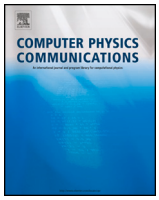
Computer Physics Communications 313 (2025) 109640

Contents lists available at ScienceDirect

Computer Physics Communications

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



Computer Programs in Physics

FLOW36: A spectral solver for phase-field based multiphase turbulence simulations on heterogeneous computing architectures

Alessio Roccon [a,b,*], Giovanni Soligo [c], Alfredo Soldati [a,b]

[a] Institute of Fluid Mechanics and Heat Transfer, TU-Wien, 1060 Vienna, Austria
[b] Polytechnic Department of Engineering and Architecture, University of Udine, 33100 Udine, Italy
[c] Complex Fluids and Flows Unit, OIST, 904-0495 Okinawa, Japan

ARTICLE INFO

The review of this paper was arranged by Prof. Weigel Martin

ABSTRACT

We present FLOW36, a GPU-ready solver for interface-resolved simulations of multiphase turbulence. The simulation framework relies on the coupling of direct numerical simulation of turbulence, used to describe the flow field, with a phase-field method, used to describe the shape and deformation of a deformable interface and the presence of surfactants. An additional transport equation for a passive scalar can be solved to describe heat transfer in multiphase turbulence. The governing equations are solved in a cuboid domain bounded by two walls along the wall-normal direction where no-slip, free-slip or fixed/moving wall boundary conditions can be applied, while periodicity is applied along the streamwise and spanwise directions. The numerical method relies on a pseudo-spectral approach where Fourier series (periodic directions) and Chebyshev polynomials (wall-normal direction) are used to discretize the governing equations in space. Equations are advanced in time using an implicit-explicit scheme. From a computational perspective, FLOW36 relies on a multilevel parallelism. The first level of parallelism relies on the message-passing interface (MPI). A second level of parallelism uses OpenACC directives and cuFFT libraries; this second level is used to accelerate the code execution when heterogeneous computing infrastructures are targeted. In this work, we present the numerical method and we discuss the main implementation strategies, with particular reference to the MPI and OpenACC directives and code portability, performance and maintenance strategies. FLOW36 is released open source under the GPLv3 license.

Program summary

Program Title: FLOW36
CPC Library link to program files: https://doi.org/10.17632/ygcn7dsb9k.1
Developer's repository link: https://github.com/MultiphaseFlowLab/FLOW36
Licensing provisions: GPLv3 License
Programming language: Modern Fortran
Nature of problem: Solving the three-dimensional incompressible Navier–Stokes equations in a Cartesian domain configured for open and closed channel flows. A phase-field method is used to describe the shape and topological changes of deformable interfaces. Additional equations are included to account for the presence of surfactants, heat transfer problems and for the transport of point-wise Lagrangian particles.
Solution method: The system of governing equations is advanced in time using an implicit-explicit strategy while the governing equations are discretized in space using a pseudo-spectral approach: Fourier series are employed along the homogeneous directions while Chebyshev polynomial along the wall-normal direction. A first order explicit Euler method is used to advance the equations for the Lagrangian particles motion. A two-dimensional pencil distributed MPI parallelization is implemented and OpenACC directives are used to execute the code on GPUs.

* Corresponding author at: Polytechnic Department, University of Udine, 33100 Udine, Italy.
E-mail address: alessio.roccon@uniud.it (A. Roccon).

https://doi.org/10.1016/j.cpc.2025.109640
Received 25 July 2024; Received in revised form 27 March 2025; Accepted 22 April 2025
Available online 29 April 2025

1. Introduction

Turbulent multiphase flows are ubiquitous in nature and our everyday life. These flow instances play a key-role in many different applications, from geophysical phenomena [1,2] to environmental and industrial processes [3–5]. With respect to single-phase turbulence, the description and modeling of multiphase turbulence is more challenging: these flows require the description of an ever-moving and deforming interface, its topological changes, and the underlying turbulent flow [6–9]. Simulations are of vital importance to obtain a better understanding of the physics of multiphase turbulence and are becoming increasingly popular in recent years: numerical investigations give access to detailed space- and time-resolved data on the flow field and on the morphology of the two phases as well as other quantities of interest.

Obtaining an accurate description of the dynamics of a turbulent multiphase flow on a discretized temporal and spatial grid is however a challenging task because of the large scale separation that characterizes these flows: scales range from the largest flow scale (of the order of the domain size), down to the Kolmogorov scale of turbulence and further down to the molecular scale of the interface. This has direct implications on the description of multiphase turbulence as the spatio-temporal resolution one can reasonably afford is limited by computing capabilities [10]. Specifically, as done in the past for single-phase turbulence [11–13], it would be desirable to perform simulations in which all scales are directly resolved, without any model. However, this approach cannot be applied to multiphase flows: the scale separation between the largest flow scale and the smallest interfacial scale is about eight to nine orders of magnitude, while the most recent high-performance computing (HPC) infrastructures can handle a maximum scale separation of about three to four orders of magnitude.

FLOW36 performs direct numerical simulations (DNS) of the Navier-Stokes equations, used to describe the flow field, coupled with a phase-field method (PFM), used to describe interfacial phenomena and surfactant concentration. Additional equations can be solved to describe the dynamic of a passive scalar and the motion of point-wise Lagrangian particles. The governing equations are solved using a pseudo-spectral method based on transforming the field variables into wave-number space via a combination of Fourier series (in the periodic streamwise and spanwise directions) and Chebyshev polynomials (in the inhomogeneous wall normal direction). The governing equations are advanced in time using an implicit-explicit scheme. The employed numerical method allows us to obtain – for each x, y location – a series of 1D Helmholtz problems along the wall-normal direction is obtained; the solution of these 1D problems can be readily parallelized using domain decomposition.

In computational fluid dynamics, a large amount of computing resources are usually required to solve complex flow problems, especially when direct numerical simulations of multiphase turbulence are involved. For this reason, FLOW36 targets Tier-0 GPU-accelerated high-performance computing infrastructures, i.e. heterogeneous computing architectures. The code relies on a multilevel approach (MPI + X) to exploit both CPU- and GPU-based computing facilities. The first level of parallelism relies on MPI and FFTW libraries [14] and it is employed when the code is executed on CPU-only computing infrastructures. A second level of parallelism employs OpenACC directives and cuFFT libraries [15] and is used to accelerate the code execution when GPU-based infrastructures are targeted. With the idea of having a single code that can be executed on different architectures, we choose here to follow the directive-based approach offered by OpenACC [16]. Among the available approaches, it has the following advantages: i) a single code has to be maintained: OpenACC directives are ignored when GPU support is not enabled; ii) in the framework of the Nvidia HPC Software Development Kit (HPC-SDK) it allows for the use of the managed memory feature, which greatly simplifies the management of the data transfers between CPU and GPU memories; iii) additional features, like the solution of new governing equations, can be easily implemented as the development time required to port new code sections to GPU architectures is largely reduced (with respect to the other available approaches). Clearly, the selected approach has also some drawbacks: i) a smaller number of tools is available to optimize and fine-tune the application; ii) as the managed memory feature relies on the page fault mechanism, the available bandwidth for host-to-device (and device-to-host) transfers is slightly smaller with respect to the case when memory transfers are explicitly defined and pinned memory buffers are used.

From a programming perspective, FLOW36 has been developed with code modularity in mind so that it can be efficiently used to study problems with very different physics. For this reason, conditional compilation directives and compilation flags are used to enable/disable the modules required. This ensures the most efficient implementation and avoids possible issues with undefined parameters and/or subroutines. Indeed, the compiler will preprocess the source-code and will skip the compilation of the code sections not required by the user. A similar strategy is also used to select the most appropriate FFT library (FFTW or cuFFT) depending on whether CPU-only or GPU-accelerated computing infrastructures are used.

The paper is organized as follows. In Section 2, the governing equations are presented; in Section 3 the numerical method is detailed. Then, in Section 4, the parallelization strategy is presented and strong scaling results are reported. In Section 5, benchmark flows are used to demonstrate the accuracy of the approach and discretization used. Finally, we draw our conclusions in Section 6.

2. Governing equations

To describe the dynamics of the system, direct numerical simulations (DNS) of the Navier-Stokes equations, used to describe the flow field, are coupled with a phase-field method (PFM), used to describe interfacial phenomena. The PFM is based on the Cahn-Hilliard (CH) equation, which describes the interface position and ensures conservation of the phase-field variable through a thermodynamically-consistent formulation. If surfactants are present, a second CH-like equation can be employed to describe their concentration. The CH model is well suited to the pseudo-spectral discretization thanks to the smooth transition between the two bulk values attained by the phase-field variable. The PFM offers similar advantages to level-set methods: the interface normals and curvature can be accurately computed without resorting to more sophisticated algorithms, resulting in a more accurate calculation of surface tension and overall a lower magnitude of spurious currents [17,18]. In addition, with the formulation presented here, the conservation of each phase is greatly improved, becoming comparable to volume of fluid methods. The Eulerian formulation of the PFM allows to implicitly account for interface breakage and coalescence, a feature that requires instead advanced interface breaking and merging algorithms in front-tracking methods. An additional transport equation is used to describe the advection and diffusion of a passive scalar and a Lagrangian tracker is used to simulate point-wise particles. In the following, a brief description of each governing equation is provided; all equations are provided in dimensionless form, in the same formulation they are implemented in FLOW36.

2.1. Navier-Stokes equations

To describe the flow field of the multiphase system, a one-fluid approach is employed and a single set of Navier-Stokes equations is solved in the entire computational domain [7,9]. We consider two incompressible and Newtonian phases that can have different densities and viscosities. With these assumptions, the Navier-Stokes equations read as:

$$\nabla \cdot \mathbf{u} = 0\,, \tag{1}$$

$$\rho(\phi)\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \frac{1}{Re_\tau}\nabla\cdot\left[\mu(\phi)(\nabla\mathbf{u} + \nabla\mathbf{u}^T)\right] + \frac{\rho\mathbf{g}}{Fr^2} + \mathbf{f}_\sigma\,, \tag{2}$$

where $\mathbf{u} = (u, v, w)$ is the velocity vector, $\rho(\phi)$ and $\mu(\phi)$ are the local density and viscosity, p is the pressure field, $\mathbf{g}$ is the gravity vector, $\mathbf{f}_\sigma$ represents the surface tension forces.

The non-dimensional local values of density and viscosity in the domain as a function of the phase-field are defined as follows [19–22]:

$$\rho(\phi) = 1 + (\rho_r - 1)\frac{\phi + 1}{2}, \qquad \text{with} \qquad \rho_r = \frac{\rho_2}{\rho_1}, \tag{3}$$

$$\mu(\phi) = 1 + (\eta_r - 1)\frac{\phi + 1}{2}, \qquad \text{with} \qquad \mu_r = \frac{\mu_2}{\mu_1}, \tag{4}$$

where ρ_1 and ρ_2 (η_1 and η_2) are the densities (viscosities) of the carrier (phase 1) and dispersed (phase 2) phases, respectively. When matched properties are considered or a single-phase flow is examined, density and viscosity are uniform and unitary everywhere, i.e. $\rho(\phi) = 1$ and $\mu(\phi) = 1$.

The surface tension forces are here computed using a continuum-surface stress approach as follows [23,24]:

$$\mathbf{f}_\sigma = \frac{3Ch}{\sqrt{8}We}\nabla \cdot [f_\sigma(\psi)\bar{\tau}_c], \tag{5}$$

where Ch is a parameter that determines the characteristic width of the thin interfacial layer between the two phases; $\bar{\tau}_c = |\nabla\phi|^2\mathbf{I} - \nabla\phi \otimes \nabla\phi$ is the Korteweg tensor used to model surface tension forces [25] and $f_\sigma(\psi)$ is the equation of state that describes the surface tension reduction induced by the local surfactant concentration. This expression accounts for both the normal and tangential (Marangoni) components of the interfacial forces. When surfactant is not considered $f_\sigma(\psi) \equiv 1$ and the typical expression for clean interfaces is retrieved [26–28].

The friction Reynolds number, Re_τ, represents the ratio between the inertial and viscous forces (computed using the phase 1 properties as a reference) and is defined as:

$$Re_\tau = \frac{\rho_1 u_\tau h}{\mu_1}, \tag{6}$$

where $u_\tau = \sqrt{\tau_w/\rho_1}$ is the friction velocity and h the half-height of the channel. The Froude number is defined as:

$$Fr = \frac{u_\tau}{\sqrt{|\mathbf{g}|h}}, \tag{7}$$

where $|\mathbf{g}|$ is the magnitude of the gravitational acceleration. Finally, the Weber number, We, represents the ratio between inertial and surface tension forces and is defined as:

$$We = \frac{\rho_1 u_\tau^2 h}{\sigma_0}, \tag{8}$$

where σ_0 is the surface tension of a clean (surfactant-free) interface.

2.2. Phase-field method

The phase-field method formulation here employed relies on two scalar order parameters to describe the shape of a deformable interface and of the surfactant concentration [26,29–31]. A first order parameter, the phase field, ϕ, describes the shape and position of the interface. A second order parameter, ψ, is used to track the concentration of surfactant. The phase field variable is uniform in the bulk of the two phases ($\phi = \pm 1$) and undergoes a smooth transition following a hyperbolic tangent profile throughout the thin transition layer; the equilibrium concentration of surfactant is instead uniform in the bulk of the two phases and reaches its maximum value at the interface.

The time evolution of the two order parameters is described by two Cahn-Hilliard-like equations:

$$\frac{\partial\phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = \frac{1}{Pe_\phi}\nabla^2\mu_\phi, \tag{9}$$

$$\frac{\partial\psi}{\partial t} + \mathbf{u}\cdot\nabla\psi = \frac{1}{Pe_\psi}\nabla\cdot\left[\psi(1-\psi)\nabla\mu_\psi\right], \tag{10}$$

where Pe_ϕ and Pe_ψ are the phase field and the surfactant Péclet numbers and μ_ϕ and μ_ψ are the corresponding chemical potentials. The two Péclet numbers are defined as follows:

$$Pe_\phi = \frac{u_\tau h}{\mathcal{M}_\phi \beta}; \qquad Pe_\psi = \frac{u_\tau h \alpha}{\mathcal{M}_\psi \beta^2}, \tag{11}$$

where $\mathcal{M}_\phi$ and $\mathcal{M}_\psi$ are the phase field and the surfactant mobilities, while α and β are positive constants used in the dimensionless procedure. From a physical point of view, the two Péclet numbers represent the ratio between the diffusive time scale, $h^2/(\mathcal{M}_\phi\beta)$ and $h^2\alpha/(\mathcal{M}_\psi\beta^2)$, and the convective time scale, h/u_τ.

The chemical potentials μ_ϕ and μ_ψ are defined as the functional derivative of a two-order-parameter Ginzburg-Landau free-energy functional $\mathcal{F}$, which accounts for the interfacial motion and for the surfactant dynamics [29,32,26,27]. The expressions of the chemical potentials are:

$$\mu_\phi = \frac{\delta\mathcal{F}}{\delta\phi} = \phi^3 - \phi - Ch^2\nabla^2\phi, \tag{12}$$

$$\mu_\psi = \frac{\delta\mathcal{F}}{\delta\psi} = Pi\log\left(\frac{\psi}{1-\psi}\right) - \frac{(1-\phi^2)^2}{2} + \frac{\phi^2}{2E_x}, \tag{13}$$

where Pi and E_x are dimensionless parameter that control the surfactant dynamics and which are defined as:

$$Pi = \frac{\kappa T\alpha}{\beta^2}; \qquad E_x = \frac{\beta}{w}, \tag{14}$$

where T is the absolute temperature, while α, β, κ and w are parameters of the free energy functional. Please refer to Soligo et al. (2019) [26] for further details on the phase-field methodology.

2.3. Transport equation for a passive scalar

To describe the advection and diffusion of a passive scalar such as in heat transfer problems, an additional transport equation can be solved. Considering a generic scalar quantity, θ, the corresponding transport equation reads as follows:

$$\frac{\partial\theta}{\partial t} + \mathbf{u}\cdot\nabla\theta = \frac{1}{Pe_\theta}\nabla^2\theta \tag{15}$$

where θ represents the scalar field (e.g. temperature).

The dimensionless number that appears in the transport equation is the Péclet number of the passive scalar:

$$Pe_\theta = \frac{u_\tau h}{\alpha} = \frac{u_\tau h}{\nu_1}\frac{\nu_1}{\alpha} = Re_\tau Pr, \tag{16}$$

where α is the diffusivity of the scalar quantity and $\nu_1 = \mu_1/\rho_1$. The Péclet number represents the ratio between convective and diffusive time scales. This dimensionless number can be also rewritten as the product between the shear Reynolds number and the Prandtl number: $Pr = \nu_1/\alpha$.

2.4. Lagrangian particle tracking

A Lagrangian particle tracker is implemented in the code FLOW36 for spherical, sub-Kolmogorov-sized particles. Within the point-particle approximation, the equations of motion of the particles read as:

$$\frac{\partial\mathbf{x}_p}{\partial t} = \mathbf{u}_p, \tag{17}$$

$$\frac{\partial\mathbf{u}_p}{\partial t} = \frac{\mathbf{u}_{f,p} - \mathbf{u}_p}{St}C_d + \frac{1}{Re_\tau Fr^2}\left(\frac{\rho_p}{\rho_1} - 1\right)\mathbf{g}. \tag{18}$$

The subscript p denotes particle quantities (e.g., $\mathbf{x}_p$ the particle position, or $\mathbf{u}_p$ the particle velocity), whereas the subscript f, p denotes fluid quantities at the particle position (e.g., $\mathbf{u}_{f,p}$ denotes the fluid velocity at the particle position). Fourth-order Lagrangian interpolation is adopted to interpolate quantities defined on the Eulerian grid onto the particle position. The Stokes number, $St = \tau_p/\tau_f$, is defined as the ratio between

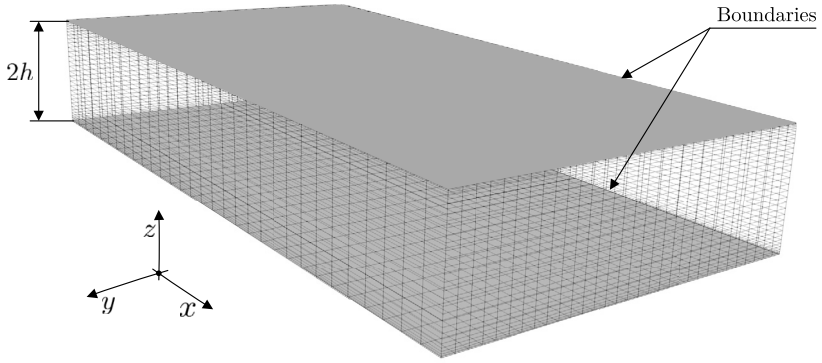


Fig. 1. Sketch of the cuboid domain employed in FLOW36. The two boundaries are located at $z = \pm h$, while arbitrary dimensions can be defined along the homogeneous periodic directions x and y. Collocation points for all variables are equally spaced along the homogeneous directions while they are stretched along the wall-normal direction, coarser at the channel center and finer near the two walls (Chebyshev-Gauss-Lobatto quadrature points).

the particle relaxation time, $\tau_p = \rho_p d_p^2/18\mu_1$, and the fluid characteristic time, $\tau_f = h/u_\tau$.

The drag correction C_d is defined as:

$$C_d = \begin{cases} 1, & \text{Stokes Drag} \\ 1 + 0.15 Re_p^{0.687}, & \text{Schiller-Naumann correction [33]} \end{cases} \tag{19}$$

where $Re_p = \rho_1 ||\mathbf{u}_{f,p} - \mathbf{u}_p|| d_p/\mu_1$ is the particle Reynolds number (d_p is the particle diameter). The latter term of equation (18) accounts for the balance of gravity and buoyancy on the particle; Fr is the Froude number (equation (7)) and $\mathbf{g}$ is the unit-vector gravity acceleration (being $|\mathbf{g}|$ its magnitude).

In the present form, particles and fluid are one-way coupled: the fluid velocity affects the motion of the particles, however, there is no back-reaction on the fluid due to the presence of the particles and particle-particle interactions are not considered either. These assumptions hold true in the dilute regime, where particles back-reaction on the fluid is negligible and the probability of particle-particle collisions is low. Extension to two-way coupling (i.e., including the back-reaction on the fluid) is straightforward: the subroutines to redistribute the forcing due to the particles onto the fluid are already implemented and only the model for the particle back-reaction is missing. Section 4.1.1 discusses the parallelization approach for the Lagrangian particle tracker and the possibility to extend the present algorithm to two- and four-way coupling.

The Lagrangian particle tracking presented here has already been used in previous works, where an additional forcing term was implemented to simulate particle adhesion to an interface [34,35].

3. Numerical method

The governing equations (1)-(2)-(9)-(10) and (15) are solved in a channel geometry using a pseudo-spectral method, Fig. 1. In particular, the equations are discretized using Fourier series in the streamwise and spanwise directions (x and y) and Chebyshev polynomials along the wall-normal direction (z). This implies that x and y directions are periodic while no-slip, free-slip or imposed velocity boundary conditions can be imposed at the two walls. This limits the possible configurations that can be simulated to Couette and Poiseuille flows and combinations thereof. The governing equations are advanced in time using an IMplicit–EXplicit (IMEX) scheme: the linear diffusive term of the equations is integrated using an implicit scheme, whereas the non-linear term is integrated using an explicit scheme. In the following, after a brief description of the variables collocation, the spatial and temporal discretization is detailed for each governing equation listed above.

3.1. Collocation points, grid resolution and dual-grid approach

In the context of pseudo-spectral methods, all Eulerian variables are defined on the collocation points. Specifically, the velocity vector $\mathbf{u}$, the phase field variable ϕ, the surfactant concentration ψ and the scalar concentration field θ are defined on the same Cartesian grid with $N_x \times N_y \times N_z$ grid points. In FLOW36, it is also possible to define variables on different grid resolutions (the so-called dual-grid approach) with respect to that used to solve the Navier-Stokes equations, which is the reference grid. This is possible by specifying the expansion factors M_x, M_y and M_z. Therefore, in this latter case, the specified variable is defined on a grid having a resolution equal to $M_x N_x \times M_y N_Y \times (M_z(N_z - 1) + 1)$. In the version of FLOW36 available in the online repository, only the surfactant concentration field can be defined on a more refined grid. Nevertheless, with minor code modifications, also the phase-field and scalar transport equations can be ported on a more refined grid [36,37]. In particular, to upscale the solution of a governing equation from the reference grid to the refined grid, the subroutines for the "physical-to-spectral" and "spectral-to-physical" transforms must be replaced with their `_fg` counterparts, which handle transforms on the refined grid. In addition, the corresponding array dimensions should be also updated. Fluid data and coupling terms have to be computed similarly to how is done for the surfactant: the coupling term is computed on the finer grid and then transformed onto the coarser grid, whereas the fluid data is available on the coarser grid and has to be transformed onto the finer grid to be used in the transport equation. Transformation between the two different grids occurs in spectral space: zero padding is used to transform a variable from coarse to fine grid, and elimination of the high-wavenumber spectral modes coupled with the 2/3 de-aliasing rule is used to transform a variable from fine to coarse grid.

3.2. Navier-Stokes equations

Before applying the numerical discretization, the Navier-Stokes equations are recasted in a more compact form collecting all the non-linear terms in the term $\mathbf{S}$:

$$\nabla \cdot \mathbf{u} = 0; \tag{20}$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{S} - \nabla p + \frac{1}{Re_\tau}\nabla^2 \mathbf{u}; \tag{21}$$

where the term $\mathbf{S} = (S_x, S_y, S_z)$ is defined as:

$$\mathbf{S} = -\frac{\rho_r - 1}{2}(\phi + 1)\frac{\partial \mathbf{u}}{\partial t} - \left(1 + \frac{\rho_r - 1}{2}(\phi + 1)\right)\mathbf{u} \cdot \nabla \mathbf{u} - \Pi + \frac{1}{Re_\tau}\nabla \cdot \left(\frac{\mu_r - 1}{2}(\phi + 1)(\nabla \mathbf{u} + \nabla \mathbf{u}^T)\right) +$$

$$+\frac{1}{Fr^2}\left(1+\frac{\rho_r-1}{2}(\phi+1)\right)\mathbf{g}+\frac{3}{\sqrt{8}}\frac{Ch}{We}\nabla\cdot\left(\overline{\tau}_c f_\sigma(\psi)\right). \tag{22}$$

The Navier–Stokes equations are not directly solved but are rewritten in the so-called wall-normal velocity–vorticity formulation [11,38,39]. Instead of three 2^{nd} order equations for each component of the velocity, a 4^{th}-order equation for the wall-normal component of the velocity $w = \mathbf{u}\cdot\mathbf{k}$ (being $\mathbf{k}$ the unit vector of the wall-normal direction) and a 2^{nd}-order equation for the wall normal vorticity $\omega_z = (\nabla\times\mathbf{u})\cdot\mathbf{k}$ are obtained. To obtain the governing equations for the wall-normal velocity-vorticity formulation, we first take the curl of the Navier-Stokes equations. The pressure gradient term vanishes thanks to the identity $\nabla\times\nabla p = 0$, and a transport equation for the vorticity vector, $\boldsymbol{\omega}$, is obtained:

$$\frac{\partial\boldsymbol{\omega}}{\partial t} = \nabla\times\mathbf{S}+\frac{1}{Re_\tau}\nabla^2\omega. \tag{23}$$

By taking again the curl of the vorticity transport equation, we obtain the following 4-th order equation for the velocity vector:

$$\frac{\partial\nabla^2\mathbf{u}}{\partial t} = \nabla^2\mathbf{S}-\nabla(\nabla\cdot\mathbf{S})+\frac{1}{Re_\tau}\nabla^4\mathbf{u}. \tag{24}$$

We solve here for the wall-normal components of the vorticity, ω_z, and velocity, w. Hence, instead of the Navier-Stokes equations expressed in term of primitive variables, the following governing equations are solved:

$$\frac{\partial\omega_z}{\partial t} = \frac{\partial S_y}{\partial x}-\frac{\partial S_x}{\partial y}+\frac{1}{Re_\tau}\nabla^2\omega_z, \tag{25}$$

$$\frac{\partial(\nabla^2 w)}{\partial t} = \nabla^2 S_z-\frac{\partial}{\partial z}\left(\frac{\partial S_x}{\partial x}+\frac{\partial S_y}{\partial y}+\frac{\partial S_z}{\partial z}\right)+\frac{1}{Re_\tau}\nabla^4 w, \tag{26}$$

complemented by the continuity equation (1) and the definition of wall-normal vorticity:

$$\omega_z = \frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}. \tag{27}$$

The four equations just obtained are then discretized in time and space; the hat notation is used to identify the variable in Fourier space (pseudo-spectral discretization in space). For the 4^{th}-order equation for the wall-normal velocity we obtain:

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{\partial^2}{\partial z^2}-k_{xy}^2\right)\hat{w} = &\left(\frac{\partial^2}{\partial z^2}-k_{xy}^2\right)\hat{S}_z\\ &-\frac{\partial}{\partial z}\left(ik_x\hat{S}_x+ik_y\hat{S}_y+\frac{\partial}{\partial z}\hat{S}_z\right)\\ &+\frac{1}{Re_\tau}\left(\frac{\partial^2}{\partial z^2}-k_{xy}^2\right)\left(\frac{\partial^2}{\partial z^2}-k_{xy}^2\right)\hat{w}\end{aligned} \tag{28}$$

where k_x and k_y are the corresponding wave-number (along x and y, respectively) and $k_{xy}^2 = k_x^2+k_y^2$; the discrete derivative along the wall-normal direction (Chebyshev polynomials) is identified with $\partial/\partial z$. We can now apply the IMEX temporal discretization scheme employing a Crank-Nicholson scheme for the linear part and an Adams-Bashforth scheme for the non-linear part. After some algebraic manipulation, we obtain:

$$\left(\frac{\partial^2}{\partial z^2}-\lambda^2\right)\left(\frac{\partial^2}{\partial z^2}-k_{xy}^2\right)\hat{w}^{n+1} = \frac{\hat{H}^n}{\gamma} \tag{29}$$

where $\hat{H}^n$ is an historical term that depends only on the previous time steps, $\gamma = \Delta t/2Re_\tau$ is a numerical coefficient and $\lambda^2 = (1+\gamma k_{xy}^2)/\gamma$. We can introduce the auxiliary variable:

$$\hat{\theta}_w = \left(\frac{\partial^2}{\partial z^2}-k_{xy}^2\right)\hat{w}^{n+1}, \tag{30}$$

and the 4^{th}-order equation can be split in two 2^{nd}-order equations:

$$\left(\frac{\partial^2}{\partial z^2}-\lambda^2\right)\hat{\theta}_w = \frac{\hat{H}^n}{\gamma}, \tag{31}$$

$$\left(\frac{\partial^2}{\partial z^2}-k_{xy}^2\right)\hat{w}^{n+1} = \hat{\theta}_w. \tag{32}$$

These two equations represent a series of 1D problem for each (k_x, k_y) wavenumber pair that can be solved along the wall-normal direction using a Chebyshev-Tau scheme with appropriate boundary conditions. For a no-slip boundary, we set:

$$\hat{w} = 0; \qquad \frac{\partial\hat{w}}{\partial z} = 0, \tag{33}$$

while for a free-slip boundary:

$$\hat{w} = 0; \qquad \frac{\partial^2\hat{w}}{\partial z^2} = 0. \tag{34}$$

It is important to observe that boundary conditions cannot be directly obtained for the auxiliary variable $\hat{\theta}$. To circumvent this issue, an influence matrix technique is employed. Once this series of problem is solved, the value of the wall-normal velocity at the step $n+1$ is obtained.

The same procedure can be applied to the transport equation for the wall-normal component of the vorticity. Introducing the spatial discretization, we obtain:

$$\frac{\partial\hat{\omega}_z}{\partial t} = ik_x\hat{S}_y-ik_y\hat{S}_x+\frac{1}{Re_\tau}\left(\frac{\partial^2}{\partial z^2}-k_{xy}^2\right)\hat{\omega}_z. \tag{35}$$

We can now apply the IMEX temporal discretization scheme (Crank-Nicholson for the linear part and Adams-Bashforth for the non-linear part). It is worth noticing that density and viscosity variations – except for those in the definition of the linear diffusive terms – are handled explicitly, as they are collected in the non-linear term $\mathbf{S}$, equation (22). Since a wall-normal velocity-vorticity formulation is used to solve the Navier-Stokes equations (to avoid solving a pressure Poisson equation with non-constant coefficients when non-unitary density ratios are considered), this approach is the most suitable. An alternative would be iterative methods; however, these significantly increase computational cost. For interface-resolved simulations of bubbly flows (density ratio $\rho_r < 1$), we found that while explicitly handling density and viscosity variations impose a constraint on the maximum time step, this approach remains more efficient than the use of iterative solvers. Due to the explicit treatment of the above mentioned terms, for density ratios typical of air/water bubbly flows ($\rho_r \simeq 0.001$), the time step should be reduced by approximately an order of magnitude compared to the matched properties case.

Equation (35) is first discretized using the same IMEX scheme presented above; after some algebraical manipulations we obtain:

$$\left(\frac{\partial^2}{\partial z^2}-\beta^2\right)\hat{\omega}_z^{n+1} = -\frac{1}{\gamma}\left[ik_x\hat{H}_y^n-ik_y\hat{H}_x^n\right], \tag{36}$$

where $\beta^2 = (1+\gamma k_{xy}^2)/\gamma$. The series of 1D problems obtained can be solved using a Chebyshev-Tau algorithm with appropriate boundary conditions. For a no-slip boundary, one can set:

$$\hat{\omega}_z = 0, \tag{37}$$

while for a free-slip boundary:

$$\frac{\partial\hat{\omega}_z}{\partial z} = 0. \tag{38}$$

Once obtained the solution for the wall-normal velocity and vorticity at the time step $n+1$, the streamwise and spanwise components (u and v) of the velocity vector can be obtained by exploiting the definition of wall-normal vorticity (in Fourier space), which is:

$$-ik_x\hat{u}^{n+1}+ik_y\hat{v}^{n+1} = \frac{\partial\hat{w}^{n+1}}{\partial z}, \tag{39}$$

and the continuity equation (in Fourier space):

$$-ik_y\hat{u}^{n+1}+ik_x\hat{v}^{n+1} = \hat{\omega}_z^{n+1}. \tag{40}$$

After completing these steps, the complete velocity field at the new time iteration is obtained. With the velocity-vorticity formulation, there is no need to solve a computationally expensive Poisson equation for the pressure, as done instead in standard projection-correction methods [38]. If needed, the pressure field can be computed in post-processing from the velocity field.

3.3. Cahn-Hilliard equation for the phase-field

We start by rewriting the Cahn-Hilliard equation for the phase-field variable in a more compact form isolating the non-linear terms in S_ϕ:

$$\frac{\partial \phi}{\partial t} = S_\phi + \frac{s}{Pe_\phi}\nabla^2\phi - \frac{Ch^2}{Pe_\phi}\nabla^4\phi\,. \tag{41}$$

$$S_\phi = -\mathbf{u}\cdot\nabla\phi + \frac{1}{Pe_\phi}\left[\nabla^2\phi^3 - (1+s)\nabla^2\phi\right] \tag{42}$$

The parameter s in equation (42) is a numerical coefficient used to perform the splitting of the Laplace operator [40,41]. By applying the spatial discretization to the Cahn-Hilliard equation, we obtain:

$$\frac{\partial \hat{\phi}}{\partial t} = \hat{S}_\phi + s\frac{Ch^2}{Pe_\phi}\left(\frac{\partial^2}{\partial z^2} - k_{xy}^2\right)\hat{\phi} + \frac{Ch^2}{Pe_\phi}\left(\frac{\partial^2}{\partial z^2} - k_{xy}^2\right)\left(\frac{\partial^2}{\partial z^2} - k_{xy}^2\right)\hat{\phi}\,.$$

We can now apply the IMEX temporal discretization scheme using an implicit Euler method for the linear term and an Adams-Bashforth scheme for the non-linear part. After some algebraic manipulation, the following discrete equation is obtained:

$$\left[\frac{1}{\gamma_\phi} - s\left(\frac{\partial^2}{\partial z^2} - k_{xy}^2\right) + \left(\frac{\partial^2}{\partial z^2} - k_{xy}^2\right)^2\right]\hat{\phi}^{n+1} = \frac{\hat{H}_\phi^n}{\gamma_\phi} \tag{43}$$

where the historical term $\hat{H}_\phi^n$ has been introduced and $\gamma_\phi = (\Delta t Ch^2)/Pe_\phi$. We can split this 4^{th} order equation in two equivalent 2^{nd} order equations by choosing a proper value for the numerical coefficient s. Specifically, by solving the second order polynomial equation, and looking for two coincident solutions, we obtain:

$$s = \sqrt{\frac{4Pe_\phi Ch^2}{\Delta t}}\,. \tag{44}$$

This value guarantees the maximum numerical stability [40] and introducing the auxiliary variable $\theta_\phi = s\phi/2 + \nabla^2\phi$, two 2^{nd} order equations are obtained:

$$\left(\frac{\partial^2}{\partial z^2} - \delta^2\right)\hat{\theta}_\phi = \frac{\hat{H}_\phi^n}{\gamma_\phi}\,, \tag{45}$$

$$\left(\frac{\partial^2}{\partial z^2} - \delta^2\right)\hat{\phi}^{n+1} = \hat{\theta}_\phi\,, \tag{46}$$

where $\delta^2 = k_{xy}^2 - s/2$. These two equations can be solved using a Chebyshev-Tau algorithm with appropriate boundary conditions. Specifically, at the top and bottom boundaries, we set:

$$\frac{\partial \hat{\phi}}{\partial z}(\pm h) = 0\,; \qquad \frac{\partial^3 \hat{\phi}}{\partial z^3}(\pm h) = 0\,, \tag{47}$$

which gives the conservation of the phase-field variable:

$$\frac{\partial}{\partial t}\int_\Omega \phi d\Omega = 0\,, \tag{48}$$

where Ω is the domain considered. The use of these two boundary conditions also imposes a 90° contact angle at the wall.

3.4. Cahn-Hilliard-like equation for the surfactant concentration

We consider now the Cahn-Hilliard-like equation for the surfactant concentration, which can be rewritten in a more compact form:

$$\frac{\partial \psi}{\partial t} = S_\psi + \frac{Pi}{Pe_\psi}\nabla^2\psi\,, \tag{49}$$

where the term S_ψ is defined as follows:

$$S_\psi = -\mathbf{u}\cdot\nabla\psi + \frac{1}{Pe_\psi}\nabla\cdot\left[\psi(1-\psi)\nabla\left(-\frac{(1-\phi^2)^2}{2} + \frac{\phi^2}{2Ex}\right)\right]. \tag{50}$$

The transport equation for the surfactant is a 2^{nd} order equation. Applying the spectral representation, we obtain:

$$\frac{\partial \hat{\psi}}{\partial t} = \hat{S}_\psi + \frac{Pi}{Pe_\psi}\left(\frac{\partial^2}{\partial z^2} - k_{xy}^2\right)\hat{\psi}\,. \tag{51}$$

We can now apply the IMEX temporal integration scheme using an implicit Euler method for the linear term and an Adams-Bashforth scheme for the non-linear part. After some algebraic manipulation, we derive:

$$\left(\frac{\partial^2}{\partial z^2} - \imath^2\right)\hat{\psi}^{n+1} = -\frac{\hat{H}_\psi^n}{\gamma_\psi}\,, \tag{52}$$

where $\gamma_\psi = (\Delta t Pi)/Pe_\psi$, $\hat{H}_\psi^n$ is an historical term and $\imath^2 = (\gamma_\psi k_{xy}^2 + 1)/\gamma_\psi$. The equation above is solved using a Chebyshev-Tau algorithm with the following boundary conditions:

$$\frac{\partial \psi}{\partial z}(\pm h) = 0\,. \tag{53}$$

This leads to the conservation of the surfactant concentration on the entire computational domain Ω.

$$\frac{\partial}{\partial t}\int_\Omega \psi d\Omega = 0 \tag{54}$$

3.5. Transport equation for a passive scalar

We consider here the transport equation for the passive scalar. This equation can be also rewritten in a more compact form:

$$\frac{\partial \theta}{\partial t} = S_\theta + \frac{1}{Pe_\theta}\nabla^2\theta\,, \tag{55}$$

where S_θ is defined as follows:

$$S_\theta = -\mathbf{u}\cdot\nabla\theta\,. \tag{56}$$

Introducing the spectral discretization for the variable θ, we obtain:

$$\frac{\partial \hat{\theta}}{\partial t} = \hat{S}_\theta + \frac{1}{Pe_\theta}\left(\frac{\partial^2}{\partial z^2} - k_{xy}^2\right)\hat{\theta}\,. \tag{57}$$

We can now apply the IMEX time integration scheme using a Crank-Nicholson scheme for the linear term and an Adams-Bashforth scheme for the non-linear part. After some algebra, we obtain:

$$\left(\frac{\partial^2}{\partial z^2} - \tau^2\right)\hat{\theta}^{n+1} = -\frac{\hat{H}_\theta^n}{\gamma_\theta}\,, \tag{58}$$

where $\gamma_\theta = \Delta t/2Pe_\theta$ and $\tau^2 = (1 + \gamma_\theta k_{xy}^2)/\gamma_\theta$. This equation can be solved using a Chebyshev-Tau algorithm with general boundary condition at the two walls:

$$a\hat{\theta} + b\frac{\partial \hat{\theta}}{\partial z}(+h) = c \qquad d\hat{\theta} + e\frac{\partial \hat{\theta}}{\partial z}(-h) = f\,, \tag{59}$$

where a, b, c, d, e and f are constant value coefficients. By setting these coefficients, Dirichlet (type I), Neumann (type II) or mixed boundary conditions can be applied at either wall.

3.6. Lagrangian particle tracking

The equation for the position and velocity of the particle, respectively equations (17) and (18), are advanced in time using an explicit Euler scheme. The new position at the step $n+1$ is computed as:

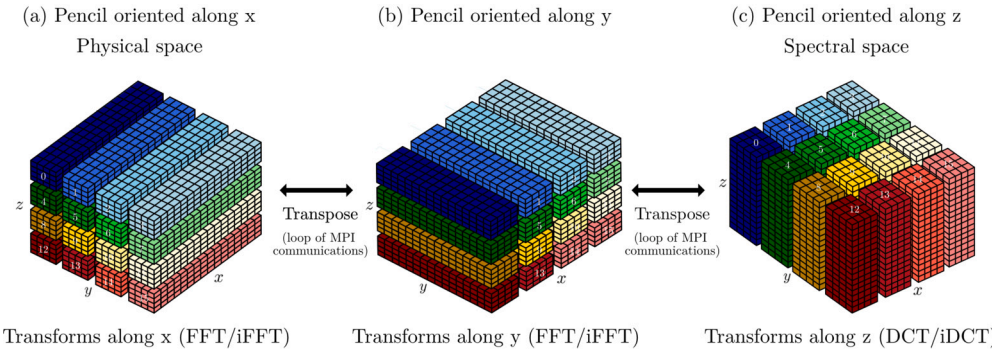


Fig. 2. Two-dimensional domain decomposition used to divide the workload among the different MPI tasks. Each color corresponds to a different MPI task, numbered from zero to 15. In physical space, the domain is divided along the y and z directions (pencils orientated along the x-direction), while in Fourier space it is divided along the x and y directions (pencil orientated along the z-direction). Transpositions (i.e. loops of MPI communications) are required to change the pencil orientation and to compute the transform along the different directions: along x in the configuration shown in panel a, along y in the configuration shown in panel b, and along z in the configuration shown in panel c. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{u}_p^n, \tag{60}$$

while the velocity at the step $n+1$ is obtained as follows:

$$\mathbf{u}_p^{n+1} = \mathbf{u}_p^n + \Delta t \mathbf{f}_p^n, \tag{61}$$

where $\mathbf{f}_p^n$ is the right-hand side of equation (18) (i.e. the sum of the forces applied on the particle). These equations are solved for all the particles. The time-step used to advance the particle equation is equal to the time-step used to advance the Eulerian fields in time; if necessary ($St \ll 1$) a smaller time-step can be used by defining the number of sub-iterations of the particle tracking to perform within one simulation time-step. Within these sub-iterations the Eulerian fields are not updated. To obtain the fluid velocity at the particle position a fourth-order Lagrangian polynomial is used.

Periodic boundary conditions are enforced at the periodic boundaries (homogeneous directions), whereas a perfectly elastic rebound is imposed at the walls.

4. Implementation of the parallelization strategy

The numerical scheme presented above is implemented in FLOW36. The code is written using Fortran 2003 and the main parallelization backbone relies on an MPI paradigm. On top of the MPI backbone, OpenACC directives and cuFFT libraries are used to accelerate the code execution on GPU-accelerated infrastructures.

4.1. First level of parallelization: MPI

The parallelization backbone of the code relies on an MPI approach; the overall workload is divided among the different MPI tasks using a 2D domain decomposition (pencil decomposition). Within this strategy, the whole domain is split in so-called pencils: the domain is divided along two out of three directions and each sub-domain is assigned to a different MPI process. In physical space, the domain is divided along the y and z directions (pencils oriented along the x-direction), while in Fourier space it is divided along the x and y directions (pencils oriented along the z-direction). This change in the pencil orientation is needed when taking the transforms: to compute the Fourier or Chebyshev transforms each process must hold all the points in the transform direction. Thus, when in physical space, at first Fourier transforms are taken in the x direction (Fig. 2a), then the parallelization changes in order to have all the points along the y direction. The domain is divided between the x and z directions when taking the Fourier transforms along y (Fig. 2b). At this point, each $x-y$ plane holds all the Fourier modes at a certain distance from the wall. Then the parallelization is again changed, switching to a domain decomposition along the x and y directions, thus each MPI process holds all the points in the z direction at a certain (k_x, k_y) wavenumber pair (parallelization in Fourier space). Finally, Chebyshev transforms are taken in the z direction (Fig. 2c).

The only MPI communications occur when the pencils are transposed. Non-linear terms are computed in physical space and transformed in Fourier space to avoid the computation of convolution integrals, thus a change of pencil orientation (and relative MPI communications) is required to compute the transforms along the three directions (from spectral to physical and vice-versa). After the calculation of all the non-linear terms, a system of Helmholtz problems along the wall-normal direction (z) is solved at each (k_x, k_y) wavenumber pair independently. Finally, it is worth mentioning that when the dual-grid approach is employed (i.e. when the solution of a governing equation is performed on a more refined grid with respect to the standard grid used for the Navier-Stokes equations), additional MPI communications are required to redistribute the Fourier modes upon spectral interpolation between the two grids: zero-padding of the highest wavenumbers is used when interpolating onto the fine grid and deletion of the highest wavenumbers when interpolating back onto the reference grid.

4.1.1. Lagrangian particle tracker parallelization

The parallelization of the Lagrangian particle tracker relies on a shared-memory implementation of MPI, available since MPI version 3.0, [42]. The code runs on $N+1$ shared-memory regions (i.e., nodes on a HPC system), with N shared-memory regions dedicated to the Eulerian solver (flow, phase field, surfactant, passive scalar) and one shared-memory region dedicated to the Lagrangian particle tracking, see Fig. 3. The MPI tasks within the N shared-memory regions assigned to the Eulerian solver are grouped into a MPI communicator (named `flow_comm`), whereas those in the remainder shared-memory region are grouped in the `part_comm` MPI communicator. The parallelization of the Eulerian solver in the `flow_comm` communicator is unchanged from what presented above, hence only the parallelization of the Lagrangian particle tracker will be reported in this section. An additional communicator `comm_comm`, comprised of `flow_comm` and one MPI task from `part_comm`, is used to transfer the Eulerian fields (fluid velocity, phase field, passive scalar,...) to the particle communicator and to retrieve the particle back-reaction.

Shared-memory windows are used to store Eulerian variables (e.g., flow velocity, phase field,...) and particle data (velocity, position and back-reaction) within the `part_comm` shared-memory communicator. All these variables are directly accessible from every MPI task belonging to `part_comm`. Access to shared variables is managed to avoid conflicts and race conditions: Eulerian variables are read-only, whereas the particle tracking workload is uniformly distributed among all MPI tasks in `part_comm` in a similar fashion to an OpenMP implementation. Race conditions on the variables containing the position and velocity of the

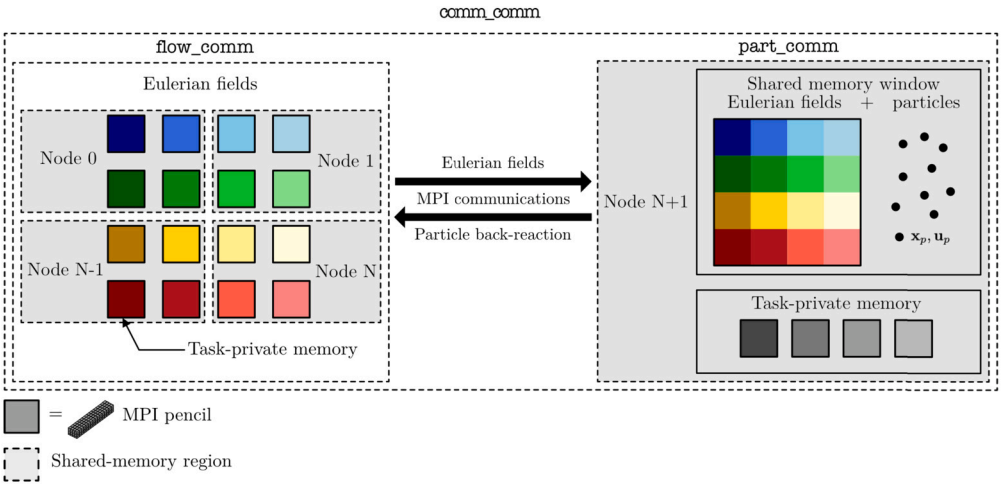


Fig. 3. Sketch showing the parallelization of the Lagrangian particle tracker. The `flow_comm` groups the N shared-memory regions working on the Eulerian solver and the `part_comm` groups the remainder shared-memory region that tracks the particles. The `comm_comm` is used to communicate data among the flow and particle communicators (flow field, phase field,... from the flow to the particle communicator and particles back-reaction from the particle to the flow communicator). Rectangles are used to show the memory management in the `flow_comm` and `part_comm`. In `flow_comm` memory is private to each MPI task (classic MPI implementation), whereas in `part_comm` Eulerian variables, particle data and particle back-reaction are stored in a shared-memory window (MPI shared-memory, accessible by all MPI tasks within `part_comm`) and all other variables are stored in the memory private to each MPI task. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

particles are prevented as each MPI task within the particle communicator operates on a unique set of particles.

A one-way-coupled Lagrangian particle tracker is currently implemented in FLOW36: the flow modifies the particle transport, whereas no back-reaction of the particles on the flow and no particle-particle collisions are considered. Extension to a two-way coupling, where the particle back-reaction on the fluid is accounted for, is straightforward: the subroutines to transfer the particle back-reaction to the flow communicator are already present and only the implementation of the physical models defining the particle back-reaction is missing. At the current stage, an efficient implementation of particle-particle interactions (four-way coupling) has not yet been considered; the current parallelization strategy has the potential for a simpler implementation, as data for all particles is readily available to all tasks in the `part_comm` communicator.

The presented parallelization strategy described here offers several advantages, among which easier particle management: the particle tracking workload is evenly distributed among all MPI tasks in `part_comm` and frequency of transfers of flow or back-reaction data is reduced as communications are grouped. This approach addresses the main weaknesses of commonly used strategies, co-located particles (i.e., each MPI task tracks particles that are within its own Eulerian subdomain) and uniformly-distributed particles (i.e., each MPI task holds a fixed number of particles and retrieves flow data from other tasks whenever needed). The co-located particles approach has the advantage of requiring no flow data communication from other MPI tasks, however comes at the cost of (frequent) transfers of particles across MPI tasks (whenever new particles enter the assigned sub-domain or particles leave it) and of an uneven workload of the particle tracking across the MPI tasks. The uniformly-distributed approach offers an even distribution of the particle tracking workload across the MPI tasks at the cost of frequent communications to retrieve flow data at the particle position. The current parallelization strategy offers an even workload among all MPI tasks in `part_comm` and requires no MPI communication to retrieve the flow data at the particle position (communications among `part_comm` and `flow_comm` are however needed to update flow and back-reaction data). The main disadvantages lie in the added complexity in the initial implementation (definition of the new MPI communicators, splitting in shared-memory regions and communication subroutines) and in the memory requirements. Since the Eulerian field obtained from the flow communicator are stored in a shared-memory window, there is no data replication among all the `part_comm` tasks, however the total memory available in the shared-memory region may become a limitation. A typical direct numerical simulation on $1024 \times 1024 \times 1025$ grid points requires about 8 GB of memory (double-precision variable) for each Eulerian variable in the shared-memory region (memory requirements for particles are negligible for particle numbers up to 10^8). Thus the size of the memory available in each shared-memory region (node) limits the maximum problem size that can be simulated when using the Lagrangian particle tracker. The implementation of the Lagrangian particle tracker pre-dates the porting of the code to GPU-accelerated architectures; as now no extension of the Lagrangian particle tracker to GPU architectures is planned. Indeed, for the number of particles typically considered, $\simeq 10^6$, the computational cost of the LPT algorithm is small (with respect to the solution of the Eulerian fields) and the use of GPUs does not provide a significative speed-up [43].

4.2. Second level of parallelization: OpenACC

On top of the MPI parallelization scheme presented above, cuFFT libraries and OpenACC directives are used to accelerate the code execution when GPUs are available. When GPU-based architectures are used, each MPI task is assigned to a specific GPU. All the computationally intensive operations are performed on the GPUs. In particular, all the transforms, Fourier and Chebyshev, are performed exploiting the highly optimized cuFFT libraries, which can be invoked using the interoperability features present in OpenACC. While the Fourier transforms are directly supported by the cuFFT libraries, algebraic manipulation are required to perform the Chebyshev transforms, as real-to-real transforms are not directly supported by the cuFFT libraries [44]. OpenACC directives (mainly kernels) are used to offload the computation of the non-linear terms, which are evaluated in physical space, as well as the solver execution. To obtain a code that can be easily compiled for CPU- and GPU-based architectures, we employ the managed memory model present in OpenACC (which exploits the CUDA unified memory feature). As in most systems CPU and GPU memories are physically separated, the use of GPUs usually requires explicit memory transfers and the handling of the copies shared between CPU and GPU memories. Thanks to the managed memory feature, memory transfers between the CPU (host) and the GPU (device) do not have to be explicitly defined and memory

Table 1
Technical specifications of the computing infrastructures employed for the strong scaling tests: LUMI-C, Vesta and Marconi-KNL for the CPU runs and Marconi-100 for the GPU runs.

System	CPU (per node)	GPU (per node)
LUMI-C (LUMI)	2 x AMD EPYC 7763 64c	-
Vesta (ALCF)	1 x IBM PowerPC A2 16c	-
Marconi-KNL (CINECA)	1 x Intel Xeon Phi 7250 68c	-
Marconi-100 (CINECA)	2 x IBM POWER9 AC922 16c	4 x NVIDIA V100

can be accessed using a single pointer from both CPU or GPU code sections.

4.3. Strong scaling tests

To evaluate the code performances and the efficiency of the implementation reported above, we analyze the strong scaling behavior of the code. The tests have been performed for both the CPU version (MPI) and GPU version of the code (MPI + OpenACC). For the CPU version of the code, we report the results obtained on LUMI-C (LUMI), Vesta (ALCF) and Marconi-KNL (CINECA) while for the GPU version, we report the results obtained on Marconi-100 (CINECA). Each of these machines is characterized by a computing architecture from a different vendor (AMD, IBM, Intel and Nvidia, respectively). Please refer to Table 1 for the technical specifications of the machines. The following compilers have been used: Cray `ftn` for LUMI-C; IBM `xlf` for Vesta; Intel `ifort` for Marconi-KNL and Nvidia `nvfortran` for Marconi-100. The tests consider the whole application, including I/O operations (performed using the MPI I/O library). The scaling results are almost independent of the number of modules activated (single-phase flow only, multiphase, multiphase with heat transfer). Indeed, the phase-field method is that is interface blind (i.e. it does not use interface reconstruction algorithms) and thus its computational cost does not depend on the interface extension [8,9]. In addition, the numerical scheme employed to solve the different governing equations (and thus the routines/modules used) are very similar in terms of MPI communication patterns and computational cost. The Lagrangian particle tracker, however, has not been included.

The strong scaling results are shown in Fig. 4. Panel *a* refers to the CPU runs while panel *b* refers to the GPU runs. Two problem sizes have been tested: $512\times512\times513$ (empty circles) and $1024\times1024\times1025$ (full circles). For the CPU runs, multithreading (when available) is used binding 1 logical thread to an MPI task while for GPU runs 1 GPU is bound to an MPI task. For the GPU runs, a different number of nodes have has used for the two problem sizes because of the different memory requirements. Specifically, the number of nodes tested allows loading the entire problem on the GPU VRAM making the performance comparison among the different cases more consistent. Analyzing the strong scaling performance exhibited by the code, we can observe that in general, good results are obtained for both CPU and GPU runs. For the CPU runs (panel *a*), excellent results are obtained and the code exhibits an excellent scaling up to 16384 and 65536 MPI tasks for the smaller and larger problem sizes, respectively. In addition, performance are retained among different machines, CPU architectures and network configurations. For GPU runs (panel *b*), performances tend to worsen as the number of MPI tasks (or nodes) is increased, especially when the smaller problem size is considered (green circles). This can be addressed to the high cost associated with the loop of MPI communications required every time a complete forward or backward transform has to be computed. Indeed, the transposition (or re-orientation) of the pencils is a classical example of all-to-all operation. The optimization of this type of operations is an open issue in multi-GPU architectures [45–47] because of i) the large amount of data that has to be transferred; ii) the massive speed-up offered by the computation of transforms on GPUs (via highly-optimized libraries, e.g. cuFFT or rocFFT).

5. Validation and benchmark

In the following, we report some examples to show the validation cases of the code and its capabilities. First, we consider a single-phase turbulent channel flow case. Second, we move to the validation of the multiphase module considering the deformation of a single drop in shear flow and the fragmentation of a swarm of surfactant-laden drops in wall-bounded turbulence. Third, we test the heat transfer module by considering the flow between a hot and a cold wall. Finally, the Lagrangian particle tracking approach is tested.

5.1. Single-phase flow

We begin by considering the case of a turbulent channel flow [11]. This flow instance has been extensively studied in literature with different numerical methods [11,48,49]. For validation purposes, we consider a shear Reynolds number equal to $Re_\tau = 180$, tough higher Reynolds number flows can also be simulated with FLOW36 [50,28]. The computational domain is a closed channel with dimensions $L_x \times L_y \times L_z = 4\pi h \times 2\pi h \times 2h$ corresponding to $L_x^+ \times L_y^+ \times L_z^+ = 2620 \times 1310 \times 360$ wall units. Equations are discretized with $N_x \times N_y \times N_z = 256 \times 256 \times 257$ collocation points. Periodic boundary conditions are imposed along the streamwise and spanwise directions while no-slip conditions are imposed at the two walls. For this case, only the continuity and Navier-Stokes equations are solved.

To validate the implementation of the numerical method, we compute the wall-normal behavior of the mean velocity profile (streamwise) and of the root mean square (RMS) of the three velocity components. Results are shown in Fig. 5: panel *a* shows the wall-normal behavior of the mean streamwise velocity profile while panel *b* the RMS of the three components of the velocity vector. A sketch showing the instantaneous streamwise velocity in the computational box (blue-low; red-high) is shown in the inset of panel *a*.

Analyzing the mean velocity profile (panel *a*), we can observe that the present results are in excellent agreement with the law of the wall, reported using a dotted line (linear part, $u^+ = z^+$) and dash-dotted line (logarithmic part, $u^+ = (1/\kappa)\log z^+ + 5.2$ where κ is the von-Karman constant [53]). As a reference, the results from previous works are reported: simulation results from Kim et al. (1987) [11] and Costa (2018) [51], and experimental data from Giurgiu et al. (2023) [52]. Moving to the RMS of the three velocity components (panel *b*), we observe a very good agreement between present results and the experimental and numerical results available in the literature for all the three velocity components.

5.2. Drop deformation in shear flow

To validate the implementation of the phase-field method, we consider the deformation of a drop in a linear shear flow; the simulation database reported in Soligo et al. (2020) [54] is used in this section. A single drop, with diameter $d = 0.8h$, is initialized at the channel center and the two walls move in opposite directions with constant speed, $u = \pm 1$. A sketch of the configuration employed is shown in Fig. 6. The domain considered has dimensions $L_x \times L_y \times L_z = 2\pi h \times \pi h \times 2h$ and is discretized with $N_x \times N_y \times N_z = 512 \times 256 \times 513$ collocation points. The Reynolds number has been set equal to $Re = 0.1$ in order to ensure creeping flow conditions and thus negligible inertial effects. The Reynolds number is computed using as a reference the wall velocity and the continuous phase properties (density and viscosity). We consider three different values of the surface tension (Weber number), which correspond to three different values of the capillary number: $Ca = 0.0625$, $Ca = 0.125$ and $Ca = 0.1875$. In creeping flow conditions, the capillary number – ratio between viscous forces and surface tension forces – is the most suitable parameter and it can be computed from the Weber and Reynolds number (simulation input for FLOW36):

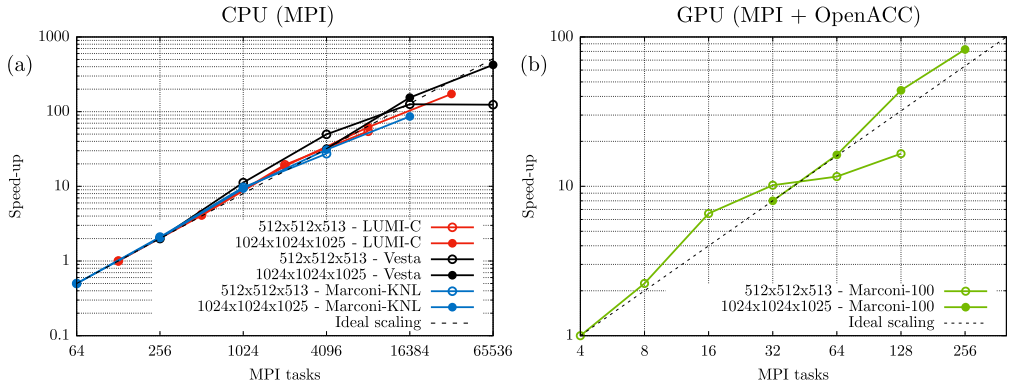


Fig. 4. Strong scaling results obtained on CPU-based architectures (panel *a*) and GPU-accelerated computing infrastructures (panel *b*). The following Tier-0 HPC clusters have been used for the CPU runs: LUMI-C (red), Vesta (black) and Marconi-KNL (blue) while for the GPU runs: Marconi-100 (green). For the CPU runs, each logical thread is bound to an MPI task while for GPU runs each GPU is bound to an MPI task. Two different problem sizes have been considered: $512 \times 512 \times 513$ (empty circles) and $1024 \times 1024 \times 1025$ (full circles). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

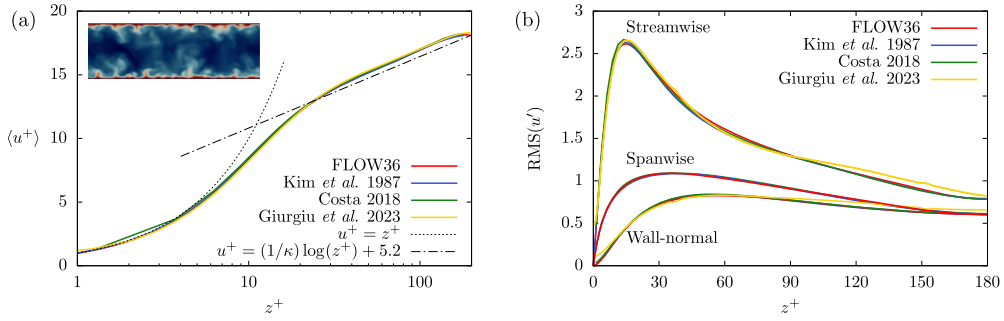


Fig. 5. Wall-normal behavior of mean velocity profile (panel *a*) and root-mean square (RMS) of the velocity vector (panel *b*). The results are compared against the numerical results of Kim et al. (1987) [11], Costa (2018) [51] and against the experimental results of Giurgiu et al. (2023) [52]. The classical law of the wall, $u^+ = z^+$ and $u^+ = (1/\kappa)\log(z^+) + 5.2$, is also reported, where κ is the von-Karman constant [53]. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

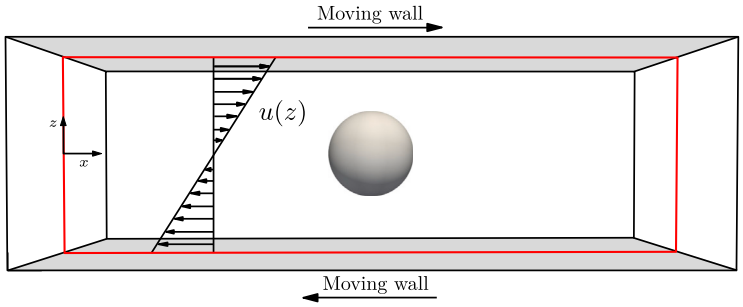


Fig. 6. Sketch of the computational domain used to analyze the deformation of a single droplet in shear flow. A spherical droplet with diameter $d = 0.8h$ is initialized in the center of the channel. The two walls move in opposite directions with constant speed, $u = \pm 1$. The domain has dimensions $L_x \times L_y \times L_z = 2\pi h \times \pi h \times 2h$.

$$Ca = \frac{We}{Re}\frac{d}{2h}. \tag{62}$$

Regarding the phase-field parameters, the accurate description of the steep gradients at the interface requires a minimum of 3 grid points across the interface for clean-interface simulations (surfactant-free). To meet this requirement, the Cahn number has been set equal to $Ch = 0.025$. The Péclet number for the phase field, Pe_ϕ, is determined based on the scaling $Pe_\phi = 3/Ch = 120$. The initial condition for the simulation is a linear velocity profile along the wall-normal direction (see Fig. 6) for the streamwise component of the velocity (u) while the other two velocity components (v and w) are set equal to zero. A spherical droplet is initialized at the center of the computational domain; the equilibrium profile of the phase field is initialized across the interface.

$$\phi(x, y, z) = \tanh\left(\frac{s - d/2}{\sqrt{2}Ch}\right) \tag{63}$$

The variable s is the distance from the center of the droplet and d the droplet diameter. No-slip boundary conditions for the flow field and no-flux boundary conditions for the phase-field are imposed at the two walls.

Once defined the initial and boundary conditions, the simulation starts and the governing equations are advanced in time. The shear flow starts to deform and elongate the droplet along the shear direction. At the same time, surface tension forces act to restore the spherical shape of the droplet (minimal energy configuration). In the limit of low capillary number, i.e. when surface tension forces are strong enough to avoid

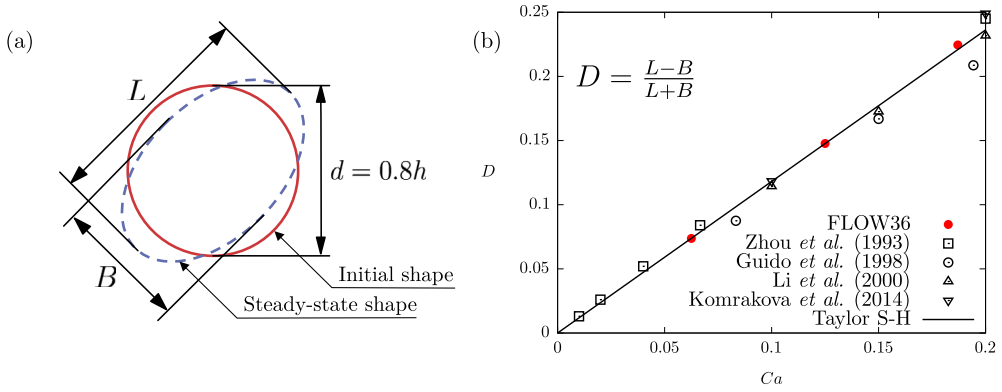


Fig. 7. Panel *a* shows a sketch of the initial shape (red) and steady-state deformed shape (dashed blue) for a drop in shear flow at low capillary numbers. The minor and major axis, L and B, are also highlighted. Panel *b* shows the values of the deformation parameter obtained from FLOW36 (red) compared against previous numerical and experimental works: black squares refer to the simulations of Zhou et al. (1993) [55], circles to the experiments of Guido et al. (1998) [56], upward triangles to the simulations of Li et al. (2000) [57] and downward triangles to the simulations of Komrakova et al. (2014) [58]. The theoretical relation of Shapira & Haber (1990) [59] is also shown. FLOW36 results are reproduced from Soligo et al. (2020) [54].

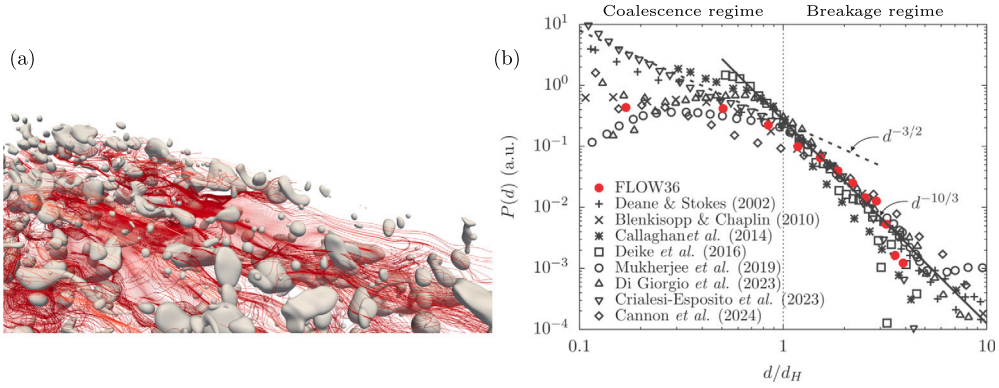


Fig. 8. Panel *a* shows a rendering of a swarm of large and deformable drops in a turbulent channel flow. The interface is identified as the iso-contour $\phi = 0$ (white); streamlines are used to visualize the chaotic turbulent motion of the flow. Panel *b* shows the resulting drop size distribution (red circles, reproduced from Soligo et al. (2019) [27], case FG). Archival data obtained from previous experiments [60–62] and simulations [63–67] is also reported using black symbols. The drop diameter is normalized using the Kolmogorov-Hinze scale for each case (estimated when not enough information is provided) while the distributions are reported in arbitrary units due to the different normalizations used.

the breakage of the droplet, a new steady-state shape for the droplet is obtained, as shown in the blue dashed line of Fig. 7*a*. The deformation of this steady-state configuration can be characterized by the deformation parameter, $D = (L - B)/(L + B)$, being L and B are the major and minor axis of deformation of the droplet (evaluated on a $x - z$ plane that passes through the center of the droplet).

The results obtained for the deformation parameter for three values of the capillary number considered are shown in Fig. 7*b*. Present results are shown with red symbols (reproduced from Soligo et al. (2020) [54]) while archival literature results are shown with black symbols: squares refer to the simulations of Zhou et al. (1993) [55], circles refer to the experiments of Guido et al. (1998) [56], upward triangles refer to the simulations of Li et al. (2000) [57] and downward triangles to the simulations of Komrakova et al. (2014) [58]. Finally, the analytical relation of Shapira & Haber (1990) [59] is shown with a black solid line. We can observe that an excellent agreement is obtained between the present results and the analytical relation (black line) as well as with previous numerical and experimental results. This benchmark can be also extended to surfactant-laden drops, see Soligo et al. (2020) [54] for further details.

5.3. Coalescence and breakage of drops in turbulence

To further validate and demonstrate the capabilities of FLOW36, we consider the injection of a swarm of surfactant-laden drops released in a turbulent channel flow. This configuration is the one adopted in Soligo et al. (2019) [27] (labeled FG in the original paper). The computational domain is a closed channel with dimensions $L_x \times L_y \times L_z = 4\pi h \times 2\pi h \times 2h$ corresponding to $L_x^+ \times L_y^+ \times L_z^+ = 3770 \times 1885 \times 600$ wall units. Equations are discretized on a grid with $N_x \times N_y \times N_z = 2048 \times 1024 \times 1025$ collocation points. The simulation is performed at a fixed shear Reynolds number of $Re_\tau = 300$ and the Weber number has been set equal to $We = 3$; the surfactant elasticity number has been set equal to $\beta_s = 4.00$. The simulation starts by releasing 256 spherical drops in a turbulent flow field (obtained from a preliminary DNS of a single-phase turbulent channel flow). After an initial transient, where drops start to break and coalesce following complex dynamics, a new equilibrium situation is reached in which a balance between coalescence and breakage events is attained.

Fig. 8*a* shows a qualitative rendering of the steady-state configuration at $t^+ = 3000$. The flow moves from left to right (along the streamwise direction x) and a 3D rendering of the systems is shown. The interface of the drops (iso-contour $\phi = 0$) is shown in white while streamlines

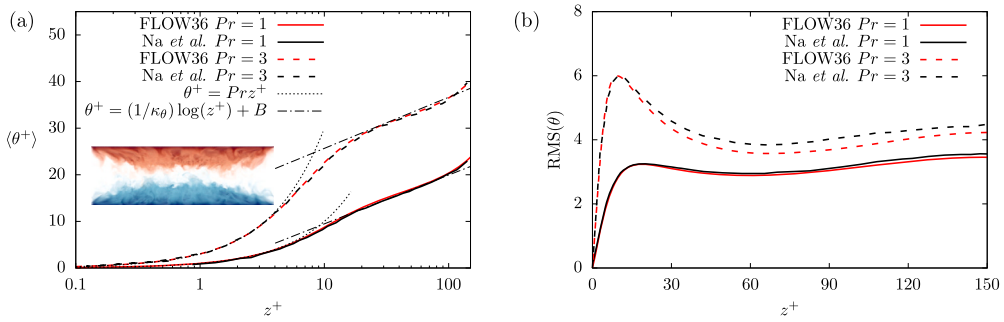


Fig. 9. Wall-normal behavior of the mean temperature profile (panel *a*) and root mean square (RMS) of the temperature fluctuations (panel *b*). A sketch showing the configuration considered and the temperature structures inside the channel is shown in the inset for $Pr = 3$. Continuous lines refer to $Pr = 1$ while dashed lines refer to $Pr = 3$. The simulation results and the analytical profiles reported in the work of Na et al. (1999) [72] are also shown. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

of the turbulent fluctuations are used to visualize the turbulent flow field. We can observe the wide range of scales and shapes that characterize the drops: from very small and almost undeformed spherical droplets to very large drops characterized by a complex three-dimensional shape. To quantify the number of small and large drops, we compute the drop size distribution. Results are shown in Fig. 8*b* with red circles. The drop diameter is normalized using the Kolmogorov-Hinze scale, which identifies the critical diameter below which a drop/bubble will not undergo breakage according to the KH framework [68,69], while the distributions are reported in arbitrary units for the sake of comparison with archival literature data. The analytical scaling laws [70], $d^{-3/2}$ and $d^{-10/3}$, for the coalescence-dominated regime (drop smaller than the KH scale) and breakage-dominated regime (drop larger than the KH scale) are also reported as reference with dashed and continuous lines, respectively. Archival literature data on drop/bubble size distribution obtained from experiments and simulations of drop/bubble fragmentation in turbulent flows are also reported. Specifically, the datasets are obtained from: i) experiments of Deane & Stokes (2002) [60], Blenkinsopp & Chaplin [61] and Callaghan et al. 2014 [62]; ii) simulations of Deike et al. (2016) [63], Mukherjee et al. (2019) [64], Di Giorgio et al. (2023) [65], Crialesi-Esposito et al. (2023) [66] and Cannon et al. (2024) [67]. In general, we can observe that in the breakage-dominated regime (drops larger than the Kolmogorov-Hinze scale), FLOW36 results are in good agreement with previous experimental and numerical results. This confirms the capabilities of interface capturing methods (e.g. phase-field method) in describing the breakage of drops and bubbles [8,9,71]. Moving to the coalescence-dominated regime (drops smaller than the Kolmogorov-Hinze scale), we can observe that experimental and numerical data are more scattered and is difficult to infer on the correct behavior. This discrepancy can be traced back to the difficulty in describing coalescence events, which are controlled by the small-scale physics of the interface. Please refer to Soligo et al. (2021) [9] for further discussion on this point, which is beyond the scope of the present paper. FLOW36 results reported in Fig. 8 refer to a matched-density and viscosity case, as also considered in most of simulation results reported. Results for non-unitary density and viscosity ratios can be found in Roccon et al. (2017) [21] and Mangani et al. (2022) [22].

5.4. Heat transfer in turbulent channel flow

We consider here the validation of the heat transfer module by studying the dynamic of a passive scalar in a turbulent channel flow. The computational domain is a closed channel with dimensions $L_x \times L_y \times L_z = 4\pi h \times 2\pi h \times 2h$ corresponding to $L_x^+ \times L_y^+ \times L_z^+ = 1885 \times 942 \times 300$ wall units. The governing equations are discretized on a grid with $N_x \times N_y \times N_z = 256 \times 256 \times 257$ collocation points. The shear Reynolds number is set equal to $Re_\tau = 150$ to match the cases reported by Na et al. 1999 [72]. Heat transfer is driven by the temperature difference between the two walls: no-slip boundary conditions are imposed for the flow field while the temperature of the two walls is imposed. Heat is considered as a passive scalar and thus the temperature field does not influence the density and/or viscosity of the fluid [73].

Two different values of the Prandtl number are considered: $Pr = 1$ and $Pr = 3$. Simulations start from a flow field obtained from a previous direct numerical simulation of a turbulent channel flow and with a zero temperature field. After a transient, the temperature reaches a new steady-state configuration where heat is supplied from the top of the domain (hot wall) and is discharged at the bottom (cold wall). To validate the implementation of the heat transfer module, we compute the mean temperature profiles and the root mean square (RMS) of the temperature field. Results are shown in Fig. 9 for the two values of the Prandtl number considered: $Pr = 1$ (continuous) and $Pr = 3$ (dashed). The simulation results obtained by Na et al. (1999) [72] are also reported as well as the analytical profiles. The analytical profiles are obtained using the following parameters: $k_\theta = 0.22$ and $B = -1.0$ for $Pr = 1$ and $k_\theta = 0.21$ and $B = 14.7$ for $Pr = 3$ [72,74]. For both statistics, we observe a very good agreement with previously published data and with the theoretical profiles. Overall, this confirms the correct implementation of the heat transfer module in FLOW36 and the accuracy of the pseudo-spectral discretization.

5.5. Particle-laden turbulent channel flow

Finally, we present the validation of the Lagrangian particle tracking algorithm. For this purpose, we consider the benchmark reported by Marchioli et al. (2008) [75], whose parameters are briefly recalled here. We consider a turbulent channel flow at a shear Reynolds number equal to $Re_\tau = 150$. The computational domain is a closed channel with dimensions $L_x \times L_y \times L_z = 4\pi h \times 2\pi h \times 2h$ corresponding to $L_x^+ \times L_y^+ \times L_z^+ = 1885 \times 942 \times 300$ wall units. The governing equations are discretized on a grid with $N_x \times N_y \times N_z = 128 \times 128 \times 129$ collocation points. The turbulent flow is laden with 3 sets of particles; each set is composed by 10^5 particles and it is characterized by a different Stokes number, ratio between the particle relaxation time and the flow time scale. In particular, we consider the following Stokes numbers: $St = 1$, $St = 5$ and $St = 25$. The particle-to-fluid density ratio is set equal to $\rho_p/\rho_1 = 769$ for all cases. For this test case, the only force acting on the particle is the Stokes drag, adjusted using the Schiller-Naumann correction [33] to account for the particle Reynolds number effect. The simulation starts from a uniform random distribution of particles and after a transient (about $\Delta t^+ = 20000$), the particles reach a new equilibrium distribution.

Fig. 10*a* shows a rendering of the particles (blue) in the turbulent channel flow (rendering of the instantaneous streamwise velocity). From the qualitative picture, we can notice how particles tend to accumulate in the near-wall region of the channel. To characterize the accumulation, we compute the concentration of particles along the wall-normal direction. To be consistent with Marchioli et al. (2008) [75], Lagrangian

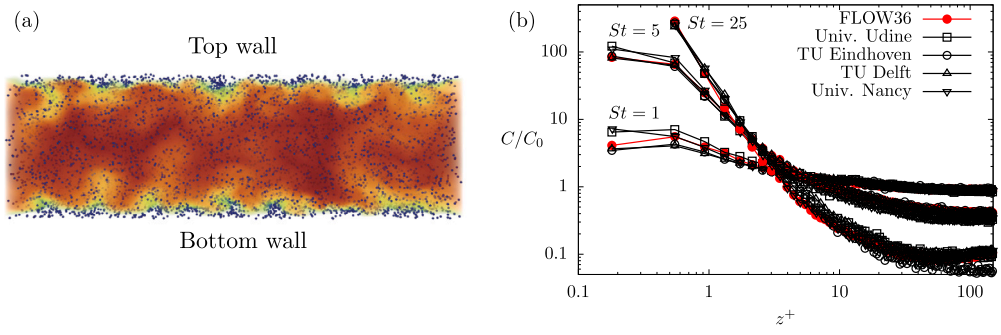


Fig. 10. Panel *a* shows a volume rendering of 10^5 Lagrangian particles dispersed in a turbulent channel flow for $St = 1$. Three sets of 10^5 particles are injected; particles are characterized by three values of the Stokes number: $St = 1$, $St = 5$ and $St = 25$ and a density ratio equal to $\rho_p/\rho_1 = 769$. The accumulation of the particles near the two walls can be qualitatively appreciated. Panel *b* shows the resulting concentration profiles along the wall-normal direction for the three Stokes numbers considered. Present results (red) are in good agreement with those included in the benchmark reported in Marchioli et al. (2008) [75].

statistics are computed by averaging over $N_s = 193$ wall-parallel fluid slabs distributed non-uniformly along the wall-normal direction. The thickness of the s-th slab, Δz^+, was obtained by means of hyperbolic-tangent binning with a stretching factor equal to $\gamma = 1.7$:

$$\Delta z^+(s) = \frac{Re_\tau}{\tanh(\gamma)} \left[\tanh\left(\frac{\gamma s}{N_s}\right) - \tanh\left(\gamma \frac{s-1}{N_s}\right) \right]. \tag{64}$$

A particle is assigned to a slab if its center is located inside the slab. A time window equal to $\Delta t^+ = 2000$ is used to compute the statistics. In Fig. 10*b*, the resulting concentration profile is shown normalized by the initial concentration profile C_0 (uniform concentration) using red circles. The concentration profiles reported in Fig. 6*a* of Marchioli et al. (2008) [75] are also included as reference. These profiles refer to the different research groups (and codes) that contributed to the benchmark: University of Udine (labeled UUD in the original paper), Technische Universiteit Eindhoven (labeled TUE in the original paper), Technische Universiteit Delft (labeled TUD in the original paper) and University of Nancy (HPU in the original paper). We can observe that the present results (red circles) well align with the reference concentration profiles. Some minor differences can be only observed in the near-wall region; these differences can be traced back to the different schemes used for the flow field interpolation, particle tracking, and flow solver.

6. Conclusions

We detail the characteristics and features of FLOW36, a pseudo-spectral code suitable for large-scale simulations of multiphase flows on heterogeneous computing architectures. The code employs direct numerical simulation of the Navier-Stokes equations coupled with a phase-field method to describe interface shape, topological changes and the presence of surfactants. Additionally, it features transport equations to model heat transfer problems and Lagrangian particle tracking in multiphase turbulence.

The governing equations are solved using a pseudo-spectral method. In particular, the Eulerian variables are transformed into wavenumber space via Fourier representations in the periodic homogeneous directions, x and y, and Chebyshev polynomials in the wall-normal (non-homogeneous) direction, z. Non-linear terms are evaluated in the physical space thus avoiding costly convolution operations in the Fourier space. The system of governing equations is advanced in time using an implicit-explicit strategy.

The code has been developed with the specific goals of simplifying code maintenance, ensuring portability of the code performances, and unifying both CPU- and GPU-ready versions. To achieve these ambitious goals, the code relies on two levels of parallelism: i) a 2D domain decomposition via MPI is used to divide the workload among the MPI tasks (CPU-only version); ii) a hybrid MPI+X parallelization, which builds on the existing MPI 2D domain decomposition and relies on OpenACC directives and CUDA libraries (cuFFT) to accelerate code execution on GPU-based computing infrastructures (CPU+GPU version). The code exploits the managed memory feature (based on the CUDA Unified memory concept) to facilitate maintenance and to avoid the explicit definition of data transfers between CPU and GPU memories (when GPU-accelerated computing infrastructures are used). This feature heavily simplifies the development of new modules/features, their debugging and validation.

Future developments will focus on the optimization of FLOW36 for GPU-accelerated architectures. In particular, the focus will be on improving strong scaling results in GPU-accelerated clusters, employing for instance the cuDecomp library [76] as well as extending the support to GPUs from different vendors exploiting the unified memory feature recently introduced for AMD GPU architectures [77,78] together with the rocFFT library [79].

CRediT authorship contribution statement

Alessio Roccon: Writing – original draft, Validation, Supervision, Software, Methodology, Conceptualization. **Giovanni Soligo:** Visualization, Validation, Software, Methodology, Conceptualization. **Alfredo Soldati:** Supervision, Resources, Project administration, Funding acquisition, 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

We acknowledge the CINECA award, under the ISCRA initiative, for the availability of high-performance computing resources and support (projects HP10CH4PTZ and HP10BUJEO5). We acknowledge the EuroHPC Joint Undertaking for awarding access to the supercomputer Marconi-100 (project PRACE-DEV-2022D01-069) and LUMI-C (project EHPC-EXT-2022E01-003). We also acknowledge the Argonne Leadership Computing Facility (ALCF) for awarding access to Vesta via Director's Discretionary Allocation Program. AR and AS gratefully acknowledge financial support from the European Union-NextGenerationEU PNRR M4.C2.1.1 - PRIN 2022, "The fluid dynamics of interfaces: mesoscale models for bubbles, droplets, and membranes and their coupling to large scale flows" 2022R9B2MW - G53C24000810001. The authors also acknowledge the TU Wien University Library for financial support through its Open Access Funding Program.

Data availability

The data presented in this study are openly available in the following Figshare repository: https://doi.org/10.6084/m9.figshare.26232683.

References

[1] B. Jähne, H. Haußecker, Air-water gas exchange, Annu. Rev. Fluid Mech. 30 (1) (1998) 443–468.
[2] R. Pereira, I. Ashton, B. Sabbaghzadeh, J. Shutler, R. Upstill-Goddard, Reduced air-sea CO_2 exchange in the Atlantic Ocean due to biological surfactants, Nat. Geosci. 11 (2018) 492–496.
[3] E. Paul, V. Atiemo-Obeng, S. Kresta, Handbook of Industrial Mixing: Science and Practice, John Wiley & Sons, 2004.
[4] L.L. Schramm, E.N. Stasiuk, D.G. Marangoni, Surfactants and their applications, Annu. Rep. Prog. Chem., Sect. C, Phys. Chem. 99 (2003) 3–48.
[5] R.A. Shaw, Particle-turbulence interactions in atmospheric clouds, Annu. Rev. Fluid Mech. 35 (1) (2003) 183–227.
[6] S. Elghobashi, Direct numerical simulation of turbulent flows laden with droplets or bubbles, Annu. Rev. Fluid Mech. 51 (1) (2019) 217–244.
[7] A. Prosperetti, G. Tryggvason, Computational Methods for Multiphase Flow, Cambridge University Press, 2009.
[8] A. Roccon, F. Zonta, A. Soldati, Phase-field modeling of complex interface dynamics in drop-laden turbulence, Phys. Rev. Fluids 8 (2023) 090501.
[9] G. Soligo, A. Roccon, A. Soldati, Turbulent flows with drops and bubbles: what numerical simulations can tell us – Freeman scholar lecture, J. Fluids Eng. 143 (2021) 080801.
[10] M. Gorokhovski, M. Herrmann, Modeling primary atomization, Annu. Rev. Fluid Mech. 40 (2008) 343–366.
[11] J. Kim, P. Moin, R. Moser, Turbulence statistics in fully developed channel flow at low Reynolds number, J. Fluid Mech. 177 (1) (1987) 133–166.
[12] P. Orlandi, Fluid Flow Phenomena: a Numerical Toolkit, vol. 55, Springer Science & Business Media, 2000.
[13] R.S. Rogallo, P. Moin, Numerical simulation of turbulent flows, Annu. Rev. Fluid Mech. 16 (1) (1984) 99–137.
[14] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, in: Special Issue on "Program Generation, Optimization, and Platform Adaptation", Proc. IEEE 93 (2) (2005) 216–231.
[15] Nvidia, cuFFT library, http://docs.nvidia.com/cuda/cufft.
[16] OpenACC working group, The OpenACC application programming interface, Retriev. March 26 (2011) 2019.
[17] S. Popinet, Numerical models of surface tension, Annu. Rev. Fluid Mech. 50 (2018) 1–28.
[18] S. Mirjalili, M.A. Khanwale, A. Mani, Assessment of an energy-based surface tension model for simulation of two-phase flows using second-order phase field methods, J. Comput. Phys. 474 (2023) 111795.
[19] H. Ding, P. Spelt, C. Shu, Diffuse interface model for incompressible two-phase flows with large density ratios, J. Comput. Phys. 226 (2) (2007) 2078–2095.
[20] J. Kim, Phase-field models for multi-component fluid flows, Commun. Comput. Phys. 12 (3) (2012) 613–661.
[21] A. Roccon, M. De Paoli, F. Zonta, A. Soldati, Viscosity-modulated breakup and coalescence of large drops in bounded turbulence, Phys. Rev. Fluids 2 (2017) 083603.
[22] F. Mangani, G. Soligo, A. Roccon, A. Soldati, Influence of density and viscosity on deformation, breakage, and coalescence of bubbles in turbulence, Phys. Rev. Fluids 7 (2022) 053601.
[23] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, G. Zanetti, Modelling merging and fragmentation in multiphase flows with SURFER, J. Comput. Phys. 113 (1) (1994) 134–147.
[24] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, S. Zaleski, Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows, J. Comput. Phys. 152 (2) (1999) 423–456.
[25] D. Korteweg, Sur la forme que prennent les equations du mouvements des fluides si l'on tient compte des forces capillaires causées par des variations de densité considérables mais continues et sur la théorie de la capillarité dans l'hypothèse d'une variation continue de la densité, Arch. Neerl. Sci. Exactes Nat. 6 (1901) 1–24.
[26] G. Soligo, A. Roccon, A. Soldati, Coalescence of surfactant-laden drops by phase field method, J. Comput. Phys. 376 (2019) 1292–1311.
[27] G. Soligo, A. Roccon, A. Soldati, Breakage, coalescence and size distribution of surfactant-laden droplets in turbulent flow, J. Fluid Mech. 881 (2019) 244–282.
[28] A. Roccon, F. Zonta, A. Soldati, Turbulent drag reduction in water-lubricated channel flow of highly viscous oil, Phys. Rev. Fluids 9 (2024) 054611.
[29] M. Laradji, H. Guo, M. Grant, M. Zuckermann, The effect of surfactants on the dynamics of phase separation, J. Phys. Condens. Matter 4 (1992) 6715–6728.
[30] S. Engblom, M. Do-Quang, G. Amberg, A. Tornberg, On diffuse interface modeling and simulation of surfactants in two-phase fluid flow, Commun. Comput. Phys. 14 (4) (2013) 879–915.
[31] R. van der Sman, M. Meinders, Analysis of improved lattice Boltzmann phase field method for soluble surfactants, Comput. Phys. Commun. 1999 (2016) 12–21.
[32] S. Komura, H. Kodama, Two-order-parameter model for an oil-water-surfactant system, Phys. Rev. E 55 (2) (1997) 1722–1727.
[33] L. Schiller, A drag coefficient correlation, Z. Ver. Dtsch. Ing. 77 (1933) 318–320.
[34] A. Hajisharifi, C. Marchioli, A. Soldati, Particle capture by drops in turbulent flow, Phys. Rev. Fluids 6 (2021) 024303.
[35] A. Hajisharifi, C. Marchioli, A. Soldati, Interface topology and evolution of particle patterns on deformable drops in turbulence, J. Fluid Mech. 933 (2022) A41.
[36] M. Schenk, G. Giamagas, A. Roccon, A. Soldati, F. Zonta, Computationally efficient and interface accurate dual-grid phase-field simulation of turbulent drop-laden flows, J. Fluids Eng. 146 (12) (2024).
[37] F. Mangani, A. Roccon, F. Zonta, A. Soldati, Heat transfer in drop-laden turbulence, J. Fluid Mech. 978 (2024) A12.
[38] C. Speziale, On the advantages of the vorticity-velocity formulation of the equations of fluid dynamics, J. Comput. Phys. 73 (2) (1987) 476–480.
[39] A. Soldati, S. Banerjee, Turbulence modification by large-scale organized electrohydrodynamic flows, Phys. Fluids 10 (7) (1998) 1742–1756.
[40] V. Badalassi, H. Ceniceros, S. Banerjee, Computation of multiphase systems with phase field models, J. Comput. Phys. 190 (2) (2003) 371–397.
[41] P. Yue, J. Feng, C. Liu, J. Shen, A diffuse-interface method for simulating two-phase flows of complex fluids, J. Fluid Mech. 515 (1) (2004) 293–317.
[42] Message Passing Interface Forum, A Message-Passing Interface Standard, Version 3.0, High-Performance Computing Center Stuttgart, Germany, 2012.
[43] J. Sweet, D. Richter, D. Thain, GPU acceleration of Eulerian–Lagrangian particle-laden turbulent flow simulations, Int. J. Multiph. Flow 99 (2018) 437–445.
[44] A. Roccon, A GPU-ready pseudo-spectral method for direct numerical simulations of multiphase turbulence, in: Procedia, 2024.
[45] K. Czechowski, C. Battaglino, C. McClanahan, K. Iyer, P.-K. Yeung, R. Vuduc, On the communication complexity of 3D FFTs and its implications for exascale, in: Proceedings of the 26th ACM International Conference on Supercomputing, 2012, pp. 205–214.
[46] L. Dalcin, M. Mortensen, D.E. Keyes, Fast parallel multidimensional FFT using advanced MPI, J. Parallel Distrib. Comput. 128 (2019) 137–150.
[47] K. Ravikumar, D. Appelhans, P. Yeung, GPU acceleration of extreme scale pseudo-spectral simulations of turbulence using asynchronism, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019, pp. 1–22.
[48] R.D. Moser, J. Kim, N.N. Mansour, Direct numerical simulation of turbulent channel flow up to $Re_\tau = 590$, Phys. Fluids 11 (4) (1999) 943–945.
[49] M. Bernardini, S. Pirozzoli, P. Orlandi, Velocity statistics in turbulent channel flow up to $Re_\tau = 4000$, J. Fluid Mech. 742 (2014) 171–191.
[50] F. Zonta, P. Sichani, A. Soldati, Interaction between thermal stratification and turbulence in channel flow, J. Fluid Mech. 945 (2022) A3.
[51] P. Costa, A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows, Comput. Math. Appl. 76 (8) (2018) 1853–1862.
[52] V. Giurgiu, G.C.A. Caridi, M. Alipour, M. De Paoli, A. Soldati, The TU Wien turbulent water channel: flow control loop and three-dimensional reconstruction of anisotropic particle dynamics, Rev. Sci. Instrum. 94 (9) (2023).
[53] T. Von Kármán, Mechanical Similitude and Turbulence, Reprint from Nachrichten von der Gesellschaft der Wissenschaften zu Gottingen, 1931.
[54] G. Soligo, A. Roccon, A. Soldati, Deformation of clean and surfactant-laden droplets in shear flow, Meccanica 55 (2020) 371–386.
[55] H. Zhou, C. Pozrikidis, The flow of suspensions in channels: single files of drops, Phys. Fluids 5 (1993) 311–324.
[56] S. Guido, M. Villone, Three-dimensional shape of a drop under simple shear flow, J. Rheol. 42 (1998) 395–415.
[57] J. Li, Y. Renardy, M. Renardy, Numerical simulation of breakup of a viscous drop in simple shear flow through a volume-of-fluid method, Phys. Fluids 12 (2) (2000) 269–282.
[58] A. Komrakova, O. Shardt, D. Eskin, J. Derksen, Lattice Boltzmann simulations of drop deformation and breakup in shear flow, Int. J. Multiph. Flow 59 (2014) 24–43.
[59] M. Shapira, S. Haber, Low Reynolds number motion of a droplet in shear flow including wall effects, Int. J. Multiph. Flow 16 (2) (1990) 305–321.
[60] G. Deane, M. Stokes, Scale dependence of bubble creation mechanisms in breaking waves, Nature 418 (6900) (2002) 839.
[61] C.E. Blenkinsopp, J.R. Chaplin, Bubble size measurements in breaking waves using optical fiber phase detection probes, IEEE J. Ocean. Eng. 35 (2) (2010) 388–401.
[62] A.H. Callaghan, M.D. Stokes, G.B. Deane, The effect of water temperature on air entrainment, bubble plumes, and surface foam in a laboratory breaking-wave analog, J. Geophys. Res., Oceans 119 (11) (2014) 7463–7482.
[63] L. Deike, W. Melville, S. Popinet, Air entrainment and bubble statistics in breaking waves, J. Fluid Mech. 801 (2016) 91–129.
[64] S. Mukherjee, A. Safdari, O. Shardt, S. Kenjereš, H.E.A. Van den Akker, Droplet-turbulence interactions and quasi-equilibrium dynamics in turbulent emulsions, J. Fluid Mech. 878 (2019) 221–276.
[65] S. Di Giorgio, S. Pirozzoli, A. Iafrati, On coherent vortical structures in wave breaking, J. Fluid Mech. 947 (2022) A44.
[66] M. Crialesi-Esposito, S. Chibbaro, L. Brandt, The interaction of droplet dynamics and turbulence cascade, Commun. Phys. 6 (1) (2023) 5.
[67] I. Cannon, G. Soligo, M. Rosti, Morphology of clean and surfactant-laden droplets in homogeneous isotropic turbulence, J. Fluid Mech. 987 (2024) A31.

[68] A.N. Kolmogorov, The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers, Proc., Math. Phys. Eng. Sci. 434 (1890) (1991) 9–13.
[69] J. Hinze, Fundamentals of the hydrodynamic mechanism of splitting in dispersion processes, AIChE J. 1 (3) (1955) 289–295.
[70] C. Garrett, M. Li, D. Farmer, The connection between bubble size spectra and energy dissipation rates in the upper ocean, J. Phys. Oceanogr. 30 (9) (2000) 2163–2171.
[71] S. Mirjalili, S.S. Jain, M. Dodd, Interface-capturing methods for two-phase flows: an overview and recent developments, Cent. Turbul. Res. Ann. Res. Briefs 2017 (117–135) (2017) 13.
[72] Y. Na, D.V. Papavassiliou, T.J. Hanratty, Use of direct numerical simulation to study the effect of Prandtl number on temperature fields, Int. J. Heat Fluid Flow 20 (3) (1999) 187–195.
[73] F. Zonta, C. Marchioli, A. Soldati, Modulation of turbulence in forced convection by temperature-dependent viscosity, J. Fluid Mech. 697 (2012) 150–174.
[74] F. Zonta, C. Marchioli, A. Soldati, Direct numerical simulation of turbulent heat transfer modulation in micro-dispersed channel flow, Acta Mech. 195 (2008) 305–326.
[75] C. Marchioli, A. Soldati, J. Kuerten, B. Arcen, A. Taniere, G. Goldensoph, K. Squires, M. Cargnelutti, L. Portela, Statistics of particle dispersion in direct numerical simulations of wall-bounded turbulence: results of an international collaborative benchmark test, Int. J. Multiph. Flow 34 (9) (2008) 879–893.
[76] J. Romero, P. Costa, M. Fatica, Distributed-memory simulations of turbulent flows on modern GPU systems using an adaptive pencil decomposition library, in: Proceedings of the Platform for Advanced Scientific Computing Conference, 2022, pp. 1–11.
[77] H. Chu, AMD heterogeneous uniform memory access, in: Proceedings of the APU 13th Developer Summit, 2013, pp. 11–13.
[78] Z. Jin, J.S. Vetter, Evaluating unified memory performance in HIP, in: 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2022, pp. 562–568.
[79] AMD, rocFFT library, https://github.com/ROCmSoftwarePlatform/rocFFT.