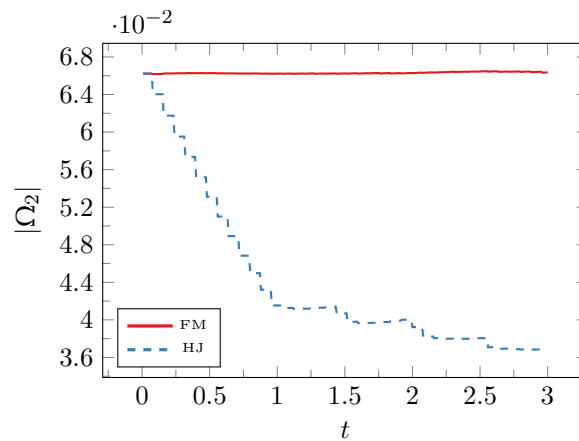
367 We have presented in this paper a comprehensive numerical framework for the simulation of
 368 multifluid flows. This framework is based on level-set methods solved by a (possibly high-order) finite
 369 element method. The explicit coupling between the level-set and the fluid has proven to be efficient

(a) Shape at final time ($t = 3$) in the vertical $x - z$ plane.(b) z_c center-of-mass vertical component.

(c) Vertical velocity.

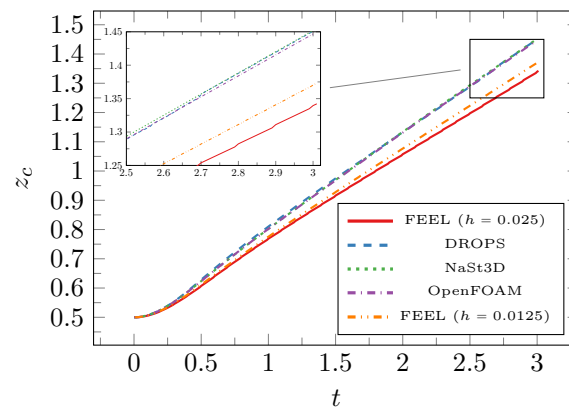
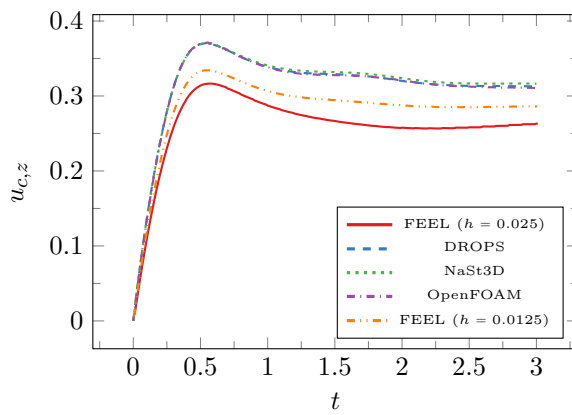


(d) Sphericity.

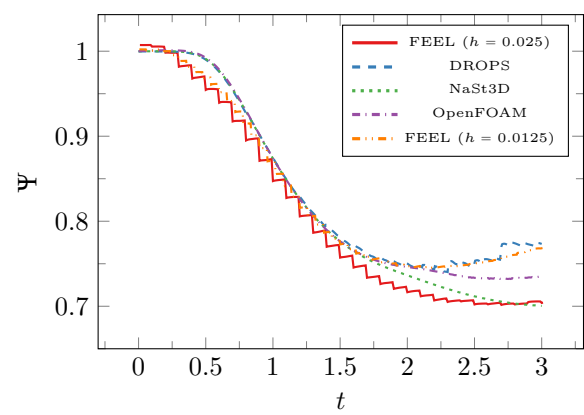


(e) Mass.

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 $h = 0.0175$.

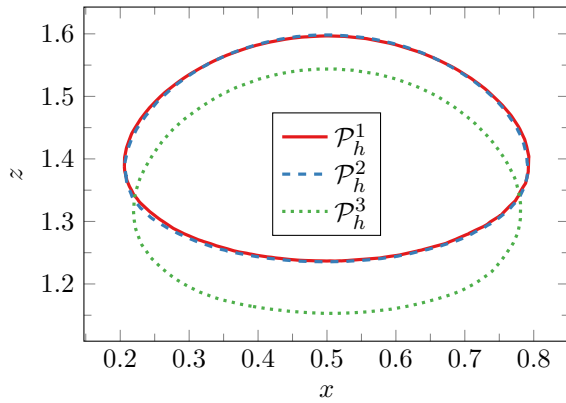
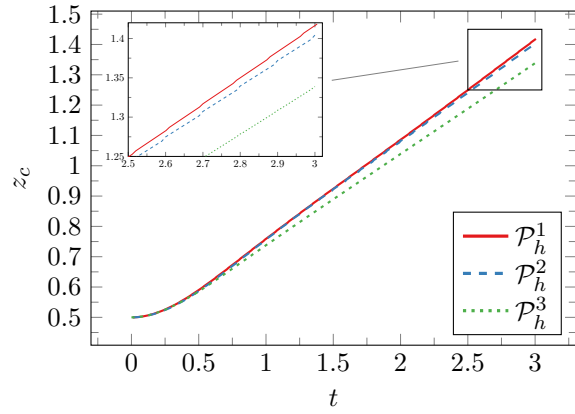
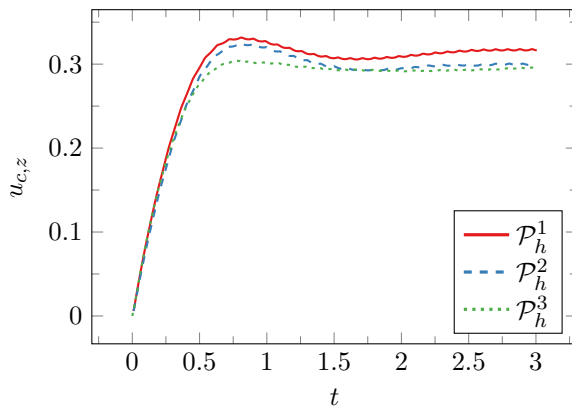
(a) z_c center-of-mass vertical component.

(b) Vertical velocity.

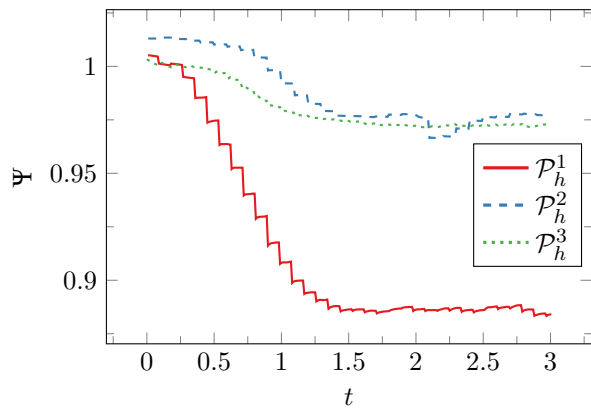


(c) Sphericity.

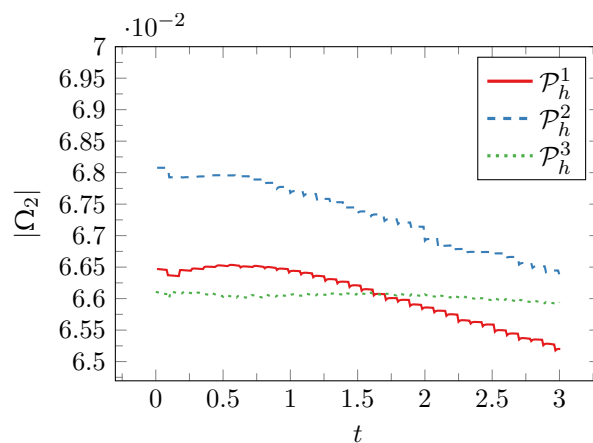
Figure 11. Comparison between our results (denoted FEEL) and the ones from [40] for the test case 2 (the skirted drop).

(a) Shape at final time ($t = 3$) in the vertical $x - z$ plane.(b) z_c center-of-mass vertical component.

(c) Vertical velocity.

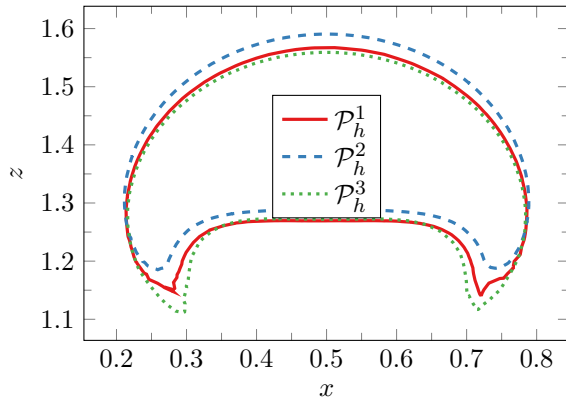
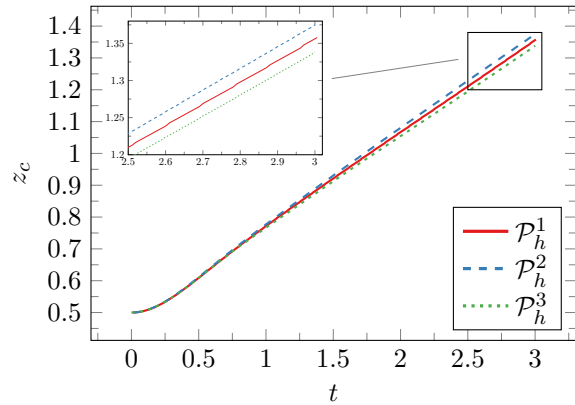
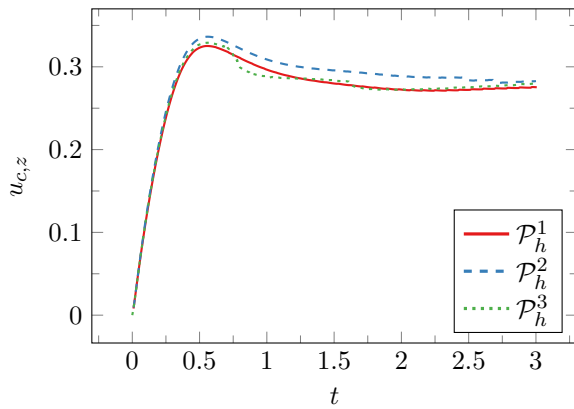


(d) Sphericity.

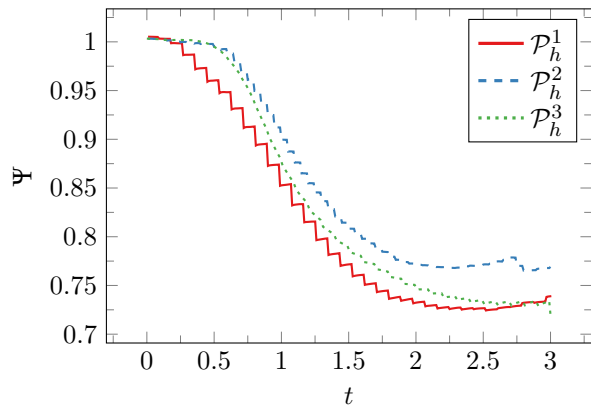


(e) Mass.

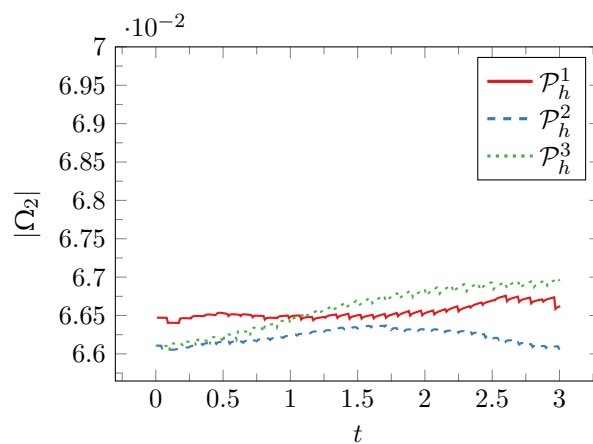
Figure 12. Comparison between \mathcal{P}_h^1 , \mathcal{P}_h^2 and \mathcal{P}_h^3 simulations for the ellipsoidal test case ($h = 0.02$).

(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 ($h = 0.02$).

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

372 The framework has been implemented within the FEEL++ toolboxes and leverages the efficiency
373 of the library to run on large numbers of processors in parallel. It also features user-friendly interfaces,
374 and allows for easy model setup and parametrization using the JSON and CFG standard formats. The
375 use of state-of-the-art metaprogramming techniques allows to seamlessly perform simulations in two or
376 three dimension, and to increase the polynomial order of the finite elements. We highlight again that
377 both the implementation and the benchmarks configuration files are open-source and available online
378 [10,11].

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

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

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

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

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

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

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

409

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

- 421 7. Prud'Homme, C.; Chabannes, V.; Doyeux, V.; Ismail, M.; Samake, A.; Pena, G. Feel++: A
422 Computational Framework for Galerkin Methods and Advanced Numerical Methods.
- 423 8. Prud'homme, C.; Chabannes, V.; Pena, G. Feel++: Finite Element Embedded Language in C++. Free
424 Software available at <http://www.feelpp.org>. Contributions from A. Samake, V. Doyeux, M. Ismail and
425 S. Veys.
- 426 9. Prud'homme, C. A domain specific embedded language in C++ for automatic differentiation, projection,
427 integration and variational formulations. *Scientific Programming* **2006**, *14*.
- 428 10. FEEL++ Consortium. Multifluid Flow Benchmarks. <http://docs.feelpp.org/cases/0.105/multifluid/>.
429 Accessed: 2018-08-31.
- 430 11. FEEL++ Consortium. Feel++ Github. <https://github.com/feelpp/feelpp>. Accessed: 2018-08-31.
- 431 12. Sethian, J. *Level Set Methods and Fast Marching Methods*; Cambridge University Press, 1996.
- 432 13. Stanley Osher, R.F. *Level Set Methods and Dynamic Implicit Surfaces*; Springer, S.S. Antman, J.E.
433 Marsden, L. Sirovich.
- 434 14. Rouy, E.; Tourin, A. A viscosity solutions approach to shape-from-shading. *SIAM Journal on Numerical
435 Analysis* **1992**, *29*, 867–884.
- 436 15. Sussman, M.; Smereka, P.; Osher, S. A Level Set Approach for Computing Solutions to
437 Incompressible Two-Phase Flow. *Journal of Computational Physics* **1994**, *114*, 146 – 159.
438 doi:<http://dx.doi.org/10.1006/jcph.1994.1155>.
- 439 16. Sethian, J.A. *Level set methods and fast marching methods: evolving interfaces in computational geometry,
440 fluid mechanics, computer vision, and materials science*; Vol. 3, Cambridge university press, 1999.
- 441 17. Chang, Y.C.; Hou, T.; Merriman, B.; Osher, S. A level set formulation of Eulerian interface capturing
442 methods for incompressible fluid flows. *Journal of computational Physics* **1996**, *124*, 449–464.
- 443 18. Cottet, G.H.; Maitre, E. A level set method for fluid-structure interactions with immersed surfaces.
444 *Mathematical models and methods in applied sciences* **2006**, *16*, 415–438.
- 445 19. Caldini Queiros, C.; Chabannes, V.; Ismail, M.; Pena, G.; Prud'Homme, C.; Szopos, M.; Tarabay, R.
446 Towards large-scale three-dimensional blood flow simulations in realistic geometries **2013-06**. pp. 17,
447 Accepted in ESAIM: Proc.
- 448 20. Chabannes, V.; Ismail, M.; Prud'Homme, C.; SZOPOS, M. Hemodynamic simulations in the cerebral
449 venous network: A study on the influence of different modeling assumptions. *Journal of Coupled Systems
450 and Multiscale Dynamics* **2015**, *3*, 23–37. doi:10.1166/jcsmd.2015.1062.
- 451 21. Daversin, C.; Prudhomme, C.; Trophime, C. Full 3D MultiPhysics Model of High Field
452 PolyHelices Magnets. *IEEE Transactions on Applied Superconductivity* **2016**, *26*, 1–4.
453 doi:10.1109/TASC.2016.2516241.
- 454 22. Daversin, C.; Prud'Homme, C. Simultaneous Empirical Interpolation and Reduced Basis Method:
455 Application to Non-linear Multi-Physics Problem. In *Model Reduction of Parametrized Systems*; Springer,
456 2017; pp. 17–35.
- 457 23. Quarteroni, A.; Quarteroni, S. *Numerical models for differential problems*; Vol. 2, Springer, 2009.
- 458 24. Brooks, A.N.; Hughes, T.J. Streamline upwind/Petrov-Galerkin formulations for convection dominated
459 flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer methods in
460 applied mechanics and engineering* **1982**, *32*, 199–259.
- 461 25. Franca, L.P.; Hughes, T.J. Two classes of mixed finite element methods. *Computer Methods in Applied
462 Mechanics and Engineering* **1988**, *69*, 89 – 129. doi:[http://dx.doi.org/10.1016/0045-7825\(88\)90168-5](http://dx.doi.org/10.1016/0045-7825(88)90168-5).
- 463 26. Hughes, T.J.; Franca, L.P.; Hulbert, G.M. A new finite element formulation for
464 computational fluid dynamics: VIII. The galerkin/least-squares method for advective-diffusive
465 equations. *Computer Methods in Applied Mechanics and Engineering* **1989**, *73*, 173 – 189.
466 doi:[http://dx.doi.org/10.1016/0045-7825\(89\)90111-4](http://dx.doi.org/10.1016/0045-7825(89)90111-4).
- 467 27. Hauke, G. A simple subgrid scale stabilized method for the advection–diffusion–reaction equation.
468 *Computer Methods in Applied Mechanics and Engineering* **2002**, *191*, 2925–2947.
- 469 28. Douglas, J.; Dupont, T. Interior penalty procedures for elliptic and parabolic Galerkin methods. In
470 *Computing methods in applied sciences*; Springer, 1976; pp. 207–216.
- 471 29. Chaple, R.P.B. *Numerical stabilization of convection-diffusion-reaction problems*; Delft University of
472 Technology, 2006.

- 473 30. Codina, R. Comparison of some finite element methods for solving the diffusion-convection-reaction
474 equation. *Computer Methods in Applied Mechanics and Engineering* **1998**, *156*, 185–210.
- 475 31. Winkelmann, C. Interior penalty finite element approximation of Navier-Stokes equations and application
476 to free surface flows. PhD thesis, 2007.
- 477 32. Tezduyar, T.E.; Osawa, Y. Finite element stabilization parameters computed from element matrices and
478 vectors. *Computer Methods in Applied Mechanics and Engineering* **2000**, *190*, 411–430.
- 479 33. Franca, L.P.; Valentin, F. On an improved unusual stabilized finite element method for the
480 advective–reactive–diffusive equation. *Computer Methods in Applied Mechanics and Engineering*
481 **2000**, *190*, 1785–1800.
- 482 34. Balay, S.; Gropp, W.D.; McInnes, L.C.; Smith, B.F. Efficient Management of Parallelism in
483 Object-Oriented Numerical Software Libraries. *Modern Software Tools in Scientific Computing*.
484 Birkhauser Press, 1997, pp. 163–202.
- 485 35. Balay, S.; Abhyankar, S.; Adams, M.F.; Brown, J.; Brune, P.; Buschelman, K.; Dalcin, L.; Eijkhout, V.;
486 Gropp, W.D.; Kaushik, D.; Knepley, M.G.; May, D.A.; McInnes, L.C.; Mills, R.T.; Munson, T.; Rupp,
487 K.; Sanan, P.; Smith, B.F.; Zampini, S.; Zhang, H.; Zhang, H. PETSc Users Manual. Technical Report
488 ANL-95/11 - Revision 3.9, Argonne National Laboratory, 2018.
- 489 36. Doyeux, V. Modélisation et simulation de systèmes multi-fluides. Applications aux écoulements sanguins.
490 PhD thesis, 2014. Thèse de doctorat dirigée par Peyla, Philippe et Ismail, Mourad Physique Grenoble
491 2014.
- 492 37. Chabannes, V.; Pena, G.; C.Prud’homme. High order fluid structure interaction in 2D and 3D.
493 Application to blood flow in arteries. Fifth International Conference on Advanced COmputational
494 Methods in ENgineering (ACOMEN 2011), 2011.
- 495 38. Hysing, S.; Turek, S.; Kuzmin, D.; Parolini, N.; Burman, E.; Ganesan, S.; Tobiska, L. Quantitative
496 benchmark computations of two-dimensional bubble dynamics. *International Journal for Numerical*
497 *Methods in Fluids* **2009**, *60*, 1259–1288. doi:10.1002/flid.1934.
- 498 39. Doyeux, V.; Guyot, Y.; Chabannes, V.; Prud’Homme, C.; Ismail, M. Simulation of two-fluid flows using a
499 finite element/level set method. Application to bubbles and vesicle dynamics. *Journal of Computational*
500 *and Applied Mathematics* **2013**, *246*, 251–259.
- 501 40. Adelsberger, J.; Esser, P.; Griebel, M.; Groß, S.; Klitz, M.; Rüttgers, A. 3D incompressible two-phase flow
502 benchmark computations for rising droplets. Proceedings of the 11th World Congress on Computational
503 Mechanics (WCCM XI), Barcelona, Spain, 2014, Vol. 179.
- 504 41. Patankar, S.V.; Spalding, D.B. A calculation procedure for heat, mass and momentum transfer in
505 three-dimensional parabolic flows. *International journal of heat and mass transfer* **1972**, *15*, 1787–1806.