CaNS-Fizzy: A GPU-accelerated finite difference solver for turbulent two-phase flows

Giandomenico Lupo [1*¶], **Peter Wellens** [2], **and Pedro Costa** [1*]

1 Delft University of Technology, Department of Process & Energy, Delft, The Netherlands **2** Delft University of Technology, Department of Marine & Transport Technology, Delft, The Netherlands ¶ Corresponding author * These authors contributed equally.

Summary

CaNS-Fizzy – Fizzy for short – is a GPU-accelerated numerical solver for massively-parallel Direct Numerical Simulations (DNS) of incompressible two-phase flows. A DNS enables direct access to all flow quantities, resolved in time and space at all relevant continuum scales. The resulting numerical experiments provide complete data sets for the analysis of the detailed mechanisms underlying the flow, particularly the interaction between the chaotic and multi-scale dynamics of turbulence and the interface movement and deformation. The insights gained can guide the design and operation of various applications, such as boiling heat transfer, liquid-liquid extraction, gas-liquid reactors, absorption and stripping columns, distillation columns, liquid combustion appliances, in all of which the rate of heat and mass transfer between phases is proportional to the interfacial area. Fizzy's two-phase capabilities were implemented using the efficient, GPU-accelerated Navier-Stokes solver CaNS (Costa, 2018) as a base.

Statement of need

Fizzy is suited for large-scale direct numerical simulations of canonical incompressible two-phase flows, from simple laminar cases to bubble/droplet-laden turbulent suspensions. These flows may be computationally expensive due to the stringent resolution requirements imposed by the direct solution of immersed interfaces dispersed throughout the domain. This demands efficient use of the capabilities of modern computing systems; Fizzy has been developed to include key desirable features that enable this objective: a one-fluid formulation of the two-phase flow governing equations, the use of a fast direct solver for the pressure Poisson equation, and an efficient distributed GPU porting with an interface capturing strategy that is suitable for GPU acceleration. In addition to the momentum transfer and interface capturing, the code has the capability to solve heat transfer in both fluid phases, and thermal convection based on the Oberbeck-Boussinesq approximation. Finally, the code has been extensively validated with several benchmark cases that demonstrate the different features of the solver, which are incorporated in the continuous integration workflows of the repository. The GPU capabilities differentiate Fizzy from other commonly used open-source state-of-the-art incompressible two-phase flow solvers such as boilingFoam (Municchi et al., 2024) and Basilisk (Popinet, 2009, 2015), which are more suited to smaller-scale direct numerical simulations in complex geometries. The recently published FLOW36 (Roccon et al., 2025) is another GPU-ready code with similar features to Fizzy, differing in the interface capturing scheme and in the use of a pseudo-spectral instead of a finite difference approach.

Mathematical model

A one-fluid formulation of the two-phase flow is employed, solving a single set of governing equations for both phases in the whole domain. The incompressible Navier-Stokes equation, the heat transport equation and the phase indicator transport equation are evolved in time to compute the velocity and pressure, temperature and phase indicator fields respectively. The latter identifies the regions of the domain occupied by either phase: it is continuous and smooth over the whole domain, and the interface between phases is diffuse. The thermophysical and transport properties (density, viscosity, thermal conductivity, specific heat capacity) are linearly mapped over the phase indicator field, and thus also continuous and smooth. The surface tension at the interface is included as a Continuous Surface Force (CSF) (Brackbill et al., 1992) in the Navier-Stokes equation.

Methods and Implementation strategy

The governing equations are spatially discretized with a second-order finite difference scheme on a 3D Cartesian grid; a staggered grid arrangement is used for the velocity field, while all other quantities are stored at the cell centers; time integration is based on a low-storage three-step Runge-Kutta scheme. The incompressible Navier-Stokes equation is solved with a pressure correction scheme to enforce mass conservation, which yields a variable coefficient Poisson equation for the pressure correction: a splitting technique adapted from (Dong & Shen, 2012) and (Dodd & Ferrante, 2014) transforms this equation into a constant coefficient Poisson equation (Frantzis & Grigoriadis, 2019), enabling the use of the fast direct FFT solver of the CaNS code. Fizzy also allows for solving the conventional variable-coefficients problem using a geometric multigrid method through the Hypre library. The interface capturing scheme for the phase indicator transport equation can be chosen between the Accurate Conservative Diffuse Interface (ACDI) scheme (Jain, 2022) and a tailored flavour of the THINC algebraic Volume-of-Fluid method (Xie & Xiao, 2017). Both methods share a diffuse interface representation of the phase interface that allows for continuous and smooth mapping of the physical fields across the interface. The ACDI method requires no explicit interface reconstruction thanks to an interface regularization flux; for the VoF method the interface geometry in each cell is simplified to allow analytic calculation of the interface-cell intersection and of the advection fluxes at cell faces: thus in both methods the computational load is kept constant regardless of the local interface topology, making the algorithm particularly suited for parallelization on GPU architecture. For the ACDI method, the momentum equation includes the flux associated with the diffuse interface regularization, which allows for a mass–momentum consistent discretization and enables stability at high density contrasts between phases. The heat equation similarly includes the enthalpy flux associated with the interface regularization.

The code is written in modern Fortran, and is parallelized using MPI and OpenACC directives for GPU kernel offloading and host/device data transfer. As in CaNS, Fizzy leverages cuDecomp (Romero et al., 2022) for distributed memory calculations in pencil domain decompositions, and cuFFT for computing Fourier transforms. These libraries are designed to work on NVIDIA GPUs; in the future Fizzy will support other GPU hardware, following updates on CaNS. On CPUs, the code uses 2DECOMP&FFT (Rolfo et al., 2023) and FFTW to perform the same operations.

Users can design and run a simulation by specifying the physical and computational parameters in a simple Fortran namelist input file. The code uses a modular, procedural design which makes extensions with different numerical methods or physical phenomena easy to develop. In the short term, we aim to allow for alternative schemes for spatial and temporal discretization, and introduce different interface capturing schemes, e.g. geometric VoF.

Finally, the code was designed so that important new computational features in the parent solver CaNS (e.g. porting efforts to other architectures) are easily propagated to Fizzy.
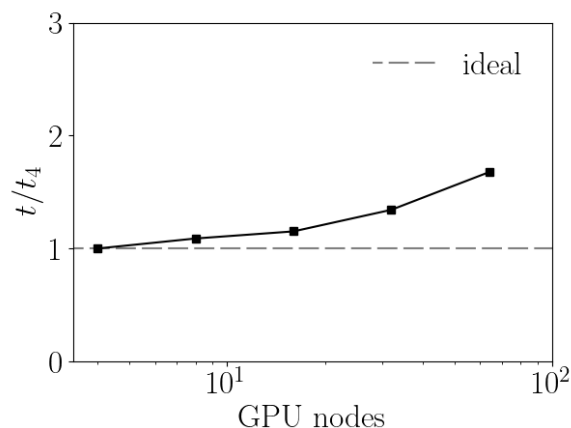
Examples

Figure 1 illustrates examples of two-phase flows simulated with this solver. The left panel shows how forced homogeneous isotropic turbulence breaks and deforms the interface of a liquid-liquid emulsion in a triperiodic domain; the right panel shows a hot gas bubble rising in a cold liquid, attaining the typical skirt shape while cooling down and simultaneously heating up the surrounding liquid in its wake.
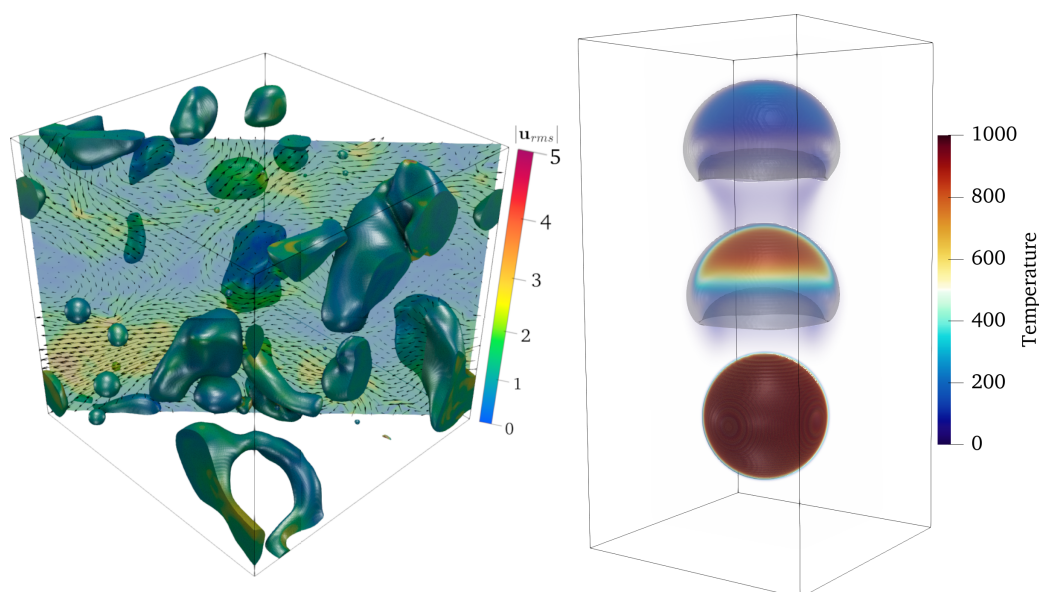
Figure 1: (Left) Simulation of a liquid-liquid emulsion in a three-periodic domain with sustained homogeneous isotropic turbulence. The colour represents the velocity magnitude. A diagonal plane with in-plane velocity vectors is shown. (Right) Three successive snapshots of a hot gas bubble rising in a cold liquid. A coloured volume rendering of the temperature field is shown both in the gas bubble and in the liquid wake.

Computational performance

Fizzy is tailored for large-scale simulations that exploit the computational capacity of modern GPU clusters with full GPU occupancy. The most relevant metric of the parallel efficiency for such scenario is a weak scaling test that determines the penalty in increased wall-clock time occurring when the problem size is increased alongside the computational resources. The liquid-liquid emulsion in homogeneous isotropic turbulence case of Figure 1 (Left) has been used for this test: the size of the computational domain has been extended in one direction linearly with the number of GPU nodes employed. The test has been carried out on the GPU partition of the supercomputer Leonardo from Cineca, Italy; each computing node is equipped with four NVIDIA A100 SXM6 64GB GPUs, and is able to fit at full memory a 1024^3 ($\sim$ 1 billion grid cells) computational box. Figure 2 shows the performance penalty for the ACDI method, as the problem domain size (i.e. the number of spatial degrees of freedom) is increased from occupying 4 nodes (16 GPUs) to 64 nodes (256 GPUs): the 16 times larger computation takes only about 1.7 times longer than the original 4-node computation. A similar performance is obtained using the algebraic VoF method. The key contributor to the parallel performance is the interface capturing approach which prevents thread divergence in GPU kernels, as the computational load is independent of the local interface morphology. Indeed, very little sensitivity of the wall-clock time per iteration to the amount of interface area is observed, even for unsteady evolution of the interface with drastic topology changes during break-up events.

Lupo et al. (2025). CaNS-Fizzy: A GPU-accelerated finite difference solver for turbulent two-phase flows. *Journal of Open Source Software*, *10*(112), 8076. https://doi.org/10.21105/joss.08076.

Figure 2: Weak scaling performance on GPU nodes at full memory. The vertical axis shows the wall-clock time normalized by the four-node case.

Acknowledgements

References

Brackbill, J. U., Kothe, D. B., & Zemach, C. (1992). A continuum method for modeling surface tension. *Journal of Computational Physics*, *100*(2), 335–354. https://doi.org/10.1016/0021-9991(92)90240-Y

Costa, P. (2018). A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows. *Computers & Mathematics with Applications*, *76*(8), 1853–1862. https://doi.org/10.1016/j.camwa.2018.07.034

Dodd, M. S., & Ferrante, A. (2014). A fast pressure-correction method for incompressible two-fluid flows. *Journal of Computational Physics*, *273*, 416–434. https://doi.org/10.1016/j.jcp.2014.05.024

Dong, S., & Shen, J. (2012). A time-stepping scheme involving constant coefficient matrices for phase-field simulations of two-phase incompressible flows with large density ratios. *Journal of Computational Physics*, *231*(17), 5788–5804. https://doi.org/10.1016/j.jcp.2012.04.041

Frantzis, C., & Grigoriadis, D. G. E. (2019). An efficient method for two-fluid incompressible flows appropriate for the immersed boundary method. *Journal of Computational Physics*, *376*, 28–53. https://doi.org/10.1016/j.jcp.2018.09.035

Jain, S. S. (2022). Accurate conservative phase-field method for simulation of two-phase flows. *Journal of Computational Physics*, *469*, 111529. https://doi.org/10.1016/j.jcp.2022.111529

Municchi, F., Markides, C. N., Matar, O. K., & Magnini, M. (2024). Computational study of bubble, thin-film dynamics and heat transfer during flow boiling in non-circular microchannels. *Applied Thermal Engineering*, *238*, 122039. https://doi.org/10.1016/j.applthermaleng.2023.122039

Popinet, S. (2009). An accurate adaptive solver for surface-tension-driven interfacial flows.

Journal of Computational Physics, *228*(16), 5838–5866. https://doi.org/10.1016/j.jcp.2009.04.042

Popinet, S. (2015). A quadtree-adaptive multigrid solver for the serre–green–naghdi equations. *Journal of Computational Physics*, *302*, 336–358. https://doi.org/10.1016/j.jcp.2015.09.009

Roccon, A., Soligo, G., & Soldati, A. (2025). FLOW36: A spectral solver for phase-field based multiphase turbulence simulations on heterogeneous computing architectures. *Computer Physics Communications*, *313*, 109640. https://doi.org/10.1016/j.cpc.2025.109640

Rolfo, S., Flageul, C., Bartholomew, P., Spiga, F., & Laizet, S. (2023). The 2DECOMP&FFT library: An update with new CPU/GPU capabilities. *Journal of Open Source Software*, *8*(91), 5813. https://doi.org/10.21105/joss.05813

Romero, J., Costa, P., & Fatica, M. (2022). Distributed-memory simulations of turbulent flows on modern GPU systems using an adaptive pencil decomposition library. *Proceedings of the Platform for Advanced Scientific Computing Conference*. https://doi.org/10.1145/3539781.3539797

Xie, B., & Xiao, F. (2017). Toward efficient and accurate interface capturing on arbitrary hybrid unstructured grids: The THINC method with quadratic surface representation and gaussian quadrature. *Journal of Computational Physics*, *349*, 415–440. https://doi.org/10.1016/j.jcp.2017.08.028