Computer Physics Communications 316 (2025) 109804

Contents lists available at ScienceDirect

Computer Physics Communications

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

Computer Programs in Physics

MHIT36: A phase-field code for GPU simulations of multiphase homogeneous isotropic turbulence

Alessio Roccon [a,b,*], Lea Enzenberger [b], Domenico Zaza [b], Alfredo Soldati [a,b]

[a] Polytechnic Department of Engineering and Architecture, University of Udine, 33100 Udine, Italy
[b] Institute of Fluid Mechanics and Heat Transfer, TU-Wien, 1060 Vienna, Austria

ARTICLE INFO

The review of this paper was arranged by Prof. W. Jong

Dataset link: https://doi.org/10.6084/m9.figshare.28082252

Keywords:
Phase-field method
Direct numerical simulation
Finite difference
GPU-computing
Domain decomposition

ABSTRACT

We present MHIT36, a GPU-tailored solver for interface-resolved simulations of multiphase turbulence. The framework couples direct numerical simulation (DNS) of the Navier–Stokes equations, which describe the flow field, with a phase-field method to capture interfacial phenomena. Simulations are performed in a cubic domain with periodic boundary conditions applied in all three spatial directions. The governing equations are discretized using a second-order finite difference scheme. The Navier–Stokes equations are integrated with an explicit fractional-step method, and the resulting pressure Poisson equation is solved using a fast Fourier transform (FFT)-based approach. The accurate conservative diffuse interface (ACDI) formulation is used to describe the transport of the phase-field variable. From a computational standpoint, MHIT36 employs a two-dimensional domain decomposition to distribute the workload across MPI tasks. The cuDecomp library is used to perform pencil transpositions and halo exchanges, while the cuFFT library and OpenACC directives are leveraged to offload the remaining computational kernels to the GPU. This parallelization strategy enables MHIT36 to achieve an excellent scaling efficiency on 1024 GPUs, while maintaining a structure that is easy to extend and modify. MHIT36 is released open source under the MIT license.

Program summary
Program Title: MHIT36
CPC Library link to program files: https://doi.org/10.17632/yb2dt99swr.1
Developer's repository link: https://github.com/MultiphaseFlowLab/MHIT36
Licensing provisions: MIT License
Programming language: Modern Fortran
Nature of problem: Solving the three-dimensional incompressible Navier-Stokes equations in a triply-periodic box. A phase-field method based on the accurate conservative diffuse interface (ACDI) formulation is used to describe the shape and topological changes of the interface.
Solution method: The system of governing equations is advanced in time using an explicit strategy while the governing equations are discretized in space using a second-order finite difference approach. A fractional step is used to solve the Navier-Stokes equations and an FFT-based method is used to solve the resulting Poisson equation for pressure. The parallelization relies on a 2D domain decomposition strategy and all intra- and inter-node communications are handled by the cuDecomp strategy. The cuFFT library and OpenACC directives are used to entirely offload code execution to GPUs.

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 much more challenging: these flows require the accurate representation of a continuously evolving interface, including its topological changes and interaction with turbulence [6–9]. High-fidelity simulations are of vital im-

* 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.2025.109804
Received 14 May 2025; Received in revised form 14 July 2025; Accepted 2 August 2025

Available online 7 August 2025

portance 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 dispersed multiphase flow – as for instance an emulsion – on a discretized temporal and spatial grid is 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 that governs interface-interface interactions. 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 for single-phase turbulence [11–13], it would be desirable to perform simulations in which all scales are directly resolved, without any model. This brute force approach, however, 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. In this context, performing finely resolved simulations is highly desirable as it allows to improve the description of very thin interfacial structures and alleviate the inherent drawbacks present in this type of simulations due to the impossibility to model and resolve the wide range of scales involved. The need of performing simulations with large grid resolution requires the use of a large amount of computing resources. For this reason, MHIT36 is tailored towards GPU-accelerated high-performance systems, which constitute the most common and efficient computing architecture currently available in pre-exascale and exascale computing infrastructures.

MHIT36 performs interface-resolved simulations of dispersed multiphase flows. Within the one-fluid formulation, the code combines direct numerical simulation (DNS) to solve the incompressible Navier-Stokes equations with a phase-field method (PFM) based on the conservative Allen-Cahn equation to capture interfacial dynamics. The governing equations are discretized in space using a second-order finite difference method, while time integration is performed using the explicit two-step Adams-Bashforth scheme. The incompressibility constraint is enforced using the fractional-step method, originally introduced by Chorin [14] and Temam [15] to solve the primitive-variable formulation of the Navier-Stokes equations. At each time step, the momentum equations for the fluid are advanced in time neglecting the contribution of the pressure gradient, thus yielding an intermediate velocity field that does not satisfy the incompressibility constraint. Then, a Poisson equation is solved for an auxiliary scalar field, commonly interpreted as the pressure or a pressure correction. The gradient of this scalar field is used to correct the velocity and project the intermediate solution onto the divergence-free space, thereby obtaining the solenoidal velocity at the new time step.

The code has been developed from scratch and targets GPU architectures. It uses the MPI library as a bootstrap to perform the initial 2D domain decomposition (slabs or pencils) and then leverages the highly efficient cuDecomp library [16] to handle all inter- and intra-node communication tasks, including pencil/slab transpositions and halo updates. The cuDecomp library supports seven different communication backends and has specifically been developed for GPU-based computing architectures. In addition, it features auto-tuning capabilities and it is thus able to identify the most suitable domain decomposition and communication backend before running the main application. Fast Fourier transforms and the computations of the different terms present in the two equations are offloaded to GPUs using OpenACC directives and cuFFT libraries [17]. The chosen approach allows for obtaining an efficient and easy-to-modify code that can also be extended from the currently implemented flow configuration of homogeneous isotropic turbulence to other flow instances (e.g., open and closed channel flows, square duct flows). In addition, the solution of new governing equations (e.g., for description of heat and mass transfer, surfactants, etc.) can also be included as the code is developed with a modular structure in mind.

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 implementation of the parallelization strategy is presented and the strong scaling and code profiling results are reported. In Section 5, benchmarks for single- and multiphase flows are used to demonstrate the accuracy and the capabilities of MHIT36. Finally, we draw our conclusions in Section 6.

2. Governing equations

To describe the dynamics of the system, direct numerical simulation (DNS) of the Navier-Stokes equations, used to describe the flow field, are coupled with a second-order phase-field method (PFM), used to describe the shape and deformation of the interface as well as coalescence and breakage phenomena.

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 with matched density and viscosity. Under these assumptions, the continuity and Navier-Stokes equations read as:

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

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} + \mathbf{f}_\sigma + \mathbf{f}_{ABC}\,, \tag{2}$$

where $\mathbf{u} = (u, v, w)$ is the velocity vector, p is the pressure field, $\mathbf{f}_\sigma$ represents the surface tension forces and $\mathbf{f}_{ABC}$ is the turbulence forcing. The surface tension forces are evaluated using an energy-based surface tension model [18]. Specifically, the interfacial forces are computed as follows:

$$\mathbf{f}_\sigma = \mu_\phi \nabla \phi\,, \tag{3}$$

where the chemical potential μ_ϕ is defined as follows:

$$\mu_\phi = \frac{6\sigma}{\epsilon}\psi' - 6\sigma\epsilon\nabla^2\phi\,, \tag{4}$$

where σ is the surface tension and ψ' is the free energy functional [18, 8], whose expression is:

$$\psi' = \phi(1-\phi)(1-2\phi)\,. \tag{5}$$

The use of this surface-tension model minimizes the number of communications required (e.g., halo updates) as well as reduces the magnitude of the spurious currents induced by surface tension forces [18].

To sustain the turbulence motion, an ABC forcing scheme [19,20] is employed, which is defined as follows:

$$\mathbf{f}_{ABC} = [C\sin(\kappa_F z) + B\cos(\kappa_F y)]\mathbf{i} \tag{6}$$

$$+ [A\sin(\kappa_F x) + C\cos(\kappa_F z)]\mathbf{j} \tag{7}$$

$$+ [B\sin(\kappa_F y) + A\cos(\kappa_F x)]\mathbf{k}\,, \tag{8}$$

where A, B and C are three coefficients, κ_F is the forced wavenumber and $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are the unit vector along the three directions.

2.2. Phase-field method

For the description of the interface, we employ here the accurate conservative diffuse-interface/phase-field method (ACDI/ACPF) [21]. The phase-field method relies on an Eulerian variable, ϕ, which is constant in the bulk of the two phases ($\phi = 0$ and $\phi = 1$) and undergoes a smooth transition across a thin interfacial layer [22,8]. The time evolution of

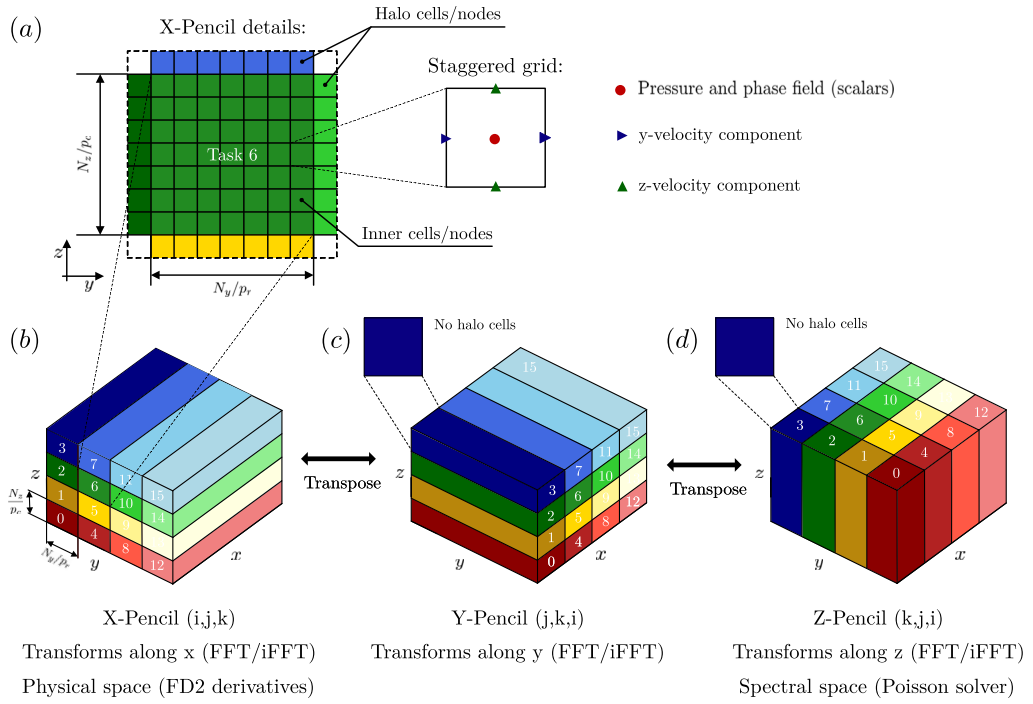

Fig. 1. The domain is decomposed along the y and z directions in $p_r \times p_c$ pencils, respectively. Each color corresponds to a different MPI task, numbered from 0 to 15 ($p_r = 4$ and $p_c = 4$). In physical space, the domain is divided along the y and z directions (X-pencil) while in Fourier space (i.e., after the 3D FFT) the domain is decomposed along the x and y directions (Z-pencil). All derivatives are evaluated in physical space using a second-order finite difference approach and only in this configuration halo cells are present (X-pencil, panels *a* and *b*). In contrast, when the pressure Poisson equation is solved using the FFT-based solver, in spectral space, no halo cells are present as they are not required (Y- and Z-pencils, panels *c* and *d*). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

the phase field variable is described by a modified version of the conservative Allen-Cahn equation [23,24,21], which reads as follows:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{u}\phi) = \nabla \cdot \left[\Gamma \left(\epsilon \nabla \phi - \frac{1}{4} \left[1 - \tanh^2 \left(\frac{\psi}{2\epsilon} \right) \right] \frac{\nabla \psi}{|\nabla \psi|} \right) \right], \tag{9}$$

where ϵ is a numerical parameter that controls the characteristic length-scale of the thin transition layer and ψ is a smooth function defined as follows:

$$\psi = \epsilon \log \left(\frac{\phi + \xi}{1 - \phi + \xi} \right), \tag{10}$$

where ξ is a small number used to avoid an undefined solution when ϕ approaches the bulk values. On the right-hand side, we can distinguish the diffusive and sharpening terms, which allow for preserving the interfacial profile during the computation, e.g., an hyperbolic tangent profile. The strength of these two terms is tuned via the numerical parameter Γ, which should be set accordingly [24,25]. We adopt this formulation of the phase-field method because, with appropriate selection of the parameters ϵ and Γ, it guarantees a bounded solution for the phase-field variable [24]. This property is crucial for accurately evaluating quantities that depend directly on the local value of the phase-field variable [8,26]. In addition, because this method relies on a second-order partial differential equation, its numerical discretization is more suitable for second-order finite difference schemes—unlike the Cahn–Hilliard equation, which involves a fourth-order term and typically requires higher-order discretizations or ad hoc numerical techniques [8,27].

3. Numerical method

The governing equations [(1)–(2)–(9)] are solved in a triply periodic domain using a fixed, uniform, staggered grid, as shown in the inset of Fig. 1*a*. Velocity components are defined at cell faces, while scalar fields such as pressure and the phase field are defined at cell centers. Spatial derivatives are evaluated using a second-order central difference scheme. Time integration of the momentum equation is carried out using a fractional-step method [14,15], with the provisional velocity field computed via the Adams–Bashforth scheme. Likewise, the phase-field equation is advanced in time using the same scheme. An explicit time integration strategy has been adopted for all governing equations to maximize performance on GPUs, which are particularly efficient for operations on independent data sets characteristic of explicit schemes. Similar strategies have also been adopted in other GPU-oriented codes or codes for simulations of multiphase flows [28–30].

3.1. Navier-Stokes equations

The Navier-Stokes and continuity equations are solved using a projection-correction method [14], reported below in a semi-discrete form:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = \alpha \mathcal{A}^n - \beta \mathcal{A}^{n-1}, \tag{11}$$

$$\nabla^2 p^{n+1} = \frac{\nabla \cdot \mathbf{u}^*}{\Delta t}, \tag{12}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \frac{\Delta t}{\rho} \nabla p^{n+1}, \tag{13}$$

where $\alpha = 1.5$ and $\beta = 0.5$ are the coefficients of the Adams-Bashforth scheme; the operator $\mathcal{A}^n$ and $\mathcal{A}^{n-1}$ includes the convective, diffusive and surface tension terms, computed at the current and previous time steps. For the first time step, where $\mathcal{A}^{n-1}$ is not defined, an Euler explicit scheme is used by setting $\alpha = 1$ and $\beta = 0$. To solve the resulting Poisson pressure equation in the context of the fractional step method, different methods are available: from iterative solvers – based for instance on the multi-grid method – up to direct solvers where, by exploiting periodic directions, the resulting Poisson equation is decoupled along one or more directions [28]. We follow here the latter approach: by leveraging

periodic boundary conditions in all spatial directions, the source term (right-hand side of equation (12)) is transformed into Fourier space using a three-dimensional Fast Fourier Transform (FFT). This ensures the solution of the Poisson equation with machine precision accuracy. In spectral space, the Poisson equation reduces to an algebraic relation, allowing the solution to be obtained via direct division by the squared wavevector magnitude. To ensure a unique solution for pressure, the zero wavenumber (mean) mode is set to zero. The inverse FFT is then applied to recover the solution for the pressure field in physical space. This method provides spectral accuracy and computational efficiency. Details on the parallel implementation of the Poisson solver can be found in Section 4. The type of Poisson solver employed can be easily extended to other types of flow configurations, e.g., channel flow or square ducts. Specifically, by changing the type of transforms performed (or avoiding the FFT operation), different combinations of pressure boundary conditions can be applied [28,31,32] and thus different flow geometries can be simulated without requiring major modifications to the code parallelization. This ensures that code performance and scalability of the MHIT36 code are retained.

3.2. Phase-field method

To advance the solution of the governing equation for the phase-field variable, an Adams–Bashforth scheme is used. The type of scheme adopted ensures that a bounded solution of the phase-field variable is obtained [24]. The governing equation in a semi-discrete form reads as follows:

$$\frac{\phi^{n+1}-\phi^{n}}{\Delta t}=\alpha\mathcal{B}^{n}-\beta\mathcal{B}^{n-1}\,, \tag{14}$$

where the terms $\mathcal{B}^{n}$ and $\mathcal{B}^{n-1}$ include the advection, diffusion and sharpening terms evaluated at the step n and $n-1$. All these terms are treated explicitly, which introduces a limitation on the maximum allowable time step. In particular, using the ACDI formulation [21], the maximum time step that can be used reads as follows:

$$\Delta t^{phase}_{max}=\frac{\Delta x}{u_{max}\epsilon^{*}}\,, \tag{15}$$

where $\epsilon^{*}=\epsilon/\Delta x$ and u_{max} is the maximum value of the velocity in the domain. This limitation is comparable to that imposed by the explicit treatment of the viscous terms in the Navier–Stokes equations. Indeed, for an Adams–Bashforth scheme, the maximum stable Courant–Friedrichs–Lewy (CFL) number that can be used for simulations of turbulent flow range from 0.3 to 0.5 [29,33]. Thus, the resulting maximum time step is:

$$\Delta t^{flow}_{max}=\frac{\mathrm{CFL}_{max}\Delta x}{u_{max}}\,. \tag{16}$$

By considering a unitary value of ϵ^{*}, we obtain $\Delta t^{phase}_{max}=\Delta x/u_{max}$ and thus the two time step restrictions are similar. Overall, the explicit treatment significantly improves the efficiency of the implementation as it eliminates the need for implicit solvers (and associated pencil transposes), which use can degrade performance on GPU architectures due to the global coupling and thus the additional communication required.

4. Implementation of the parallelization strategy

The domain is decomposed into smaller subdomains (pencils) in the y and z directions, the number of pencils along the two directions is p_r (y direction) and p_c (z direction), as shown in Fig. 1*b*. The domain decomposition (slabs or pencils) is performed using the MPI library (bootstrapping); during this phase, each pencil is assigned to a MPI task and in turn to a specific GPU. OpenACC directives are used to offload the computation of all the terms, which are evaluated in physical space using a second-order finite difference scheme. Likewise, all the Fourier transforms are performed exploiting the highly optimized cuFFT library [17], which can be invoked using the interoperability features present in OpenACC. To obtain a code that can be easily modified, 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. 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 can be accessed using a single pointer from both CPU and GPU code sections. In terms of performance, this choice has a minor effect (less than 2%) as most of the host-to-device and device-to-host memory transfers occur only during the initialization.

Communications between processors are required to perform halo updates and during the execution of the pressure Poisson solver, which, being based on FFTs, requires pencil transpositions to perform FFT along all three directions. Specifically, halo cells – extra layers of cells surrounding the boundaries of a pencil – must be updated whenever a variable is modified, as they are required to evaluate derivatives in physical space at the boundaries of the pencil. This operation requires communications with the neighboring pencils (i.e., processes); as illustrated in Fig. 1*a*. Communication is also required to perform a 3D FFT as each process must hold all the points along the transform direction. Starting from X-pencil (physical space, Fig. 1*b*), the first FFT is performed along the x direction. Following this, the parallelization is rearranged so that each process holds all the data along the y direction (Y-pencils, Fig. 1*c*), and the FFT is then performed in the y direction. Finally, the parallelization is changed again to align with the z direction (X-pencils, Fig. 1*d*), and the FFT is applied along the z direction.

All these communications – halo updates and pencil transpositions – are handled by the cuDecomp library [34]. The library supports multiple communication backends and includes autotuning capabilities to optimize performance and scalability on different combinations of network configurations and GPU architectures. Autotuning can be used to identify the most performing decomposition strategy (autotune process grid) as well as to also identify the most performing communication backend (full autotuning).

4.1. Strong scaling

To evaluate the code performance and the efficiency of the implementation reported above, we analyze the strong scaling behavior of the code. The tests have been performed on two different pre-exascale systems: i) booster module of Leonardo [35] (CINECA) ii) ACC partition of Marenostrum5 [36] (BSC). Refer to Table 1 for the technical specifications of the two clusters. For both machines, the Nvidia `nvfortran` compiler has been used. The scaling tests have been executed considering the single-phase solver (solution of the mass conservation and Navier-Stokes equations only) as well as the multiphase solver (solution of mass conservation and Navier-Stokes equations as well as transport equation of the phase-field variable).

The strong scaling results are shown in Fig. 2. The left panel shows the scaling of the single-phase solver, while panel *b* shows the strong scaling results of the multiphase solver. Four problem sizes have been tested: $256\times256\times256$ (blue), $512\times512\times512$ (red), $1024\times1024\times1024$ (orange) and $2048\times2048\times2048$ (violet). The autotune process grid is kept enabled during the scaling test, and the cuDecomp library defines, via some preliminary tests (before each run), the most suitable domain decomposition strategy. In particular, the library defines the number of pencils to be used along the y and z directions (i.e., the parameters p_r and p_c, see Fig. 1*a*). In addition, it is also possible to enable the full autotuning mode: the cuDecomp library not only defines the domain decomposition strategy but also the communication backend to be employed [16]. This second autotuning option has not been used here and for all the scaling tests the MPI backend (not pipelined) has been used. Analyzing the strong scaling performance exhibited by the code, we can observe that in general, very good results are obtained on both machines. In particular, the code exhibits an excellent scaling up to

Table 1
Technical specifications of the computing infrastructures employed for the strong scaling tests: booster partition of Leonardo and ACC partition of Marenostrum5.

System	CPU (per node)	GPU (per node)
Leonardo booster (CINECA)	1 x Intel Xeon Phi 8358 32c	4 x NVIDIA A100
Marenostrum5 - ACC (BSC)	2 x Intel Sapphire Rapids 8460Y 40c	4 x NVIDIA H100

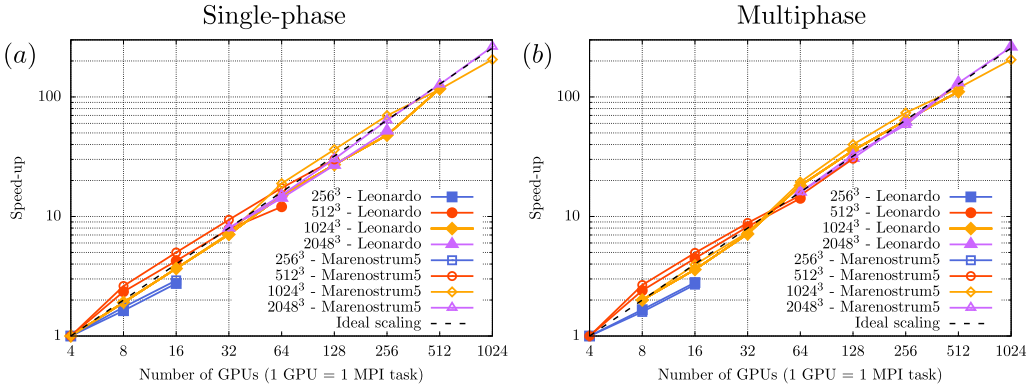

Fig. 2. Strong scaling results obtained on two GPU-accelerated computing infrastructures: Leonardo and Marenostrum5. Panel *a* shows the scaling of the single-phase solver and panel *b* of the multiphase solver (single-phase and phase-field). Four different problem sizes have been considered: 256^3 (blue), 512^3 (red), 1024^3 (orange) and 2048^3 (violet). Results obtained on Leonardo (booster) are reported using filled symbols while those obtained on Marenostrum5 (ACC partition) are represented with empty symbols.

512/1024 GPUs when the two larger problem sizes are considered. Interestingly, both the single-phase and multiphase solvers exhibit a similar scaling behavior. Indeed, the phase-field method is an interface blind method and thus its computational cost does not depend on the interface extension [8,9]. Specifically, the method does not rely on interface reconstruction algorithms or non-linear advection schemes, which can hinder scalability on GPU-based computing architectures.

It is worth to observe that scaling performance is retained among the two different machines tested and with different generations of GPU architectures (Ampere and Hopper). In terms of pure performance (i.e., time elapsed per iteration), the difference between the two architectures – Ampere (A100, Leonardo) and Hopper (H100, Marenostrum5) – is about 10% in favor of the newer GPU architecture. This difference is rather contained because the technical specifications of the two GPUs are identical in terms of memory bandwidth, which is the main bottleneck in CFD codes [37,38]. Indeed, both cards offer a peak bandwidth of about 1.6 TB/s [35,36] and are based on a HBM2e memory technology.

The efficiency of the present GPU implementation is further demonstrated through performance benchmarks comparing CPU and GPU executions. These tests evaluate the time required for a single time step, using all available cores on a single node (32 on Leonardo and 80 on Marenostrum) against the time taken by the 4 GPUs available per node (4 on both systems). For both machines, the GPU implementation achieves a speedup of approximately 17× compared to the CPU counterpart; these results align with the speed-up achieved by codes based on similar numerical methods [39].

4.2. Weak scaling

Having analyzed the strong scaling performance of the code, we now turn to analyze the weak scaling results. The tests were conducted on the same two clusters used for the strong scaling benchmarks, employing the Nvidia `nvfortran` compiler. As with the strong scaling analysis, results are presented for both the single-phase solver—which involves solving the mass conservation and Navier–Stokes equations—and the multiphase solver, which additionally includes the transport equation for the phase-field variable. The results are shown in Fig. 3. The left panel shows the weak scaling performance of the single-phase solver, while panel *b* presents the weak scaling of the multiphase solver. Two problem sizes per node have been tested: 256^3 (blue) and 512^3 (red), corresponding to approximately 4 and 32 million grid points per MPI task, respectively (with 4 MPI tasks per node, one per GPU). First, we observe that both the single-phase and multiphase solvers exhibit similar behavior in terms of weak scaling efficiency. In general, better efficiency is achieved when using the larger problem size per node (full and empty circles), as the increased computational workload helps to partially hide communication overhead. On both machines, the scaling efficiency remains satisfactory even when using 512 or 1024 GPUs. The main bottleneck limiting efficiency at scale arises from the pencil transpose operations required by the Poisson solver. These operations involve all-to-all communication, which is a well-known scalability challenge in numerical methods relying on direct FFT-based solvers [28,40–43,34]. Here, the use of the cuDecomp library provides a significant advantage: its autotuning capabilities help mitigate this bottleneck by selecting the optimal pencil decomposition strategy and communication backend for a given computing system and problem size.

4.3. Code profiling

To gain further insights into the time distribution across the different code sections, Fig. 4 shows the time spent in each section, normalized by the total runtime and expressed as a percentage. The results have been obtained considering a problem size equal to 512^3 executed on a single-node of the Leonardo supercomputer. Panel *a* refers to the single-phase solver, while panel *b* shows the breakdown for the multiphase solver. In both cases, the majority of computational time is devoted to solving the pressure Poisson equation, which involves performing FFTs and the associated pencil transpositions needed to compute FFTs along all three spatial directions. This is typical of solvers based on the fractional step method, where the Poisson equation solution is a major computational bottleneck [38,34]. Transpose operations, which require all-to-all communication, account for approximately 25% of the total time per time step in both solver versions. In contrast, halo updates—which also involve inter- and intra-node communication—have a significantly smaller impact. These updates involve only nearest-neighbor exchanges with relatively small data volumes and contribute less than 7% to the total runtime for both solvers. For the multiphase solver, halo

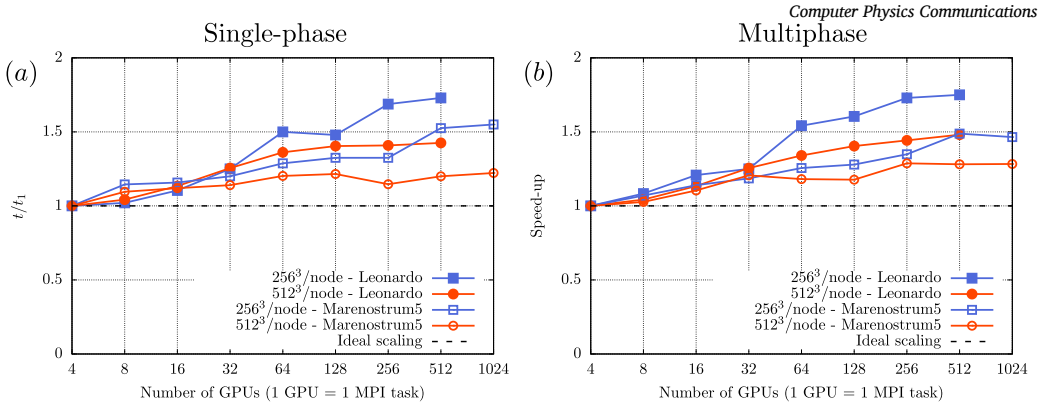

Fig. 3. Weak scaling results obtained on two GPU-accelerated computing infrastructures: Leonardo and Marenostrum5. Panel *a* refers to the single-phase solver while panel *b* to the multiphase solver. Two problem sizes have been considered: 256^3 points per node (blue) and 512^3 grid points per node (red). This corresponds to about 4 and 32 million grid points per MPI task (4 for each node), respectively. Results obtained on Leonardo (booster) are reported using filled symbols while those obtained on Marenostrum5 (ACC partition) are represented with empty symbols.

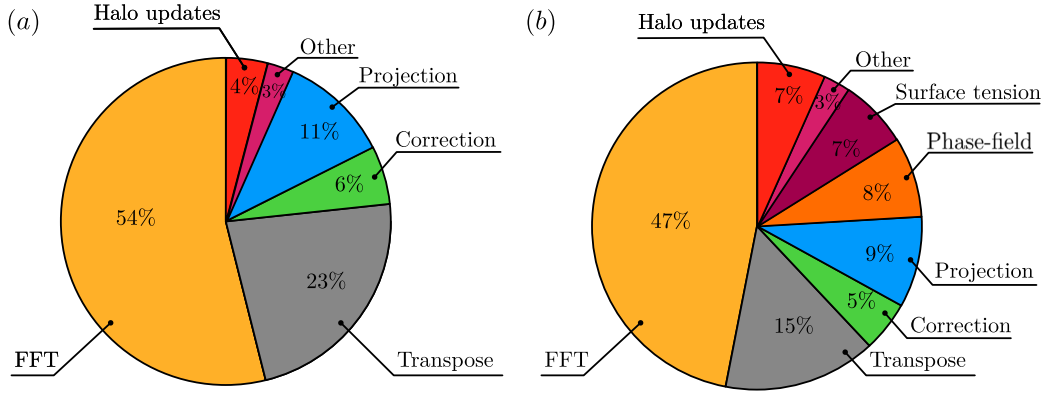

Fig. 4. Visualization of the time spent in the different code-sections with respect to the total time for the single-phase solver (panel *a*) and for the multiphase solver (panel *b*). The results refer to a grid size equal to 512^3 executed on 4 GPUs on Leonardo.

updates have a slightly greater impact due to the additional communications required for solving the phase-field transport equation and computing surface tension forces. Finally, the overhead introduced by the phase-field method is relatively modest. As shown in the pie chart of the multiphase solver, only 15% of the total runtime is used to advance the phase-field transport equation and to evaluate surface tension forces.

While the results presented in Fig. 4 refer to a problem size of 512^3 executed on a single node (4 GPUs), it is worth discussing how the relative cost of the different code sections changes when larger problem sizes or a greater number of GPUs are used. For the single-phase flow solver, the projection and correction steps exhibit good scalability, as their computational cost scales proportionally with the number of points per task. Similarly, halo updates involve only communication with neighboring tasks and therefore do not represent a significant communication bottleneck. In contrast, the Poisson solver—which accounts for a substantial portion of the total time due to the combined cost of FFTs and pencil transposes—exhibits only satisfactory scalability. This is because the transpose step requires all-to-all communication, a known scalability bottleneck [28,34,41,43]. This trend is also evident in the weak scaling results, where the relative cost of the Poisson solver increases with the number of MPI tasks. Turning to the multiphase solver, the picture does not change remarkably compared to the single-phase case when scaling to larger problem sizes or more GPUs. Indeed, the phase-field code sections demonstrate good scalability: the kernels involved are similar to those used in the projection and correction steps, and only a few additional halo exchanges are required.

5. Validation and benchmark

In the following, we report the validation tests of the code and some results from production runs to illustrate the capabilities of MHIT36. First, we validate the implementation of the single-phase solver. Second, we move to the validation of the phase-field module by considering different tests, from the Laplace equation to the breakage of drops in homogeneous isotropic turbulence.

5.1. Taylor-Green vortex

The first validation test we consider is the temporal evolution of a Taylor-Green vortex. For this benchmark, only the Navier–Stokes equations are solved, with no external forcing applied. The simulation is performed in a triply periodic box with dimensions $L\times L\times L = 2\pi\times 2\pi\times 2\pi$. The viscosity is set to $\nu = 1/1600$ and the fluid density is $\rho = 1$ and the resulting Reynolds number is equal to $Re_\Gamma = 1/\nu = 1600$ [28,38,44]. The following initial condition for the velocity field is adopted:

$$
\begin{aligned}
u(x,y,z) &= \frac{2}{\sqrt{3}}\sin\left(\theta+\frac{2\pi}{3}\right)\sin(x)\cos(y)\cos(z)\,,\\
v(x,y,z) &= \frac{2}{\sqrt{3}}\sin\left(\theta-\frac{2\pi}{3}\right)\cos(x)\sin(y)\cos(z)\,,\\
w(x,y,z) &= \frac{2}{\sqrt{3}}\sin(\theta)\cos(x)\cos(y)\sin(z)\,,
\end{aligned}
\tag{17}
$$

where θ is a free parameter that defines a family of possible initial conditions and is set to zero [44]. The smooth initial velocity field generates

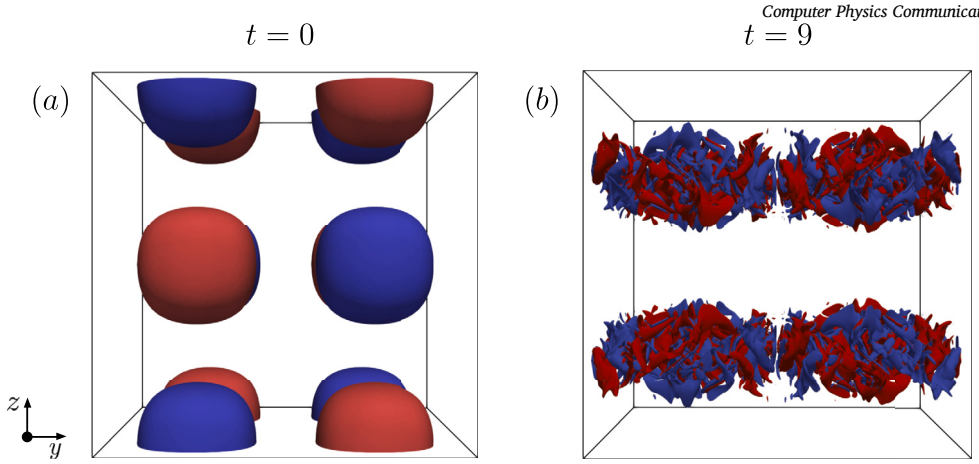

Fig. 5. Iso-contour of the z-component of the vorticity $\omega_z = \partial v/\partial x - \partial u/\partial y$ for the Taylor–Green vortex benchmark. The left panel refers to the initial time instant ($t = 0$) while the right panel to the time instant where the maximum dissipation rate is attained ($t = 9$). The two iso-contours shown refer to $\omega_z = \pm 2$ (panel a) and $\omega_z = \pm 20$ (panel b) and are rendered in red and blue, respectively.

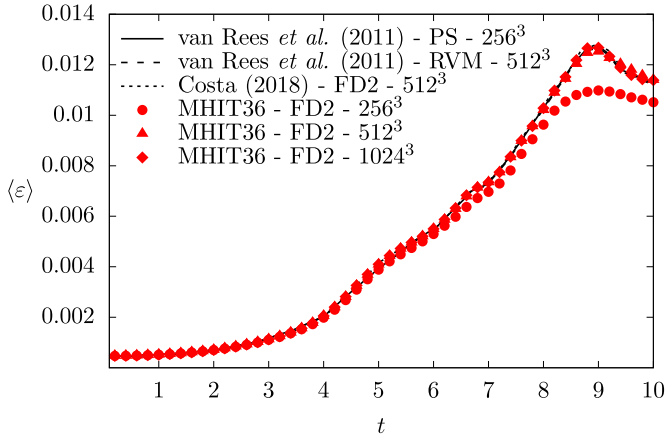

Fig. 6. Time evolution of the average dissipation rate for the Taylor–Green vortex benchmark. Archival literature results obtained by van Rees et al. (2011) [44] (pseudo-spectral and remeshed vortex method) and Costa (2018) [28] (finite difference) are reported as reference. Results obtained from MHIT36 using three different grid resolutions are reported using red symbols: 256^3 (circles), 512^3 (triangles), 1024^3 (diamonds).

vorticity through the vortex-stretching mechanism, leading to the formation of small-scale vortex structures. This process is illustrated in Fig. 5, which shows iso-contours of the vorticity magnitude at the initial time ($t = 0$) in panel a and at the time of maximum energy dissipation ($t = 9$), in panel b.

We compare the results obtained from MHIT36 against those available in archival literature [28,44,38]. Fig. 6 shows the time evolution of the viscous dissipation, averaged over the entire domain. Archival literature results are shown in black: a continuous and a dashed line are used to identify the results obtained by van Rees et al. (2011) [44] using a pseudo-spectral and a remeshed vortex method while results obtained by Costa (2018) [28] using a second-order finite difference method are reported using a dotted line. Results obtained from MHIT36 using three different grid resolutions (256^3, 512^3 and 1024^3) are shown using different red symbols. Comparing the results, we observe that with a grid resolution equal or larger than 512^3, present results converge to archival literature results and an excellent agreement is found. The smaller grid resolution (256^3) is not sufficient to describe the small-scale vortical structures generated in the latter stages of the simulation. It is worth observing that the results of van Rees et al. (2011) [44] obtained using the pseudo-spectral method exhibit a higher order of convergence as the method accuracy allows for the recovery of the correct behavior with a lower grid resolution. Indeed, the results of Costa (2018) [44], which are obtained using a second-order finite difference method, employ a grid resolution equal to 512^3, thus conforming to present results. The same grid resolution is also required for the remeshed vortex method to converge to reference results. Notably, for a given grid resolution, the computational cost associated with a pseudo-spectral method is higher than the one of a second-order finite-difference method and thus the two methods have a comparable accuracy/computational cost ratio. Specifically, using the same grid resolution, using the pseudo-spectral solver present in the cuDecomp library examples folder, we observe that it requires about 7 times the time required by MHIT36 for a single time step. Clearly, this is only a rough indication as many factors can influence performance, especially using GPU-based architectures.

5.2. Homogeneous isotropic turbulence

We consider now the homogeneous isotropic turbulence flow case. This is a well-established setup that has been largely validated and tested using many different methods. This flow instance considers a triply-periodic box having dimensions $L \times L \times L = 2\pi \times 2\pi \times 2\pi$. We replicate here the flow configuration adopted by Crialesi-Esposito et al. (2023) [45]. Specifically, we set the viscosity equal to $\nu = \mu/\rho = 0.006$, and the flow is forced using the following parameters: $A = B = C = 1$ and $\kappa_F = 2$. Two different grid resolutions have been used: $N \times N \times N = 256 \times 256 \times 256$ and $N \times N \times N = 512 \times 512 \times 512$. The flow is initialized using the Taylor-Green flow used for the previous test case. After a short transient, memory of the initial condition is lost, and a steady-state configuration for the turbulent flow is obtained. In this new configuration, the energy injected via the ABC forcing balances out with the viscous energy dissipation. The mean flow induced by the ABC forcing is removed at every time step in order to have a proper homogeneous isotropic turbulence flow.

To validate the results obtained, we first evaluate some macroscopic properties of the flow, such as the mean viscous dissipation and the associated Taylor and Kolmogorov length scales. The Taylor and Kolmogorov length scales and the corresponding Taylor Reynolds number have been computed as follows:

$$\lambda = \left(\frac{15 u_{rms} \nu}{\varepsilon}\right)^{1/2}, \qquad \eta_k = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4}, \qquad Re_\lambda = \frac{\lambda u_{rms}}{\nu}, \tag{18}$$

where u_{rms} identifies the root mean square of the velocity fluctuations.

The resulting values are reported in Table 2 as well as reference data obtained from previously published literature [45–47]. Analyzing present results, we observe a very good agreement with the reference data for all the macroscopic parameters for both grid resolutions. In-

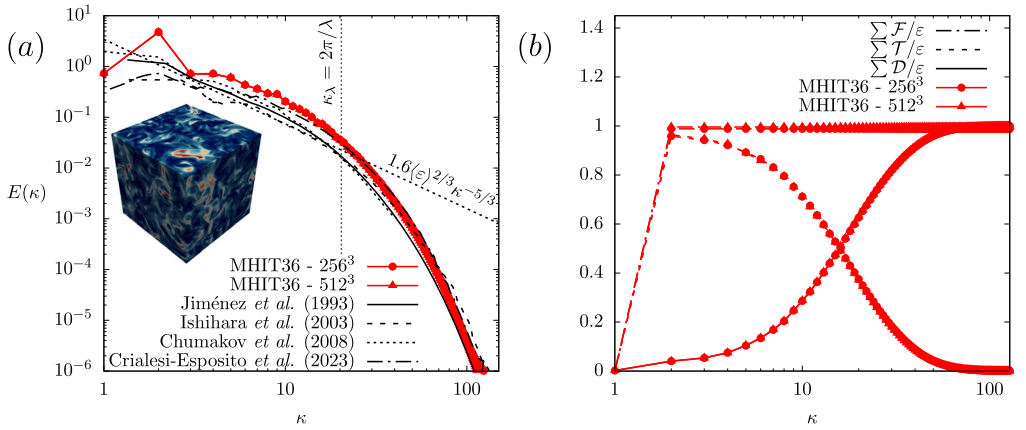

Fig. 7. Panel *a* shows the power energy spectrum for the homogeneous isotropic turbulence flow for the two grid resolutions tested: 256^3 (circles) and 512^3 (triangles). Results are compared against archival literature data at roughly the same Taylor Reynolds number: Jiménez et al. (1993) [46], Ishihara et al. (2003) [47], Chumakov et al. (2008) [48] and Crialesi-Esposito et al. (2023) [45]. Panel *b* shows the cumulative scale-by-scale energy budget; the contributions are shown using different line styles: injected energy (dash-dotted), viscous dissipation (continuous) and energy transfer (dashed). The inset shows a rendering of the turbulent kinetic energy.

Table 2
Macroscopic parameters obtained from the simulation of homogeneous isotropic turbulence. Archival literature results are also reported as reference.

Case	$\langle\varepsilon\rangle$	u_{rms}	λ	η_k	Re_λ
MHIT36 - 256^3	2.955	1.76	0.308	0.016	94
MHIT36 - 512^3	3.002	1.80	0.316	0.016	95
Crialesi-Esposito et al. (2023) [45]	3.059	1.81	0.314	0.016	95
Jiménez et al. (1993) [46]			0.311	0.016	94
Ishihara et al. (2003) [47]			0.327	0.017	94

deed, for this value of the Taylor Reynolds number, even the smaller grid resolution (256^3) is sufficient to accurately resolve all the flow scales. In particular, for the smaller grid resolution, we obtain $K\eta_k \simeq 2.1 > 2$ with $K = N/2$ (N being the grid resolution), thus ensuring that the resolution used is adequate [46].

Moving to more detailed statistics, we start by considering the kinetic energy spectrum, shown in Fig. 7*a*. Results obtained from MHIT36 are reported using red colors (circles for 256^3 and triangles for 512^3) while archival literature results[1] [45–48] are reported using black lines and different line styles. Comparing the results, we can observe a good agreement between all the different energy spectra. Differences in the y-values can be traced back to the different combination of forcing, density, and viscosity values used in the archival literature data. We can also appreciate the emergence of an inertial range that spans from the wavenumber associated with energy injection ($\kappa_F = 2$ in this case) down to approximately the wavenumber linked to the Taylor length scale, $\kappa_\lambda = 2\pi/\lambda$, covering about one decade. In the inertial range, the present energy spectra (red) match well with the theoretical scaling law $E(\kappa) = C\langle\varepsilon\rangle^{2/3}\kappa^{-5/3}$ being $C \simeq 1.6$ the Kolmogorov constant [49,50]. Finally, the peak observed for $\kappa = 2$ corresponds to the forced wavenumber ($\kappa_F = 2$) and it is also visible in other works where the ABC forcing is employed.

To further confirm the accurate description of the flow field, we compute the cumulative scale-by-scale (SBS) energy budget. The balance equation for the cumulative SBS energy budget can be obtained by integrating the SBS energy budget over a sphere (i.e., for all $|\kappa_i| < \kappa$) [51]. In this way, we obtain the following equation:

$$\frac{\partial \sum_{|\kappa_i|<\kappa} E(\kappa_i)}{\partial t} = \sum_{|\kappa_i|<\kappa} \mathcal{T}(\kappa_i) + \sum_{|\kappa_i|<\kappa} \mathcal{D}(\kappa_i) + \sum_{|\kappa_i|<\kappa} \mathcal{F}(\kappa_i), \tag{19}$$

where the first term on the right-hand side indicates the energy flux from the largest flow scales down to the dissipative scales, the second term represents the energy dissipated by the viscous forces and the third term represents the energy injected to sustain turbulence. At steady-state, when the injected energy balances out the energy dissipated by the viscous forces, the sum of these three terms is zero. The cumulative value of these three terms as a function of the wavenumber is shown in Fig. 7*b* using different line styles: injected energy (dash-dotted), viscous dissipation (continuous), and energy transfer (dashed). We can observe that the energy injection term (dot-dashed) is zero at $\kappa = 1$ and increases to a unitary value for $\kappa = 2$ as the flow is forced at $\kappa_F = 2$. Beyond $\kappa = 2$, the energy injection term contribution is zero as no energy is injected at smaller scales. The energy transfer mechanism (dashed) exhibits its peak for $\kappa = 2$ and then decays and approaches zero for larger wave numbers. Indeed, being an energy transfer mechanism, it has no active contribution to the energy balance. Finally, the viscous dissipation (continuous) starts from zero and increases for larger wave numbers where viscous forces become important, and its value reaches a unitary value. Present results confirm the capabilities of MHIT36 in describing all flow scales accurately as well as satisfying the energy balance of the system. Cumulative SBS trends are also in agreement with data from prior studies [51,52].

5.3. Enright test case

To validate the implementation of the phase-field method (ACDI formulation), we start by considering an advection test. We employ the Enright three-dimensional test case [53,54]. We consider a unitary computational domain $L \times L \times L = 1 \times 1 \times 1$. A sphere with diameter $D = 0.3L$ and located at $\mathbf{x_c} = (0.35, 0.35, 0.35)$ is advected by a time-varying velocity field. The velocity field is prescribed over space and time and it is defined as follows:

$$\begin{aligned} u(x,y,z) &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/T), \\ v(x,y,z) &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/T), \\ w(x,y,z) &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/T), \end{aligned} \tag{20}$$

where the last temporal pre-factor defines the temporal modulation of the velocity field. For the present case, we select $T = 3$. This flow stretches the sphere into a thin sheet, creating two bending and spiraling tongues. The maximum deformation occurs for $t = T/2 = 1.5$, where the temporal cosine pre-factor fully elongates the sphere, mak-

[1] With reference to the work of Chumakov et al. (2008), the data reported here is not present in the original manuscript but has been obtained using the code HIT3D, developed by Chumakov and available at https://github.com/schumakov/hit3d.

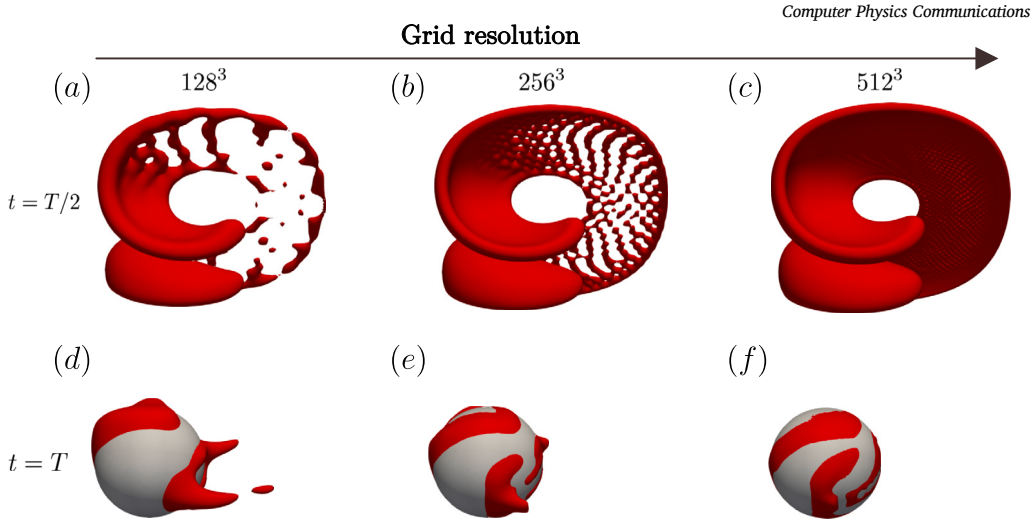

Fig. 8. Iso-contour of the phase-field variable ($\phi = 0.5$) for different grid resolutions (left to right) at two different times: $t = T/2 = 1.5$ (top row) and $t = T = 3.0$ (bottom row). In the plots reported in the bottom row, the initial spherical shape ($t = 0$) is also shown in gray. By increasing the grid resolution, the initial spherical droplet shape can be recovered more accurately and the thin interfacial structures obtained for $t = T/2$ can be better described.

ing it challenging for numerical methods to capture the resulting very thin interfacial structure. Beyond this point, the flow reverses, and the sphere is expected to recover its original spherical shape for $t = T = 3.0$.

We execute this benchmark by considering three different grid resolutions, from 128^3 up to 512^3; the phase-field parameters have been set equal to $\Gamma = 1$ and $\epsilon/\Delta x = 1$. The parameter $\epsilon/\Delta x$ controls the number of grid points present across the interface and specifically, by setting $\epsilon/\Delta x = 1$, from 4 to 5 grid points are present across the thin interfacial layer (identified as the region where the phase-field smoothly increases from $\phi = 0.1$ to $\phi = 0.9$. The Enright test case results are shown in Fig. 8 and each column refers to a different grid resolution. The top row refers to $t = T/2$ (maximum drop deformation) while the bottom row refers to $t = T$. In the bottom row, the initial spherical shape of the drop is also reported in gray as a reference. From the results shown, we can observe that with the larger grid resolution (512^3), the thin interfacial structures obtained for $t = T/2 = 1.5$ can be captured without inducing numerical breakage. In addition, we can also notice that as the grid resolution is increased, the initial spherical shape of the drop can be better recovered and only marginal deviations can be observed, Fig. 8*f*. This confirms the accuracy of the method from the advection and mass conservation point of view. Present results are also in agreement with those obtained by Pirozzoli et al. (2025) [54] using a similar phase-field formulation and by Enright et al. (2002) [53] using the particle level-set method.

5.4. Laplace pressure jump

To validate the implementation of the surface tension forces and thus the coupling between the phase-field method and the momentum and mass conservation equations, we consider a spherical drop in a quiescent fluid. We test here the capabilities of the code in reproducing the theoretical pressure jump, which for a three-dimensional spherical drop can be calculated as $\Delta p = 2\sigma/R$, R being the radius of the drop and σ the surface tension. For this test, we consider two fluids with the same density and viscosity and the phase-field method module is enabled.[2] Density and viscosity have been set equal to $\rho = 1$ and $\mu = 0.006$ and surface tension has been set equal to $\sigma = 1$. Simulations consider a triply-periodic box $L \times L \times L = 2\pi \times 2\pi \times 2\pi$ using a grid resolution equal to 256^3. Fig. 9*a* shows a comparison between the numerical and theoretical values of the pressure jump obtained. Different values of the ratio $\epsilon/\Delta x$ and droplet radii R/L (where L is the computational domain size) have been considered. Generally speaking, we can observe a very good agreement between theoretical and numerical results for the entire range of droplet radii and capillary widths ϵ considered. Small deviations can be observed for the smaller values of the drop radius considered and for $\epsilon/\Delta x > 1$. This small deviation is due to the fact that when the drop is small and $\epsilon/\Delta x > 1$, its size becomes comparable with the transition layer thus resulting in a loss of accuracy. The small imbalance between surface tension forces and pressure jump obtained for this case can lead to the generation of spurious currents, which are shown in Fig. 9*b*. Nevertheless, the surface tension model here employed, which avoids the explicit computation of the curvature, and the use of the ACDI formulation ensures that the magnitude of these spurious currents is minimized [18,21,55,56].

[2] Please note that this test case can be also performed without advancing in time the phase-field variable as the drop is not advected by the flow.

5.5. Coalescence and breakage of drops in turbulence

To demonstrate the capabilities of MHIT36 in describing drop-laden turbulence, we consider a swarm of large and deformable drops released in homogeneous isotropic turbulence. The configuration adopted by Crialesi-Esposito [45], originally simulated using the volume-of-fluid method, is here reproduced using MHIT36. The computational domain is a triply-periodic box with dimensions $L \times L \times L = 2\pi \times 2\pi \times 2\pi$. Equations are discretized on a grid with $N \times N \times N = 512 \times 512 \times 512$ grid points. The simulation starts from a flow field obtained from a precursor simulation of homogeneous isotropic turbulence at $Re_\lambda = 95$. For the phase-field, a spherical drop with radius $R = 1.8$ is initialized in the center of the domain; the resulting volume fraction is $\Phi = V_d/(V_d + V_c) = 9.8\%$ (where V_d is the drop volume and V_c the continuous phase volume). After an initial transient, the turbulent flow starts to deform and break the drop; the generated drops, in turn, start to break and interact with the surrounding drops until a new equilibrium situation is reached in which coalescence and breakage events alternate over time.

Fig. 10*a* shows a qualitative rendering of the configuration attained for $t = 5$. The interface of the drops (iso-contour at $\phi = 0.5$) is shown in white. 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. In addition, we can also observe the formation of thin and elongated liquid threads. To quantify the number of small and large drops, we compute the drop size distribution. Results are shown in Fig. 10*b* with red circles. The drop diameter is normalized using the Kolmogorov-Hinze scale, which identifies the critical diameter below

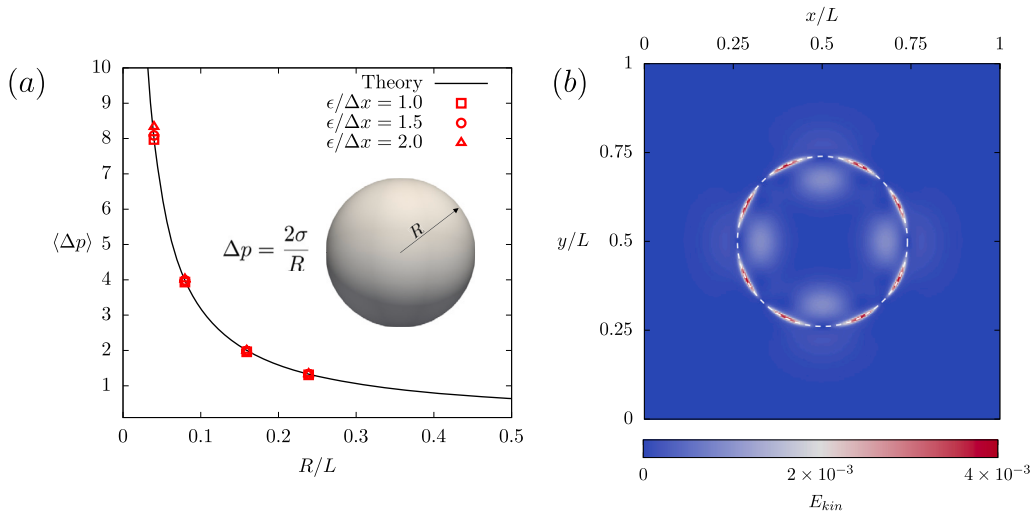

Fig. 9. Panel *a* shows the comparison between the numerical and theoretical pressure jump for a spherical drop immersed in a quiescent liquid. Different values of $\epsilon/\Delta x$ and drops radii R/L have been tested. Panel *b* shows a contour map of the spurious currents generated by the surface tension forces. The white dashed line represents the iso-contour line of $\phi = 0.5$.

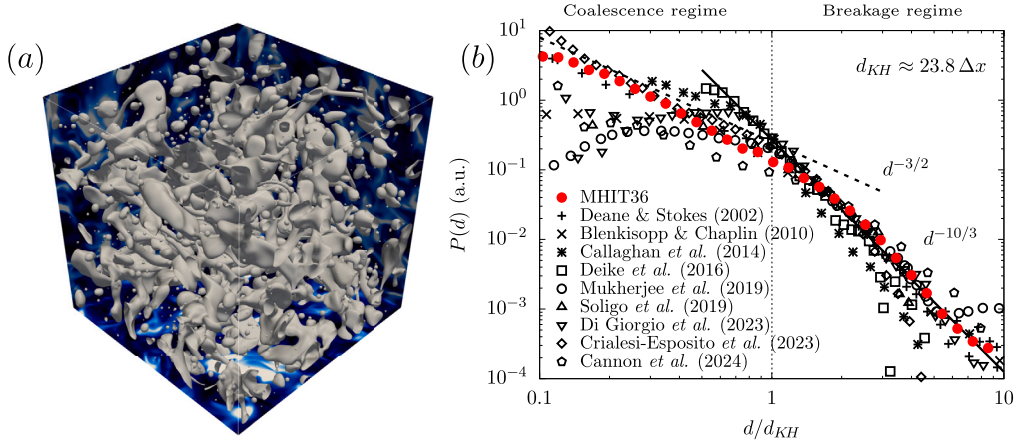

Fig. 10. Panel *a* shows a rendering of a swarm of drops in homogeneous isotropic turbulence. The interfaces of the drops is shown in white while colormaps of the turbulent kinetic energy (blue-low; white-high) are shown in the background. Panel *b* shows the resulting drop size distribution (red circles). Archival data obtained from previous experiments [57–59] and simulations [60–65] 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.

which a drop/bubble will not undergo breakage according to the KH framework [66,67], while the distributions are reported in arbitrary units for the sake of comparison with reference data from the literature. The Kolmogorov-Hinze (KH) scale has been evaluated as

$$d_{KH} = 0.725\langle\varepsilon\rangle^{-2/5}\left(\frac{\sigma}{\rho}\right)^{3/5}, \tag{21}$$

where ε is the average viscous dissipation, σ denotes the surface tension, ρ is the density of the carrier fluid, and the prefactor 0.725 is set in accordance with the original work of Hinze [67]. In Fig. 10*b*, the analytical scaling laws [68], $d^{-3/2}$ 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) [57], Blenkinsopp & Chaplin [58] and Callaghan et al. 2014 [59]; ii) simulations of Deike et al. (2016) [60], Mukherjee et al. (2019) [61], Soligo et al. (2019) [62], Di Giorgio et al. (2023) [63], Crialesi-Esposito et al. (2023) [64] and Cannon et al. (2024) [65]. We can observe that the present results (red) are in very good agreement with the theoretical scaling laws as well as with previous experimental and numerical results for both the coalescence- and breakage-dominated regimes, i.e., for both sub- and super-Kolmogorov-Hinze scale drops. The agreement extends for an entire decade of drop diameters for each regime and overall the agreement extends for two decades, i.e., from $d/d_{KH} = 0.1$ up to $d/d_{KH} = 10$. In terms of grid resolutions, the smallest drop scale reported in the plot ($d/d_{KH} = 0.1$) is described with at least three grid points while a drop with a size equal to the KH scale is described with about 24 grid points. Compared to previous studies based on the phase-field method and in particular based on the Cahn-Hilliard equation [62], the ACDI formulation here employed ensures a better conservation of small drops and thin ligaments characterized by $d/d_{KH} < 0.3$. Overall, present results confirm the capabilities of the chosen interface capturing method – the phase-field method – in accurately describing the behavior of both sub- and super-Kolmogorov-Hinze drops and matching theoretical scaling laws as well as previous

experimental and numerical studies (based on different interface capturing schemes).

6. Conclusions

We detail the characteristics and features of MHIT36, a finite-difference code tailored towards large-scale simulations of multiphase flows on GPU-based computing infrastructures. The code employs direct numerical simulation of the Navier-Stokes equations coupled with a phase-field method (ACDI formulation [21]) to describe interface shape and its topological changes (e.g., breakage and coalescence).

The governing equations are discretized in space using a second-order finite difference method. Time integration of the Navier–Stokes equations is performed via a fractional step method. First, a tentative velocity field is computed using an Adams–Bashforth scheme. Next, a Poisson equation for pressure is solved using an FFT-based approach. The resulting pressure field is then used to correct the velocity and project the intermediate solution onto the divergence-free space, yielding the velocity field at the new time step. The phase-field method – based on a modified version of the conservative Allen-Cahn equation (ACDI method) – is time advanced using an explicit integration scheme. The present methodology has been validated using benchmarks for single-phase and multiphase flows. Further validation of the code capabilities has been performed by simulating drops breakage and coalescence in homogeneous isotropic turbulence. The resulting drop size distribution shows excellent agreement with theoretical scaling laws for both sub- and super-Kolmogorov Hinze scale drops and matches well with previous numerical and experimental studies, thereby confirming code capabilities in accurately describing both regimes.

The numerical method is implemented in a Fortran code with full support for multi-GPU execution. The parallelization strategy is based on a two-dimensional domain decomposition. All inter- and intra-node communication operations (e.g., pencil transpositions and halo exchanges) are handled by the cuDecomp library [34], which ensures excellent scalability, achieving up to 95% efficiency on 1024 GPUs. The remainder of the solver is offloaded to GPUs using OpenACC directives, along with the use of managed memory. Future developments will target the simulation of systems with high dispersed volume fractions (e.g., emulsions), including the integration of modules for surfactant transport [69] and the implementation of multi-marker methods [70,71].

CRediT authorship contribution statement

Alessio Roccon: Writing – original draft, Visualization, Validation, Software, Methodology, Data curation, Conceptualization. **Lea Enzenberger:** Writing – review & editing, Visualization, Validation, Software, Data curation. **Domenico Zaza:** Writing – review & editing, Visualization, Validation, Software, Methodology, Data curation. **Alfredo Soldati:** Writing – review & editing, 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

The authors gratefully acknowledge the Austrian Science Fund (FWF) [Modeling And sImulation of emulsiOns (MAIO), https://doi.org/10.55776/PAT9292123], and the Italian Ministry for Research [The fluid dynamics of interfaces: mesoscale models for bubbles, droplets, and membranes and their coupling to large scale flows, PRIN Project No. 2022R9B2MW_PE8]. We acknowledge the EuroHPC Joint Undertaking for awarding access to the supercomputer Leonardo (project EHPC-REG-2024R01-063) and Marenostrum5 (project EHPC-BEN-2025B04-078). This work was completed in part at the CINECA Open Hackathon, part of the Open Hackathons program and the authors would like to thank Matt Bettencourt, Josh Romero, Laura Bellentani and Alessio Piccolo.

Data availability

The data presented in this study are openly available in the following Figshare repository: https://doi.org/10.6084/m9.figshare.28082252.

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, ASME 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] A.J. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comput. 19 (1968) 745–762.
[15] R. Temam, Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (ii), Arch. Ration. Mech. Anal. 33 (1969) 377–385.
[16] J. Romero, P. Costa, M. Fatica, Distributed-memory simulations of turbulent flows on modern GPU systems using an adaptive pencil decomposition library, in: PASC Proceedings, 2022, pp. 1–11.
[17] Nvidia, cuFFT library, http://docs.nvidia.com/cuda/cufft.
[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] A. Libin, G. Sivashinsky, Long wavelength instability of the ABC-flows, Q. Appl. Math. 48 (4) (1990) 611–623.
[20] O. Podvigina, A. Pouquet, On the non-linear stability of the 1: 1: 1 ABC flow, Phys. D: Nonlinear Phenom. 75 (4) (1994) 471–508.
[21] S.S. Jain, Accurate conservative phase-field method for simulation of two-phase flows, J. Comput. Phys. 469 (2022) 111529.
[22] S. Mirjalili, S.S. Jain, M. Dodd, Interface-capturing methods for two-phase flows: an overview and recent developments, Cent. Turbul. Res. Annu. Res. Briefs 2017 (117–135) (2017) 13.
[23] P.-H. Chiu, Y.-T. Lin, A conservative phase field method for solving incompressible two-phase flows, J. Comput. Phys. 230 (1) (2011) 185–204.
[24] S. Mirjalili, C.B. Ivey, A. Mani, A conservative diffuse interface method for two-phase flows with provable boundedness properties, J. Comput. Phys. 401 (2020) 109006.
[25] S.S. Jain, A. Mani, P. Moin, A conservative diffuse-interface method for compressible two-phase flows, J. Comput. Phys. 418 (2020) 109606.
[26] S. Mirjalili, A. Mani, Consistent, energy-conserving momentum transport for simulations of two-phase flows using the phase field equations, J. Comput. Phys. 426 (2021) 109918.
[27] H.-R. Liu, C.S. Ng, K.L. Chong, D. Lohse, R. Verzicco, An efficient phase-field method for turbulent multiphase flows, J. Comput. Phys. 446 (2021) 110659.
[28] 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.
[29] M. Crialesi-Esposito, N. Scapin, A.D. Demou, M.E. Rosti, P. Costa, F. Spiga, L. Brandt, FluTAS: a GPU-accelerated finite difference code for multiphase flows, Comput. Phys. Commun. 284 (2023) 108602.

[30] I. Cannon, S. Olivieri, M.E. Rosti, Spheres and fibers in turbulent flows at various Reynolds numbers, Phys. Rev. Fluids 9 (6) (2024) 064301.
[31] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier-Stokes equations, J. Comput. Phys. 59 (2) (1985) 308–323.
[32] U. Schumann, R.A. Sweet, Fast Fourier transforms for direct solution of Poisson equation with staggered boundary conditions, J. Comput. Phys. 75 (1) (1988) 123–137.
[33] E.P. Van Der Poel, R. Ostilla-Mónico, J. Donners, R. Verzicco, A pencil distributed finite difference code for strongly turbulent wall-bounded flows, Comput. Fluids 116 (2015) 10–16.
[34] R.D. Sanhueza, J. Peeters, P. Costa, A pencil-distributed finite-difference solver for extreme-scale calculations of turbulent wall flows at high Reynolds number, arXiv preprint arXiv:2502.06296, 2025.
[35] M. Turisini, G. Amati, M. Cestari, LEONARDO: a pan-European pre-exascale supercomputer for HPC and AI applications, J. Large-Scale Res. Facil. 9 (2023).
[36] F. Banchelli, M. Garcia-Gasulla, F. Mantovani, J. Vinyals, J. Pocurull, D. Vicente, B. Eguzkitza, F.C. Galeazzo, M.C. Acosta, S. Girona, Introducing MareNostrum5: a European pre-exascale energy-efficient system designed to serve a broad spectrum of scientific workloads, arXiv preprint arXiv:2503.09917, 2025.
[37] J. Jiménez, Computing high-Reynolds-number turbulence: will simulations ever replace experiments?, J. Turbul. 4 (1) (2003) 022.
[38] P. Ouro, U. Lopez-Novoa, M.F. Guest, On the performance of a highly-scalable computational fluid dynamics code on AMD, ARM and intel processor-based HPC systems, Comput. Phys. Commun. 269 (2021) 108105.
[39] M. Xiao, A. Ceci, P. Costa, J. Larsson, S. Pirozzoli, CaLES: a GPU-accelerated solver for large-eddy simulation of wall-bounded flows, Comput. Phys. Commun. (2025) 109546.
[40] X. Zhu, E. Phillips, V. Spandan, J. Donners, G. Ruetsch, J. Romero, R. Ostilla-Mónico, Y. Yang, D. Lohse, R. Verzicco, et al., AFiD-GPU: a versatile Navier–Stokes solver for wall-bounded turbulent flows on GPU clusters, Comput. Phys. Commun. 229 (2018) 199–210.
[41] P. Yeung, K. Ravikumar, S. Nichols, R. Uma-Vaideswaran, GPU-enabled extreme-scale turbulence simulations: Fourier pseudo-spectral algorithms at the exascale using OpenMP offloading, Comput. Phys. Commun. 306 (2025) 109364.
[42] M. Lee, N. Malaya, R.D. Moser, Petascale direct numerical simulation of turbulent channel flow on up to 786k cores, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '13, 2013, 61.
[43] S. Ha, J. Park, D. You, A multi-GPU method for ADI-based fractional-step integration of incompressible Navier-Stokes equations, Comput. Phys. Commun. 265 (2021) 107999.
[44] W.M. Van Rees, A. Leonard, D.I. Pullin, P. Koumoutsakos, A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high Reynolds numbers, J. Comput. Phys. 230 (8) (2011) 2794–2805.
[45] M. Crialesi-Esposito, G. Boffetta, L. Brandt, S. Chibbaro, S. Musacchio, Intermittency in turbulent emulsions, J. Fluid Mech. 972 (2023) A37.
[46] J. Jiménez, A.A. Wray, P.G. Saffman, R.S. Rogallo, The structure of intense vorticity in isotropic turbulence, J. Fluid Mech. 255 (1993) 65–90.
[47] T. Ishihara, Y. Kaneda, High resolution DNS of incompressible homogeneous forced turbulence—time dependence of the statistics, in: Statistical Theories and Computational Approaches to Turbulence: Modern Perspectives and Applications to Global-Scale Flows, Springer, 2003, pp. 177–188.
[48] S.G. Chumakov, A priori study of subgrid-scale flux of a passive scalar in isotropic homogeneous turbulence, Phys. Rev. E 78 (3) (2008) 036313.
[49] K.R. Sreenivasan, On the universality of the Kolmogorov constant, Phys. Fluids 7 (11) (1995) 2778–2784.
[50] P.K. Yeung, Y. Zhou, Universality of the Kolmogorov constant in numerical simulations of turbulence, Phys. Rev. E 56 (2) (1997) 1746.
[51] M. Crialesi-Esposito, M.E. Rosti, S. Chibbaro, L. Brandt, Modulation of homogeneous and isotropic turbulence in emulsions, J. Fluid Mech. 940 (2022) A19.
[52] K. Singh, A. Komrakova, Comparison of forcing schemes to sustain homogeneous isotropic turbulence, Phys. Fluids 36 (1) (2024).
[53] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (1) (2002) 83–116.
[54] S. Pirozzoli, S. Di Giorgio, D. Rossi, Efficient implementation of the Allen-Cahn phase-field method for material interface tracking, Comput. Fluids 301 (1) (2025) 106768.
[55] S. Popinet, Numerical models of surface tension, Annu. Rev. Fluid Mech. 50 (2018) 1–28.
[56] D. Rossi, S. Di Giorgio, S. Pirozzoli, Comparative analysis of volume of fluid and phase–field methods for numerical simulations of two-phase flows, Int. J. Multiph. Flow (2025) 105245.
[57] G. Deane, M. Stokes, Scale dependence of bubble creation mechanisms in breaking waves, Nature 418 (6900) (2002) 839.
[58] 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.
[59] 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.
[60] L. Deike, W. Melville, S. Popinet, Air entrainment and bubble statistics in breaking waves, J. Fluid Mech. 801 (2016) 91–129.
[61] 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.
[62] 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.
[63] S. Di Giorgio, S. Pirozzoli, A. Iafrati, On coherent vortical structures in wave breaking, J. Fluid Mech. 947 (2022) A44.
[64] M. Crialesi-Esposito, S. Chibbaro, L. Brandt, The interaction of droplet dynamics and turbulence cascade, Commun. Phys. 6 (1) (2023) 5.
[65] I. Cannon, G. Soligo, M. Rosti, Morphology of clean and surfactant-laden droplets in homogeneous isotropic turbulence, J. Fluid Mech. 987 (2024) A31.
[66] 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.
[67] J.O. Hinze, Fundamentals of the hydrodynamic mechanism of splitting in dispersion processes, AIChE J. 1 (3) (1955) 289–295.
[68] 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.
[69] S.S. Jain, A model for transport of interface-confined scalars and insoluble surfactants in two-phase flows, J. Comput. Phys. 515 (2024) 113277.
[70] P. Karnakov, S. Litvinov, P. Koumoutsakos, Computing foaming flows across scales: from breaking waves to microfluidics, Sci. Adv. 8 (5) (2022) eabm0590.
[71] K. Sugihara, N. Onodera, Y. Sitompul, Y. Idomura, S. Yamashita, Bubble Flow Analysis Using Multi-Phase Field Method, EPJ Web Conf., vol. 302, EDP Sciences, 2024, p. 03002.