Computer Physics Communications 308 (2025) 109460

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Computer Programs in Physics

twoPhaseInterTrackFoam: An OpenFOAM module for arbitrary Lagrangian/Eulerian interface tracking with surfactants and subgrid-scale modeling ☆

Moritz Schwarzmeier [a], Suraj Raju [a], Željko Tuković [b], Mathis Fricke [a], Dieter Bothe [a], Tomislav Marić [a,*]

[a] Mathematical Modeling and Analysis Institute, Mathematics department, TU Darmstadt, Germany
[b] Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Croatia

ARTICLE INFO

Keywords:
Finite volume
Interface tracking
ALE
Unstructured mesh
Subgrid-scale

ABSTRACT

We provide an implementation of the unstructured Finite-Volume Arbitrary Lagrangian / Eulerian (ALE) Interface-Tracking method for simulating incompressible, immiscible two-phase flows as an OpenFOAM module. In addition to interface-tracking capabilities that include tracking of two fluid phases, an implementation of a Subgrid-Scale (SGS) modeling framework for increased accuracy when simulating sharp boundary layers is enclosed. The SGS modeling framework simplifies embedding subgrid-scale profiles into the unstructured Finite Volume discretization. Our design of the SGS model library significantly simplifies adding new SGS models and applying SGS modeling to Partial Differential Equations (PDEs) in OpenFOAM.

Program summary
Program title: twoPhaseInterTrackFoam
CPC Library link to program files: https://doi.org/10.17632/6b49wb7fvd.1
Developer's repository link: https://gitlab.com/interface-tracking/twophaseintertrackfoamrelease
Licensing provisions: GPLv3
Programming language: C++
Nature of problem: Two-phase flow problems involving surface-active agents (surfactants), variable surface tension force and very sharp boundary layers.
Solution method: An OpenFOAM implementation of the Arbitrary Lagrangian / Eulerian Interface Tracking method.

1. Introduction

The unstructured Finite Volume Arbitrary Lagrangian / Eulerian Interface Tracking method in OpenFOAM (ALE-IT) [39,37] is a highly accurate method that tracks fluid phases as deforming solution domains separated by fluid interfaces in the form of deforming domain boundaries (cf. Fig. 1). Modeling fluid interfaces using domain boundaries makes it possible to directly apply jump conditions at fluid interfaces as boundary conditions. Additionally, it is possible to solve Partial Differential Equations (PDEs) on the fluid interface, by discretizing surface transport equations using the discretization of the domain boundary. Our implementation of the ALE-IT method, which we provide in this manuscript, has been successfully applied to hydrodynamically challenging two-phase flow problems [25], two-phase flows with soluble surfactant [11,12,29] and two-phase flows with interfacial mass transfer [41,30]. This type of the ALE-IT method has also been extended to handle large mesh deformations by allowing for topological changes on tetrahedral meshes, driven by mesh quality criteria [31,26]. However,

☆ The review of this paper was arranged by Prof. Andrew Hazel.

* Corresponding author.
E-mail addresses: schwarzmeier@mma.tu-darmstadt.de (M. Schwarzmeier), raju@mma.tu-darmstadt.de (S. Raju), zeljko.tukovic@fsb.unizg.hr (Ž. Tuković), fricke@mma.tu-darmstadt.de (M. Fricke), bothe@mma.tu-darmstadt.de (D. Bothe), maric@mma.tu-darmstadt.de (T. Marić).

https://doi.org/10.1016/j.cpc.2024.109460
Received 29 March 2024; Received in revised form 1 November 2024; Accepted 28 November 2024

Available online 2 December 2024

transferring the methodology from [26] into our implementation of the ALE-IT method is a complex task, which we will address in our future work.

Mass transfer across the fluid interface often results in extremely steep concentration gradients, that cannot be resolved by modern numerical simulation methods at affordable computational costs. The Subgrid-Scale (SGS) modeling approach [1,4,42] employs a local analytical solution to enhance the computation accuracy for the passive scalar transport in interfacial boundary layers, saving multiple local refinement levels in the Finite Volume mesh. This speeds up simulations significantly, in some cases enabling them in the first place, in other cases saving computational resources, while delivering results with very high accuracy. The topic of Subgrid-Scale modeling for transport processes at fluid interfaces is gaining attention and some modifications have been made to include reactive species [17] or local curvature effects [16]. While the SGS modeling addressed in this paper is implemented in the ALE-IT method, others have implemented it in the VoF method, e.g. [42,6], or in the Front Tracking method [16,8].

With this publication, we provide an open-source implementation of our ALE-IT method as an OpenFOAM module, which simplifies the development of SGS models for interfacial transport phenomena in OpenFOAM. Researchers that do not use OpenFOAM will be able to analyze our implementation details, which is very helpful for developing these methods.

The version of the ALE-IT method from this manuscript is *v1.1*, available in the GitLab repository [33] and archived as [38]. Results reported in this paper are archived as [34].

The advantages of our ALE-IT method are high simulation accuracy of two-phase flows involving incompressible fluid phases with strong differences in densities, strong influence of surface tension forces, effects of variable surface tension coefficients, transport of species concentrations on and across the fluid interface, and the Subgid-Scale modeling of mass transfer at fluid interfaces.

This version of the ALE-IT method is limited to moderate deformations of the fluid interface and does not allow for topological changes of the fluid interface, which will be addressed in our future work. Compared to other two-phase flow simulation methods, including those available in OpenFOAM, our ALE-IT method excells in accuracy, under above mentioned limitations.

The following sections cover the details of the mathematical model, equation discretization, and Subgrid-Scale modeling implemented in the version *1.1* of our ALE-IT OpenFOAM module, as well as the results for some showcases.

2. Mathematical model

2.1. Arbitrary Lagrangian/Eulerian (sharp) interface (tracking) model

We model isothermal flow of two immiscible incompressible fluids, schematically shown in Fig. 1, e.g. as a liquid and a gas phase separated by a sharp interface $\Sigma(t)$. Appropriate boundary conditions are then enforced at the fluid interface $\Sigma(t)$, see [39]. We only briefly outline the model, the reader is additionally referred to [18] for more details.

Isothermal flow of an incompressible Newtonian fluid inside an arbitrary moving control volume V bounded by a closed surface $S := \partial V$, i.e. S is the boundary of the finite volume, moving with the boundary velocity $\mathbf{v}_s$ is governed by the mass and linear momentum conservation laws:

$$\oint_S \rho \mathbf{v} \cdot \mathbf{n} \, dS = 0, \tag{1}$$

$$\frac{\partial}{\partial t} \int_V \rho \mathbf{v} \, dV + \oint_S \rho((\mathbf{v} - \mathbf{v}_s) \otimes \mathbf{v}) \cdot \mathbf{n} \, dS = \oint_S \boldsymbol{\tau} \cdot \mathbf{n} \, dS - \int_V \nabla p \, dV, \tag{2}$$

where $\mathbf{n}$ is the outward pointing unit normal on S, ρ is the fluid density, $\mathbf{v}$ is the fluid velocity, $\mathbf{v}_s$ is the velocity of surface S, and $\boldsymbol{\tau}$ is the viscous stress tensor which together with the thermodynamic pressure p' makes the Cauchy stress tensor $\mathbf{T} = -p'\mathbf{I} + \boldsymbol{\tau}$. The dynamic pressure in Eq. (2) is defined as

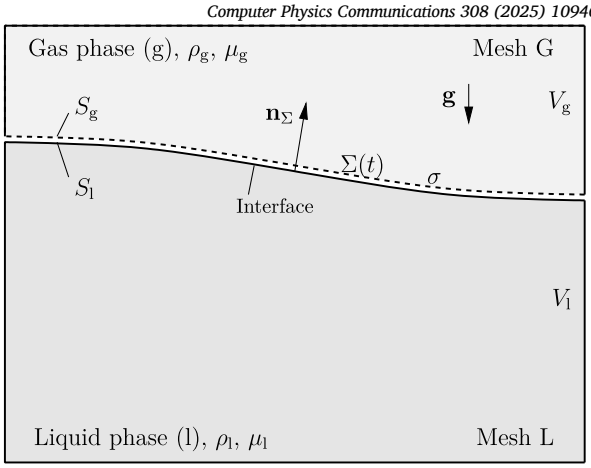


Fig. 1. Definition of the solution domain for the ALE-IT method.

$$p = p' - \rho \mathbf{g} \cdot \mathbf{r}, \tag{3}$$

where $\mathbf{g}$ is the gravitational acceleration vector and $\mathbf{r}$ is the position vector. Furthermore, the viscous stress tensor of the Newtonian fluid is given by

$$\boldsymbol{\tau} = \eta(\nabla \mathbf{v} + \nabla \mathbf{v}^T), \tag{4}$$

where η is the dynamic viscosity.

The relationship between the rate of change of the volume V and the velocity $\mathbf{v}_s$ is defined by the *geometric (space) conservation law* (GCL, see [35,10]):

$$\frac{\partial}{\partial t} \int_V dV - \oint_S \mathbf{v}_s \cdot \mathbf{n} \, dS = 0. \tag{5}$$

The formulation of the above mathematical model is a well-known ALE formulation.

2.2. Interface coupling conditions

If fluid phases are immiscible, fluid flow eqs. (1) and (2) can be applied for each phase separately, while on the interface the proper boundary conditions must be used. The relation between fluid velocities on the two sides of the interface is determined by the *kinematic condition*, which states that the velocity must be continuous across the interface:

$$\mathbf{v}_l = \mathbf{v}_g, \tag{6}$$

where $\mathbf{v}_l$ and $\mathbf{v}_g$ are the fluid velocities at the two sides (e.g. liquid and gas) of the fluid interface.

The dynamic condition follows from the momentum conservation law and states that forces acting on the fluid at the interface are in equilibrium:

$$(p_g - p_l)\mathbf{n}_\Sigma - (\boldsymbol{\tau}_g - \boldsymbol{\tau}_l) \cdot \mathbf{n}_\Sigma = \sigma \kappa \mathbf{n}_\Sigma + \nabla_s \sigma - (\rho_g - \rho_l)(\mathbf{g} \cdot \mathbf{r})\mathbf{n}_\Sigma, \tag{7}$$

where $\mathbf{n}_\Sigma$ is the unit normal vector on the interface, which points from the liquid phase to the gas phase, $\kappa = -\nabla_s \cdot \mathbf{n}_\Sigma$ is twice the mean curvature of the interface, σ is the surface tension coefficient and $\nabla_s \sigma$ is the gradient of the surface tension coefficient, where $\nabla_s = \nabla(\mathbf{I} - \mathbf{n}_\Sigma \mathbf{n}_\Sigma)$ is the surface gradient operator. The last term on the right-hand side of eq. (7) appears due to transformation of the thermodynamic pressure to the dynamic pressure at the interface. The pressure jump across the in-

terface is obtained by taking the normal component of the force balance (eq. (7)):

$$p_g - p_l = \sigma\kappa - 2\left(\eta_g - \eta_l\right)\nabla_s \cdot \mathbf{v} - (\rho_g - \rho_l)(\mathbf{g}\cdot\mathbf{r}), \tag{8}$$

The second term on the right-hand side of eq. (8) represents the jump of the normal viscous force across the interface, expressed through the surface divergence of the interface velocity. For example, the normal viscous force at the interface can be expressed as (see [7]):

$$(\mathbf{n}_\Sigma \otimes \mathbf{n}_\Sigma) : \boldsymbol{\tau} = 2\eta(\mathbf{n}_\Sigma\mathbf{n}_\Sigma) : \nabla\mathbf{v} = -2\eta\nabla_s \cdot \mathbf{v}, \tag{9}$$

where the following identity, being valid for an incompressible fluid flow ($\nabla \cdot \mathbf{v} = 0$), is used:

$$\nabla \cdot \mathbf{v} = \nabla_s \cdot \mathbf{v} + (\mathbf{n}_\Sigma \otimes \mathbf{n}_\Sigma) : \nabla\mathbf{v}. \tag{10}$$

By taking the tangential component of the force balance (eq. (7)) one obtains a relation between the normal derivative of the tangential velocity on the two sides of the interface:

$$\eta_g\left(\nabla\mathbf{v}_t \cdot \mathbf{n}_\Sigma\right)_g - \eta_l\left(\nabla\mathbf{v}_t \cdot \mathbf{n}_\Sigma\right)_l = -\nabla_s\sigma - \left(\eta_g - \eta_l\right)\left(\nabla_s v_n\right), \tag{11}$$

where it is taken into account that the tangential component of the viscous force at the interface can be expressed as follows:

$$(\boldsymbol{\tau} \cdot \mathbf{n}_\Sigma) \cdot (\mathbf{I} - \mathbf{n}_\Sigma \otimes \mathbf{n}_\Sigma) = \eta\left(\nabla\mathbf{v}_t \cdot \mathbf{n}_\Sigma + \nabla_s v_n\right), \tag{12}$$

where $\mathbf{v}_t = (\mathbf{I} - \mathbf{n}_\Sigma \otimes \mathbf{n}_\Sigma) \cdot \mathbf{v}$ is the tangential velocity component and $v_n = \mathbf{n}_\Sigma \cdot \mathbf{v}$ is the normal velocity component at the interface. A non-zero gradient $\nabla_s\sigma$ of the surface tension coefficient can occur for example due to a nonuniform distribution of surfactant at the interface or due to the presence of a temperature gradient.

2.3. Surfactant transport

If impact of surfactant on the flow field is to be analyzed, it is necessary to extend the mathematical model by an equation which governs the transport of surfactant in the liquid bulk and an equation which governs the transport of surfactant along the interface. The bulk surfactant transport equation in ALE form reads as follows:

$$\frac{\partial}{\partial t}\int_V c\, dV + \oint_S (\mathbf{v} - \mathbf{v}_s)c \cdot \mathbf{n}\, dS - \oint_S (D\nabla c) \cdot \mathbf{n}\, dS = 0, \tag{13}$$

where c is the volume-averaged bulk surfactant concentration and D is the bulk surfactant diffusion concentration.

The governing equation for surfactant transport along a surface patch S attached to the fluid interface ($S \subset \Sigma(t)$) and bounded by a closed curve ∂S which can move along the interface reads:

$$\frac{\partial}{\partial t}\int_S \Gamma\, dS + \oint_{\partial S} (\mathbf{v} - \mathbf{b}) \cdot \mathbf{m}\,\Gamma\, dL - \oint_{\partial S} (D_\Gamma\nabla_s\Gamma) \cdot \mathbf{m}\, dL = \int_S s_\Gamma\, dS, \tag{14}$$

where Γ is the surfactant concentration at the interface, Γ_∞ is the saturated surfactant concentration under the given thermodynamic conditions, $\mathbf{m}$ is the outward pointing unit bi-normal on ∂S, $\mathbf{b}$ is the velocity at which the curve ∂S moves along the interface, L is the arc length measured along ∂S, D_s is the diffusion coefficient of the surfactant along the interface and s_Γ is the source/sink of surfactants per unit area due to adsorption and desorption.

For the sake of simplicity, we focus in this paper on the prominent Langmuir's kinetics law [21,5,29] for the transfer of surfactant between the bulk and the interface due to adsorption and desorption:

$$s_\Gamma = k_a\left[c_l(\Gamma_\infty - \Gamma) - \beta\Gamma\right], \tag{15}$$

where k_a and β are the parameters of the adsorption and desorption kinetics, respectively, and c_l is the value of the bulk surfactant concentration at the interface. The implementation allows for a straightforward extension to cope with other sorption models. The normal derivative of bulk surfactant concentration at the interface is given by following expression:

$$(\nabla c \cdot \mathbf{n})_\Sigma = -\frac{s_\Gamma}{D}. \tag{16}$$

In order to account for the surfactant bulk-interface mass transfer balance (eq. (16)), the normal derivative of the surfactant concentration at the interface needs to be calculated with sufficient accuracy. Due to convection along the interface and typically small surfactant diffusivities, extremely thin boundary layers appear and make this calculation a challenging step. At this point, use of Subgrid-Scale modeling for improved flux calculation as described in section 4 is beneficial.

The surface tension is related to the surfactant concentration on the interface and given for the Langmuir sorption model with its adsorption isotherm by the following equation of state

$$\sigma = \sigma_0 + \mathcal{R}T\Gamma_\infty \ln\left(1 - \frac{\Gamma}{\Gamma_\infty}\right), \tag{17}$$

where σ_0 is the surface tension of a clean interface, $\mathcal{R}$ is the universal gas constant and T is the temperature [21,5,29]. Equation (17) can also be written as

$$\sigma = \sigma_0\left[1 + \beta_s \ln\left(1 - \frac{\Gamma}{\Gamma_\infty}\right)\right], \tag{18}$$

where $\beta_s = \frac{\mathcal{R}T\Gamma_\infty}{\sigma_0}$ is the elasticity number.

3. Finite volume and finite area discretisation and equation-system solution

3.1. Discretisation of the computational domain

The cell-centered Finite Volume Method (FVM, [39]) is used to discretise the bulk fluid models (volumetric transport equations) and the face-centered Finite Area Method (FAM, [39]) is used to discretise the surfactant transport model, a transport equation on the fluid interface (moving surface). This discretization is sketched in Figs. 5 and 7 and can be seen for a case with a bubble in Fig. 19. The finite volume/area discretisation is based on the integral form of the conservation equation over a fixed or moving control volume/area. The discretisation procedure is divided into two parts: discretisation of the computational domain and the equation discretisation. Here we provide a brief description of both discretisation methods together with a description of the interface tracking method, while more details can be found in [39,37,36].

The time interval is split into a finite number of time-steps Δt and the equations are solved in a time-marching manner. The computational space is divided into a finite number of polyhedral control volumes (CV) or cells bounded by polygons. The cells do not overlap and fill the spatial domain completely. Fig. 2 shows a polyhedral control volume V_P around the computational point P located in its centroid, face f with area S_f, face unit normal vector $\mathbf{n}_f$ and the centroid N of a neighboring CV sharing the face f. The geometry of the CV is fully determined by the position of its vertices.

In case the solution domain is changing in time, e.g., when the fluid interface deforms, the finite volume mesh is adjusted to the time-varying position or shape of the solution domain boundaries using a deforming mesh approach. The internal CV-vertices are moved based on the prescribed motion of the boundary vertices, while the topology of the mesh stays unchanged. In the implemented moving mesh interface tracking solver, a finite-volume automatic mesh motion solver is applied, where the Laplace equation is solved for mesh-point displacements with variable diffusion coefficient, using the polyhedral unstructured cell-centered FVM.

The polygonal cell-faces which coincide with the interface constitute the finite area mesh on which the surfactant transport equation is discretised. Fig. 3 shows a sample polygonal control area S_P around

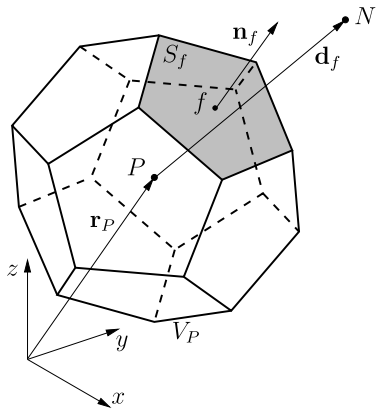


Fig. 2. Polyhedral control volume (cell) and the notation used (adopted from [39]).

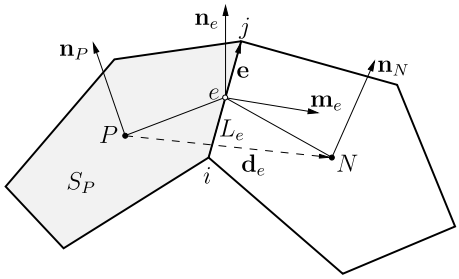


Fig. 3. Polygonal control area.

the computational point P located in its centroid, the edge e, the edge length L_e, the edge unit bi-normal vector $\mathbf{m}_e$ and the centroid N of the neighboring control area sharing the edge e. The bi-normal $\mathbf{m}_e$ is perpendicular to the edge normal $\mathbf{n}_e$ and to the edge vector $\mathbf{e}$.

3.2. Discretisation of volumetric equations

The second-order collocated FV discretisation of an integral conservation equation transforms the surface integrals into sums of face integrals and approximates them and the volume integrals to the second order accuracy. Error estimates for the unstructured FVM have been analyzed in detail, e.g., in [19,20]. We rely on second-order divergence and Laplace term discretization and second-order interpolation schemes, denote approximated quantities with $\approx$ and omit the error terms in the description below for clarity, to place focus on the discretization. The spatially discretised form of the momentum eq. (2) for the moving control volume V_P reads:

$$\frac{\partial\left(\mathbf{v}_P V_P\right)}{\partial t}+\sum_f(\dot{V}_f-\dot{V}_{s,f})\mathbf{v}_f = \sum_f \nu_f\left[(\nabla\mathbf{v})\cdot\mathbf{n}\right]_f S_f+(\nabla p)_P V_P, \tag{19}$$

where the subscripts $_P$ and $_f$ represent the cell-center and face-center values. The volume flow rate due to fluid flow through the cell-face, $\dot{V}_f^{n+1}=(\mathbf{n}\cdot\mathbf{v}S)_f^{n+1}$ must satisfy the discretised mass conservation law, where the superscript $^{n+1}$ represents the new time-step, while the volume flux due to grid motion, $\dot{V}_{s,f}^{n+1}$, must satisfy the discretised GCL, eq. (5). The evaluation of the mesh-face volume fluxes is described in [13,37].

The face-center values of all dependent variables are calculated using linear interpolation of the neighboring cell-center values. The exception is the face value of the dependent variable in the convection term [$\mathbf{v}_f$ in eq. (19)] which must be calculated using some of the bounded convection discretisation schemes available in OpenFOAM.

The face-normal derivative of the velocity $[\mathbf{n}\cdot\nabla\mathbf{v}]_f$ is discretised using the linear scheme with non-orthogonal and skewness correction (see Fig. 2):

$$(\nabla\mathbf{v}\cdot\mathbf{n})_f\approx\frac{\mathbf{v}_N-\mathbf{v}_P}{d_{\mathrm{n},f}}+\frac{(\nabla\mathbf{v})_N\cdot\mathbf{k}_N-(\nabla\mathbf{v})_P\cdot\mathbf{k}_P}{d_{\mathrm{n},f}}, \tag{20}$$

where $d_{\mathrm{n},f}=\mathbf{n}_f\cdot\mathbf{d}_f$, $\mathbf{k}_P=(\mathbf{I}-\mathbf{n}_f\otimes\mathbf{n}_f)\cdot(\mathbf{r}_f-\mathbf{r}_P)$ and $\mathbf{k}_N=(\mathbf{I}-\mathbf{n}_f\otimes\mathbf{n}_f)\cdot(\mathbf{r}_N-\mathbf{r}_f)$. The first term on the right-hand side of eq. (20) is treated implicitly, while the correction term is treated explicitly.

The cell-face volumetric fluid flux in the non-linear convection term is treated explicitly after the discretisation, i.e. its values from the previous iteration are used. The cell-center gradient of the dependent variables used for the calculation of the explicit source terms and the non-orthogonal correction in eq. (20), is obtained by using the least-squares fit [9]. This method produces a second-order accurate gradient irrespective of the mesh quality.

The **temporal discretisation** of all equations is performed by using the second-order accurate implicit three time level scheme [13] referred to as the `backward` scheme in OpenFOAM. All terms of eq. (19) are evaluated at the new time instance $t^{n+1}=t^n+\Delta t$ and the temporal derivative is discretised by using two old-time levels:

$$\left(\frac{\partial\mathbf{v}}{\partial t}\right)^{n+1}\approx\frac{3\mathbf{v}^{n+1}-4\mathbf{v}^n+\mathbf{v}^{n-1}}{2\Delta t}, \tag{21}$$

where $\mathbf{v}^{n+1}=\mathbf{v}(t^n+\Delta t)$, $\mathbf{v}^n=\mathbf{v}(t^n)$ and $\mathbf{v}^{n-1}=\mathbf{v}(t^n-\Delta t)$. One can also use other temporal discretisation schemes available in OpenFOAM.

When eqs. (20) and (21) are substituted into eq. (19) and the convection discretisation scheme is applied, the discretised form of the momentum eq. (2) can be written in the form of a linear algebraic equation, which for cell P reads:

$$a_P\,\mathbf{v}_P^{n+1}+\sum_N a_N\,\mathbf{v}_N^{n+1}=\mathbf{r}_P+(\nabla p)_P, \tag{22}$$

where the diagonal coefficient a_P, the off-diagonal coefficients a_N and the source term $\mathbf{r}_P$ can be found in [37].

The mathematical model of fluid flow is solved using a segregated solution procedure where the discretised momentum equation is solved decoupled from the discretised pressure equation. The discretised pressure Poisson equation, obtained by combining the discretised momentum and continuity equations, reads:

$$\sum_f\left(\frac{1}{a}\right)_f(\mathbf{n}\cdot\nabla p)_f^{n+1}\,S_f^{n+1}=\sum_f\mathbf{n}_f^{n+1}\cdot\left(\frac{\mathbf{H}}{a}\right)_f S_f^{n+1}, \tag{23}$$

where the $(1/a)_f$ and $(\mathbf{H}/a)_f$ terms are calculated by using the temporally consistent Rhie-Chow interpolation procedure proposed in [37].

The face normal derivative of the pressure at the left hand side of eq. (23) is discretised using eq. (20) applied on the pressure field. The absolute volume fluid flux $\dot{V}_f^{n+1}$ through the cell face f is calculated as follows:

$$\dot{V}_f^{n+1}=\mathbf{n}_f^{n+1}\cdot\left[\left(\frac{\mathbf{H}}{a}\right)_f-\left(\frac{1}{a}\right)_f(\nabla p)_f^{n+1}\right]S_f^{n+1}. \tag{24}$$

Volume flux calculated in this manner will satisfy the discretised continuity equation if the pressure field satisfies the pressure eq. (23).

3.3. Discretisation of surface equations

Applying a second-order collocated finite area method, the surfactant transport eq. (14) can be discretised on the moving control area S_P (see Fig. 3) as follows:

$$\frac{\partial\left(\Gamma_P S_P\right)}{\partial t}+\sum_e(\mathbf{m}\cdot\mathbf{v})_e L_e\Gamma_e=\sum_e D_{\Gamma,e}(\mathbf{m}\cdot\nabla_s\Gamma)_e L_e+(s_\Gamma)_P S_P, \tag{25}$$

where the subscripts $_P$ and $_e$ represent the face-center and edge-center values, and it is assumed that finite-area points move in normal direction which means that $(\mathbf{m}\cdot\mathbf{b})_e=0$.

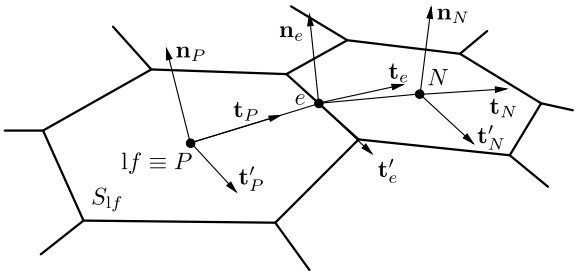


Fig. 4. Edge-based local orthogonal coordinate system whose axis are aligned with the orthogonal unit vectors $\mathbf{n}$, $\mathbf{t}$ and $\mathbf{t}'$, where vector $\mathbf{t}$ is tangential to the geodetic line $\overline{PeN}$.

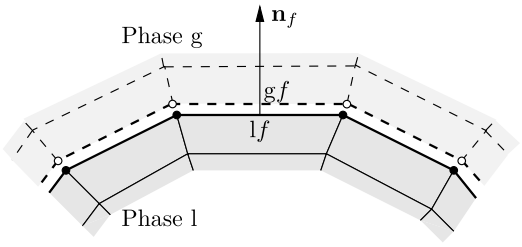


Fig. 5. Representation of the interface with the mesh boundary faces.

The edge-center tangential velocity $\mathbf{v}_{\mathrm{t}}$ is calculated using following linear interpolation formula:

$$\mathbf{v}_{\mathrm{t},e} = (\mathbf{T}_e)^{\mathrm{T}} \cdot \left[e_x \mathbf{T}_P \cdot \mathbf{v}_{\mathrm{t},P} + (1 - e_x) \mathbf{T}_N \cdot \mathbf{V}_{\mathrm{t},N} \right], \tag{26}$$

where e_x is the interpolation factor which is calculated as the ratio of the geodetic distances $\overline{eN}$ and $\overline{PeN}$ (see Fig. 3):

$$e_x = \frac{\overline{eN}}{\overline{PeN}}, \tag{27}$$

and $\mathbf{T}_P$, $\mathbf{T}_N$ and $\mathbf{T}_e$ are the tensors of transformation from the global Cartesian coordinate system to the $(\mathbf{t}'_e, \mathbf{t}_e, \mathbf{n}_e)$ edge-based local coordinate system defined in Fig. 3. The convection term in eq. (25) is discretised using the linear upwind discretisation scheme by taking into account geodetic distances between the neighboring control area centers. The diffusion term is discretised using the central differencing scheme with non-orthogonal correction [19]:

$$(\mathbf{m} \cdot \nabla_s \psi)_e = \underbrace{|\boldsymbol{\Delta}_e| \frac{\psi_N - \psi_P}{L_{PN}}}_{\text{Orthogonal contribution}} + \underbrace{\mathbf{k}_e \cdot (\nabla_s \psi)_e}_{\text{Non-orthogonal correction}}, \tag{28}$$

where $\mathbf{k}_e = \mathbf{m}_e - \boldsymbol{\Delta}_e$, $\boldsymbol{\Delta}_e = \frac{\mathbf{t}_e}{\mathbf{t}_e \cdot \mathbf{m}_e}$, L_{PN} is the geodetic distance $\overline{LeP}$ and $\mathbf{t}_e$ is the unit tangential vector to the geodetic line $\overline{PeN}$ at the point e (see Fig. 4).

3.4. Interface tracking procedure

The numerical modeling of the two-phase fluid flow with a sharp interface is performed using a moving mesh interface tracking procedure. The computational mesh consists of two separate parts, where each of the meshes covers only one of the considered two fluid phases, see Fig. 1. The two meshes are in contact over two geometrically equal surfaces, S_{l} and S_{g}, at the boundary between the phases, i.e. the interface. Surface S_{l} represents the liquid side of the interface, and surface S_{g} represent the gas side of the interface. Each surface is defined by a set of boundary faces, see Fig. 5, where each face $\mathrm{l}f$ on the surface S_{l} has a corresponding geometrically equal face $\mathrm{g}f$ on the surface S_{g}. Matching of the two meshes at the interface is assumed in order to make the explanation of the interface tracking method clearer, and is not required in general.

Coupling of flow equations between fluid phases is performed by applying adequate boundary conditions at the boundary faces which define the side l and g of the interface. At the side l the pressure p_{l} and normal velocity derivative $(\nabla \mathbf{v} \cdot \mathbf{n})_{\mathrm{l}}$ are specified, while at the side g the velocity $\mathbf{v}_{\mathrm{g}}$ is specified and normal derivative of dynamic pressure is set to zero, $(\mathbf{n} \cdot \nabla p)_{\mathrm{g}} = 0$. The boundary conditions are calculated using the kinematic and dynamic conditions as follows:

1. The value of dynamic pressure specified on the face $\mathrm{l}f$ (see Fig. 5) is calculated from the dynamic pressure at the face $\mathrm{g}f$ using eq. (8) as follows:

$$\begin{aligned} p_{\mathrm{l}f} = {} & p_{\mathrm{g}f} - (\rho_{\mathrm{l}} - \rho_{\mathrm{g}}) \mathbf{g} \cdot \mathbf{r}_{\mathrm{l}f} \\ & - (\sigma\kappa)_{\mathrm{l}f} - 2 \left(\eta_{\mathrm{l}} - \eta_{\mathrm{g}} \right) (\nabla_s \cdot \mathbf{v})_{\mathrm{l}f}, \end{aligned} \tag{29}$$

   where $p_{\mathrm{g}f}$ is the dynamic pressure at the face $\mathrm{g}f$ calculated by extrapolation from the fluid g, and $\mathbf{r}_{\mathrm{l}f}$ is the position vector of the face center $\mathrm{l}f$. The surface divergence of the velocity vector $(\nabla_s \cdot \mathbf{v})_{\mathrm{l}f}$ at $\mathrm{l}f$ is calculated using the surface Gauss integral theorem [40]. The procedure for calculating the surface force $(\sigma\kappa)_{\mathrm{l}f}$ is described later in this section.
2. The normal velocity derivative specified at $\mathrm{l}f$ is calculated from the normal velocity gradient at $\mathrm{g}f$ using eq. (11), as follows:

$$\begin{aligned} (\nabla \mathbf{v} \cdot \mathbf{n})_{\mathrm{l}f} = {} & \frac{\eta_{\mathrm{g}}}{\eta_{\mathrm{l}}} (\mathbf{I} - \mathbf{n}_{\mathrm{l}f} \otimes \mathbf{n}_{\mathrm{l}f}) \cdot \left[(\nabla \mathbf{v})_{\mathrm{g}f} \cdot \mathbf{n}_{\mathrm{l}f} \right] \\ & + \frac{1}{\eta_{\mathrm{l}}} (\nabla_s \sigma)_{\mathrm{l}f} - \mathbf{n}_{\mathrm{l}f} (\nabla_s \cdot \mathbf{v})_{\mathrm{l}f} \\ & + \frac{(\eta_{\mathrm{g}} - \eta_{\mathrm{l}})}{\eta_{\mathrm{l}}} \left(\nabla_s \mathbf{v}_{\mathrm{n}} \right)_{\mathrm{l}f}, \end{aligned} \tag{30}$$

   where $\mathbf{n}_{\mathrm{l}f} = -\mathbf{n}_{\mathrm{g}f}$ is the unit normal of the face $\mathrm{l}f$. Equation (30) is derived using the identity $\nabla \mathbf{v} \cdot \mathbf{n} + \mathbf{n} (\nabla_s \cdot \mathbf{v}) = (\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \cdot (\nabla \mathbf{v} \cdot \mathbf{n})$. The surface gradient of the normal velocity component $(\nabla_s \mathbf{v}_{\mathrm{n}})_{\mathrm{l}f}$ at $\mathrm{l}f$ is calculated using the surface Gauss integral theorem. Calculating the tangential surface force $(\nabla_s \sigma)_{\mathrm{l}f}$ is described later in this section.
3. According to the kinematic condition (6), the tangential velocity component specified on $\mathrm{g}f$ is transferred from $\mathrm{l}f$:

$$(\mathbf{v}_{\mathrm{t}})_{\mathrm{g}f} = (\mathbf{v}_{\mathrm{t}})_{\mathrm{l}f}. \tag{31}$$

   The normal velocity component specified on $\mathrm{g}f$ is calculated from the condition of zero net mass flux, $(\dot{V}_{\mathrm{g}f} - \dot{V}_{\mathrm{g}f}) = 0$, i.e.

$$(\mathbf{v}_{\mathrm{n}})_{\mathrm{g}f} = -\frac{\dot{V}_{\mathrm{g}f}}{S_{\mathrm{g}f}} \mathbf{n}_{\mathrm{l}f}, \tag{32}$$

   where $\dot{V}_{\mathrm{g}f}$ is the volume flux of the face $\mathrm{g}f$. Since the displacement of mesh points on the side g is equal to the displacement on the side l of the interface, $\mathbf{u}_{\mathrm{g}i} = \mathbf{u}_{\mathrm{l}i}$, the same is valid for the volume fluxes of the faces $\mathrm{l}f$ and $\mathrm{g}f$:

$$\dot{V}_{\mathrm{g}f} = \dot{V}_{\mathrm{l}f}. \tag{33}$$

The specification of interface boundary conditions using the above procedure is done at the beginning of each outer iteration. In general, at the end of an outer iteration, the net volume flux through the boundary side l is different from zero, i.e.

$$[\dot{V}^{\mathrm{p}}_{\mathrm{l}f} - (\dot{V}_s)^{\mathrm{p}}_{\mathrm{l}f}] \neq 0, \tag{34}$$

where $\dot{V}^{\mathrm{p}}_{\mathrm{l}f}$ is the fluid volume flux through the face $\mathrm{l}f$ and $(\dot{V}_s)^{\mathrm{p}}_{\mathrm{l}f}$ is the cell-face volume flux through the same face, both obtained in the previous outer iteration. In order to correct the net volume flux, the interface points must be moved to accomplish the following volume flux corrections:

$$\dot{V}'_{\mathrm{l}f} = [\dot{V}^{\mathrm{p}}_{\mathrm{l}f} - (\dot{V}_s)^{\mathrm{p}}_{\mathrm{l}f}]. \tag{35}$$

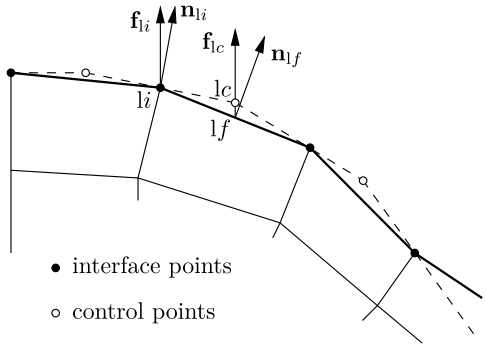


Fig. 6. Definition of the interface using control points.

The displacement of the interface points is calculated based on the procedure proposed in [28], where a control point lc is attached to each face lf at the side l of the interface as is shown in Fig. 6. The corrected position of the interface points is calculated using the following procedure:

1. Calculate the volume $\delta V'_{\mathrm{l}f}$ which face lf sweeps on the way from the current to the corrected position in order to cancel the net mass flux:

$$\delta V'_{\mathrm{l}f} = \frac{2}{3}\dot{V}'_{\mathrm{l}f}\,\Delta t, \tag{36}$$

   where $\dot{V}'_{\mathrm{l}f}$ is the volume flux correction for face lf, eq. (35).
2. Using the above, the displacement of control points in the direction $\mathbf{f}_{\mathrm{l}c}$ is:

$$h'_{\mathrm{l}c} = \frac{\delta V'_{\mathrm{l}f}}{S^{\mathrm{p}}_{\mathrm{l}f}\,\mathbf{n}^{\mathrm{p}}_{\mathrm{l}f}\cdot\mathbf{f}_{\mathrm{l}c}}, \tag{37}$$

   where $S^{\mathrm{p}}_{\mathrm{l}f}$ and $\mathbf{n}^{\mathrm{p}}_{\mathrm{l}f}$ are the area and unit normal of the face lf in the previous iteration and $\mathbf{f}_{\mathrm{l}c}$ is the control point displacement direction. The new corrected positions of the control points are calculated according to the following expression:

$$\mathbf{r}^{n}_{\mathrm{l}c} = \mathbf{r}^{\mathrm{p}}_{\mathrm{l}c} + h'_{\mathrm{l}c}\,\mathbf{f}_{\mathrm{l}c}, \tag{38}$$

   where $\mathbf{r}^{\mathrm{p}}_{\mathrm{l}c}$ is the position vector of the control point lc before the correction.
3. The new position of the interface mesh point li is obtained by projection to the plane which is laid over the corresponding control points using the least square method.

3.5. Calculation of interface curvature and surface tension

Regardless of the approach used for tracking the interface between the phases in a multiphase fluid flow, the implementation of surface tension is always demanding.

Let us assume the interface is discretised with an unstructured surface mesh consisting of arbitrary polygonal control areas. The surface tension force acting on the control area S_{lf} (see Fig. 7) can be expressed by the following equation:

$$\mathbf{F}^{\sigma}_{\mathrm{l}f} = \oint_{\partial S_{\mathrm{l}f}} \sigma\mathbf{m}\,dL = \sum_{e}\int_{L_e}\sigma\mathbf{m}\,dL = \sum_{e}\sigma_e\mathbf{m}_e L_e, \tag{39}$$

where σ_e and $\mathbf{m}_e$ are the surface tension and bi-normal unit vector at the center of the edge e and L_e is the length of the edge e. If the total surface tension force for each control area in the mesh is calculated using eq. (39), then the total surface tension force for a closed surface will be exactly zero if the unit bi-normals $\mathbf{m}_e$ for two control areas sharing the edge e are parallel and have opposite direction.

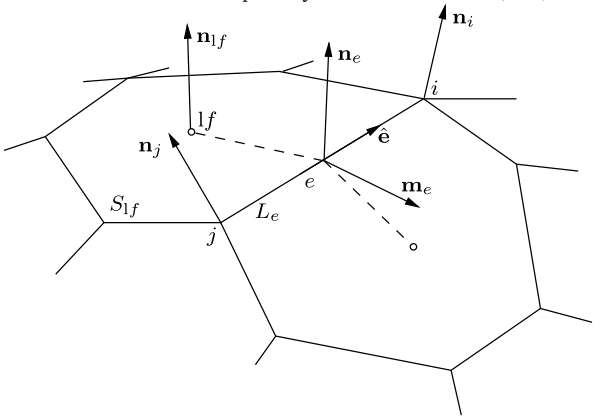


Fig. 7. Control area $S_{\mathrm{l}f}$ at the interface.

It remains to decompose the surface tension force $\mathbf{F}^{\sigma}_{\mathrm{l}f}$ into the tangential component $(\nabla_s\sigma)_{\mathrm{l}f}$ used in eq. (30), and the normal component $(\kappa\sigma)_{\mathrm{l}f}\mathbf{n}_{\mathrm{l}f}$ used in eq. (29). Using the surface Gauss integral theorem, the surface tension force acting on the control area $S_{\mathrm{l}f}$ can be expressed by the following equation:

$$\mathbf{F}^{\sigma}_{\mathrm{l}f} = \int_{S_{\mathrm{l}f}}\nabla_s\sigma\,dS + \int_{S_{\mathrm{l}f}}\kappa\sigma\mathbf{n}\,dS. \tag{40}$$

When the right hand side of eq. (40) is approximated by the mid-point role and the result of the discretisation is equalised with the right hand side of eq. (39), the following expression is obtained:

$$(\nabla_s\sigma)_{\mathrm{l}f} + (\kappa\sigma)_{\mathrm{l}f}\,\mathbf{n}_{\mathrm{l}f} = \frac{1}{S_{\mathrm{l}f}}\sum_{e}\sigma_e\mathbf{m}_e L_e. \tag{41}$$

Hence, the tangential component of the surface tension force acting on the control area $S_{\mathrm{l}f}$ is equal to the tangential component of the right hand side of eq. (41):

$$(\nabla_s\sigma)_{\mathrm{l}f} = \frac{1}{S_{\mathrm{l}f}}(\mathbf{I}-\mathbf{n}_{\mathrm{l}f}\mathbf{n}_{\mathrm{l}f})\cdot\sum_{e}\sigma_e\mathbf{m}_e L_e \tag{42}$$

and its normal component is equal to the respective normal component:

$$(\kappa\sigma)_{\mathrm{l}f}\mathbf{n}_{\mathrm{l}f} = \frac{1}{S_{\mathrm{l}f}}(\mathbf{n}_{\mathrm{l}f}\otimes\mathbf{n}_{\mathrm{l}f})\cdot\sum_{e}\sigma_e\mathbf{m}_e L_e. \tag{43}$$

If the surface tension coefficient is constant ($\sigma = \text{const.}$), eq. (42) will give a tangential component of the surface tension force equal to zero if the normal unit vector of the control area S_{lf} satisfies the following equation:

$$\kappa_{\mathrm{l}f}\mathbf{n}_{\mathrm{l}f} = \frac{1}{S_{\mathrm{l}f}}\sum_{e}\mathbf{m}_e L_e \tag{44}$$

or, if $\kappa_{\mathrm{l}f} \neq 0$:

$$\mathbf{n}_{\mathrm{l}f} = \frac{\sum_{e}\mathbf{m}_e L_e}{|\sum_{e}\mathbf{m}_e L_e|}. \tag{45}$$

With eqs. (42) to (44) we shall formulate a procedure for the calculation of the surface tension force which ensures that the total surface tension force on a closed surface will be exactly zero. Unfortunately, the fulfillment of this condition is not sufficient for successful application of surface tension forces in the calculation. Specifically, unphysical fluid flow near the interface arises due to local (rather than global) inaccuracy in the calculation of surface tension forces.

From eq. (39) one can see that the accuracy of surface tension force calculation depends on the accuracy of calculation of the bi-normal unit vector $\mathbf{m}_e$ which is calculated using the following expression:

$$\mathbf{m}_e = \hat{\mathbf{e}} \times \frac{\mathbf{n}_i + \mathbf{n}_j}{2}, \tag{46}$$

where $\hat{\mathbf{e}}$ is the unit vector parallel with edge e and $\mathbf{n}_i$ and $\mathbf{n}_j$ are the interface normal unit vectors in points i and j (see Fig. 7). Using eqs. (44) and (46) one obtains the exact value of curvature of the control area S_{lf} if the points of the control area lie on the surface of the sphere.

3.6. Solution procedure

Based on the described interface tracking method, one can now define the solution procedure for the Navier-Stokes system on a moving mesh, which may be used for simulating two-phase fluid flow using a moving mesh interface tracking method. The procedure consists of the following steps:

1. For the new time instance $t = t^{n+1}$, initialize the values of all dependent variables with the corresponding values from the previous time instance;
2. Define the displacement directions for the interface mesh points and the control points;
3. Start of outer iteration loop:
   (a) Update pressure and velocity boundary conditions at the interface;
   (b) Assemble and solve the discretised momentum equation eq. (19) on the mesh with the current shape of the interface. The pressure field, face mass fluxes and volume fluxes are used from the previous (outer) iteration;
   (c) The velocity field obtained in the previous step is used for the assembly of the discretised pressure eq. (23). After the pressure equation is solved, new absolute mass fluxes through the cell faces are calculated. The net mass flux through an interface is generally different from zero;
   (d) In order to compensate the net mass flux obtained in the previous step, the interface displacement is calculated using the interface points displacement procedure defined in the previous section;
   (e) The interface points displacement is used as a boundary condition for the solution of the mesh motion problem. After mesh displacement, the new face volume fluxes are calculated using the current points positions and the position from the previous time instance;
   (f) Convergence is checked and if the residual levels and the net mass flux through the interface do not satisfy the prescribed accuracy, the procedure is returned to step (a).
4. If the final time instance is not reached, return to step 1.

4. Subgrid-scale model

4.1. Motivation

The simulation of realistic gas-liquid systems is still a huge challenge, e.g. when one would like to simulate an industrial scale bubble column reactor. The nature of this application is multi-scale. One of the smallest scales occurring is that of concentration boundary layers of the rising bubbles at realistic, i.e. high, Schmidt numbers. With conventional FV-methods these scales have to be resolved, what is still unfeasible for an application as the one mentioned above. This motivated the first paper to discuss SGS modeling [1]. In SGS modeling, an analytical function is used to describe the profile of a scalar field inside its boundary layer and to use this SGS information for improved flux computation.

The method has been developed further since and has been applied in the context of Volume of Fluid [42,17,4] as well as in interface tracking frameworks [29,30,41]. An overview on the state of Subgrid-Scale modeling was also given by [43]. Recently, the approach was taken up in [16], where local curvature and tangential convection effects have been included.

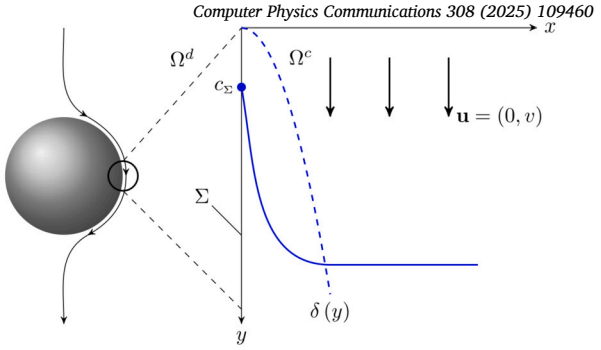


Fig. 8. The simplified model for species transfer across a bubble surface. [42] adapted from [4].

4.2. Analytical model

The SGS model we implement is based upon analytical solution of a simplified problem, depicted for a bubble in Fig. 8. The idea behind the SGS modeling is to zoom in on the interface, so that the curvature gets negligibly small, allowing the analytical model therefore to assume a (locally) flat boundary. This assumption is adequate for scenarios with a concentration boundary layer thickness small in comparison to the radius of curvature of the given boundary.

At the interface Σ of the model problem (see Fig. 8), a constant and homogeneous concentration c_Σ is prescribed, which can be computed from Henry's law under the sensible assumption of a well-mixed gas-side concentration, as is pointed out by [4].

The velocity in the substitute model problem is parallel to the interface. No velocity gradient normal to the boundary exists in the model problem ($\partial v/\partial x = 0$). There could be a non-zero gradient in the boundary-parallel direction, in which case the boundary layer thickness δ in eq. (48) would change accordingly. There is no velocity component normal to the boundary.

In the application within ALE-IT, the boundary geometry and the velocity field have to be resolved. In a co-moving reference frame, the interface neighboring cell's velocity should be nearly parallel to the interface and with the assumption of zero interface-normal velocity gradient the next cells velocity should be nearly identical to it. When a thin scalar boundary layer exists, the SGS can model the scalar fluxes normal to the boundary without resolving the scalar boundary layer itself.

Further assumptions of the substitute problem are a given, constant value (e.g. zero concentration) upstream ($y < 0$) of the domain and far away from the boundary ($x \to \infty$) as well as negligible diffusion in streamwise direction. We also assume quasi-steadiness and the transported scalar being passively transported.

Applying these assumptions to the advection-diffusion eq. (13) with the sorption term $s^\Sigma + [\![\boldsymbol{j} \cdot \boldsymbol{n}_\Sigma]\!] = 0$ on $\Sigma(t)$ and Fick's law $\boldsymbol{j} = -D\nabla c$ and v being the boundary-parallel velocity, lead to the expression

$$v\frac{\partial c}{\partial y} = D\frac{\partial^2 c}{\partial x^2} \quad \text{for } x > 0 \text{ and } y > 0 \tag{47}$$

with the boundary conditions

$$c(x, y = 0) = c_\infty, \quad c(x \to \infty, y > 0) = c_\infty \quad \text{and} \quad c(x = 0, y > 0) = c_{|\Sigma}.$$

The analytical solution to this problem is

$$c(x, y) = c_{|\Sigma} + (c_\infty - c_{|\Sigma}) \cdot \operatorname{erf}\left(\frac{x}{\delta(y)}\right) \qquad \text{with } \delta(y) = \sqrt{4Dy/v}. \tag{48}$$

The boundary layer thickness δ is considered a free parameter, which in IT will be fitted for each interface-attached cell individually as described in section 4.3.2. The fitting selects the locally best approximation from the family of given error-functions in eq. (48).

From the analytical solution for the fitted parameter, we can approximate a more accurate boundary-normal gradient of the scalar concen-

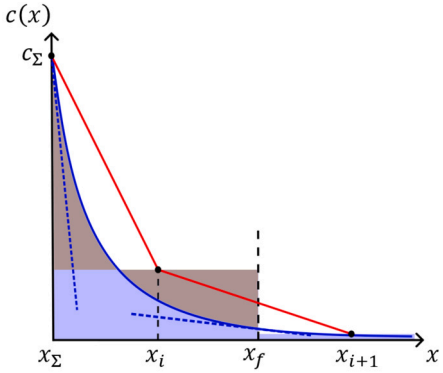


Fig. 9. The concentration profile and linear/corrected gradient next to a boundary. Adapted from [42]. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

tration at the interface as well as at the first cell faces normal to the boundary, x_f in Fig. 9. These gradients are used to scale the convective and diffusive scalar fluxes in the Finite Volume discretization, keeping the discretization of convective and diffusive operators implicit. Scaling is only applied in interface normal direction and described in detail in sections 4.3.3 and 4.3.4.

Without the SGS, the maximally second-order unstructured Finite Volume convective and diffusive fluxes are underestimated at the interface and overestimated at the first cell faces normal to the boundary. Fig. 9 visualizes the linear gradient calculation in red and the actual gradients at the interface and the first cell faces normal to the boundary with the dashed blue lines.

Note that in the algorithm the concentration in the first cell is not altered by the SGS. Instead the gradients are used to scale the diffusivities at the respective locations, as described in sections 4.3.3 and 4.3.4.

Details of the SGS model can be found in [4,42,30]. It has been applied to VoF by [42], where the implementation becomes more complicated.

4.3. Algorithm

4.3.1. Equation discretization in finite volume solvers

The species transport eq. (13) is discretised using the unstructured Finite Volume Method, e.g. using the backward scheme as described above for Navier-Stokes equations,

$$\frac{3c_P^{n+1}V_P^{n+1} - 4c_P^{n}V_P^{n} + c_P^{n-1}V_P^{n-1}}{\Delta t} + \sum_f \Phi_f c_f^{n+1} = \sum_f D_f (\nabla c)_f^{n+1} \cdot \boldsymbol{S}_f, \tag{49}$$

where V_P is a control volume/cell, $\Phi_f = \boldsymbol{S}_f \cdot (\mathbf{v} - \boldsymbol{w})_f$ is the face flux and the superscripts $^{n+1}$, ^{n} and $^{n-1}$ symbolize the new, current and preceding timestep. The formula is written for a constant time step size. The subscript $_P$ and $_f$ represent the value at the cell centers, respectively the face centers.

For the SGS model to take effect in the computation, we will modify the diffusivity coefficient and the face fluxes in the equation above.

4.3.2. Computation of model parameter boundary layer thickness

The model parameter boundary layer thickness δ is computed by iteratively comparing the amount of the scalar in the first cell adjacent to the boundary in the simulation, $\overline{\eta}_c$, with the amount that would be contained in said cell for a given δ, η_{SGS}. In Fig. 9 both light brown areas have to be the same size or the area enclosed by the real concentration profile and the cell centered FV-value has to be the same. Mathematically this is being expressed in this expression:

$$\overline{\eta}_c = \frac{\overline{c} - c_{|\Sigma}}{c_\infty - c_{|\Sigma}} \overset{!}{=} \frac{1}{V}\int_V \eta(x/\delta)\, dV = \eta_{SGS} \tag{50}$$

with

$$\eta(x,y) = \frac{c(x,y) - c_{|\Sigma}}{c_\infty - c_{|\Sigma}} = \mathrm{erf}(x/\delta(y)). \tag{51}$$

Using the derivative of η_{SGS} with respect to δ, as described in [30], we use a Newton-bisection method until eq. (50) is fulfilled within a given tolerance. For details like the initial value and the algorithm in detail the reader is referred to the previously cited article. We ensure numerical robustness of the Newton-Bisection root finding approach by limiting the next value during the search to a value larger than 2^{-52}, a floating point number `SMALL` (machine epsilon) used in OpenFOAM to stabilize floating-point arithmetic. While we recommend this to be always used, we have implemented it as an optional switch. For both, the advective and diffusive flux, the same idea is utilized: the scaling of the fluxes according to the boundary layer thickness and therefore the theoretical values given by the SGS.

4.3.3. Computation of diffusive fluxes

With D_f being the face-centered molecular diffusivity, S_f being the area magnitude of the finite volume face and $\partial_n c$ being the surface normal gradient of the scalar c, the face-centered diffusive flux F_f^D is given by

$$F_f^D = -D_f S_f (\partial_n c)_f. \tag{52}$$

In the following, we use the superscript ^{SGS} to indicate the value is obtained from, or modified by, the SGS model. The SGS model solves the analytical model for the actual surface normal gradient of c, which will be $(\partial_n c)_f^{SGS}$. However, we solve the transport equation for c implicitly to retain a high-degree of numerical stability and scale the diffusion coefficient to account for the influence of the SGS model. Therefore, we state that the flux computed with the molecular (physical) diffusivity coefficient and the normal gradient from the SGS model is equal to the flux computed from the implicit unstructured FVM gradient and a scaled diffusion coefficient, i.e. meaning

$$F_f^D = -D_f S_f (\partial_n c)_f^{SGS} \overset{!}{=} D^{SGS} S_f (\partial_n c)_f^{o}, \tag{53a}$$

which can be easily re-arranged into

$$D_f^{SGS} = \frac{D_f (\partial_n c)_f}{(\partial_n c)^{SGS}}. \tag{53b}$$

This scaling is applied to the cell faces at the boundary as well as at first cell faces normal to the boundary. For both respective locations the diffusivity parameter needs to be modified at the corresponding position, x_Σ and x_f in Fig. 9. There are exceptions, when the correction is not employed, the details of this can be found in [30].

4.3.4. Computation of advective fluxes

In similar manner to the diffusion, the advective flux is given by

$$F_f^c := c_f^{SGS} F_f = c_f F_f^{SGS} \Phi, \tag{54}$$

where F_f^c is the volumetric flux of the concentration c at the face f, and $F_f := \mathbf{v}_f \cdot \mathbf{S}_f$ is the volumetric flux. Since no convection of the species normal to the interface can take place, the scaling for advective fluxes is only applied to the first cell faces normal to the boundary. Equation (54) analogously to our procedure for the diffusive flux, leads to

$$\Phi^{SGS} = \frac{c^{SGS}}{c}\Phi. \tag{55}$$

Exception handling is done similar to diffusion fluxes, of which the details can be found in [30].

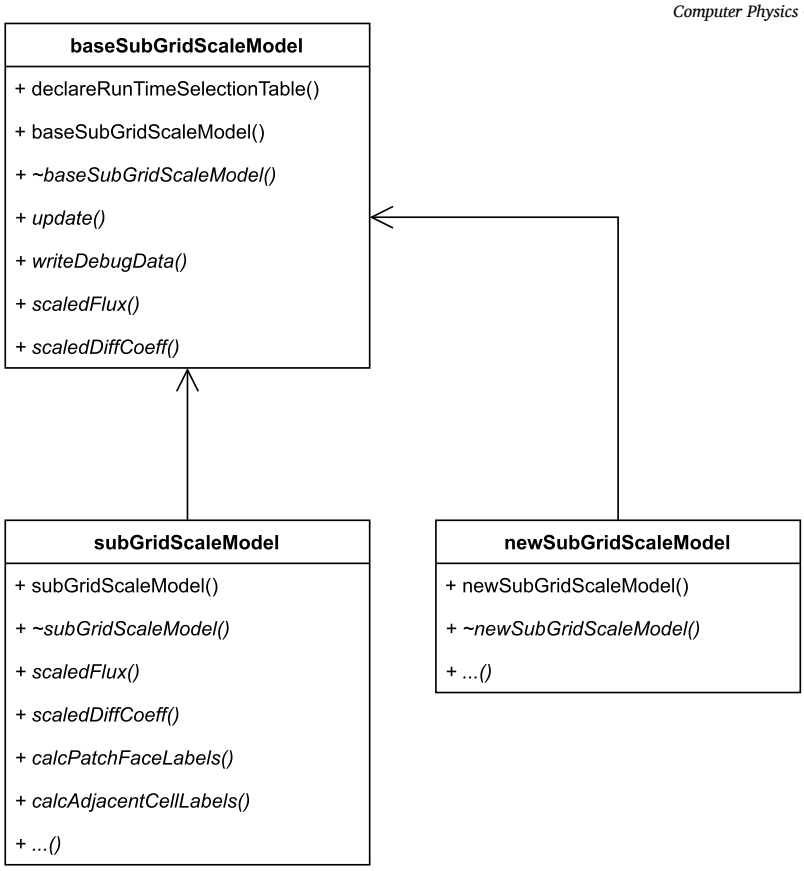


Fig. 10. Class diagram of the Subgrid-Scale model, also showing a potential new model.

5. Software design of the SGS model library

The software design is closely related to the model as described in section 4.

The first benefit of our implementation is a strict adherence to the OpenFOAM design principles, which enable easy library extension with new models alternative to the SGS [30] and their Run-Time Selection (RTS) [22] using configuration files. The implementation is depicted in Fig. 10, already showing how a new model `newSubGridScaleModel` would be implemented.

The second benefit of our software design is the possibility to apply SGS models to any Partial Differential Equation (PDE) discretized in OpenFOAM, without requiring extensive modification of existing solvers. Our SGS model library can be applied to any part of the domain boundary with a minimal requirement of a prismatic boundary-adjacent mesh with at least three layers of cells in the boundary layer.

Our design of the SGS model implements an OpenFOAM library with a class hierarchy and RTS [22]. A base class which does nothing is present, so an option remains of not using SGS modeling. The SGS model is implemented in a derived class, with virtual functions implementing the functionality from [30]. All functionality needed by the SGS is either accessed from core OpenFOAM libraries or is contained within the SGS model implementation, which ensures the modularity of the SGS model. An alternative SGS model will require similar addressing and geometric mesh information in the boundary layer and the implementation provided in the original SGS model can be re-used.

To enable the user of the SGS a good understanding of the effects of the SGS model and to make accessible insights for further developments of the SGS model, we implemented the possibility to output various SGS model parameters into OpenFOAM `volScalarFields`. These include the model parameter δ or an indication of the exception handling, what represents the reason the SGS model might not be employed. These can easily be inspected visually, an example is given in Fig. 14, where the diffusion coefficient after the SGS scaling is visualized. One can see, that on the inflow side, the scaling is strong and of opposite effect for the boundary faces versus the faces between adjacent cell and the next outer cell layer. On the outflow side in contrast, there is little effect to be seen. The third layer represents the un-modified diffusion coefficient. Visualizing multiple operating figures of the SGS enables insights into and deepens the understanding of the simulation case and the effect of the SGS to it.

An exemplified code for the modification of the PDE in an OpenFOAM solver for applying an SGS model from our model library is shown in Listing 1.

The SGS requires some values as user input, that are read from the configuration file *transportProperties* in OpenFOAM, with the relevant SGS sub-dictionary shown in Listing 2.

The SGS model can be activated with the keyword-pair `type SubGridScale` for a mesh boundary patch specified by the keyword `patchName`. Additional required user input is the far-field concentration, defined with the keyword `cInfinity`. Optional parameters are, whether to output additional parameters from the SGS model as an OpenFOAM `volScalarField` and the level of verbosity from 0 to 3 to print to the solver log. 0 corresponds to a quiet mode and 3 prints very extensive information of almost all processes and results within the SGS algorithm and calculations. As these settings affect simulation performance and disk consumption, the first defaults to `false` while the later affects log readability and defaults to `1`, which gives limited log-output. These visualizing capabilities as well the rapid info statement level enable insights to the user, but they also help with future model development.

Our framework is continuously and automatically tested, cf Appendix A.

Listing 1: A minimal example of the application of the SGS model to a PDE solver in OpenFOAM.

```
#include "baseSubGridScaleModel.H"
int main(int argc, char *argv[])
{
    ...

    autoPtr<baseSubGridScaleModel> sgsPtr =
        baseSubGridScaleModel::New(transportProperties.subDict("SGS"));

    while (simple.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;
        while (simple.correctNonOrthogonal())
        {
            // Compute SGS data needed for scaling.
            sgsPtr->update(psi);
            fvScalarMatrix psiEqn
            (
                fvm::ddt(psi)
              + fvm::div(sgsPtr->scaledFlux(phi, psi), psi)
              - fvm::laplacian(sgsPtr->scaledDiffCoeff(D, psi), psi)
             ==
                fvOptions(psi)
            );

            psiEqn.relax();
            fvOptions.constrain(psiEqn);
            psiEqn.solve();
            fvOptions.correct(psi);
        }
        runTime.write();
    }
}
```

Listing 2: *transportProperties* sub-dictionary for the SGS model.

```
SGS
{
    // Choose the SGS model: "SubGridScale" or "inactive"
    type            SubGridScale;

    // patch to apply SGS to
    patchName       freeSurface;

    // far-field concentration
    cInfinity       cInfinity [0 -3 0 0 1 0 0] 0;

    // (optional:) Switch, whether to limit next
    // search point to be bigger than SMALL
    iSGS            false;

    // (optional:) write parameters for visualization
    // of SGS parameters
    visualizeParameters true;

    // (optional:) verbosity level for info statement output (0-3)
    infoLvl         1;
}
```

6. Results

6.1. Error norms and rate of convergence

For verification cases, we employ up to three different error norms. The L_1-, L_2- and L_∞-norms are calculated by

$$L_1 = \sum_i \left| y_{i,num} - y_{i,ref} \right| \tag{56}$$

$$L_2 = \sqrt{\sum_i \left(y_{i,num} - y_{i,ref} \right)^2} \tag{57}$$

$$L_\infty = \max \left| y_{i,num} - y_{i,ref} \right| \tag{58}$$

where y_{num} is the numerically computed value, y_{ref} the reference value, either from an analytical solution of the given problem or a value to be assumed correct.

The rate of convergence is used to describe, how fast a numerical method converges towards a mesh-independent solution. The rate of convergence for two given cases with different discretization length is approximated as

$$c = \frac{\log \frac{e_1}{e_2}}{\log \frac{h_1}{h_2}}, \tag{59}$$

where c is the rate of convergence, e is the error, computed as $e = y_{num} - y_{ref}$, and h is the typical discretisation length. For the cases, where we compute a rate of convergence in this article, the edge length of the surface mesh is the typical discretization length. The subscripts $_1$ and $_2$ denote the two different cases, where h_1 is the bigger discretisation length.

6.2. Flat plate test case

The smallest test for model development and code verification is what we call the flat plate test case. It implements the assumptions made in section 4.2 and was already implemented by e.g. [30]. The test consists of a flat plate with a prescribed constant concentration of $1\,\mathrm{mol/m^3}$. A velocity parallel to the plate is prescribed for the whole domain, including the boundaries so there is no velocity boundary layer at the plate. Initially and in the inflow, the concentration is set to zero, the length of the plate is $0.005\,\mathrm{m}$ and the fluid velocity is $0.1\,\mathrm{m/s}$. The timestep of $0.0002\,\mathrm{s}$ is used for a physical duration of $0.1\,\mathrm{s}$, which is double the time the fluid needs to be transported through the domain. Fig. 11 visualizes the setting.

With the possibilities described in section 5, it can be verified that fundamental parts of the algorithm as described in section 4.3 are implemented correctly, like the boundary layer thickness δ and the diffusion correction. Beneficially, there is an analytical solution to this model for comparison. For these reasons the test enables insights into the capabilities of the SGS model and the implementation thereof. This simple test

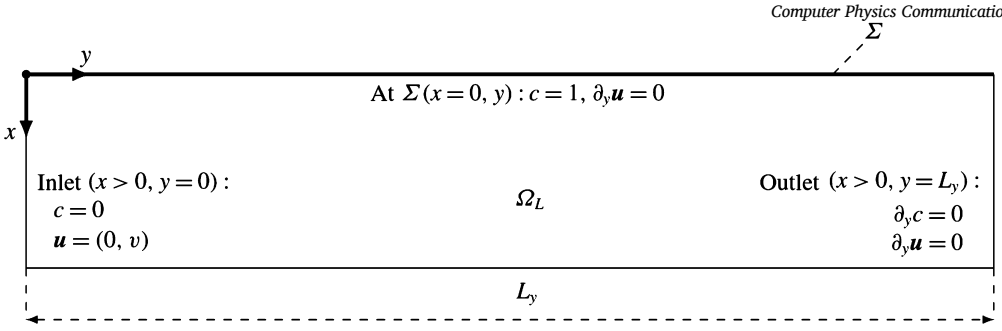


Fig. 11. Set-up of the flat plate test. Figure taken from [30].

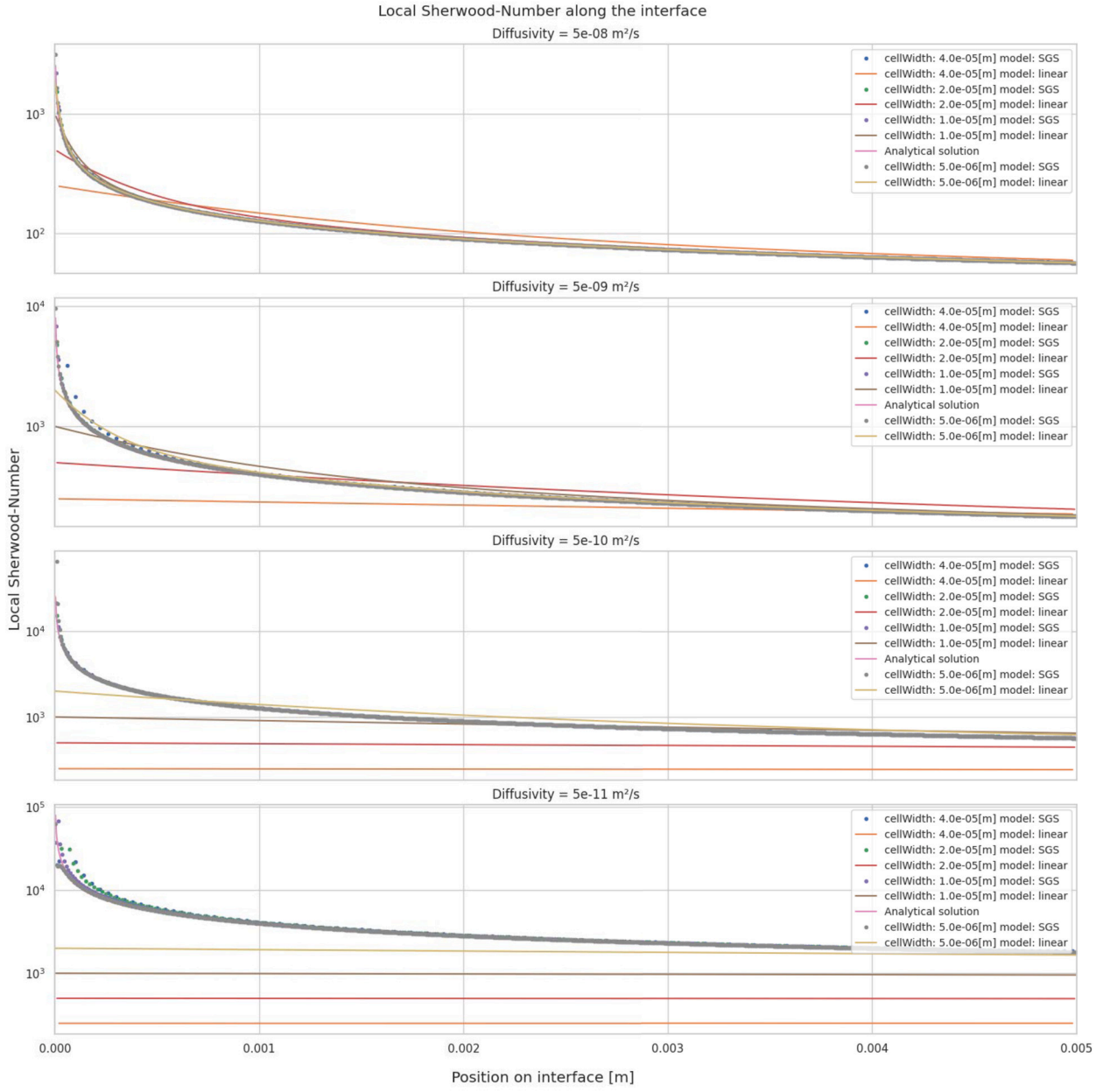


Fig. 12. Results for the parameter study over of the flat plate test case.

case does circumvent most of the error-handling and also convective flux correction must not have an effect on the outcome.

In Fig. 12 the local Sherwood numbers of a parameter study along the boundary surface are displayed. The parameter study has four levels of mesh refinement, where the cell size halves between each execution from 40 μm to 5 μm, four different diffusivity levels spanning four orders of magnitude from $5 \times 10^{-8}\,\mathrm{m}^2/\mathrm{s}$ to $5 \times 10^{-11}\,\mathrm{m}^2/\mathrm{s}$ and every of these setups is run with and without active SGS model. Taking the length of

the plate as reference length L, the Peclet number is within the interval $[10^4, 10^7]$.

It is apparent, how without the SGS model, the Sherwood number cannot be computed in cases with a boundary layer that is small in comparison to the cell size, i.e. when the cell size is big and the diffusivity is small. In contrast to the linear interpolation with the SGS model the Sherwood number is captured reasonably well in all cases. Note in Fig. 12, that at some locations the Sherwood number with the application of the SGS model is actually smaller than without it.

6.3. Mass transfer at a spherical fluid particle

The second test case for the SGS model mimics a moving droplet or bubble. It uses the axisymmetric solution of velocity field from Satapathy and Smith in [32] for low Reynolds numbers. The computational domain is a wedge with prescribed velocity. This test case was also implemented by and we take the reference solution from [30]. This test case introduces additional complexity compared to section 6.2, since there is flow non-parallel to the interface, therefore violating some of the assumptions made when deriving the SGS mathematical model, as described in section 4.2. From the visualization capabilities we can see in the rear part of the "bubble" the exception handling partially taking place, as there the concentration rises, not conforming to the assumption of a thin boundary layer (with parallel flow) made in the derivation of the SGS. As in section 6.2 the inflow and initial concentration is set to $0\,\mathrm{mol/m^3}$ and the interface is set to have a constant concentration of $1\,\mathrm{mol/m^3}$. The simulated physical duration is 0.6 s and the results shown here represent the final state.

To see the best results with the SGS, it is important to set the interpolation schemes to `limitedLinear phi 1.0` for the concentration c and to `linear` for the interpolation of the scaled diffusion coefficient as well as for the flux of U in the *fvSchemes*-file.

We run a parameter study consisting of four different diffusivities from $1\times10^{-8}\,\mathrm{m^2/s}$ to $1\times10^{-11}\,\mathrm{m^2/s}$, four different mesh resolutions with the meshing parameter $\mathrm{N} = [62, 124, 248, 496]$ as the number of faces along the half-circled interface. Every study is run with and without employing the SGS and with a constant timestep of 0.0002 s. The bubble diameter of 2 mm serves as reference length for the Sherwood number. It can be observed from Fig. 13 that without deploying the SGS, the local Sherwood number is almost constant and except for the rear part of the bubble too low in case of small diffusivities and coarse meshes. When the mesh resolution is fine and the diffusivity is high, the boundary layer is not as thin (in comparison to the mesh cell size), and therefore the results of linear modeling and SGS modeling are similar.

In Fig. 14 the diffusion coefficient after scaling with the SGS is displayed. In the cell layer adjacent to the interface, the diffusion coefficient at the interface is visible, whilst in the next outer cell layer the scaled diffusion coefficient at the faces between interface-adjacent cells and that same cell is visualized. The third row of cells represents the uncorrected diffusion coefficient for reference.

The biggest correction is applied in the front part of the bubble, where the flow impinges onto the interface and the boundary layer is very thin, therefore the derivative in surface normal direction of the concentration is very big. In the rear part of the bubble the boundary layer has grown and the main direction of the flow is no longer parallel to the surface, making the surface concentration gradient in interface-normal direction much smaller and reasonable good represented by linear interpolation. Therefore less correction is needed. To understand what is happening, we would like to point the reader to Fig. 9 again.

6.4. Marangoni-induced migration of an air bubble in silicone oil

The velocity of a bubble/droplet within an unbounded fluid subjected to uniform temperature gradient ($\partial T/\partial z$) and zero gravity field ($g = 0$) is defined by following expression [44]:

$$v^{T}_{\mathrm{b,YBG}} = \frac{2}{\left(2+\frac{\lambda_i}{\lambda_o}\right)\left(2+3\frac{\mu_i}{\mu_o}\right)}\frac{R}{\mu_o}\frac{\partial\sigma}{\partial T}\frac{\partial T}{\partial z}, \tag{60}$$

where R is the radius of the bubble, T is the fluid temperature, σ is the surface tension coefficient, μ_i is the dynamic viscosity of the internal fluid, μ_o is the dynamic viscosity of the outside fluid and λ is the thermal conductivity of the fluid.

In the case the Marangoni convection is caused by a specified fixed gradient of surfactant concentration along the interface, the bubble migration velocity can be expressed based on eq. (60) as follows, see [27]:

$$v^{\Gamma}_{\mathrm{b,YBG}} = \frac{2}{3\left(2+3\frac{\mu_i}{\mu_o}\right)}\frac{R}{\mu_o}\frac{\partial\sigma}{\partial\Gamma}\frac{\partial\Gamma}{\partial z}, \tag{61}$$

where Γ is the concentration of surfactants at the interface.

A numerical simulation is performed for an air bubble of radius $r_b = 1$ mm moving through the silicone oil. Relevant properties of both phases are listed in Table 1. The bubble is assumed to be of spherical shape and rigid and is fixed during the simulation. The spatial computational domain consists of a volume of the spherical bubble filled with air and volume around the bubble bounded by the spherical surface of radius $r = 20r_\mathrm{b}$ and filled with silicone oil. The spatial domain is discretised by the unstructured mesh consisting of 979 520 hexahedral cells.

The flow inside and outside of the bubble is driven by Marangoni forces due to the nonuniform distribution of surfactant concentration along the interface, which is fixed during the simulation. The distribution of surfactant along the bubble surface is defined by following linear function:

$$\frac{\Gamma}{\Gamma_\infty} = \frac{z+2r_\mathrm{b}}{L}, \tag{62}$$

where Γ_∞ is the saturated surfactant concentration and L is the length scale, whose value is set to 525 mm. The equation of state is also linear and reads as follows:

$$\sigma = \sigma_\mathrm{s}\left(1-\beta_\mathrm{s}\frac{\Gamma}{\Gamma_\infty}\right), \tag{63}$$

where $\beta_\mathrm{s} = \frac{RT\Gamma_\infty}{\sigma_\mathrm{s}}$ is the elasticity number which amounts to 2 in this study.

The unsteady computation is performed with time-step size $\Delta t = 0.01$ s using the first order accurate `Euler` temporal discretisation scheme, until steady state is reached. The gradient of the surface tension coefficient as well as the gradients of volume fields is calculated using the least square method. The diffusion terms in the momentum and pressure equations are discretised using the skew-corrected central differencing scheme, while the convective term in the momentum equation is discretised using the linear-upwind scheme. At the outer side of the spatial domain the velocity and normal derivative of pressure are set to zero and at the interface corresponding kinematic and dynamics conditions are enforced.

The bubble rise velocity is calculated based on numerically determined volume flow rates through the faces at the interface as follows:

$$v_\mathrm{b} = \frac{\sum_f \mathrm{pos}(\phi_{\mathrm{i,f}})\phi_{\mathrm{i,f}}}{\sum_f \mathrm{pos}(\mathbf{k}\cdot\mathbf{S}_{\mathrm{i,f}})(\mathbf{k}\cdot\mathbf{S}_{\mathrm{i,f}})}, \tag{64}$$

where $\phi_{\mathrm{i,f}}$ is the volume flow rate of the fluid through the face f at the interface, $\mathbf{S}_{\mathrm{i,f}}$ is the area vector of the face f at the interface and $\mathbf{k}$ is the unit direction vector of bubble motion. The bubble velocity calculated using eq. (64) based on simulation results amounts to $v_b = 0.001\,325\,\mathrm{m/s}$, while the corresponding analytical (eq. (61)) gives the migration velocity as $v^{\Gamma}_{\mathrm{b,YBG}} = 0.001\,329\,5\,\mathrm{m/s}$. The velocity field inside and around the bubble along the plane $y = 0$ is shown in Fig. 15.

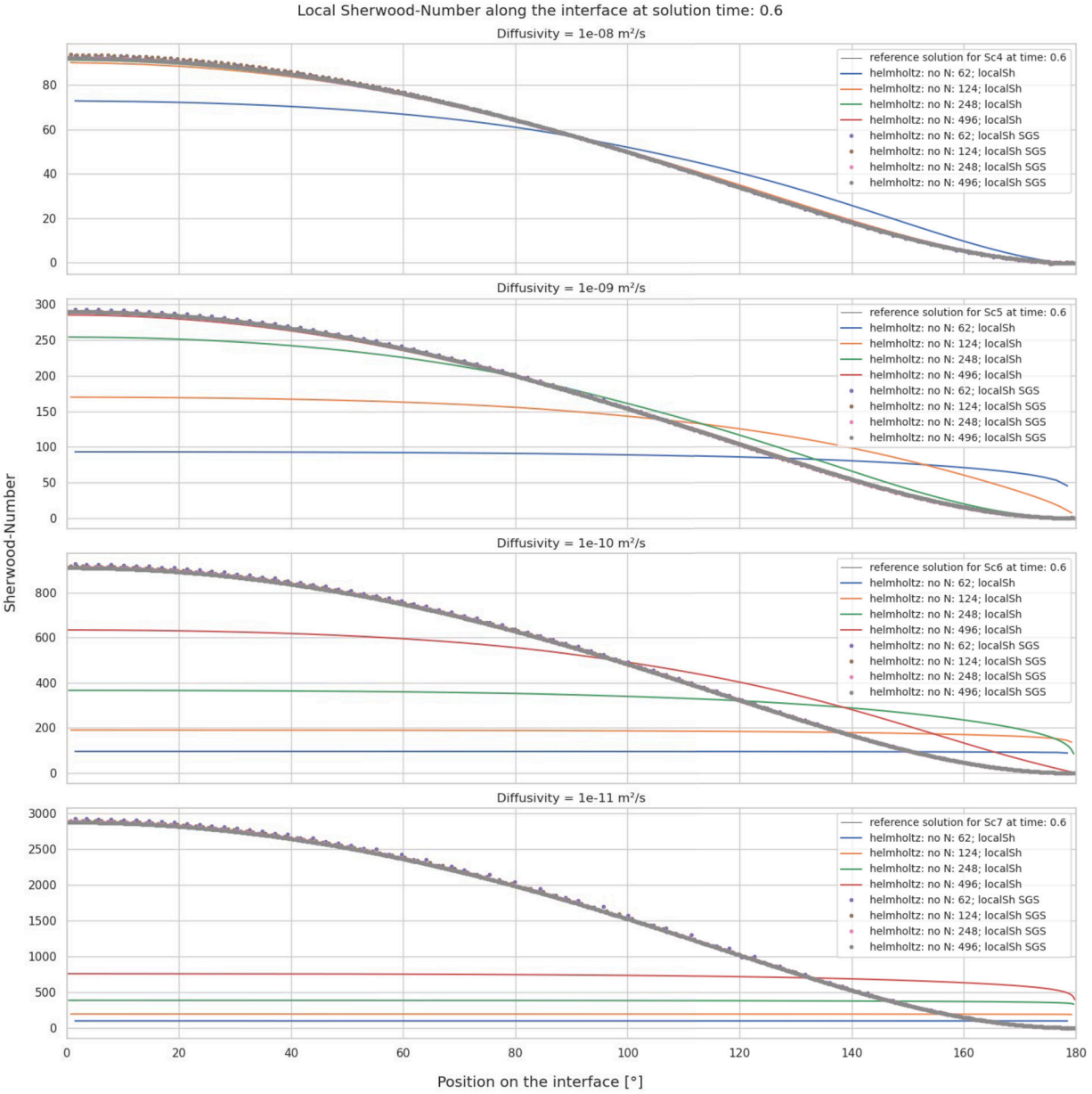


Fig. 13. Local Sherwood numbers of a parameter study for Schmidt-numbers from 1×10^4 to 1×10^7.

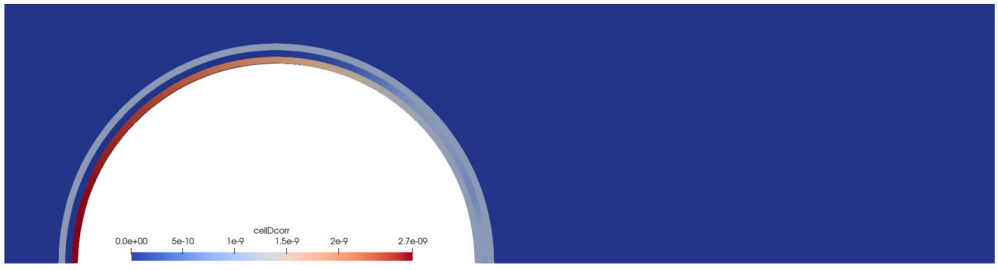


Fig. 14. Visualization of corrected diffusion coefficients for the coarsest mesh with a diffusivity of $1 \times 10^{-9}\,\mathrm{m}^2/\mathrm{s}$.

6.5. Passive scalar transport on the interface

We test the framework's ability of adequate transport of a species along an interface with a rotating droplet with a prescribed concentration. The droplet is represented in 2-D. The test setup follows the test done by [2,3]. We prescribe the initial concentration distribution and the velocity. For the evaluation we compute one rotation, what equals one second with our prescribed velocity referring to an angular velocity $\omega = 1/s$.

The initial distribution is given by

$$\Gamma(\theta, \tau = 0) = \frac{\cos\theta + 1}{2}\Gamma_0, \tag{65}$$

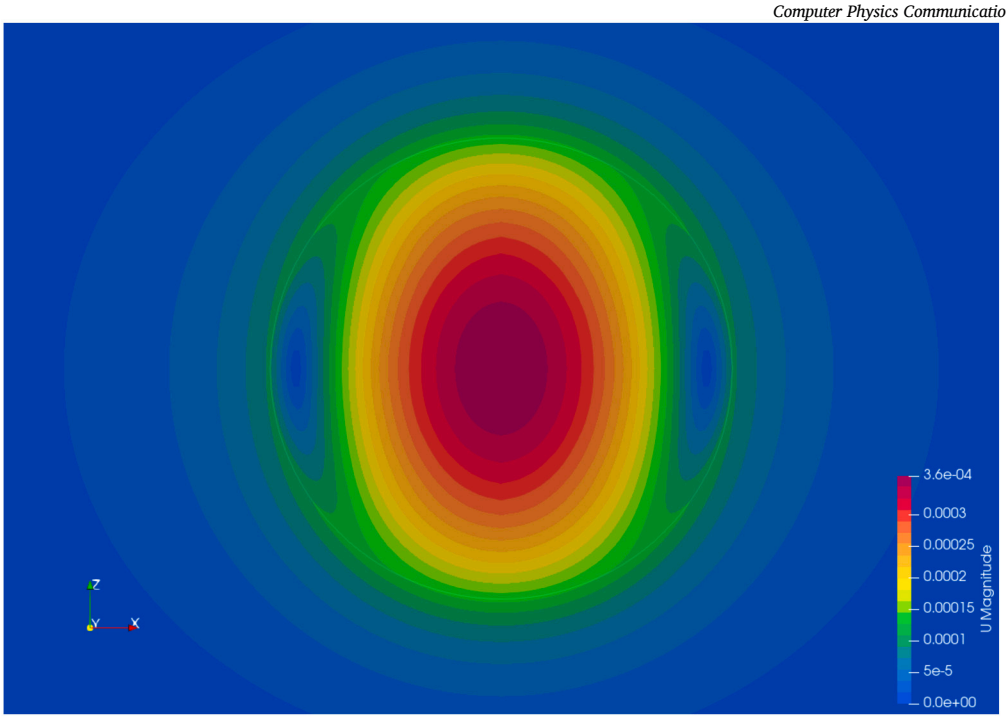


Fig. 15. Marangoni-induced velocity magnitude inside and around the migrating bubble along the plane $y = 0$.

Table 1
Properties of air and silicone oil.

	ρ [kg/m^3]	μ [Pa s]	σ [N/m]
Air	1.8×10^{-5}	1.2	0.02
Oil	955	0.191	0.02

where Γ is the surface concentration, τ is the time, θ is the angle and Γ_0 is the maximum initial concentration. After a time τ the expected distribution from an analytical approach is given by

$$\Gamma(\theta, \tau) = \frac{e^{-D_I \tau} \cos(\theta - \omega\tau) + 1}{2} \Gamma_0, \tag{66}$$

where D_I is the diffusion in interface tangential direction.

For proper evaluation a mesh refinement study with a parameter variation of the diffusivity is conducted. The results are visualized in Fig. 16. We use a meshing parameter $N = [24, 32, 40, 56, 80, 96, 120]$, corresponding to the number of surface cells along the interface. The timestep is set to 0.0005 s regardless of the mesh size. The surface diffusivity is varied between $0\,\mathrm{m}^2/\mathrm{s}$ and $0.1\,\mathrm{m}^2/\mathrm{s}$.

Visually it is difficult to see a difference between the discretizations in Fig. 16, except for very coarse mesh resolutions. In Fig. 17 the error norms and their rate of convergence are shown. The rate of convergence of the L_∞-norm is close to 2, for the L_2-norm it is close to 1.5 and for the L_1-norm is around 1 for the resolutions analyzed. The rates of convergence shown in Fig. 17b are computed from the mesh resolution, at which they are shown and the next finer one. With the low error norms and their respective rate of convergence we deduct, that our framework is well able to handle transport along an interface.

6.6. Expanding fluid interface

To test the frameworks conservation of surfactant mass for an expanding fluid interface, we implement a test case, that mimics an expanding droplet like it was done by [2,3]. The initial distribution of species is described and there is no diffusion happening. The initial distribution is the same as in eq. (65). The droplet is represented by a wedge that is inflated with a prescribed velocity and thereby the surface is stretched. The analytical solution to this setup is given by

$$\Gamma(\theta, r(\tau)) = \frac{\cos\theta + 1}{2} \Gamma_0 \frac{r_0^2}{r(\tau)^2}, \tag{67}$$

where $r(\tau) = \sqrt[3]{3V_0(\tau + \tau_0)}$, where $\tau_0 = 1/3s$ and r is the radius, respectively r_0 is the initial radius.

Fig. 18 shows, how the species concentration per area at the interface has decreased after 2.5 s of bubble growth. The concentration matches precisely the expected result for all mesh resolutions tested. We conclude, that our framework is fit to simulate phenomena involving surface expansion/shrinking.

6.7. Bubble rising in surfactant solution

To showcase the framework's capabilities, we add a complex test case. This test case is a re-calculation of the case from [30]. It models a single air bubble rising in quiescent water with a surfactant, and there is experimental data to validate our simulation.

The previous test cases have shown in an isolated way, that the framework is able to resolve the boundary layer (sections 6.2 and 6.3), handle chemically induced changes in surface tension (section 6.4), compute surfactant transfer on the interface (section 6.5) and conserve surfactant mass on the interface, when the interface area changes (section 6.6). For this test case all of these capabilities are needed.

As we will compare our results with [30], we strive to use exactly the same settings in [30]. However, [30] have used an ALE-IT implementation in another OpenFOAM version, which introduces some differences in the solution algorithm, addressed below. The bubble diameter is 1.45 mm, and it is initialized as a sphere with zero velocity. The fluid properties used can be found in Table 2. The timestep size used is 5×10^{-6} s and the mesh contains 979 520 cells, where the interface is being discretized with 9280 faces. It is displayed in Fig. 19, where on the right-hand side, the interface mesh is displayed in dark grey. The mesh is created with OpenFOAM's `blockMesh` utility and, therefore, is different than the mesh in [30], but it has similar discretization lengths. Also, the newly introduced switch to limit the boundary layer thickness

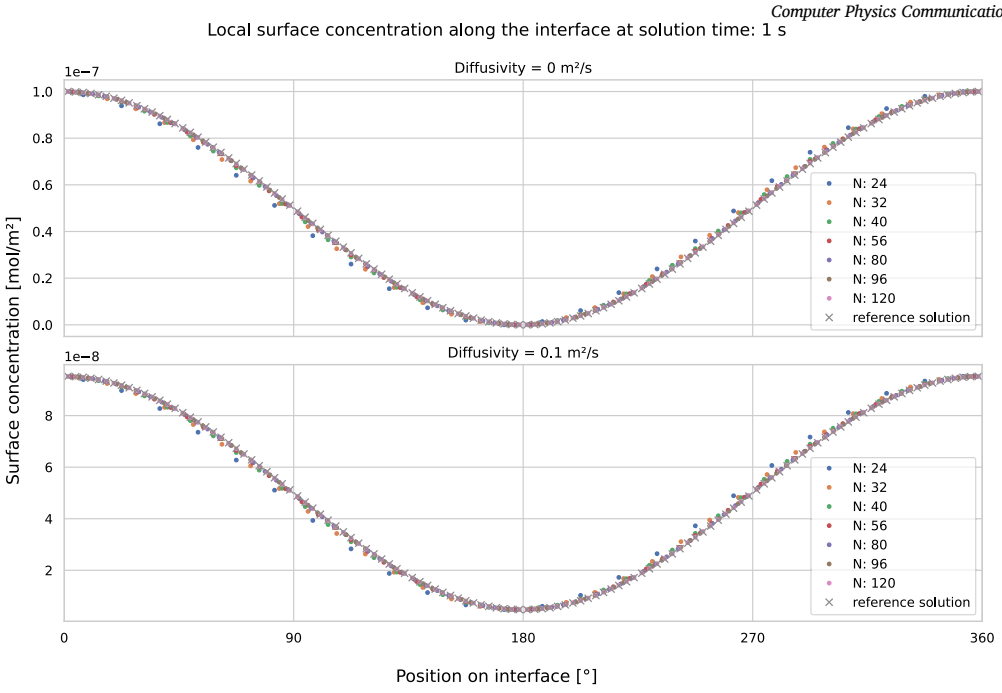
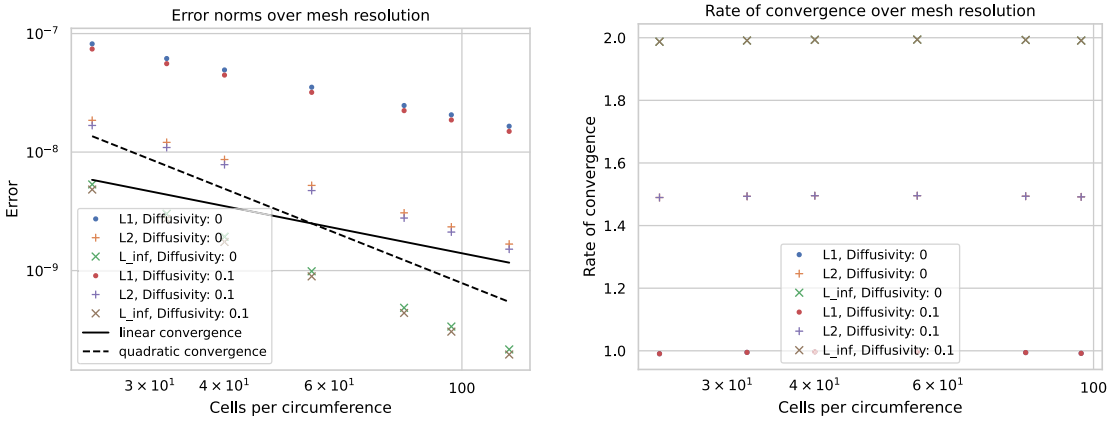


Fig. 16. Results from the rotating droplet parameter study.



(a) Error norms

(b) Rate of convergences

Fig. 17. Errors of the rotating droplet parameter study.

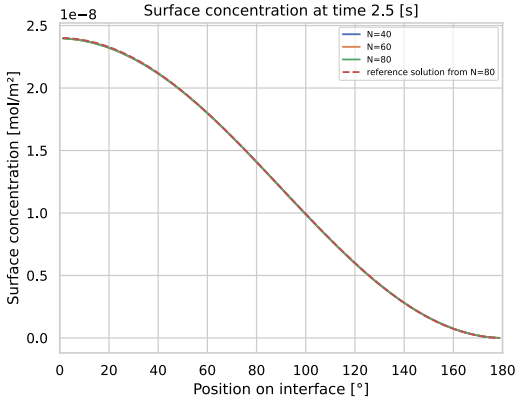


Fig. 18. Surface concentration after 0.5 s for different mesh resolutions.

to a minimum of 1×10^{-15} m is de-activated to perform a more direct comparison with [30].

Table 2
Fluid properties to model water ($^+$) and air ($^-$).

ρ^+ [kg/m^3]	μ^+ [kg/ms]	ρ^- [kg/m^3]	μ^- [kg/ms]	σ_0 [N/m]
997.3	9.3×10^{-4}	1.1965	1.83×10^{-5}	0.0724

To track the bubble's motion a moving reference frame (MRF) is used. This MRF follows the bubble, so it stays in the center of the computational domain and the mesh is less deformed. The domain is now moving and accelerating like the bubble, making a correction in the momentum equation (adding $\rho \alpha_{MRF}$ as well as a modified boundary condition for the velocity ($v_{out} = -v_{MRF}$) necessary.

The surfactant modeled is the non-ionic dodecyl-dimethyl-phosphine-oxide ($C_{12}DMPO$). Its properties can be found in Table 3. The sorption process is modeled with the fast Langmuir model and the thin concentration boundary layer at the interface is treated with the SGS model.

We model 4 different cases, where each one starts with a clean interface. The liquid phase is initialized with a surfactant concentration

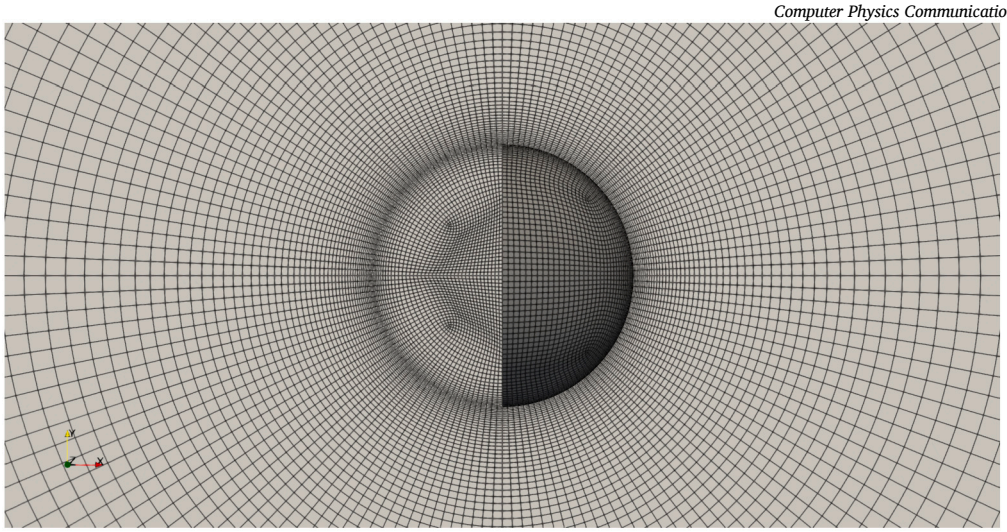


Fig. 19. Undeformed mesh and bubble at the beginning of the simulation.

Table 3
Surfactant ($C_{12}DMPO$) properties, including fast Langmuir adsorption model parameters.

c_∞^Σ [mol/m^2]	a_L [mol/m^3]	D [m^2/s]	D^Σ [m^2/s]	T [K]
4.17×10^{-6}	4.85×10^{-3}	5×10^{-10}	5×10^{-7}	296

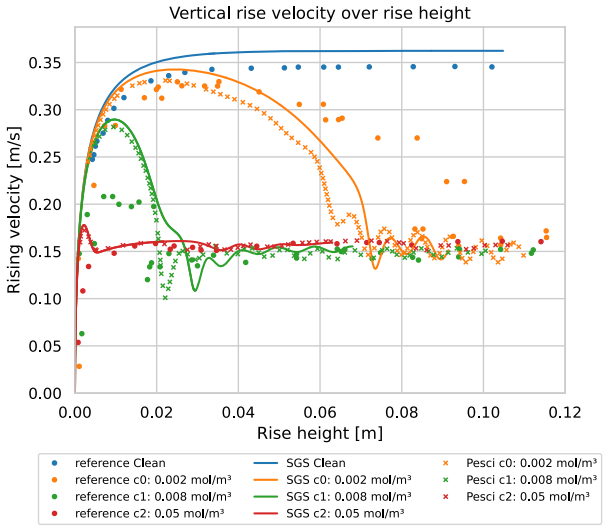


Fig. 20. Rising velocities over rising height for different initial concentrations.

of $c_{initial} = c_\infty = [0,\ 2 \times 10^{-3},\ 8 \times 10^{-3},\ 5 \times 10^{-2}]$ mol/m^3, where c_∞ is also the concentration boundary condition for inflow into the liquid domain. There is no surfactant present in the gas phase at any point.

Fig. 20 shows the rising velocities obtained by experiments [29], previous simulations [30], as well as results of our new ALE-IT implementation. There are only minor discrepancies between the two solutions obtained by ALE-IT simulations with SGS, that are far exceeded by the differences to the experimental results. Small discrepancies between two different ALE-IT methods are caused by the differences in the numerical setup, as well as uncertainties related especially to the surfactant properties, like the surface diffusion coefficient, different meshing, slightly different numerical schemes, and the Finite Element solver used for the mesh motion in [30]. Both sets of results show the characteristic features of over- and undershooting for all initial concentration levels and a good overall agreement with experimental data.

Fig. 21 shows the surfactant distribution in the bulk phases and on the surface after 0.1 s for the 3 different cases with surfactant. While a thin boundary layer can be observed in all cases, the area where it detaches and the amount accumulated on the interface vary greatly. In all cases, the surfactant accumulates at the rear part, where it immobilizes the surface, as can be concluded from the low velocities there in Fig. 20. This changes the flow field significantly. In the clean case, there is no visible detachment of the bubble wake, whereas in the case with the highest concentration, there is detachment and a recirculation zone bigger than the bubble itself. Another major difference between the cases is the shape of the bubble: the clean bubble deforms into a spheroidal form, whereas in the cases with high concentrations the bubble remains almost spherical.

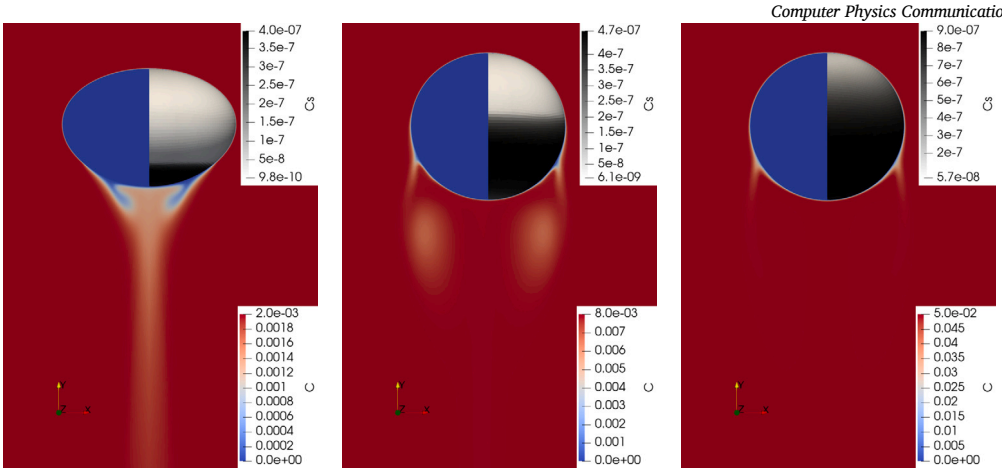
We have shown for a rising bubble in Figs. 20 to 22, that our framework can effectively capture the expected results in the rising velocity of the bubble, the surfactant concentration in the bulk and on the interface, as well as their influence on the bubble shape.

7. Conclusions

We provide a thoroughly automatically tested implementation of the unstructured Finite-Volume Arbitrary Lagrangian / Eulerian (ALE) Interface-Tracking (IT) method for simulating incompressible, immiscible two-phase flows with surfactants and Subgrid-Scale (SGS) modeling of interfacial mass transfer as an OpenFOAM module.

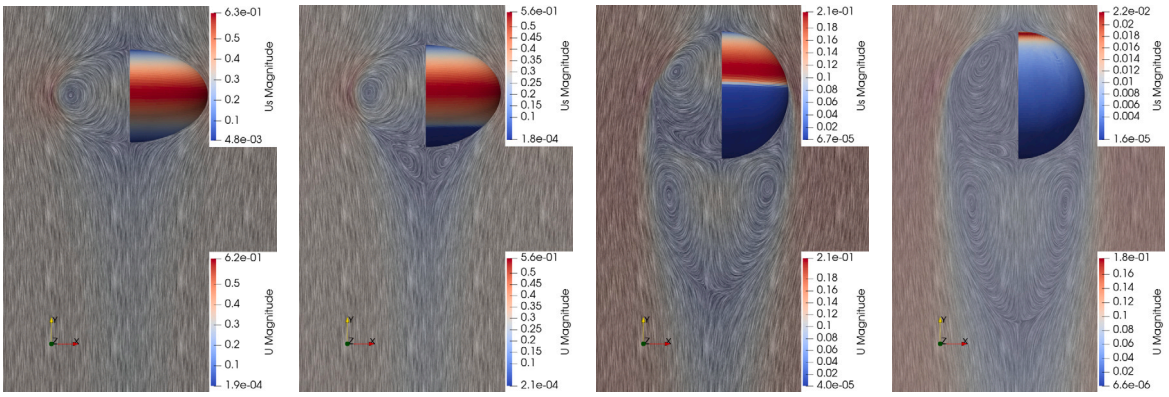
We provide a modular and easily extendible SGS model hierarchy within an ALE-IT OpenFOAM module, with configurable output parameters for in-depth analysis. The SGS model's ability to accurately capture thin boundary layers and steep gradients was demonstrated, alongside the critical need for flux correction not only at the interfaces but also across adjacent cell boundaries. Furthermore, the Interface Tracking module's proficiency in accurately simulating surfactants on moving fluid interfaces and their impact on surface tension and Marangoni flow was demonstrated. We demonstrate the capabilities of our ALE-IT implementation for simulating complex hydrodynamics with mass transfer by simulating a rising bubble with soluble surfactants. Our framework is able to reproduce the expected results, such as the rising velocity of the bubble, the surfactant concentration in the bulk and on the interface, as well as the surfactants influence on the bubble shape.

Ongoing work includes ensuring consistent contact line kinematics [14,15] for the ALE-IT framework to improve simulation accuracy for contact line kinematics and dynamics, and extensions towards tetrahedral meshes for stronger mesh deformation and topological changes [26].



(a) $c_\infty = 2 \times 10^{-3}\,\mathrm{mol/m^3}$ (b) $c_\infty = 8 \times 10^{-3}\,\mathrm{mol/m^3}$ (c) $c_\infty = 5 \times 10^{-2}\,\mathrm{mol/m^3}$

Fig. 21. Surfactants on the surface and in the bulk phases at t = 0.1 s for different concentration levels.



(a) $c_\infty = 0\,\mathrm{mol/m^3}$ (b) $c_\infty = 2 \times 10^{-3}\,\mathrm{mol/m^3}$ (c) $c_\infty = 8 \times 10^{-3}\,\mathrm{mol/m^3}$ (d) $c_\infty = 5 \times 10^{-2}\,\mathrm{mol/m^3}$

Fig. 22. Velocities on the surface and in the bulk phases at t = 0.1 s for different concentration levels.

CRediT authorship contribution statement

Moritz Schwarzmeier: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Suraj Raju:** Writing – original draft, Software, Methodology. **Željko Tuković:** Writing – original draft, Validation, Supervision, Software, Methodology, Investigation, Conceptualization. **Mathis Fricke:** Supervision, Funding acquisition, Conceptualization. **Dieter Bothe:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization. **Tomislav Marić:** Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Project administration, Methodology, Investigation, Funding acquisition, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Moritz Schwarzmeier and Tomislav Marić would like to thank the Federal Government and the Heads of Government of the Länder, as well as the Joint Science Conference (GWK), for their funding and support within the framework of the NFDI4Ing consortium. Funded by the German Research Foundation (DFG) - project number 442146713.

Funded by the German Research Foundation (DFG) – Project-ID 265191195 – SFB 1194.

Part of this work was funded by the Hessian Ministry of Higher Education, Research, Science and the Arts - cluster project Clean Circles.

The authors gratefully acknowledge the computing time provided to them on the high-performance computer Lichtenberg II at TU Darmstadt, funded by the German Federal Ministry of Education and Research (BMBF) and the State of Hesse.

Appendix A. Automatic testing

We have set up our test cases in a Continuous Integration (CI) pipeline. The pipeline is depicted in Figs. A.23 and A.24. In our approach we follow the practices presented in [23,24]. This means the tests are run and evaluated automatically in the CI with Jupyter Notebooks. They also produce the results and graphs included in this work. A job in a succeeding stage to the downstream pipeline has been omitted from Fig. A.24 for the sake of readability. Said job collects and bundles all those artifacts from the two previous stages.

In [23,24] CI is focused on tests and their automatic evaluation. We have added tests to the CI pipeline, that check certain additional aspects of the repository, like e.g. formatting and the formal correctness of the *CITATION.cff*-file. We also build an image within the CI with OpenFOAM and other needed packages. This job can not be seen in Fig. A.23, since it is only run once a new OpenFAOM version is available. It is located before the trigger job for the downstream pipelines.

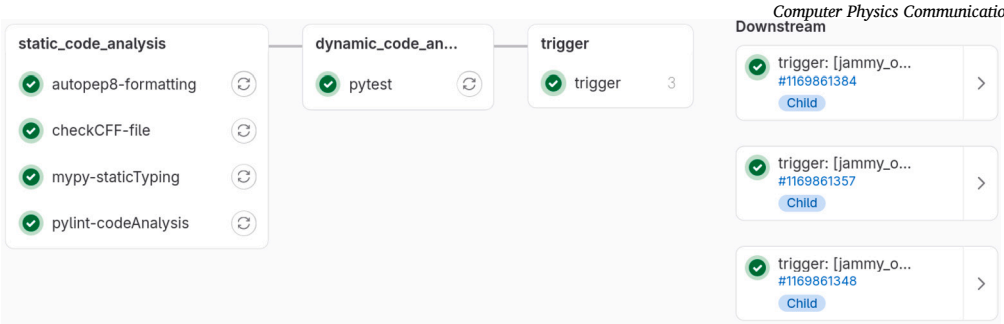


Fig. A.23. Upstream CI pipeline.

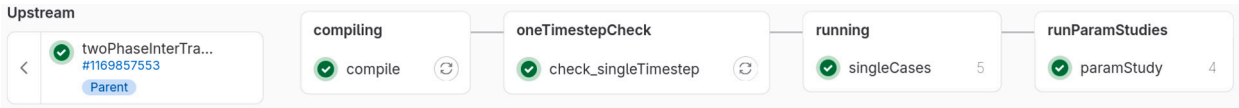


Fig. A.24. Downstream CI pipeline.

We further expanded the CI to be able to test the code for multiple OpenFOAM versions, which at the moment is being done for the latest three versions OpenFOAM `v2312`, `v2306` and `v2212`. In feature-branches only the latest version is tested to save resources.

Another feature of the CI pipeline is the addition of a test, that checks the OpenFOAM log-files for warnings and errors, where tutorials added to the repository are automatically detected and run for a single timestep. This almost ensures, that there are no errors in the set-up of test cases and would also uncover some of errors in the code, that can only be detected, when OpenFOAM is run. The automatisation of this test in finding and running all simulation cases is especially valuable. Since running all of the simulations for only one timestep needs a very limited amount of time it can be checked, that changes in the code don't corrupt any case before committing and pushing to the central repository. The automated finding of cases also prevents forgetting to add a simulation case to the CI (to this testing method) and the automated execution and error finding gently forces all participants of the repository to set up their cases to run flawlessly and fully automatised, e.g. they need to include everything needed to run a case in the *Allrun* script. This ensures all the cases provided with the code can be run without debugging.

Another topic not covered in the previously mentioned references is that for repetitive tasks, like reading dictionary entries, reading files or calculating error norms needed for the evaluation a set of python functions is provided. This set of functions is also tested automatically with the testing framework pytest as part of the CI pipeline and precedes the more computationally expensive OpenFOAM-specific jobs, as can be seen in Fig. A.23.

Appendix B. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2024.109460.

Data availability

We cite the PID of the research data archive published on the TU-datalib data repository at TU Darmstadt, we link the manuscript with the publicly available GitLab repository.

References

[1] A. Alke, D. Bothe, M. Kröger, B. Weigand, D. Weirich, H. Weking, Direct numerical simulation of high Schmidt number mass transfer from air bubbles rising in liquids using the volume-of-fluid-method, Ercoftac Bull. 82 (2010) 5–10.
[2] T. Antritter, Numerical Simulation of Coupled Wetting and Transport Phenomena in Inkjet Printing, PhD thesis, Technische Universität Darmstadt, Darmstadt, 2022.
[3] T. Antritter, T. Josyula, T. Marić, D. Bothe, P. Hachmann, B. Buck, T. Gambaryan-Roisman, P. Stephan, A two-field formulation for surfactant transport within the algebraic volume of fluid method, Comput. Fluids (ISSN 0045-7930) (March 2024) 106231, https://doi.org/10.1016/j.compfluid.2024.106231.
[4] D. Bothe, S. Fleckenstein, A volume-of-fluid-based method for mass transfer processes at fluid particles, Chem. Eng. Sci. (ISSN 0009-2509) 101 (2013) 283–302, https://doi.org/10.1016/j.ces.2013.05.029.
[5] H. Brenner, Interfacial Transport Processes and Rheology, Butterworth-Heinemann Series in Chemical Engineering, Elsevier Science, ISBN 9781483292274, 2013.
[6] K. Cai, G. Huang, Y. Song, J. Yin, D. Wang, A sub-grid scale model developed for the hexahedral grid to simulate the mass transfer between gas and liquid, Int. J. Heat Mass Transf. (ISSN 0017-9310) 181 (December 2021) 121864, https://doi.org/10.1016/j.ijheatmasstransfer.2021.121864.
[7] K.P. Chen, W. Saric, H.A. Stone, On the deviatoric normal stress on a slip surface, Phys. Fluids A 12 (12) (2000) 3280–3281.
[8] C.M.Y. Claassen, S. Islam, E.A.J.F. Peters, N.G. Deen, J.A.M. Kuipers, M.W. Baltussen, An improved subgrid scale model for front-tracking based simulations of mass transfer from bubbles, AIChE J. (ISSN 1547-5905) 66 (4) (2020) e16889, https://doi.org/10.1002/aic.16889, _eprint https://onlinelibrary.wiley.com/doi/pdf/10.1002/aic.16889.
[9] I. Demirdžić, S. Muzaferija, Numerical method for coupled fluid flow, heat transfer and stress analysis using unstructured moving meshes with cells of arbitrary topology, Comput. Methods Appl. Mech. Eng. 125 (1995) 235–255.
[10] I. Demirdžić, M. Perić, Space conservation law in finite volume calculations of fluid flow, Int. J. Numer. Methods Fluids 8 (1988) 1037–1050.
[11] K. Dieter-Kissling, M. Karbaschi, H. Marschaldenl, A. Javadi, R. Miller, D. Bothe, On the applicability of drop profile analysis tensiometry at high flow rates using an interface tracking method, Colloids Surf. A, Physicochem. Eng. Asp. 441 (2014) 837–845, https://doi.org/10.1016/j.colsurfa.2012.10.047.
[12] K. Dieter-Kissling, H. Marschall, D. Bothe, Direct numerical simulation of droplet formation processes under the influence of soluble surfactant mixtures, Comput. Fluids 113 (2015) 93–105, https://doi.org/10.1016/j.compfluid.2015.01.017.
[13] J.H. Ferziger, M. Perić, Computational Methods for Fluid Dynamics, Springer Verlag, Berlin-New York, 1995.
[14] M. Fricke, Matthias Köhne, D. Bothe, On the kinematics of contact line motion, PAMM (ISSN 1617-7061) 18 (1) (December 2018) e201800451, https://doi.org/10.1002/pamm.201800451.
[15] M. Fricke, T. Marić, D. Bothe, Contact line advection using the geometrical volume-of-fluid method, J. Comput. Phys. (ISSN 1090-2716) 407 (July 2020), https://doi.org/10.1016/j.jcp.2019.109221.
[16] M. Grosso, G. Bois, A. Toutant, Thermal boundary layer modelling for heat flux prediction of bubbles at saturation: a priori analysis based on fully-resolved simulations, Int. J. Heat Mass Transf. (ISSN 0017-9310) 222 (May 2024) 124980, https://doi.org/10.1016/j.ijheatmasstransfer.2023.124980.
[17] D. Gründing, S. Fleckenstein, D. Bothe, A subgrid-scale model for reactive concentration boundary layers for 3D mass transfer simulations with deformable fluid interfaces, Int. J. Heat Mass Transf. (ISSN 0017-9310) 101 (October 2016) 476–487, https://doi.org/10.1016/j.ijheatmasstransfer.2016.04.119.
[18] J.C. Slattery, L. Sagis, E.-S. Oh, Interfacial Transport Phenomena, Springer US, Boston, MA, ISBN 978-0-387-38438-2, 2007.
[19] H. Jasak, Error analysis and estimation for finite volume method with applications to fluid flows, PhD thesis, Imperial College, University of London, 1996.
[20] F. Juretić, A.D. Gosman, Error analysis of the finite-volume method with respect to mesh type, Numer. Heat Transf., Part B, Fundam. (ISSN 1040-7790) 57 (6) (June 2010) 414–439, https://doi.org/10.1080/10407791003685155.

[21] P.A. Kralchevsky, K.D. Danov, N.D. Denkov, Chemical physics of colloid systems and interfaces, in: K.S. Birdi (Ed.), Handbook of Surface and Colloid Chemistry, second expanded and updated edition, CRC Press, New York, ISBN 978-0-429-12509-6, 2002, p. 208.

[22] T. Marić, J. Höpken, K.G. Mooney, The OpenFOAM technology primer, https://doi.org/10.5281/zenodo.4630596, March 2021, Version OpenFOAM-v2012.1.

[23] T. Marić, D. Gläser, J.P. Lehr, I. Papagiannidis, B. Lambie, C. Bischof, D. Bothe, A research software engineering workflow for computational science and engineering, https://doi.org/10.48550/arXiv.2208.07460, August 2022.

[24] T. Marić, D. Gläser, J.P. Lehr, I. Papagiannidis, B. Lambie, Christian Bischof, D. Bothe, A pragmatic workflow for research software engineering in computational science, https://doi.org/10.48550/ARXIV.2310.00960, 2023.

[25] H. Marschall, S. Boden, C. Lehrenfeld, Carlos J. Falconi D., U. Hampel, A. Reusken, M. Wörner, D. Bothe, Validation of interface capturing and tracking techniques with different surface tension treatments against a Taylor bubble benchmark problem, Comput. Fluids (ISSN 0045-7930) 102 (October 2014), https://doi.org/10.1016/j.compfluid.2014.06.030.

[26] S. Menon, K.G. Mooney, K.G. Stapf, D.P. Schmidt, Parallel adaptive simplical remeshing for deforming domain CFD computations, J. Comput. Phys. (ISSN 0021-9991) 298 (October 2015) 62–78, https://doi.org/10.1016/j.jcp.2015.05.044.

[27] M. Muradoglu, G. Tryggvason, A front-tracking method for computation of interfacial flows with soluble surfactants, J. Comput. Phys. 227 (4) (February 2008) 2238–2262, https://doi.org/10.1016/j.jcp.2007.10.003.

[28] S. Muzaferija, M. Perić, Computation of free-surface flows using the finite-volume method and moving grids, Numer. Heat Transf., Part B 32 (1997) 369–384.

[29] C. Pesci, H. Marschall, T. Kairaliyeva, V. Ulaganathan, R. Miller, D. Bothe, Experimental and computational analysis of fluid interfaces influenced by soluble surfactant, in: Dieter Bothe, Arnold Reusken (Eds.), Transport Processes at Fluidic Interfaces, Springer International Publishing, Cham, ISBN 978-3-319-56602-3, 2017, pp. 395–444.

[30] C. Pesci, A. Weiner, H. Marschall, D. Bothe, Computational analysis of single rising bubbles influenced by soluble surfactant, J. Fluid Mech. (ISSN 0022-1120) 856 (709–763) (2018) 1469–7645, https://doi.org/10.1017/jfm.2018.723.

[31] S. Quan, J. Lou, D.P. Schmidt, Modeling merging and breakup in the moving mesh interface tracking method for multiphase flow simulations, J. Comput. Phys. (ISSN 0021-9991) 228 (7) (April 2009) 2660–2675, https://doi.org/10.1016/j.jcp.2008.12.029.

[32] R. Satapathy, W. Smith, The motion of single immiscible drops through a liquid, J. Fluid Mech. 10 (4) (1961) 561, https://doi.org/10.1017/S0022112061000366, ISSN 0022-1120, 1469-7645.

[33] M. Schwarzmeier, S. Raju, Z. Tuković, T. Marić, Twophaseintertrackfoam source code repository, GitLab, https://gitlab.com/interface-tracking/twophaseintertrackfoamrelease/-/tree/v1.1?ref_type=tags, 2024. (Accessed 1 November 2024).

[34] M. Schwarzmeier, S. Raju, Z. Tuković, T. Marić, twoPhaseInterTrackFoam - data & results v1, 2024-11-01, https://doi.org/10.48328/tudatalib-1396.2.

[35] P.D. Thomas, C.K. Lombard, Geometric conservation law and its application to flow computations on moving grids, AIAA J. 17 (1979) 1030–1037.

[36] Ž. Tuković, H. Jasak, Simulation of free-rising bubble with soluble surfactant using moving mesh finite volume/area method, in: Proceedings of 6th International Conference on CFD in Oil & Gas, Metallurgical and Process Industries, no. CFD08-072, 2008.

[37] Ž. Tuković, M. Perić, H. Jasak, Consistent second-order time-accurate non-iterative PISO-algorithm, Comput. Fluids 166 (April 2018) 78–85, https://doi.org/10.1016/j.compfluid.2018.01.041.

[38] Z. Tuković, M. Schwarzmeier, S. Raju, T. Marić, twoPhaseIntertRackFoam - code, 2024-11-01, https://doi.org/10.48328/tudatalib-1395.2.

[39] Ž. Tuković, H. Jasak, A moving mesh finite volume interface tracking method for surface tension dominated interfacial fluid flow, Comput. Fluids 55 (February 2012) 70–84, https://doi.org/10.1016/j.compfluid.2011.11.003.

[40] C.E. Weatherburn, Differential Geometry in Three Dimension, Cambridge University Press, London, 1972.

[41] P.S. Weber, H. Marschall, D. Bothe, Highly accurate two-phase species transfer based on ale interface tracking, Int. J. Heat Mass Transf. 104 (2017) 759–773.

[42] A. Weiner, D. Bothe, Advanced subgrid-scale modeling for convection-dominated species transport at fluid interfaces with application to mass transfer from rising bubbles, J. Comput. Phys. (ISSN 0021-9991) 347 (2017) 261–289, https://doi.org/10.1016/j.jcp.2017.06.040.

[43] A. Weiner, D. Gründing, D. Bothe, Computing mass transfer at deformable bubbles for high Schmidt numbers, Chem. Ing. Tech. (ISSN 1522-2640) 93 (1–2) (2021) 81–90, https://doi.org/10.1002/cite.202000214.

[44] N.O. Young, J.S. Goldstein, M.J. Block, The motion of bubbles in a vertical temperature gradient, J. Fluid Mech. 6 (03) (October 1959) 350, https://doi.org/10.1017/s0022112059000684.