ISSN No:-2456-2165

Algorithm for Droplet Motion around Weber Number with Fortran

Masood Hemati, Nikolay Alekseevich Zabelin

Abstract:- We provide a scalable GPU code for droplet motion in biphasic streams. This code solves the Navier-Stokes incompressibility equation for two-fluid systems with a direct Poisson FFT solver based on the pressure equation. The interface between the two fluids is shown by the Volume Volume (VoF) method, which is suitable for complex flows due to its capacity to manage topological changes. The energy equation is explicitly solved and coupled with the momentum equation via the Bosinsk approximation. This code is modularly designed to be able to use different numerical methods independently, modify existing procedures, and combine new ones simply and consistently. FluTAS is written in Fortran and uses the MPI / OpenMP combination in the CPU-only version in parallel, accelerating GPU execution with OpenACC instructions. The two dominant forces affecting droplet fracture, drag force, and surface tensile force are confirmed using two benchmarks: pressure distribution on a cylindrical surface in uniform flow and oscillation of a square drop under surface tensile force. The results show that the failure process occurs in two steps. During the first stage, the droplets are stretched and thinned perpendicular to the direction of fluid flow. In the second stage, isolated points appear on the surface of the droplets which are attributed to the unstable growth of surface waves. The topology of the droplet after failure depends on the value of the Weber number: the larger the Weber number, the more isolated points on the surface of the droplets.

Keywords:- *Two-phase flows, Fluid volume method, Turbulence in multiphase flows, High-performance computing, OpenACC instructions.*

I. INTRODUCTION

Multiphase currents exist everywhere in many fields, from environmental currents to industrial applications. The interaction between phases plays a prominent role in the formation and evolution of clouds [2], sediment transport [3, 4], ocean sprays, and bubble production [5], and genes are one of the major challenges of ambient fluid mechanics [6]. These currents are also very important in several industrial applications, such as pharmacy, transportation, food processing, and electricity generation [7]. From a theoretical point of view, the main problem in analyzing multiphase flows depends on the nature of their multiscale, because the longitudinal scale of the interface is the order of the path without the mean, while in most applications the normal longitudinal scale is several times Of a larger size (-10 105-106).

In the case of multiphase perturbation, where most of our interests and applications are, both experimental research and numerical simulations have been widely used over the past thirty years and have led to significant contributions to a variety of problems and configurations: Case in point, filled particles. Flows and sediment transport are discussed in [8, 4], bubble and droplet flow are studied in [9, 10, 11], and oceanic sprays and bubbles as specified in [5]. However, as previously discussed in [10] and despite recent advances in instantaneous bubble/drop shape measurements [12, 13, 14], there is still a lack of experimental data for measuring the instantaneous velocity fields of both carriers and scattered phases as well as for turbulent kinetic energy and dissipation near the interface locations. These constraints disappear when dealing with numerical simulations, and so, in recent decades, solved simulations of multiphase flow interfaces have become a central investigative tool. Despite the advantages, numerical simulations continue to be limited to simple configurations and medium-scale separations, making it difficult to choose the appropriate method for the complete solution of the two-phase interface. As discussed in [15], there is now a consensus that appropriate numerical methods for performing simulations solved with a multiphase flow interface should have the following characteristics:

- capable of applying mass, momentum, and kinetic energy survival at a discrete surface;
- mismatch in material properties, the magnitude of which depends on the application, and
- Manage complex and possibly arbitrary topological changes.

Among the four groups of numerical methods for multiphase currents, front tracking (FT) [16], fluid volume (VoF) [17], phase-field (PFM) [18], surface set (LS) [19], the minimum There is a type of each that has the above numerical properties and gives researchers and scientists the freedom to choose their preferred numerical tool (see [17, 20, 21] for review).

However, it becomes clearer that another desirable feature of any numerical method is its simple adaptation to allow the implementation of mass parallel simulations, especially in accelerated architectures.

As the computing power of GPUs increases, several HPCs have now shifted to GPU-only and GPU accelerated architectures. This process of parallelization of GPU numeric codes makes it a mandatory requirement for fluid mechanics rather than a simple advantage. This attempt has already been made for single-phase code, where at least three open-source codes for incompressible and fully compressed simulations can be run on acceleration architectures: AFiD [23], STREAMS [24], and the accelerated version of CanS [25]. Conversely, in the multiphase counterpart, despite the high availability of CPU-based open source code, PARIS Simulator [26],

ISSN No:-2456-2165

TBFsolver [27], FS3D [28], NGA2 [29], Basilisk [30], and MFC [31] to name a few, limited effort has been devoted to adapting them to hybrid architectures.

In this work, we aim to fill this gap and provide FluTAS (Fluid Accelerator Solver), a code for massive direct numerical simulations in multi-GPU and multi-CPU architectures that target incompressible multiphase streams, optionally with Heat transfer. The numerical solution of these flows is usually performed using finite difference methods in ambiguous variable arrangement and involves solving the Poisson equation to apply constraints on velocity divergence. In this context, FluTAS uses the Navier-Stokes CanS solver [32] and its GPU extension [25] as a basis, whose key feature is a general implementation that includes all possible homogeneous pressure boundary conditions that can be used FFT-based elliptical solvers [33]. The version in our group has been validated in [34] and is widely used in various types of multiphase configurations, both for laminar flows [35, 36, 37] and for turbulent flows [38, 39, 40, 41] Used. Note that it extends to phase shift currents [42] as well as to handle low-density (almost Mach Mach) multi-phase currents [43, 44].

This paper is organized as follows. In §2, we introduce the governing equations for an incompressible two-fluid system. Details of the VoF discretization method, the energy equation, and the Navier-Stokes solver are provided in §3, while the standard criteria for code validation are discussed in §4. Then, parallelization for GPU acceleration is provided along with scaling tests in §5 and §6. The code potentials are shown in two requested simulations of multiphase turbulence: emulsions in homogeneous isotropic turbulence (HIT) and two-phase thermal convection (see § 7).

II. GOVERNING EQUATIONS

We consider a two-phase system of immiscible incompressible Newtonian fluids (e.g., a gas-liquid system). The two phases are bounded by an extremely small interface through which momentum and energy can be transmitted. To describe the system, we define a phase H indicator function that distinguishes two phases at position x and time t:

$$H(x,t) = \{1 \;\; if\; x\epsilon\Omega_1 \; 0 \;\; if\; x\epsilon\Omega_2 \qquad (1)$$

Where Ω_1 and Ω_2 are domains corresponding to phases 1 and 2. We can use H to define the thermophysical properties in the whole field $\Omega = \Omega_1 \cup \Omega_2$ as follows:

$$\xi(\mathbf{x},t) = \xi_1 H(\mathbf{x},t) + \xi_2(1 - H(\mathbf{x},t)), \qquad (2)$$

Where ξi (i = 1,2) can be mass density ρi, dynamic viscosity μi, thermal conductivity ki, or specific heat capacity at a constant pressure cp, i. Henceforth, unless stated otherwise, thermophysical quantities that do not specifically refer to one of the phases are defined by the equation. (2). The evolution of the indicator function is controlled by the following topological equation:

$$\frac{\partial H}{\partial t} + \nabla \cdot (\mathbf{u}_\Gamma H) = H \nabla \cdot \mathbf{u}_\Gamma \qquad (3)$$

Where u_Γ is the interface speed. If the phase does not change, the velocity of a fluid u is continuous throughout the interface and therefore it can be used as the interface velocity in relation (3).

The equations governing the motion and energy transfer for the liquid and gas phases are coupled through suitable surface conditions [45], which are reported below in a formula called fluid or whole amplitude, in which each transfer equation, defined in Ω, is coupled [20].

$$\nabla \cdot \mathbf{u} = 0, \qquad (4)$$

$$\rho \left[\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) \right] = -\nabla p + \nabla \cdot \left[\mu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T \right) \right] + \sigma \kappa \delta_\Gamma + \hat{\rho}\mathbf{g}, \qquad (5)$$

$$\rho c_p \left[\frac{\partial T}{\partial t} + \nabla \cdot (\mathbf{u}T) \right] = \nabla \cdot (k \nabla T) \qquad (6)$$

Here, u is the velocity of the fluid, which is assumed to be continuous in Ω, p is the hydrodynamic pressure, and T is the temperature. In relation (5), σ is the surface tension, κ is the localized surface curvature and δ_Γ is the Dirac delta function, g is the gravitational acceleration and ˆρ is the volume density field that has been changed to explain the thermal effects of gravitational forces. Using the Oberbeck-Boussinesq approximation, ˆρ is pronounced as:

$$\hat{\rho} = \rho_{1,r}[1 - \beta_l (T - T_r)]H + \rho_{2,r}[1 - \beta_g (T - T_r)](1 - H), \qquad (7)$$

Where ρ_i = 1,2, r are the reference phase density and β_i = 1,2 are the thermal expansion coefficients of the liquid and gas.

A. *Numerical methodology*

The numerical solution of the governing equations (3), (4), (5), and (6) presented in Section 2 is shown on a regular Cartesian lattice with a constant distance Δx, Δy, and Δz along each direction. Marker and cell arrays are used for points of velocity and pressure [46], while all scalar fields are defined at cell centers. In each time step, the governing equations proceed in terms of time with $\Delta t^{n+1} = t^{n+1} - t^n$, where the previous time step is represented by $\Delta t^n = t^n - t^{n-1}$.

ISSN No:-2456-2165

In the following, we present the numerical discretization of the governing equations in the same order as their solution.

B. Fluid volume: MTHINC method

The first step of the time march algorithm involves reconstructing the interface and its subsequent advection. As mentioned earlier, these tasks are considered in a completely Eulerian framework using the Fluid Volume (VoF) method. Numerically, this first involves defining the volume fraction φ in each cell of the computational domain:

$$\phi = \frac{1}{V_c}\int_{V_c} H(\mathbf{x},t)dV_c \quad (8)$$

With $V_c = \Delta x\Delta y\Delta z$. Next, Equation (3) in terms of volume fraction is written as follows:

$$\frac{\partial \phi}{\partial t} + \nabla\cdot(H\mathbf{u}) = \phi\nabla\cdot\mathbf{u} \quad (9)$$

The distinctive feature of each category of the VoF method lies in the approximate method H. In this work, we use the fluid volume algebraic method based on the reconstruction of multidimensional tangential hyperbola, MTHINC [1], the main idea of which is to approximate H with a hyperbolic tangent:

$$H(\tilde{x},\tilde{y},\tilde{z}) = \frac{1}{2}\left[1+\tanh(\beta_{th}\left(\mathcal{T}(\tilde{\mathbf{x}})+d_{th}\right)\right], \quad (10)$$

where β_{th}, d_{th} are the sharpness and the normalization parameter, respectively, and (x˜, y˜, z˜) a local coordinate system $\mathbf{x}$˜ = [(x − 0.5)/Δx,(y − 0.5)/Δy,(z − 0.5)/Δz]. Employing equation (10) has two distinct advantages concerning a piecewise approximation, commonly employed in the geometric VoF methods. First, the phase indicator H can be approximated with a reconstructing polynomial T of arbitrary order straightforwardly. Then, when T is known, the resulting interface at the boundary of the two phases has a smooth but controlled thickness (with the βth parameter), which also allows the exact calculation of the normal vector n and the curvature tensor K directly from φ. Further details on the choice of T and the calculations of dth, n, and K can be found in the main article by Ii et al. [1], but for completeness, we add them with the details of the numerical implementation in Appendix A.

After the reconstruction phase, the interface is transferred using a directional splitting approach [47, 48], which involves evaluating the numerical fluxes sequentially in each direction using the latest VoF field estimate for each split. Accordingly, the three temporary fields φpi, j, k (with p = [x, y, z]) is first calculated:

(11)

$$\phi^p_{i,j,k} = \frac{\phi^s_{i,j,k} - \dfrac{1}{\Delta l^p}\left[f^p_+(\phi^s_{i,j,k}) - f^p_-(\phi^s_{i,j,k})\right]}{1-\dfrac{\Delta t^{n+1}}{\Delta l^p}\left(u^p_+ - u^p_-\right)^n},$$

$$s=[n,x,y],\ [\Delta l^x,\Delta l^y,\Delta l^z]=[\Delta x,\Delta y,\Delta z],\ [u^x,u^y,u^z]=[u,v,w$$

Where with ± p-th amine component of velocity. The calculation of the numerical fluxes f ± in Equation (11) is evaluated using the approximation of the hyperbolic tangent H as described in Appendix A. Then, the divergence correction step is applied to impose the survival of the volume of both phases at a separate level :

$$\phi^{n+1}_{i,j,k} = \phi^z_{i,j,k} - \sum_{p=x,y,z}\frac{\Delta t^{n+1}}{\Delta l^p}\phi^p_{i,j,k}(u^p_+ - u^p_-)^n \quad . \quad (12)$$

With the above approach, the survival of the offense is guaranteed until the conditions without divergence (4) are satisfied. Accordingly, if direct methods are used to solve the Poisson equation, the mass of each phase will be maintained until the machine is accurate. Another method with similar properties is presented in [49].In that case, however, the expansion of the term is explicitly considered at the denominator of Equation (11), whereas here they are used in an implicit strategy. This is at the cost of the final correction step provided by Equation (12), but with the advantage of not creating additional time step constraints (except for convection) in the color function advection.

C. Thermal effects

The next step in the time march algorithm involves advancing the temperature field using the explicit second-order Adams-Bashforth method:

$$T^{n+1} = T^n + \Delta t^{n+1}\left(f_{t,1}\mathcal{M}^n_T - f_{t,2}\mathcal{M}^{n-1}_T\right), \quad (13)$$

ISSN No:-2456-2165

Where ft, 1 = (1 + 0.5Δt^{n+1} / Δt^{n}) and ft, 2 = 0.5Δt^{n+1} / Δt^{n} are the Adams-Bashworth plan coefficients. In Equation (13), the MT operator calculates the share of advection and emission and is presented in the following semi-discrete form:

$$\mathcal{M}_T^n = -\nabla \cdot (\mathbf{u}^n T^n) + \frac{1}{\rho^{n+1} c_p^{n+1}} \nabla \cdot (k^{n+1} \nabla T^n) \quad . \qquad (14)$$

All spatial terms in Equation (14) are discrete with second-order central designs, except for the term temperature convection. The second discretization is based on the WENO5 order 5, as in reference [50].

D. Pressure correction algorithm

As the energy equation progresses, the momentum equation is solved by a second-order pressure correction [51], which is reported semi-discretely as follows:

$$\left(\frac{\mathbf{u}^{\star\star} - \mathbf{u}^n}{\Delta t^{n+1}} \right) = f_{t,1} \mathcal{M}_{\mathbf{u}}^n - f_{t,2} \mathcal{M}_{\mathbf{u}}^{n-1} + \frac{(\sigma \kappa \delta_\Gamma + \hat{\rho} \mathbf{g})^{n+1}}{\rho^{n+1}} \quad , (15)$$

$$\mathbf{u}^{\star} = \mathbf{u}^{\star\star} - \frac{\Delta t^{n+1}}{\rho_0} \left[\left(1 - \frac{\rho_0}{\rho^{n+1}} \nabla \hat{p} \right) + \nabla p^n \right], \qquad (16)$$

$$\nabla^2 \psi^{n+1} = \frac{\rho_0}{\Delta t^{n+1}} \nabla \cdot \mathbf{u}^{\star} \quad , \qquad (17)$$

$$\mathbf{u}^{n+1} = \mathbf{u}^{\star} - \frac{\Delta t^{n+1}}{\rho_0} \nabla \psi^{n+1} \quad , \qquad (18)$$

$$p^{n+1} = p^n + \psi^{n+1}, \qquad (19)$$

Where the operators M$^{n}_{u}$ and M$^{n}_{u}$$^{-1}$ in equation (15) include convective and diffuse terms calculated at the current and previous time levels and ignore the surface tension and gravitational forces that are then included as the source term. Slowly The spatial gradients in the hair are discretized with central designs. Your average u** It is then updated with the participation of conditions due to the division of time pressure, as in (16). Note that ρ_0 is the minimum value of the density field in the computational domain and $\hat{p}$ represents the time-extrapolated pressure between the current and the old-time step, i.e. $\hat{p} = (1 + \Delta t^{n+1}/\Delta t^n)p^n - (\Delta t^{n+1}/\Delta t^n)p^{n-1}$. Following [52] and contrary to [53, 54], the terms arising from the pressure splittings are included in the prediction of the velocity field (see eq. (16) before the imposition of the boundary conditions. This approach has two distinct benefits. First, it shows an incremental pressure prediction that makes it possible to achieve an almost second-order accuracy in the time pressure field [52]. It then ensures the stability of the pressure field near a solid boundary (i.e. u^{n+1} = u* = 0), in which the component of the normal pressure gradient to the boundary (i.e. ∇⊥ψ^{n+1} = 0) disappears independently Be Local density (see Equation (18)).

Next, the Poisson equation of constant coefficients (17) is solved by a special expansion method that can be used for different combinations of homogeneous pressure boundary conditions [33]. Finally, the velocity field is corrected as in Equation (18) to apply the divergence constraint (i.e., the solenoid velocity field) and to update the pressure according to Equation (19).

E. Poisson solvent

This code uses an FFT-based finite difference solver developed and implemented in DNS CanS code; See [32, 25]. The underlying numerical approach dates back to the late 1970s and has regained its popularity in recent years thanks to improvements in the hardware and software frameworks for mass data communications provided by the MPI standard and above Libraries such as 2DECOMP & FFT. In summary, this approach uses Fourier-based expansions along with two directions of amplitude that reduce the system of equations derived from the Laplace three-dimensional finite-difference second-order operator (seven non-zero diagonals) to a simple three-diagonal system. These Fourier-based expansions depend on system boundary conditions and can be calculated using FFTs, some of which process FFT pre / after / output processing (see, for example, [56]). FFT-based expansions are applied along with the x and y directions, and the resulting three-diagonal system along the z is solved using Gaussian omission. For CPU calculations, this method uses the FFTW library guru interface [57], which allows all possible combinations of discrete conversions to be performed using the same method. In GPUs, fast discrete cosine and sine transform is implemented using real-to-complex / complex-to-real FFTs from the CUFFT library, with pre-and post-processing of input and output signals to calculate the desired batch expansion [56, 25]. We refer to Refs. [32, 25] For details on this method and its implementation.

Due to the parallelization of the method in a distributed memory environment, FFT-based transformations and Gaussian deletion steps require that the data be localized along each direction for each MPI task. The domain is

ISSN No:-2456-2165

parsed using two-dimensional pencil parsing, in which mass communication is required to move around to parse two-dimensional data. These transitions are performed using the 2DECOMP & FFT library [58], which has been modified to allow GPU-GPU communication in [23, 25].

It is worth noting that, in line with recent CanS developments, the present method uses the default parsing (ie, "outside" of Poisson solver) based on partitioning along y and z, resulting in x-aligned pencils. This reduces the total number of data transitions that take place while solving the Poisson equation from 6 to 4. This approach is adopted for both the CPU and the GPU, and the operations required to solve the Poisson equation are summarized as follows:

- Perform forward FFT-based conversions along x.
- x-to-y displacement.
- Perform FFT-based conversions forward along y.
- y-to-z transition.
- Solve the triangular system using Gaussian removal along z.
- Move z-to-y.
- Perform FFT-based conversions backward along y.
- Move y-to-x.
- Perform FFT-based conversions backward along x.

In addition, to implement the GPU, when the domain is not parsed along z, the solvent explicitly reduces the number of all-to-all operations (for example, when x-y slash parsing is allowed). This effectively reduces the number of group operations from 4 to 2 (steps 2 and 8 above are ignored). This is the approach taken in the GPU implementations presented here - due to the higher memory bandwidth in GPUs, slab parsing is sufficient for computing distributed memory with a wall clock time small enough at each step. Explicit ignoring of the two without operation leads to a significant reduction in wall clock time at each stage and an overall improvement in the scalability of the parallel solvent.

F. Complete Solve Algorithm

For clarity, a step-by-step description of the entire solution procedure is presented in Algorithm 1.

Validation

The Zalesak problem represents a classical criterion for assessing the accuracy of an interface capture / tracking algorithm. It involves the rotation of the solid body of a slotted disc embedded in a two-dimensional imposed disc.

Algorithm 1 Overall solution procedure

1: φ^0, T^0, $\mathbf{u}^0$, p^0 are initialized;
2: ρ^0, μ^0, k^0 and $c^0{}_p$ are calculated using equation (2) from φ^0;
3: $n = 0$ is set,
4: **while** ($t < t_{tot}$ k $n < N_{tot}$) **do**
5: Set $n = n + 1$ and Δt^{n+1};
6: φ^{n+1} is calculated from equation (11) and (12);
7: $\mathbf{n}^{n+1}$ and κ^{n+1} are evaluated using the procedure described in Appendix A;
8: ρ^{n+1}, μ^{n+1}, k^{n+1} and C_p^{n+1} is calculated from equation (2);
9: T^{n+1} is calculated from Eq. (13);
10: $\mathbf{u}^*$ is calculated from Eq. (15) and Eq. (16);
11: ψ^{n+1} is calculated from Eq. (17);
12: $\mathbf{u}^{n+1}$ is calculated from Eq. (18); 13: p^{n+1} is computed from Eq. (19).
14: **end while**
15: End of simulation.

Velocity field u = (0.5 - y, x - 0.5). The disk can be easily defined in the Cartesian two-dimensional domain by setting the indicator function $H_{i,\ j,\ k}{}^0$ equal to 1 in the Ω_H range below.

$$\Omega_H : \left[(x-0.5)^2 + (y-0.75)^2\right] \leq 0.15^2 \cap (|x-0.5| \geq 0.025 \cup y \geq 0.85)\,. \tag{20}$$

The criterion involves comparing the deformation of the solid disk with respect to the initial shape after a complete round. The VoF equation in a two-dimensional square domain Ω = [0,1] × [0,1], discrete with four different grid distances [Δx, Δy] = [1 / Nx, 1 / Ny] with Nx × Ny = [32 × 32.64 × 64.128 × 128.256 × 256]. Periodic boundary conditions are imposed in both directions. The simulations are performed up to t = 2π (ie a complete slit disk rotation) using a constant time step Δt = t / 3200. Note that this value is selected to ensure stable temporal integrity for high-resolution grids (ie 256 × 256) and is used for larger cases. Figure 1 shows the final disk shape for different network solutions for two resolution parameters β_{th} = 2 and β_{th} = 3. Note that the greatest deviation from the original shape is in the corner areas, where the areas with the most curvature are located. In addition, the weak solution is dependent on the value of β_{th}, and the deviations between the different β_{th}s used are visible only for larger simulations.

ISSN No:-2456-2165

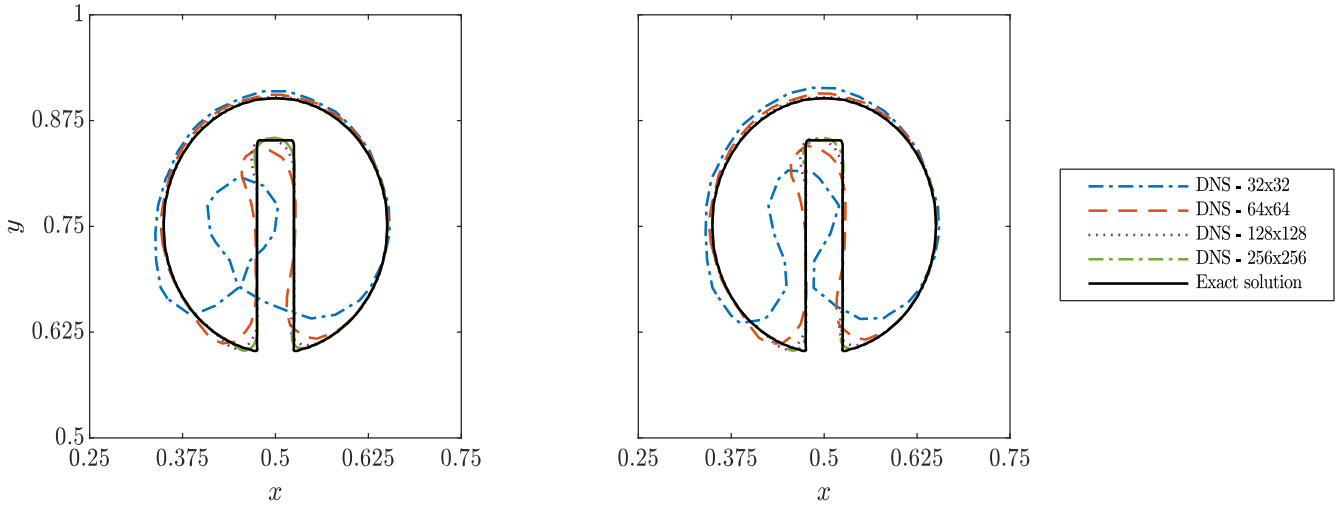

Fig. 1: Zalesak gap disk deformation after t = 2π for β_{th} = 2 (left) and β_{th} = 3 (right).

Among the several possible setups to study secondary breakup of a drop, the three most popular are (i) shock tubes, (ii) continuous jets and (iii) free falling droplets. In this study, we use the continuous air jet setup. A drop of diameter D0 is placed at t = 0 in a flow with a constant farfield velocity field, U∞. We define the aerodynamic Weber number based on the relative velocity between the liquid drop and the gas stream at t = 0, as

$$we = \frac{\rho g U_\infty^2 D_0}{\gamma}$$

where ρg is the density of the ambient gas and γ is the surface tension coefficient at the drop surface. Assuming both the liquid and the ambient gas to be incompressible, the continuity equation is given by

$$\nabla \cdot u = 0,$$

where u is the divergence free velocity field. We use a volume of fluid method which is essentially a one-fluid model for two phase flows. The governing equations for the momentum are given by the Navier–Stokes equations augmented with surface forces to implicitly account for the interfacial boundary conditions of continuity of velocity, and normal and tangential stress balance,

$$\rho(F)\left(\frac{dy}{dx} + \nabla.uu\right) = -\nabla p + \nabla.(\mu(F)D) + \gamma kn\delta_s$$

where F is the volume fraction of liquid and takes values between 0 and 1, ρ(F) = ρlF + (1 − F)ρg, μ(F) = μlF + (1 − F)μg, with ρl, ρg are liquid and gas densities, respectively, and μl and μg are liquid and gas viscosities, respectively. The deformation rate tensor is given by D = (∇u + ∇u^T)/2.

The last term in the equation accounts for the surface tension force (γ κ, where κ is the local curvature of the interface) on the interface embedded in a Eulerian grid and marked with the surface Dirac delta function, δs. The direction of the force is along the local normal (n) at the interface. The surface tension force is modelled as a volumetric force using the continuum surface force approach owing to Brackbill et al. [42]. The evolution equation for the interface is given as an advection equation in terms of the volume fraction, F,

$$\frac{\partial F}{\partial t} + u.\nabla F = 0$$

Bag mode (We ≤ 20)

ISSN No:-2456-2165

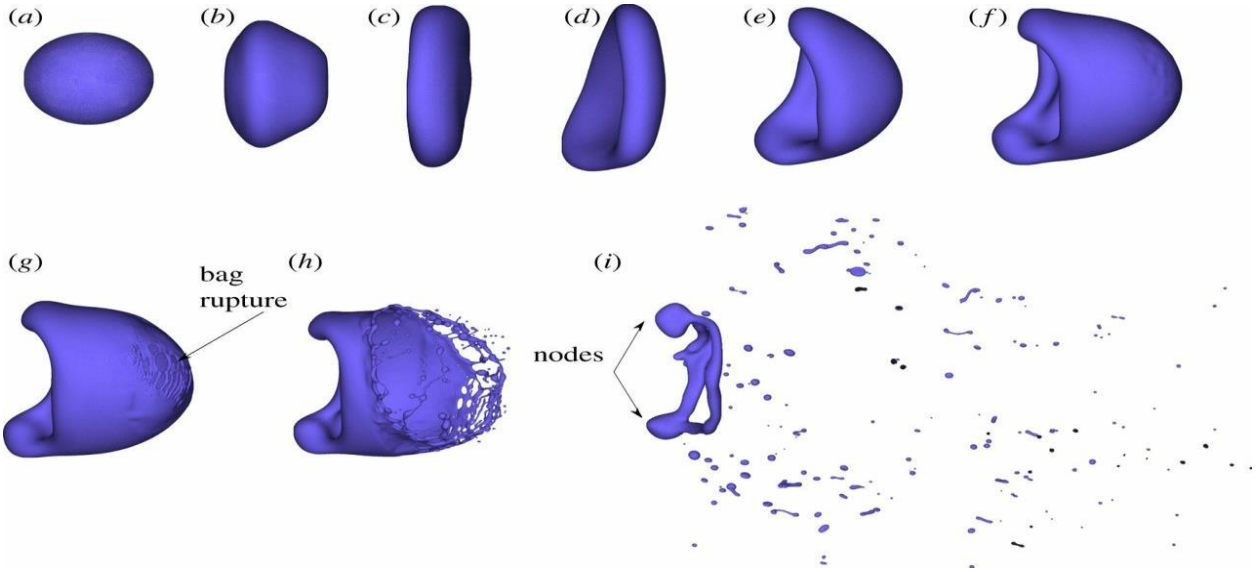

Fig. 2: Droplet motion around Weber number

Bag mode (We ≤ 40)

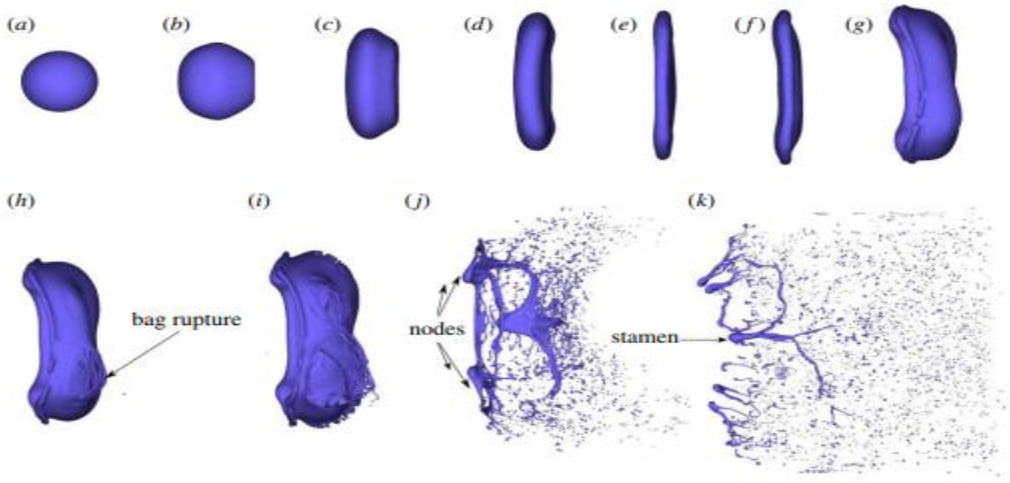

Fig. 3: Droplet motion around Weber number

Bag mode (We ≤ 80)

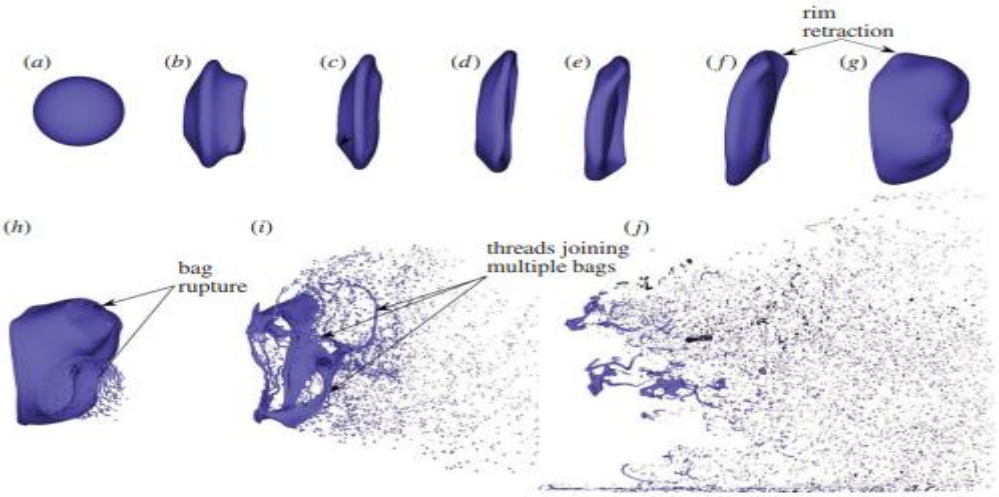

Fig. 4: Droplet motion around Weber number

Bag mode (We ≤ 120)

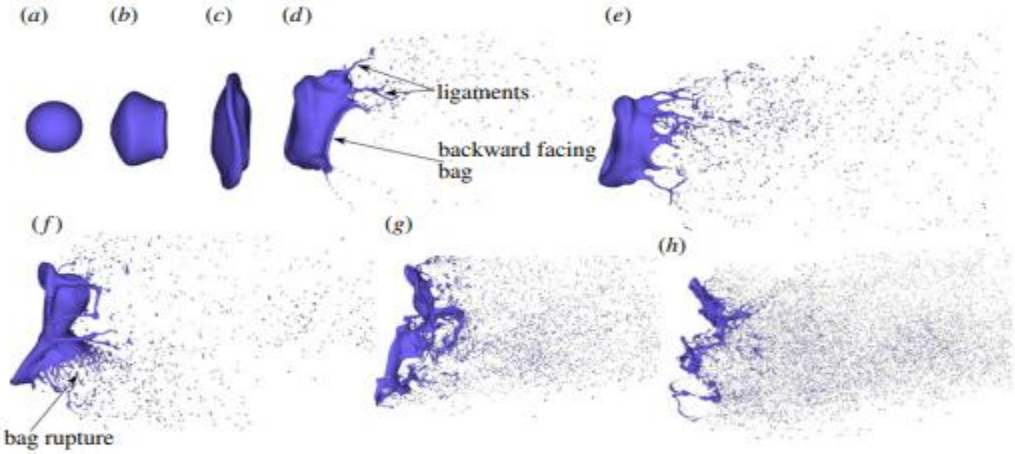

Fig. 5: Droplet motion around Weber number

Finally, to evaluate the accuracy of the answer, we calculate the L_1 norm and the convergence sequence as follows:

$$L_1 = \frac{1}{N_x N_y} \sum_{i=1}^{N_x} \sum_{y=1}^{N_z} |\phi(i,j) - \phi^0(i,j)| \tag{21}$$

$$n_{L1} = \frac{\log\left(\frac{L_{2,N}}{L_{1,N}}\right)}{\log(2)}, \tag{22}$$

Where L_1, N is the error L_1 using the network points Nx × Ny and L_1,2N is the error L_1 which is evaluated with the points of the network 2Nx × 2Ny. The results are reported in Figure 2, where one degree of convergence between the first and second order is obtained for φ, almost independent of the value used β_{th}.

To show the accuracy of the code in the presence of thermal effects, this section considers the airflow in a closed 2D square heated cavity. The cavity is heated and cooled by the vertical side walls (y-normal), while the horizontal walls are adiabatic (z-normal). In this configuration, a blood circulation is formed and maintained by a hot ascending fluid next to the heated wall and a descending cold fluid next to the cooled wall. Therefore, the flow is purely thermal and is denoted by the rail number Ra =) and the Prandtl number Pr = ν / α.

In these definitions, β is the coefficient of thermal expansion of the fluid, ν is the viscosity of the fluid, α is the thermal diffusion of the fluid, and ΔT = (T_h - T_c) is the temperature difference between the heated (T_h) and cooled (T_c) walls. The height of the hole is usually considered as the reference length (lr = Lz), while the reference velocity and time are u_r = α / l_r and. The case simulated here follows the setting presented in several studies [61, 62, 60] with Ra = 106 and Pr = 0.71. The boundaries of the sphere are solid walls and the boundary conditions are non-slip. Depending on the temperature field, the constant temperature boundary conditions are applied to the vertical walls and a zero temperature gradient is applied along the normal direction to the horizontal walls.

ISSN No:-2456-2165

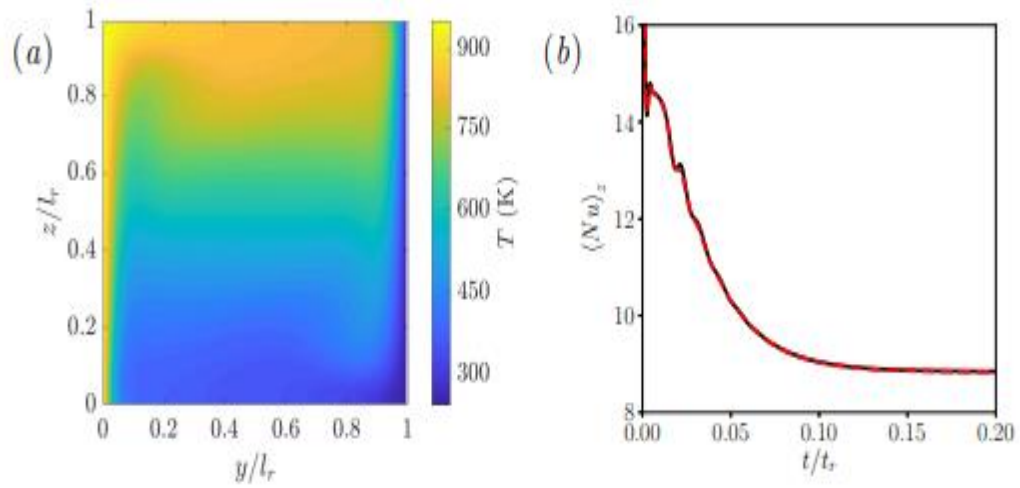

Fig. 3: (A) Temperature field contour diagram at t / tr = 0.5 (steady state) for the heated cavity test, (b) temporal evolution of the mean Nusselt number of the wall average on the heated wall. Black integrated line, present results. The red dashed line is the result of the reference from [60]

The domain is split in space using a uniform Cartesian network with 256 ১ 256 cells. Initially, the cavity air is stationary and isothermal at T0 = Tc. A constant time step Δt is used to advance the solution in time, which is obtained with Δt / tr = 5.0 × 10-7. (Figure 2a) The contour shows the temperature field at t / tr = 0.5 where the point has reached a steady state. The temperature field is characterized by thin, spatially developing boundary layers alongside thermally active vertical walls and a classified area in the central region of the cavity. The rate of heat transfer inside the cavity is expressed by the Nusselt number, which is defined as follows:

$$\text{Nu} = \frac{hl_r}{k} = \frac{l_r}{\Delta T}\nabla T\big|_w \cdot \mathbf{n}_w, \quad (23)$$

Where h is the heat transfer coefficient, k is the thermal conductivity of the fluid, the temperature gradient in each of the vertically thermally active walls, and n_w is the normal vector of the corresponding unit on the wall. (Figure 2b) shows a comparison of the temporal evolution of the nusselt hNuiz number with the mean wall on the hot wall between the current and reference results from [60]. Obviously the present results are in excellent agreement with the reference solution for the whole simulation period.

	V_{max}/u_r	W_{max}/u_r	N_{umax}	N_{umin}	$\langle N_u \rangle_z$
Ref.[60]	64.85	220.6	17.58	0.9794	8.830
Present	64.86	220.3	17.67	0.9773	8.843
%dev	0.02	0.14	0.51	0.21	0.14

Table 2: Comparison of the values of the key criteria in the steady state for the test case of the cavity with different heating. Vmax is the maximum horizontal velocity along the vertical center plate (y = 0. lr), Wmax is the maximum vertical velocity along the horizontal middle plate (z = 0.5lr), Numax and Numin are the maximum and minimum Nusselt values Heated on the wall

III. GPU CODE SYNCHRONIZATION AND GPU ACCELERATION

A. Domain parsing

This code is designed to run on multiple CPU and multi GPU architectures. For domain analysis, both slabs (1D) and pencils (2D) are allowed through the 2DECOMP library [58]. The type of parsing can be implicitly configured in a dns.in input file by dimming the two-component array (e.g. [1, n] for slabs and [n, m] for pencils). The pencil / slab direction can be arbitrarily selected, such as CanS, via the preprocessor flags -D DECOMP X, -D DECOMP Y and -D DECOMP Z, which specify the direction in which the amplitude does not decompose. This flexibility improves both CPU and GPU performance. For the CPU, the use of a pencil makes it possible to increase the number of processes used in each run (ie up to N_2 for $n_x = n_y = n_z = N$), thus reducing the solving time. In the GPU implementation, only z-pencil and x-slabs decomposition is allowed. It is recommended to use x-slabs in the GPU (ie compile with -D DECOMP X and use dims = [1, n]) because this implementation minimizes the number of all-round calls and thus the GPU- Reduces the GPU and improves performance in multi-node executables.

B. Code parallelization

Paralleling is done using MPI. When GPU acceleration is enabled, the MPI assigns a rating to each GPU. This code assumes that the MPI library is optionally "CUDA-aware", meaning that the GPU data is transmitted directly to the MPI function call, and the MPI implementation takes care of data transfer in the most efficient way. The most efficient way, if any, is to connect the GPU to the GPU using NVIDIA NVLink, which is a physical GPU-to-GPU connection that has a higher bandwidth (at least once as large) than Infiniband. Across code, all nested loops, that is, iterations across domains, use OpenACC [63], a standard portable instruction-based programming model that can execute code on multi-core CPUs as well as accelerators such as NVIDIA GPUs Run. Such a drain is not used to compile and execute only the CPU. To run FluTAS, the platform requires support

ISSN No:-2456-2165

for NVIDIA integrated memory, which has two main advantages:

- The ability to allocate and manage allocated GPU memory more than is physically present on the device;
- Ability to avoid explicit management of Host-to-Device and Device-to-Host data transfer, which makes runtime work for developers.

Both features are used in code and have been proven to be critical to efficient GPU acceleration.

C. Code performance

We now present an analysis of code performance in standard CPU-based and accelerated GPU-based architectures. GPU tests were performed on MeluXina at LuxProvide (LXP, Luxembourg) [64] and Berzelius at the National Supercomputer Center (NSC, Sweden) [65], while tests on CPUs on Tetralith were also performed by the NSC.

D. Poor and strong scaling

We first discuss the poor scaling tests for the Rayleigh-B'enard problem with the same configuration as will be discussed in .27.2. For this experiment, we start with a "base" computing grid Nx × Ny × Nz = 1024 × 512 × 256 grid points on 2 GPUs. Then, while keeping Nx and Ny constant, we increase Nz in proportion to the number of GPUs, similar to the method of the spatial group average (ie, more simulated structures to improve the convergence of large-scale statistics). As discussed in §5, we compile a slab parallelism along the z-direction using the -D DECOMP X option, which reduces the number of all-in-one operations to two. It is worth noting that although both HPCs are equipped with NVIDIA A100-40GB cards, the Berzelius has 8 GPU / node while the MeluXina has 4 GPU / node. In addition, the connection between GPUs is done through NVLink, while the node-to-node connection is done through Infiniband (IB), which is known for having less bandwidth and operating on different protocols. Hence, IB management is not easy because it requires more precise configuration from a hardware and software perspective.

This requires selecting the appropriate MPI settings and selecting compatible communication libraries, resulting in performance that may vary significantly across HPCs. For these reasons, Berzelius was used to perform low-scale tests on a node to prove IB-independent scaling and to maximize GPU-to-GPU connectivity. MeluXina, on the other hand, was used for multi-node tests to evaluate IB-dependent scaling. (Figure 3a)shows that the weak scale is linear when constrained by NVLink communications (ie without IB communications), as clearly supported by Berzelius experiments.

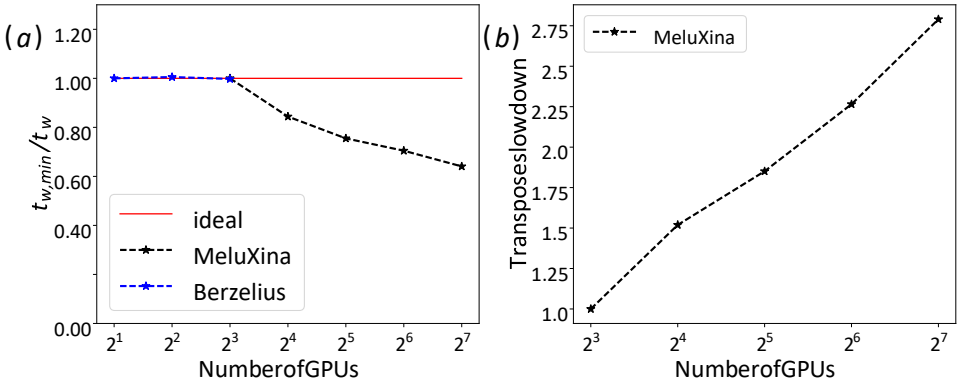

Fig. 4: For the Rayleigh-B'enard two-layer convection problem discussed in .27.2: a) Code performance in MeluXina and Berzelius, b) Speed reduction due to transfer operation. For each data set, we compute tw, min as the time in the time step in the minimum number of GPUs tested in tw, that is, the time at any given time in a given number of GPUs.

When IB communications are required (ie, data transfer from node to node), code performance is reduced. It is worth noting that, while an increasing communication overhead is provided by node-to-node communication in the IB network, the reduction in additional velocity is caused by slab parallelism. As the number of elements along the z increases, more data must be transmitted during the x-to-z shift, and the communication load increases more. This is clearly shown in(Figure 3b, where the deceleration increases with the number of GPUs. Strong scaling test results are reported in (Figure 4). Here we use two different networks, 1024 × 512 × 1024 (network-1) and 1024 × 1024 × 1024 × 1024 (network-2) for the Rayleigh-B'enard problem under discussion. In §7.2. Experiments are performed on Meluxina and Berzelius for poor massification. By keeping the size of the problem constant, the number of GPUs gradually increases to a maximum of 128, from NGPU = 16, which indicates the minimum value required to accommodate the two computing domains in the existing GPU memory.

Despite the ever-increasing speed, the code shows a gradual reduction in performance, that is, a reduction in the benefits of increasing the number of GPUs. Note, however, that more network points (eg Grid-2) will result in less performance, as higher GPU occupancy can be achieved.

ISSN No:-2456-2165

The decrease in performance observed in Figure 7 is due to two factors: an increase in communication between GPUs and a decrease in the size of the local problem, which does not affect the full computing capacity of each GPU. While these effects are present in a strong scaling test, a weak scale allows us to isolate the effects of multi-GPU communications while maintaining higher GPU saturation. Therefore, we argue that poor scale is a better tool for identifying communication bottlenecks across multiple GPUs. Conversely, robust scaling is more useful for estimating the degree of segmentation of a fixed domain while maintaining efficient use of computational resources. In general, the previous analysis suggests an important guideline for the user: in the presence of computational architectures versus an unbalanced network (e.g., a more efficient node-to-network network connection than a GPU connection in the same node), the optimal number of GPUs To be used, it should be selected as close as possible to the minimum value required to fit the computational range in the existing GPU memory.

In fact, this is not always the case with older HPC architectures that use previous generations of GPU hardware, where NVLink connections between GPUs within a node usually did not exist. For a fixed problem size, modern cards with high computing power complete the required computations faster, leaving the remainder of the computation as limited communication. In older GPUs, acceleration is slower and communication becomes the dominant component, affecting the scalability of more GPUs. Hence, the best practice is to use as few GPUs as possible. On modern units with 80GB of HBM memory, it is convenient to use an 8-way GPU node (such as the DGX A100) where possible, where all GPUs are also connected via NVLink, significantly reducing communication costs Gives.

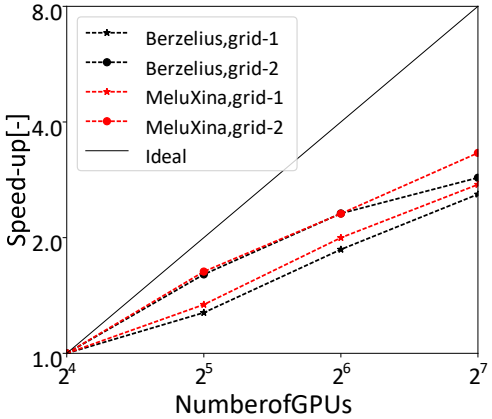

Fig. 5: Strong scaling experiment was performed on Berzelius (black dashed lines) and MeluXina (red dashed lines) clusters for two different networks: 1024 × 512 × 1024 (network-1) and 1024 × 1024 × 1024 × gr-24. The continuous black line indicates the ideal behavior desired for the robust scaling test

IV. CPU-GPU COMPARISON

As a result, we make a comparison between the code performance of a CPU and a GPU architecture. It is worth noting that such a comparison is not obvious. First, no precise and standard method has been developed to compare the two systems. Next, code performance may show large variations between different architectures, and using a hybrid CPU-GPU node to perform experiments on both can be misleading. CPU-only nodes and CPU-GPU nodes are inherently different in terms of network configuration and GPU / CPU connectivity, so neutral testing may not be performed directly on hybrid architectures (because the CPU-GPU cluster is difficult to run on CPU only Used jobs).

Therefore, the following analysis should be considered as the first approximate estimate. Here we repeat the weak scale simulation with nGPU = 8 GPU in Berzelius at nCPU = 512 CPU in Tetralith, in both cases we use slab parallelism along z. Experiments show that for GPUs the average wall clock time per step t8 is GPU = 0.191 seconds, while for CPU t512, CPU = 1.075 seconds. This results in an equivalent number of GPUs of neq = (t512, CPUnCPU) / (t8,GPUnGPU) ≈ 359.

Finally, a comparison in terms of computational load percentage for each piece of code is shown in Figure 5. As previously predicted, the displacements during the GPU simulation (panel a) represent more than half of the computational load. The remaining sections consist mainly of loop-bound stencil operations, largely utilizing GPU-offload, while CPUs (panel b) account for more than 70% of the total wall clock time at each time stage.

ISSN No:-2456-2165

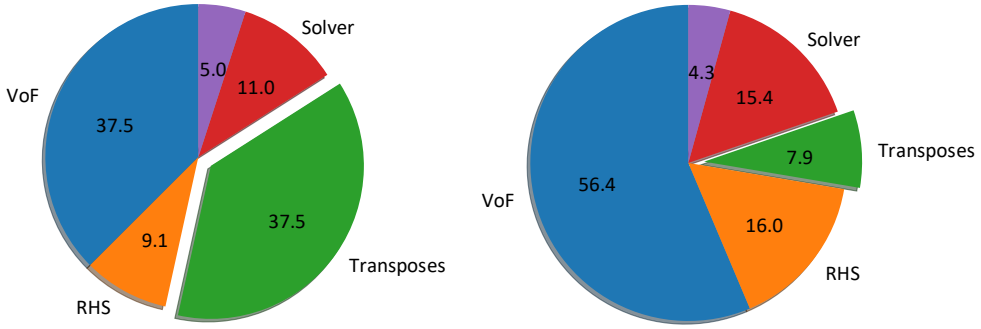

Fig. 6: Comparison of the percentage of code load in the total simulation time for GPU (panel a) and CPU (panel b). Different "slices" represent different pieces of code: 1) VoF (ie, interface reconstruction and advection, upgrade of thermophysical properties), 2) RHS (ie, discretization of governing equations), 3) displacement (ie, solver displacement operation), 4) solution (Ie Gaussian deletion only) and other items (ie correction step, divergence / time step checks, output and post-processing procedures).

V. CONCLUSIONS AND MORE PROGRESS

We provide the FluTAS code, a numerical framework for direct numerical simulations of multiphase currents with the option of heat transfer, which can be effectively implemented on standard CPU-based architectures and GPU-based accelerator machines. The open source version, released under the MIT license, includes a pressure correction algorithm for two-phase flows developed by the fluid volume algebra method (MTHINC) to record interface dynamics.

Here we provide a description of the numerical algorithm used with details of solving governing equations and interface advection. After presenting different validation criteria in single-phase and multi-phase configurations, we discuss code performance with a focus on two aspects: Has a knot, ii. Its advantages over the CPU in terms of "time to dissolution" Finally, we report the results of two configurations of fundamental benefits in multiphase turbulence: emulsions in homogeneous isotropic turbulence and rail-binard double layer convection. In the future, our goal is to improve the ability to store and carry code (both on the CPU and GPU) and to release additional modules under development, for example. Poor compressibility and phase change [43, 44]. Further efforts will be made to improve code performance across multiple GPU nodes, reducing current communication bottlenecks. To this end, it is a promising strategy proposed in [73], that is, it implements a triangular system solution for the third direction on distributed memory. The main advantage of this approach is the elimination of all-in-one operations in the Poisson solver. This development, together with future advances in software frameworks for mass data communications, will make it possible to address several multi-step problems while maintaining efficient use of computational resources.

REFERENCES

[1.] S. Ii, K. Sugiyama, S. Takeuchi, S. Takagi, Y. Matsumoto, F. Xiao, An interface capturing method with a continuous function: the thinc method with multi-dimensional reconstruction, Journal of Computational Physics 231 (5) (2012) 2328–2358.

[2.] W. W. Grabowski, L.-P. Wang, Growth of cloud droplets in a turbulent environment, Annual review of fluid mechanics 45 (2013) 293–324.

[3.] G. Seminara, Fluvial sedimentary patterns, Annual Review of Fluid Mechanics 42 (2010) 43–66.

[4.] L. Brandt, F. Coletti, Particle-laden turbulence: Progress and perspectives, Annual Review of Fluid Mechanics 54 (2021).

[5.] F. Veron, Ocean spray, Annual Review of Fluid Mechanics 47 (2015) 507–538.

[6.] T. Dauxois, T. Peacock, P. Bauer, C.-c. P. Caulfield, C. Cenedese, C. Gorl´e, G. Haller, G. N. Ivey, P. F. Linden, E. Meiburg, et al., Confronting grand challenges in environmental fluid mechanics, Physical review fluids 6 (2) (2021) 020501.

[7.] C. T. Crowe, Multiphase flow handbook, CRC press, 2005.

[8.] G. A. Voth, A. Soldati, Anisotropic particles in turbulence, Annual Review of Fluid Mechanics 49 (2017) 249–276.

[9.] F. Risso, Agitation, mixing, and transfers induced by bubbles, Annual Review of Fluid Mechanics 50 (2018) 25–48.

[10.] S. Elghobashi, Direct numerical simulation of turbulent flows laden with droplets or bubbles, Annual Review of Fluid Mechanics 51 (2019) 217–244.

[11.] V. Mathai, D. Lohse, C. Sun, Bubbly and buoyant particle–laden turbulent flows, Annual Review of Condensed Matter Physics 11 (2020) 529–559.

[12.] A. U. M. Masuk, A. Salibindla, R. Ni, A robust virtual-camera 3d shape reconstruction of deforming bubbles/droplets with additional physical constraints, International Journal of Multiphase Flow 120 (2019) 103088.

[13.] A. K. Salibindla, A. U. M. Masuk, S. Tan, R. Ni, Lift and drag coefficients of deformable bubbles in intense turbulence determined from bubble rise velocity, Journal of Fluid Mechanics 894 (2020).

[14.] A. U. M. Masuk, A. K. Salibindla, R. Ni, Simultaneous measurements of deforming hinze-

scale bubbles with surrounding turbulence, Journal of Fluid Mechanics 910 (2021).
[15.] S. Mirjalili, S. S. Jain, M. Dodd, Interface-capturing methods for twophase flows: An overview and recent developments, Center for Turbulence Research Annual Research Briefs 2017 (117-135) (2017) 13.
[16.] S. O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, Journal of computational physics 100 (1) (1992) 25–37.
[17.] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annual review of fluid mechanics 31 (1) (1999) 567–603.
[18.] D. M. Anderson, G. B. McFadden, A. A. Wheeler, Diffuse-interface methods in fluid mechanics, Annual review of fluid mechanics 30 (1) (1998) 139–165.
[19.] J. A. Sethian, P. Smereka, Level set methods for fluid interfaces, Annual review of fluid mechanics 35 (1) (2003) 341–372.
[20.] A. Prosperetti, G. Tryggvason, Computational methods for multiphase flow, Cambridge university press, 2009.
[21.] G. Soligo, A. Roccon, A. Soldati, Turbulent flows with drops and bubbles: What numerical simulations can tell us—freeman scholar lecture, Journal of Fluids Engineering 143 (8) (2021).
[22.] A. Khan, H. Sim, S. S. Vazhkudai, A. R. Butt, Y. Kim, An analysis of system balance and architectural trends based on top500 supercomputers, in: The International Conference on High Performance Computing in Asia-Pacific Region, 2021, pp. 11–22.
[23.] X. Zhu, E. Phillips, V. Spandan, J. Donners, G. Ruetsch, J. Romero, R. Ostilla-Mo´nico, Y. Yang, D. Lohse, R. Verzicco, et al., Afid-gpu: a versatile navier–stokes solver for wall-bounded turbulent flows on gpu clusters, Computer physics communications 229 (2018) 199–210.
[24.] M. Bernardini, D. Modesti, F. Salvadore, S. Pirozzoli, Streams: A highfidelity accelerated solver for direct numerical simulation of compressible turbulent flows, Computer Physics Communications 263 (2021) 107906.
[25.] P. Costa, E. Phillips, L. Brandt, M. Fatica, Gpu acceleration of cans for massively-parallel direct numerical simulations of canonical fluid flows, Computers & Mathematics with Applications 81 (2021) 502–511.
[26.] W. Aniszewski, T. Arrufat, M. Crialesi-Esposito, S. Dabiri, D. Fuster, Y. Ling, J. Lu, L. Malan, S. Pal, R. Scardovelli, et al., Parallel, robust, interface simulator (paris), Computer Physics Communications 263 (2021) 107849.
[27.] P. Cifani, J. Kuerten, B. Geurts, Highly scalable dns solver for turbulent bubble-laden channel flow, Computers & Fluids 172 (2018) 67–83.
[28.] K. Eisenschmidt, M. Ertl, H. Gomaa, C. Kieffer-Roth, C. Meister, P. Rauschenberger, M. Reitzle, K. Schlottke, B. Weigand, Direct numerical simulations for multiphase flows: An overview of the multiphase code fs3d, Applied Mathematics and Computation 272 (2016) 508–517.
[29.] O. Desjardins, G. Blanquart, G. Balarac, H. Pitsch, High order conservative finite difference scheme for variable density low mach number turbulent flows, Journal of Computational Physics 227 (15) (2008) 7125– 7159.
[30.] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, Journal of Computational Physics 228 (16) (2009) 5838– 5866.
[31.] S. H. Bryngelson, K. Schmidmayer, V. Coralic, J. C. Meng, K. Maeda, T. Colonius, Mfc: An open-source high-order multi-component, multiphase, and multi-scale compressible flow solver, Computer Physics Communications (2020) 107396doi:10.1016/j.cpc.2020.107396.
[32.] P. Costa, A fft-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows, Computers & Mathematics with Applications 76 (8) (2018) 1853–1862.
[33.] U. Schumann, R. A. Sweet, Fast fourier transforms for direct solution of poisson's equation with staggered boundary conditions, Journal of Computational Physics 75 (1) (1988) 123–137.
[34.] M. E. Rosti, F. De Vita, L. Brandt, Numerical simulations of emulsions in shear flows, Acta Mechanica 230 (2) (2019) 667–682.
[35.] F. De Vita, M. E. Rosti, S. Caserta, L. Brandt, On the effect of coalescence on the rheology of emulsions, Journal of Fluid Mechanics 880 (2019) 969–991.
[36.] F. De Vita, M. E. Rosti, S. Caserta, L. Brandt, Numerical simulations of vorticity banding of emulsions in shear flows, Soft matter 16 (11) (2020) 2854–2863.
[37.] M. E. Rosti, S. Takagi, Shear-thinning and shear-thickening emulsions in shear flows, Physics of Fluids 33 (8) (2021) 083319.
[38.] M. E. Rosti, Z. Ge, S. S. Jain, M. S. Dodd, L. Brandt, Droplets in homogeneous shear turbulence, Journal of Fluid Mechanics 876 (2019) 962–984.
[39.] M. Kozul, P. S. Costa, J. R. Dawson, L. Brandt, Aerodynamically driven rupture of a liquid film by turbulent shear flow, Physical Review Fluids 5 (12) (2020) 124302.
[40.] M. Crialesi-Esposito, M. E. Rosti, S. Chibbaro, L. Brandt, Modulation of homogeneous and isotropic turbulence in emulsions, Journal of Fluid Mechanics 940 (2022).
[41.] I. Cannon, D. Izbassarov, O. Tammisola, L. Brandt, M. E. Rosti, The effect of droplet coalescence on drag in turbulent channel flows, Physics of Fluids 33 (8) (2021) 085112.
[42.] N. Scapin, P. Costa, L. Brandt, A volume-of-fluid method for interfaceresolved simulations of phase-changing two-fluid flows, Journal of Computational Physics 407 (2020) 109251.
[43.] F. Dalla Barba, N. Scapin, A. D. Demou, M. E. Rosti, F. Picano, L. Brandt, An interface capturing method for liquid-gas flows at lowmach number, Computers & Fluids 216 (2021) 104789.

[44.] N. Scapin, F. Dalla Barba, G. Lupo, M. E. Rosti, C. Duwig, L. Brandt, Finite-size evaporating droplets in weakly compressible homogeneous shear turbulence, Journal of Fluid Mechanics 934 (2022).
[45.] M. Ishii, T. Hibiki, Thermo-fluid dynamics of two-phase flow, Springer Science & Business Media, 2010.
[46.] F. H. Harlow, J. E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, The physics of fluids 8 (12) (1965) 2182–2189.
[47.] E. G. Puckett, A. S. Almgren, J. B. Bell, D. L. Marcus, W. J. Rider, A high-order projection method for tracking fluid interfaces in variable density incompressible flows, Journal of computational physics 130 (2) (1997) 269–282.
[48.] E. Aulisa, S. Manservisi, R. Scardovelli, S. Zaleski, A geometrical areapreserving volume-of-fluid advection method, Journal of Computational Physics 192 (1) (2003) 355–364.
[49.] G. D. Weymouth, D. K.-P. Yue, Conservative volume-of-fluid method for free-surface simulations on cartesian-grids, Journal of Computational Physics 229 (8) (2010) 2853–2865.
[50.] M. Castro, B. Costa, W. S. Don, High order weighted essentially nonoscillatory weno-z schemes for hyperbolic conservation laws, Journal of Computational Physics 230 (5) (2011) 1766–1792.
[51.] A. J. Chorin, Numerical solution of the navier-stokes equations, Mathematics of computation 22 (104) (1968) 745–762.
[52.] C. Frantzis, D. G. Grigoriadis, An efficient method for two-fluid incompressible flows appropriate for the immersed boundary method, Journal of Computational Physics 376 (2019) 28–53.
[53.] S. Dong, J. Shen, 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) (2012) 5788–5804.
[54.] M. S. Dodd, A. Ferrante, A fast pressure-correction method for incompressible two-fluid flows, Journal of Computational Physics 273 (2014) 416–434.
[55.] P. N. Swarztrauber, The methods of cyclic reduction, fourier analysis and the facr algorithm for the discrete solution of poisson's equation on a rectangle, Siam Review 19 (3) (1977) 490–501.
[56.] J. Makhoul, A fast cosine transform in one and two dimensions, IEEE Transactions on Acoustics, Speech, and Signal Processing 28 (1) (1980) 27–34.
[57.] M. Frigo, S. G. Johnson, Fftw: An adaptive software architecture for the fft, in: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181), Vol. 3, IEEE, 1998, pp. 1381–1384.
[58.] N. Li, S. Laizet, 2decomp & fft-a highly scalable 2d decomposition library and fft interface, in: Cray user group 2010 conference, 2010, pp. 1–13.
[59.] S. Turek, O. Mierka, K. B¨aumler, Numerical benchmarking for 3d multiphase flow: New results for a rising bubble, in: European Conference on Numerical Mathematics and Advanced Applications, Springer, 2017, pp. 593–601.
[60.] J. Armengol, F. Bannwart, J. Xam´an, R. Santos, Effects of variable air properties on transient natural convection for large temperature differences, International Journal of Thermal Sciences 120 (2017) 63–79.
[61.] G. de Vahl Davis, I. Jones, Natural convection in a square cavity: a comparison exercise, International Journal for numerical methods in fluids 3 (3) (1983) 227–248.
[62.] M. Leal, H. Machado, R. Cotta, Integral transform solutions of transient natural convection in enclosures with variable fluid properties, International Journal of Heat and Mass Transfer 43 (21) (2000) 3977–3990.
[63.] Openacc.URL https://www.openacc.org/sites/default/files/ inline-files/OpenACC.2.7.pdf
[64.] Meluxina. URL https://luxprovide.lu/technical-structure/
[65.] Berzelius. URL https://www.nsc.liu.se/systems/berzelius/
[66.] On the non-linear stability of the 1:1:1 ABC flow, Physica D: Nonlinear Phenomena 75 (4) (1994) 471–508. doi:10.1016/0167-2789(94) 00031-X.
[67.] P. D. Mininni, A. Alexakis, A. Pouquet, Large-scale flow effects, energy transfer, and self-similarity on turbulence, Physical Review E Statistical, Nonlinear, and Soft Matter Physics 74 (1) (2006) 1–13. doi:10.1103/PhysRevE.74.016303.
[68.] C. Garrett, M. Li, D. Farmer, The connection between bubble size spectra and energy dissipation rates in the upper ocean, Journal of Physical Oceanography 30 (9) (2000) 2163–2171. doi:10.1175/ 1520-0485(2000)030<2163:TCBBSS>2.0.CO;2.
[69.] G. B. Deane, M. D. Stokes, Scale dependence of bubble creation mechanisms in breaking waves, Nature 418 (6900) (2002) 839–844. doi: 10.1038/nature00967.
[70.] F. H. Busse, On the aspect ratios of two-layer mantle convection, Physics of the Earth and Planetary Interiors 24 (4) (1981) 320–324.
[71.] F. Wilczynski, D. W. Hughes, Stability of two-layer miscible convection, Physical Review Fluids 4 (10) (2019) 103502.
[72.] H.-R. Liu, K. L. Chong, Q. Wang, C. S. Ng, R. Verzicco, D. Lohse, Twolayer thermally driven turbulence: mechanisms for interface breakup, Journal of Fluid Mechanics 913 (2021).
[73.] S. Ha, J. Park, D. You, A multi-gpu method for adi-based fractional-step integration of incompressible navier-stokes equations, Computer Physics Communications 265 (2021) 107999.
[74.] D. L. Youngs, Time-dependent multi-material flow with large fluid distortion, Numerical methods for fluid dynamics (1982).
[75.] D. L. Youngs, An interface tracking method for a 3d eulerian hydrodynamics code, Atomic Weapons

Research Establishment (AWRE) Technical Report 44 (92) (1984) 35.
[76.] G. Strang, On the construction and comparison of difference schemes, SIAM journal on numerical analysis 5 (3) (1968) 506–517.