Evolution of a double-front Rayleigh-Taylor system using a graphics-processing-unit-based high-resolution thermal lattice-Boltzmann model

P. Ripesi,[1,*] L. Biferale,[1] S. F. Schifano,[2] and R. Tripiccione[3]

[1]*Department of Physics and INFN, University of Tor Vergata, Via della Ricerca Scientifica 1, 00133 Rome, Italy*
[2]*Dipartimento di Matematica e Informatica, Università di Ferrara and INFN, Via G. Saragat 1, 44100 Ferrara, Italy*
[3]*Dipartimento di Fisica e Scienze della Terra, Università di Ferrara and INFN, Via G. Saragat 1, 44100 Ferrara, Italy*

We study the turbulent evolution originated from a system subjected to a Rayleigh-Taylor instability with a double density at high resolution in a two-dimensional geometry using a highly optimized thermal lattice-Boltzmann code for GPUs. Our investigation's initial condition, given by the superposition of three layers with three different densities, leads to the development of two Rayleigh-Taylor fronts that expand upward and downward and collide in the middle of the cell. By using high-resolution numerical data we highlight the effects induced by the collision of the two turbulent fronts in the long-time asymptotic regime. We also provide details on the optimized lattice-Boltzmann code that we have run on a cluster of GPUs.

I. INTRODUCTION

A Rayleigh-Taylor (RT) system is composed of the superposition of two layers of a single-phase fluid, with the lower lighter than the upper one and subject to an external gravity field $g > 0$. In this system, the two layers mix together until the fluid reaches "equilibrium", characterized by a completely homogeneous environment with hydrostatic density-temperature profiles. Applications span a wide range of fields, such as astrophysics [1], quantum physics related to the immiscible Bose-Einstein condensates [2,3], and oceanic and atmospheric sciences [4,5]. Historically, the first theoretical work on the stability of a stratified fluid in a gravitational field is due to Rayleigh in 1900 [6], followed about 40 years later by Taylor's work on the growth of perturbations between two fluids with different densities [7]. Since then, the Rayleigh-Taylor instability has been intensively studied theoretically, experimentally and numerically (see, e.g., the review of Dimonte *et al.* [8]). Still, many problems remain open. The RT instability amounts to two main physics problems: the initial growth of perturbations between two layers of fluid with different densities and the mixing problem related to the penetration of the perturbation front through the static fluid. The evolution of the mixing layer length, $L(t)$, follows a "free-fall" temporal law, $L(t) = \alpha g(\text{At})t^2$; At $= (\rho_1 - \rho_2)/(\rho_1 + \rho_2)$ is the Atwood number, which takes into account the density differences between the upper (ρ_1) and lower (ρ_2) layers, g is the acceleration of gravity, and α is a dimensionless coefficient, the so-called *growth rate*. Recent works [9,10] have suggested that the value of α might depend also on the initial conditions. Beside the large-scale growth of the mixing layer, small-scale statistics also have attracted the attention of many groups in recent years, both in two-dimensional (2D), three-dimensional (3D), and *quasi* 2D–3D geometries [11–14]. Moreover, different setups have been investigated including stratification [15–19] and reaction [20,21]. In spite of the progress made so far, most of the work in this area is limited to the classical

case of double-density fluids, while complex stratifications effects have not been extensively investigated. In this paper we want to present the study of an RT system which is slightly different from the ones present in the literature: we focus on the spatio-temporal evolution of a single component fluid when initially prepared with three different density layers in hydrostatic unstable equilibrium (see Fig. 1). Previous studies
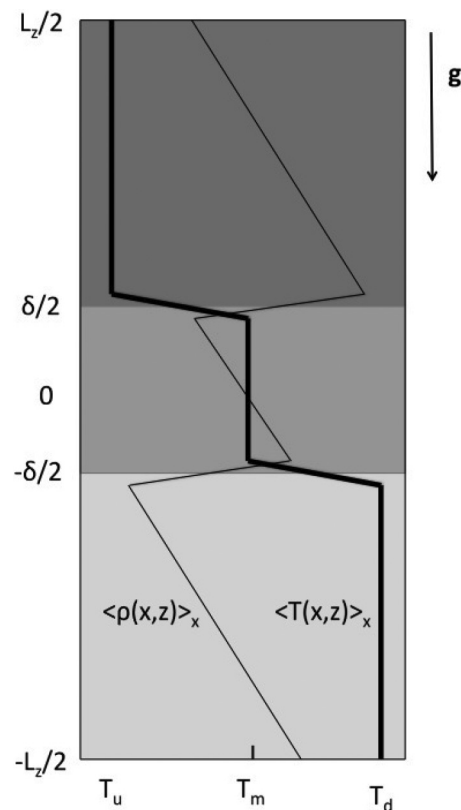

FIG. 1. Sketch of the initial configuration for the triple-temperature Rayleigh-Taylor system. Temperature in the three regions is chosen constant, while the density follows an hydrostatic profile [Eq. (1)]. The temperature jump at the interface is smoothed by a tanh profile of the order of 10 grid points. The bold and thin solid lines represent the temperature and density profiles, respectively.

*patrizio.ripesi@roma2.infn.it

on triple-density RT were limited to the case of one unstable and one stable layer [22], focusing mainly on the entrainment by the unstable flow inside the stable one.

On the contrary, we have a fully unstable density-temperature distributions with two unstable layers, and we need to take into account the nonlinear interactions of rising and falling plumes from each one of the two developing mixing layers, different from the case of the interaction of buoyant plumes with an interface [23]. Similarly, our case differs from the case of propagating fronts [24–26] because we do not have

the extra effects induced by the deflagration velocity. The setup is given by a 2D $L_x \times L_z$ tank of fluid split in three subvolumes at three different, initially homogeneous, temperatures $T_u < T_m < T_d$, where each of them is in hydrostatic equilibrium $\partial_z p_0(z) = -g\rho_0(z)$. We enforce periodic boundary conditions in the horizontal direction.

The initial hydrostatic unstable configuration is therefore given by

$$
\begin{aligned}
T_0(z) = T_d, \quad & \rho_0^d(z) = \rho_d \exp[-g(z-z_d)/T_d], \quad -L_z/2 < z < -\delta/2. \\
T_0(z) = T_m, \quad & \rho_0^m(z) = \rho_m \exp[-g(z-z_m)/T_m], \quad -\delta/2 < z < \delta/2, \\
T_0(z) = T_u, \quad & \rho_0^u(z) = \rho_u \exp[-g(z-z_u)/T_u], \quad \delta/2 < z < L_z/2,
\end{aligned}
\tag{1}
$$

where δ is the width of the middle layer at temperature T_m, and z_u, z_m, z_d are three parameters fixing the overall geometry.

Assuming that for each single domain we have $p_0(z) = T_0\rho_0(z)$, in order to be at equilibrium we require the same pressure at the interface, finding the following simple condition on the above expressions:

$$
\rho_0^d(-\delta/2)T_d = \rho_0^m(-\delta/2)T_m, \quad \rho_0^m(\delta/2)T_m = \rho_0^u(\delta/2)T_u. \tag{2}
$$

Since $T_u < T_m < T_d$, we have $\rho_u > \rho_m > \rho_d$, ensuring that we have an unstable initial condition.

From a phenomenological point of view, the problem we are going to study is the interaction between two turbulent fronts (one originated from the upper density jump and one from the lower) when they come in contact and then evolve together like a single mixed front. Turbulent fluctuations are the driving force of the RT instability, so it is of interest to study how two turbulent flows interact. Moreover, comparison with the one-front RT system is made in order to show how the presence of an "intermediate" well-mixed turbulent layer of fluid can alter the evolution of typical large-scale quantities, such as the mixing layer length and the velocity-temperature fluctuations.

The study is done using a lattice-Boltzmann-thermal (LBT) scheme [12,16] running on a GPU cluster, so our work is interesting also from the point of view of several architecture-specific optimization steps that we have applied to our computer code in order to boost its computational efficiency. This paper is organized as follows: in Sec. II we present the equations of motion; in Sec. III we describe the details of the lattice Boltzmann model (LBM) formulation and its implementation on GPUs. Section IV presents the result of a large-scale analysis and makes a comparison with the "classical" RT evolution. Conclusions close the paper in Sec. V.

II. EQUATIONS OF MOTION

The evolution of a compressible flow in an external gravity field is described by the following Navier-Stokes equations (double indexes are summed upon):

$$
D_t\rho = -\partial_i(\rho u_i)
$$

$$
\rho D_t u_i = -\partial_i P - \rho g\,\delta_{i,z} + \mu \partial_{jj} u_i \tag{3}
$$

$$
\rho c_p D_t T - D_t P = k\partial_{ii} T,
$$

where D_t is the material derivative, μ and k are molecular viscosity and thermal conductivity, respectively, c_p is the constant pressure-specific heat, and ρ, T, P, and **u** are the thermo-hydrodynamical fields of density, temperature, pressure, and velocities, respectively.

In the limit of small compressibility, the parameters depend weakly on the local thermodynamics fields, so expanding pressure around its hydrostatic value $P = p_0 + p$, with $\partial_z p_0 = -g\rho$ and $p \ll p_0$, and performing a small Mach number expansion we can write Eqs. (3) as

$$
D_t\rho = -\partial_i(\rho u_i)
$$

$$
D_t u_i = -\partial_i p/\rho + g\theta/\tilde{T}\delta_{i,z} + \nu\partial_{jj} u_i \tag{4}
$$

$$
D_t T - u_z\gamma = \kappa\partial_{ii} T,
$$

where $\tilde{T}$ is the mean temperature averaged on the whole volume, $\nu = \mu/\rho$ is kinematic viscosity, $\kappa = k/(c_p\rho)$ is thermal diffusivity, and $\gamma = g/c_p$ is the adiabatic gradient for an ideal gas. From this approximation it is clear that only temperature fluctuations $\theta = T - \tilde{T}$ force the system; assuming the adiabatic gradient is negligible, $\gamma \sim 0$, it is well known that starting from an unstable initial condition, as shown in Fig. 1, any small perturbation will lead to a turbulent mix between the cold and hot regions, developing along the vertical direction. If the adiabatic gradient is not negligible, the RT mixing does not proceed forever and stops when the mixing length becomes of the order of the stratification length scale, a further complexity that will not be studied here [15].

III. NUMERICAL METHOD

A. Thermal kinetic model

In this section, we recall the main features of the lattice-Boltzmann model (LBM) employed in the numerical simulations; for full details see Refs. [16,27,28].

For an ideal isothermal fluid, LBM [29–31] can be derived from the continuum Boltzmann equation in the BGK approximation [32], upon expansion in a Hermite velocity space of the single-particle distribution function (PDF) $f(\boldsymbol{x}, \zeta, t)$,
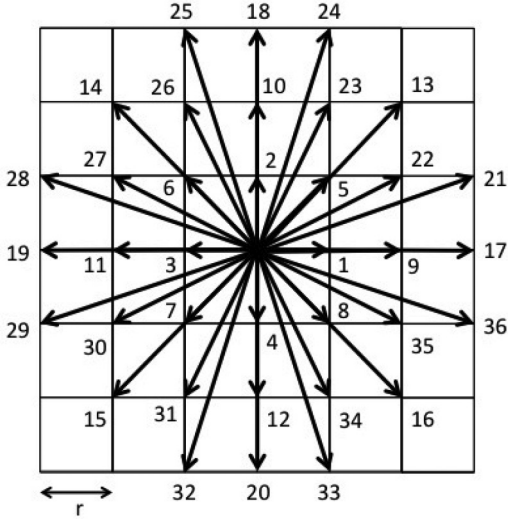
FIG. 2. Scheme of the discretized set of 37 velocities c_l that are used by our LBM to recover the hydrodynamical behavior in the long wave-number limit. $r \sim 1.1969$ is the lattice constant [16].

which describes the probability to find a particle at space-time location $(\boldsymbol{x},t)$ and with velocity ζ [33–35]. Discretization on the lattice is enforced taking a discrete finite set of velocities $\zeta \in [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_M]$, where the total number M is determined by the embedding spatial dimension and the required degree of isotropy [31]. As a result, the dynamical evolution on a discretized spatial and temporal lattice is described by a set of populations $f_l(\boldsymbol{x},t)$ with $l = 1, \ldots, M$.

It has been shown [27,28] that in two dimension with only one set of kinetic populations we need $M = 37$ fields (the so-called *D2Q37* model) to recover in the Chapman-Enskog limit the continuum thermal-hydrodynamical evolution given by Eq. (3). The set of speeds are shown in Fig. 2, while the discretized LBM evolution is given by

$$f_l(\boldsymbol{x} + \boldsymbol{c}_l \Delta t, t + \Delta t) - f_l(\boldsymbol{x},t) = -\frac{\Delta t}{\tau_{LB}}\big[f_l(\boldsymbol{x},t) - f_l^{(eq)}\big]. \quad (5)$$

The left-hand side of Eq. (5) stands for the streaming step of f_l, while the right-hand side represents the relaxation toward a local Maxwellian distribution function $f_l^{(eq)}$, with the characteristic time τ_{LB}. A novelty introduced by the mentioned algorithm is that the equilibrium distribution function directly depends on the coarse grained variables plus a shift due to the local body force term [16,28]:

$$f_l^{(eq)} = f_l^{(eq)}\bigg[\rho, \mathbf{u} + \mathbf{g}\tau_{LB}, T + \frac{\tau_{LB}(\Delta t - \tau_{LB})}{d}g^2\bigg]; \quad (6)$$

the macroscopic fields are defined in terms of the lattice Boltzmann populations as follows:

$$\rho = \sum_l f_l, \quad \rho\,\mathbf{u} = \sum_l \mathbf{c}_l f_l, \quad d\,\rho\,T = \sum_l |\mathbf{c}_l - \mathbf{u}|^2 f_l \quad (7)$$

(d is the space dimensionality). It was shown [16,27] that in order to avoid spurious terms due to lattice discretization and to recover the correct hydrodynamical description from the discretized lattice-Boltzmann variables, momentum and temperature must be renormalized.

This can be obtained by taking for momentum and temperature the following expressions:

$$\mathbf{u}^{(H)} = \mathbf{u} + \frac{\Delta t}{2}\mathbf{g}; \qquad T^{(H)} = T + \frac{(\Delta t)^2 g^2}{4D}.$$

Using these renormalized hydrodynamical fields for a 2D geometry, it is known that it is possible to recover the standard thermo-hydrodynamical equations (4) through a Chapman-Enskog expansion [16,27].

B. GPU optimized algorithm

We have optimized a code that implements the LBM described above, taking into account both performance on one GPU and scaling on a fairly large number of GPUs; our runs have been performed on a cluster based on NVIDIA C2050/C2070 GPUs. GPUs have a large number of small processing elements working in parallel and performing the same sequence of operation; in CUDA, the programming language that we have used throughout, each sequence of instructions operating on different data is called a *thread*. Optimization focuses along three lines: (1) organizing data in such a way that it can be quickly moved between GPU and memory, (2) ensuring that a large number of GPU threads operate independently on different data items with as little interference as possible, and (3) organizing data moves among GPUs minimizing the idle time of each GPU as it waits for data from another node.

Concerning data organization, we first split a lattice of size $L_x \times L_z$ on N_p GPUs along the x dimension; each GPU handles a sublattice of $L_x/N_p \times L_z$ points. Lattice data are stored in memory in column-major order, and we keep in memory two copies of the lattice: at each time step, the code reads from one copy and writes to the other. This choice requires more memory, but it allows us to map one GPU thread per lattice site and then process all threads in parallel. Arrays of LB populations are stored in memory one after the other (this is usually referred to as structure of arrays); this scheme helps *coalescing* memory accesses to different lattice sites that the GPU processes in parallel and helps increase bandwidth.

The physical lattice is surrounded by halo columns and rows; see Fig. 3. For a physical lattice of size $L_x \times L_z$, we
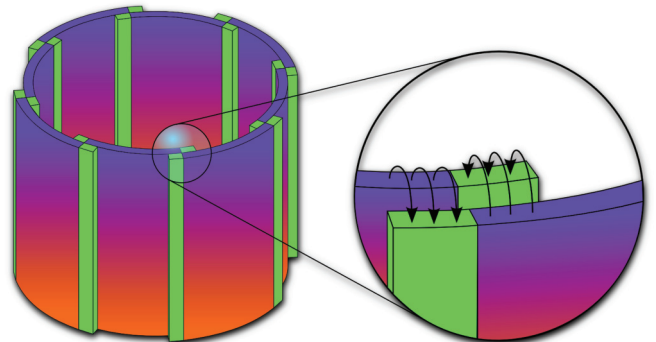

FIG. 3. (Color online) Tiling of the physical lattice with periodic boundary conditions in the x direction on several processing nodes; the picture shows the halo columns that contain copies of the lattice points processed by the neighboring processors.

allocate a grid of $NX \times NZ$ lattice points, $NX = H_x + L_x + H_x$, and $NZ = H_z + L_z + H_z$. H_x, H_z are the sizes of the halos used to establish data continuity between GPUs working on adjoining sublattices and to enforce periodic boundary conditions in the x direction. This makes the computation uniform for all sites, so we avoid thread divergences, badly impacting on performance. We have $H_x = 3$, and $H_z = 16$; the halo in Z is larger than needed by the algorithm, in order to keep data aligned (data words must be allocated in multiples of 32), and to maintain cache-line alignment in multiples of 128 bytes.

As customary for GPUs, the host starts the execution of each time step, corresponding to four main kernels: first, the periodic boundary conditions step exchanges columns of the z-halos, then three steps follow that implement (1) the free propagation (propagate) expressed by the left-hand side of (5), (2) the rigid boundary conditions (bc) at the top and bottom walls, and (i3) the collisions (collide) in the right-hand side of (5). Step propagate moves each population of each site to a different lattice site, according to its velocity vector. Computerwise this corresponds to memory accesses to sparse addresses.

Two options are available: *push* moves all populations from the *same* site to their appropriate destinations, while *pull* gathers populations from neighbor sites to the *same* destination. In the first case one has aligned memory reads and misaligned writes, while the opposite is true in the second case; we have tested both options and then settled for *pull*, which offers $\simeq 20\%$ higher bandwidth. Step bc executes after propagate in order to enforce boundary conditions at the top and bottom of the cell; it adjusts population values at sites with coordinates $z = 0, 1, 2$ and $z = L_z - 3, L_z - 2$ and $L_z - 1$, and its computational impact is fully negligible, so we do not apply significant optimization steps here. Finally, collide performs the collision phase of the LB procedure. Step collide executes in parallel on a large number of threads: no synchronization is necessary because all threads read data from one copy of the lattice (the prv array) and write to the other copy (the nxt array); prv and nxt swap their role at the following time step. Some care is needed to find the optimal number of threads: if this value is too small, available computing resources are wasted, while if it is too large there is not enough space to keep all intermediate data items on GPU registers and performance drops quickly. For each data point collide executes ≈ 7600 double-precision floating-point operations (some of which can be optimized away by the compiler); $\approx 72\%$ of them can be cast as fused multiply adds, in which the operation $a \times b + c$ is performed in just one step.

We now consider the optimization steps for data exchange between different GPUs, i.e., for step pbc of the program. In principle one first moves data from a strip of lattice points of one GPU to the halo region of the logically neighboring GPU, and then executes the streaming phase of the program (i.e., the propagate function): this means that one has to wait till data transfer has completed. They key remark is that fresh halo data is needed only when applying propagate to a small number of lattice sites (close to the halos). We then divide propagate in three concurrent *streams* (in CUDA jargon). One stream handles the most lattice points (those far away from the halos), while the two remaining streams handle
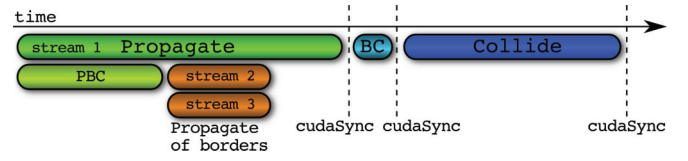

FIG. 4. (Color online) Time schedule of the main step of the complete program; the picture shows that the time associated to data transfer between processing elements can be overlapped with computation.

GPU-to-GPU communications followed by propagate on the data strips close to the halos (one process handles the halo columns at right and the other those at left). In this way the complete program follows the time pattern shown in Fig. 4 and effectively hides almost all communication overheads (up to 32 GPUs, the size of the machine that we have used for our runs).

Over the years we have developed several version of this code optimized for a number of HPC systems. As early as 2010 we had a version [36,37] for the QPACE massively parallel system [38]; more recently, we carefully optimized the code for multicore commodity CPUs [39] and many-core GPUs [40].

Table I summarizes our performance results. This table covers three generations of HPC processors: the early PowerCellX8i of QPACE (2008), the Intel Westmere CPU and the NVIDIA C2050 (2010), and the Intel Sandybridge CPU and NVIDIA K20 (2013). Note that for each processor we consider a code specifically optimized for its architecture, exploiting several levels of parallelism (vector instructions and core parallelism). Table I shows the sustained performance of the full code, the number of lattice-sites updated per second (MLUPS, a user-friendly figure-of-merit for performance), and the approximate value of the energy used to update one site.

Our numbers refer to the performance of just one processing node: this is a reasonable approximation to actual performance up to $\simeq 32$ nodes, since communication overheads can be successfully hidden, as discussed above. Table I shows a significant improvement of performances as newer processors appear; it is interesting to underline that, at a fixed time, GPUs offers two to three times better performance than multicore CPUs, while at the same time being roughly three times more energy efficient.

Further optimizations that we have applied to our code after the present work has been completed offer even higher performances: i.e., a system of two CPUs and two K20X

TABLE I. Performance comparison of our LB code among several architectures, based on the results of Refs. [36,39,40]. 2-WS (2-SB) are Intel dual-processor systems based on the Westmere (Sandy Bridge) micro-architecture; C2050 (K20X) are NVIDIA GPUs based on the Fermi (Kepler) processors.

	QPACE	2-WS	C2050	2-SB	K20X
P (GFlops)	15	60	172	166	412
MLUPS	1.9	7.7	22	21.7	124
E/site (μJ/site)	56	34	10	12	4.2

GPUs now breaks the 1 Tflops barrier for single-node sustained double-precision performance; see Ref. [41]. We conclude this section underlining that the set of optimization steps that we have applied to our 2D code are expected to be equally efficient in 3D code for the code running on each processing node. In this case, memory needs are larger, so it may be necessary to use a larger number of processing nodes, and multinode scaling would have to be studied carefully.

IV. DATA AND ANALYSIS

Here we present the results obtained from the numerical simulations of the 2D $L_x \times L_z$ RT systems putting at the middle of the vertical domain a layer of depth δ at an intermediate temperature $T_m = (T_u + T_d)/2$. For comparison, we also perform simulations for the usual RT configuration, i.e., with $T = T_u$ on the upper half and $T = T_d > T_u$ on the lower half of the cell. In this study we limit ourselves to the case of negligible stratification. Within each layer, temperature values are chosen to set $At \ll 1$. It is important to stress that the algorithm is also applicable to strongly stratified flows [37]. All physical parameters are listed in Table II. The initial width of the intermediate layer is chosen as $\delta \sim L_z/10$, in order to avoid possible confining effects by the vertical boundaries during the merging of the middle temperature layer.

Snapshots of the temperature, temperature gradient and vertical velocity are shown in Fig. 5, taken at three different time points $(t = 0, 2, 3)\tau$ during the evolution; $\tau = \sqrt{L_x/(gAt)}$ is the typical RT normalization time. Vertical temperature profiles

$$\overline{T}(z) = \frac{1}{L_x} \int dx\, T(x,z,t)$$

are also shown in Fig. 6, comparing the double-front RT with the single-front RT.

The triple-density fluid starts to merge under the effect of the instability, causing the development of two fronts in correspondence of the two temperature-density interfaces. These two fronts continue to mix separately the region of the flow between T_u and T_m (referred to as the upper front) and the region between T_m and T_d (referred to as the lower front), until they get in touch at $t \sim 2.5\tau$. At this time, the T_m layer of fluid is greatly eroded, and the flows with temperatures T_u and T_d start to interact. Unlike the classical RT system, in this case we have also the action of turbulent viscosity exerted by one front on the other. This effect leads to a slowing down in the growth of the mixing rate of the double-front experiment with respect to the one-front case.

A useful quantity to describe the mixing layer extension is the *mixing length* $L(t)$, defined as the region where the mean temperature profile is within a given range, e.g., $\overline{T}(z) \in [(1 + a)\, T_u : (1 - a)\, T_d]$ [typically $a = 0.05(T_d - T_u)$]. An alternate way to evaluate $L(t)$ uses the following integral law [1]:

$$L(t) = \int dz\, \Theta\left[\frac{\overline{T}(z,t) - T_u}{T_d - T_u}\right], \qquad (8)$$

where $\Theta[b]$ is a function with a tent map profile:

$$\begin{aligned} \Theta[b] &= 2b \quad 0 < b < 1/2, \\ \Theta[b] &= 2(1 - b) \quad 1/2 < b < 1. \end{aligned} \qquad (9)$$

There is a vast literature based on observation, dimensional analysis, and self-similar assumptions [42,43], which shows that the mixing layer length follows a quadratic evolution in time:

$$L(t) = \alpha(\text{At})g\, t^2, \qquad (10)$$

where α is a dimensionless parameter named "growth rate". Taking the square of the time derivative of Eq. (10), we have the following self-similar scaling [44]:

$$[\dot{L}(t)]^2 = 4\alpha g(\text{At})\, L(t). \qquad (11)$$

Considering our double-front problem, we have to write an expression for $L(t)$ which must take into account the different nature of the system respect to the classical case. For this purpose we use two distinct mixing lengths, one for the upper front, $L_u(t)$, and one for the lower front, $L_d(t)$, defined as

$$\begin{aligned} L_u(t) &= \int dz\, \Theta\left[\frac{\overline{T}(z,t) - T_u}{T_m - T_u}\right], \\ L_d(t) &= \int dz\, \Theta\left[\frac{\overline{T}(z,t) - T_m}{T_d - T_m}\right]. \end{aligned} \qquad (12)$$

Starting from Eq. (10), we can write its double-front counterpart as

$$L_u(t) = \alpha_u(\text{At}_u)g\, t^2, \quad L_d(t) = \alpha_d(\text{At}_d)g\, t^2, \qquad (13)$$

where $\text{At}_u = (T_m - T_u)/(T_m + T_u)$ and $\text{At}_d = (T_d - T_m)/(T_d + T_m)$ are the upper and lower Atwood numbers and α_u and α_d are the upper and lower growth rate, respectively. Within the double mixing length approach, we can define the total mixing length of the fluid as the sum of the upper and lower components, $L_{\text{tot}}(t) = L_u(t) + L_d(t)$. Using the previous expression (13) we have

$$L_{\text{tot}}(t) = \alpha_u g(\text{At}_u)\, t^2 + \alpha_d g(\text{At}_d)\, t^2 \equiv \alpha_{\text{tot}} g(\text{At})\, t^2, \qquad (14)$$

where we have introduced the definition for the total growth rate in this double-layer case as $\alpha_{\text{tot}} = (\alpha_u \text{At}_u + \alpha_d \text{At}_d)/\text{At}$

TABLE II. Parameters for the RT runs. Total Atwood number, $At = (T_d - T_u)/(T_d + T_u)$; upper Atwood number, $At_u = (T_m - T_u)/(T_m + T_u)$; lower Atwood number, $At_d = (T_d - T_m)/(T_d + T_m)$; gravity g; temperature in the upper region, T_u; temperature in the middle region, T_m; temperature in the lower region; T_d; number of independent RT evolution, N_{conf}; adiabatic length $L_{ad} = 2\Delta T/g$; maximum Rayleigh number, $Ra = \Delta T L_z^3 g/\nu\kappa$; Reynolds number, $Re = V L_z/\nu$ ($V = <v_z^2>_x^{1/2}$); Prandtl number, $Pr = \nu/\kappa$.

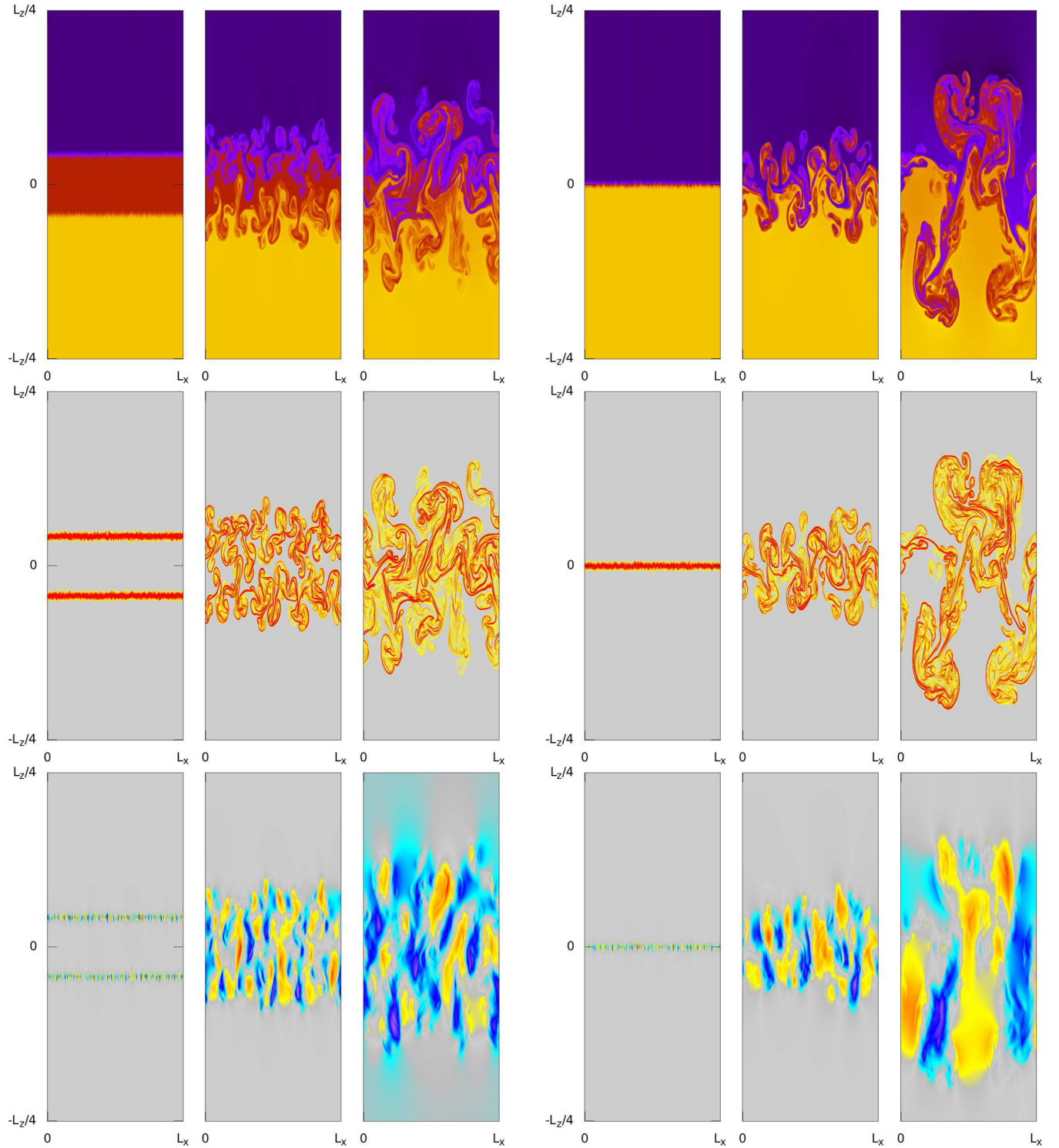| Case	At	At$_u$	At$_d$	L_x	L_z	δ	g	T_u	T_m	T_d	N_{conf}	L_{ad}	Ra	Re	Pr
Single-front	0.025			2400	6144		1.5×10^{-5}	0.975		1.025	7	6666	10^{10}	6×10^4	1
Double-front	0.025	0.0126	0.0123	2400	6144	500	1.5×10^{-5}	0.975	1	1.025	7	6666	10^{10}	6×10^4	1

FIG. 5. (Color online) Snapshot of the temperature (top row), temperature gradients (middle row), and vertical velocity (bottom row) for the double-front (left) and single-front (right) RT at $t/\tau = 0,2,3$, with $\tau = \sqrt{L_x/(g\,\mathrm{At})}$.

An important advantage of Eq. (14) is that it is local in time, so we may extract the coefficient α_{tot} by a simple evaluation of the plateau in the ratio $\dot{L}_{\mathrm{tot}}^2/L_{\mathrm{tot}}$ at each time.

In Fig. 7 we show the temporal evolution of the mixing length $L_{\mathrm{tot}}(t)$ and the growth parameter $\alpha_{\mathrm{tot}}(t)$ for the double-front and single-front RT. In order to make the analysis coherent, we apply the double map tent formulation (12) also

to the single-front case, where now the middle temperature T_m layer is "virtual". Inspection of Fig. 7 shows that, as long as the two fronts evolve separately, the mixing length has the same shape for the single and double-front cases with similar values of α_{tot}, which is a good evidence that in this regime the two fronts proceed independently and in a self-similar way. Differences arise after the two fronts
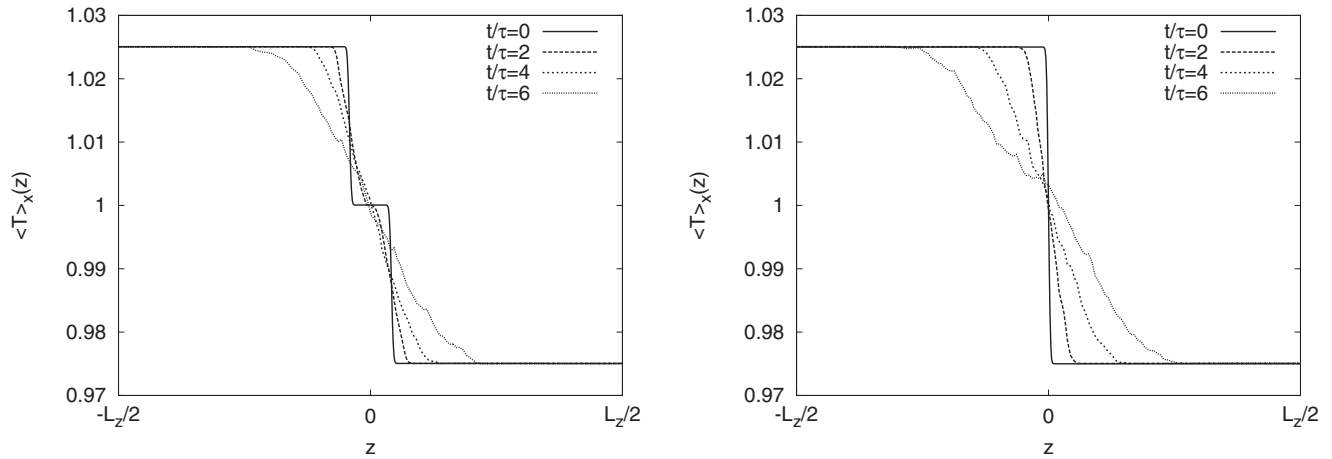
FIG. 6. Temperature profile for the double-front (left) and single-front (right) RT at various times.
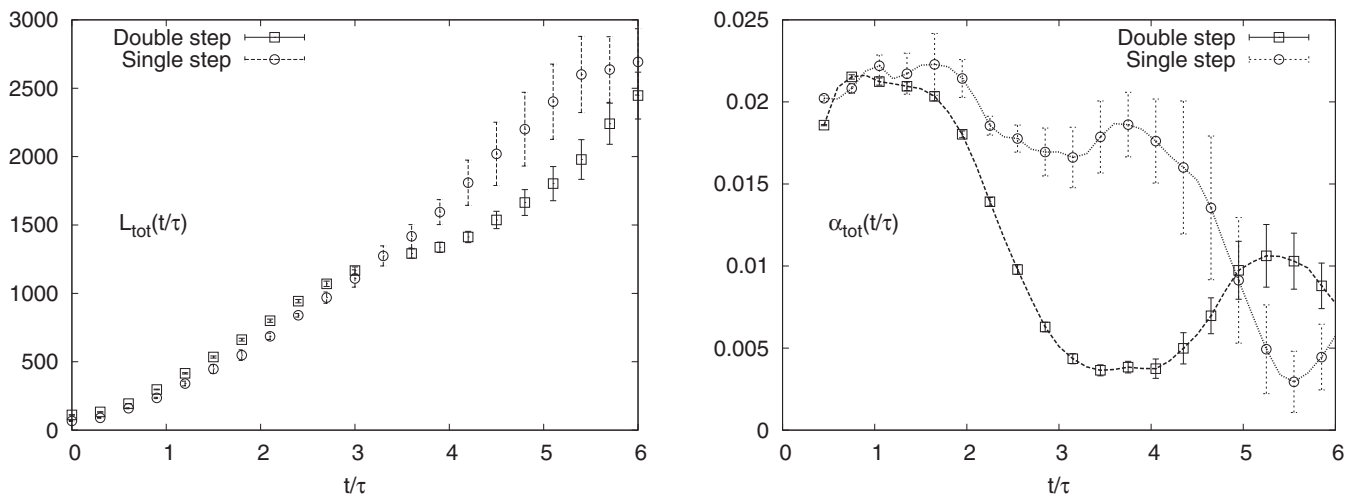

FIG. 7. Mixing layer length $L_{\text{tot}}(t/\tau)$ (left) and asymptotic growth rate $\alpha_{\text{tot}}(t/\tau)$ (right) evolution for the double (square) and single (circle) front RT.
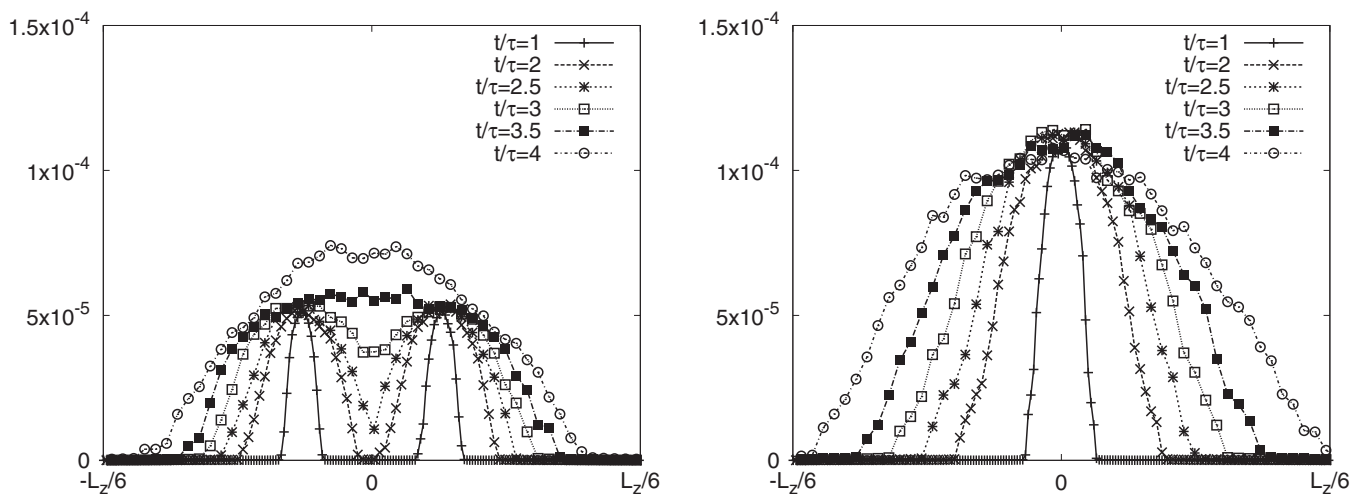

FIG. 8. Temperature fluctuations $\sqrt{\langle (T - \langle T \rangle_x)^2 \rangle_x}$ for the double-front (left) and single-front (right) RT.
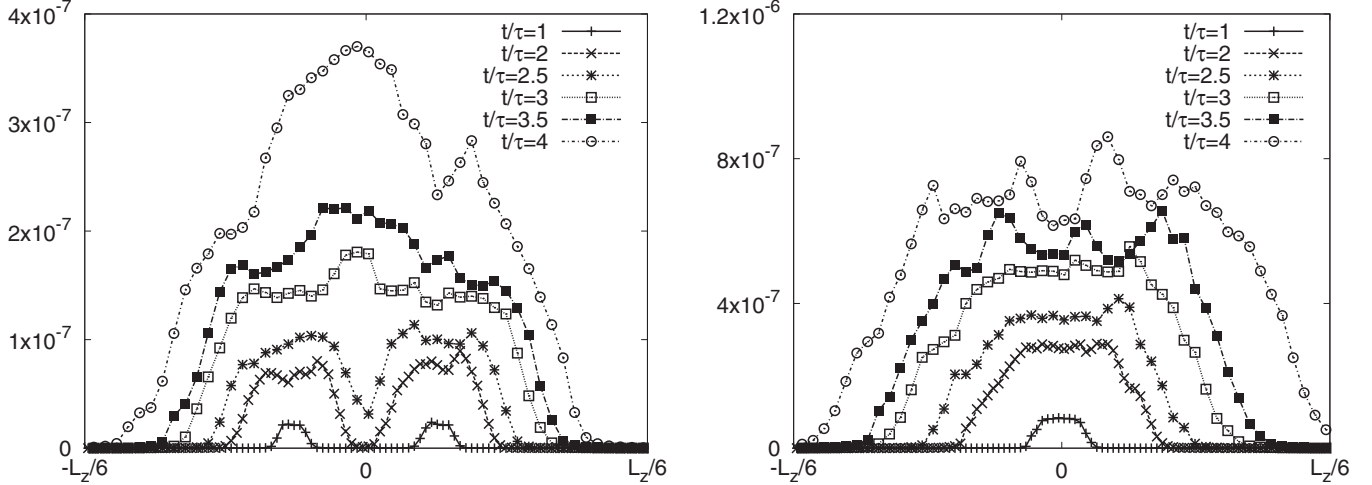
FIG. 9. Kinetic energy fluctuations $\sqrt{\langle (E_k - \langle E_k \rangle_x)^2 \rangle_x}$ for the double-front (left) and single-front (right) RT.

come in contact and start to interact. As we can see the mixing rate of the double-RT decreases with respect to the single one, as is visible from the drops of the curves of L_{tot} and α_{tot} in Fig. 7 around $t/\tau \sim 3$. Evidently the turbulent viscosity generated from one front acts on the other slowing the propagation, as the system spends some amount of energy to fill the temperature or energy gap present between the two fronts (see Figs. 8 and 9 for a one-to-one comparison of the temperature and kinetic energy fluctuations for single-front and double-front). For the classical RT system it is well known that temperature fluctuations remain constant during the development of convection (as we can see in the right panel of Fig. 8); this is also valid for the double-front case until the two turbulent volumes are separated, because we can treat each of them as an independent RT system. When they start to merge, the temperature fluctuations in the region at the center of the cell, between the two fronts, must be "reordered" and brought to the same level of the two peaks, slowing the vertical growth of the mixing layer. After the two fronts are completely merged at $t/\tau \sim 4$, the double-front system behaves like a classic one-front RT with temperatures T_u and T_d, and the mixing layer length starts to increase again with the expected rate, $L_{\text{tot}} \sim t^2$.

V. CONCLUSIONS

In this paper, we have studied a 2D Rayleigh-Taylor turbulence with an intermediate temperature layer in the middle of the vertical domain. The goal of this paper is twofold. First, we have developed a highly optimized GPU-based thermal lattice-Boltzmann algorithm to study hydrodynamical and thermal fluctuations in turbulent single-phase fluids. Second, we have applied it to study the evolution (and collision) of two turbulent RT fronts.

Our results clearly show that at the moment of the collision the vertical evolution of the two front systems slows down and that the long-time evolution is recovered only when a well-mixed region is present in the center of the mixing length. This observation clearly shows the importance on the outer environment on the evolution of any RT system.

Different initial temperature profiles would have lead to a different time for the collision between the two fronts and to a different duration of the intermediate mixing period (where the slowdown is observed). We have checked that the maximum effect is obtained in the configuration analyzed here, i.e., when two fronts of comparable kinetic energy collide. In the case when one of the two fronts is much stronger $\text{At}_u \gg \text{At}_d$ the nonlinear superposition becomes less important. Furthermore, a closer look at the spatial configurations for the one-front and two-front cases presented in Fig. 5 demonstrates that the large-scale behavior recovers a universal evolution in the asymptotic regime, while the temperature and velocity fluctuations at small scales still show important differences, suggesting the possibility of a long-term memory of the initial configuration for high-wave-number modes.

For completeness, we point out that the results presented here are related to the case of "white noise" initial conditions at the RT unstable interface with low values of the Atwood number. Differences can arise for the single-mode initial conditions and if the Atwood number is close to unity; in that case the nonlinear RT instability may be initially described as large-scale bubbles rising up in the heavy fluid with or without mixing, the latter depending on the strength of the secondary Kelvin-Helmotz instability [45–47]. A further generalization of this study to full 3D geometries and to analyze the small-scale properties of the system in the region where the two fronts collide will be presented in a future work.

[1] W. H. Cabot and A. W. Cook, Nature (London) **2**, 562 (2006).

[2] K. Sasaki, N. Suzuki, D. Akamatsu, and H. Saito, Phys. Rev. A **80**, 063611 (2009).

[3] D. Kobyakov *et al.*, Phys. Rev. A **89**, 013631 (2014).

[4] P. R. Spalart and J. H. Watmuff, J. Fluid Mech. **249**, 337 (1993).

[5] E. J. Gutman, K. C. Schadow, and K. H. Yu, Annu. Rev. Fluid Mech. **27**, 375 (1995).

[6] L. Rayleigh, *Scientific Papers*, Vol. 2 (Cambridge University Press, Cambridge, 1900), p. 200.

[7] G. I. Taylor, Proc. R. Soc. Lond. **201**, 192 (1950).

[8] G. Dimonte, D. L. Youngs, and A. Dimits, Phys. Fluids **16**, 1668 (2004).

[9] A. W. Cook and P. E. Dimotakis, J. Fluid Mech. **443**, 69 (2001).

[10] S. B. Dalziel, P. F. Linden, and D. L. Youngs, J. Fluid Mech. **399**, 1 (1999).

[11] M. Chertkov, Phys. Rev. Lett. **91**, 115001 (2003).

[12] L. Biferale *et al.*, Phys. Fluids **22**, 115112 (2010).

[13] G. Boffetta, A. Mazzino, S. Musacchio, and L. Vozzella, Phys. Rev. E **79**, 065301 (2009).

[14] G. Boffetta, F. De Lillo, and S. Musacchio, Phys. Rev. Lett. **104**, 034505 (2010).

[15] L. Biferale *et al.*, Phys. Rev. E **84**, 016305 (2011).

[16] A. Scagliarini *et al.*, Phys. Fluids **22**, 055101 (2010).

[17] E. A. Spiegel, Astrophys. J. **141**, 1068 (1965).

[18] J. Frolich and S. Gauthier, Eur. J. Mech. B/Fluids **12**, 141 (1993).

[19] F. Robinson and K. Chan, Phys. Fluids **16**, 1321 (2004).

[20] L. Biferale *et al.*, Europhys. Lett. **94**, 54004 (2011).

[21] M. Chertkov, V. Lebedev, and N. Vladimirova, J. Fluid Mech. **633**, 1 (2009).

[22] J. W. Jacobs and S. B. Dalziel, J. Fluid Mech. **542**, 251 (2005).

[23] R. I. Nokes, J. Fluid Mech. **188**, 185 (1988).

[24] V. V. Bychkov, S. M. Golberg, M. A. Liberman, and L. E. Eriksson, Phys. Rev. E **54**, 3713 (1996).

[25] J. B. Bell *et al.*, Astrophys. J. **608**, 883 (2004).

[26] M. Modestov, V. Bychkov, R. Betti, and L. E. Eriksson, Phys. Plasmas **15**, 042703 (2008).

[27] M. Sbragaglia *et al.*, J. Fluid Mech. **628**, 299 (2009).

[28] P. C. Philippi, L. A. Hegele, L. O. E. dos Santos, and R. Surmas, Phys. Rev. E **73**, 056702 (2006).

[29] R. Benzi, S. Succi, and M. Vergassola, Phys. Rep. **222**, 145 (1992).

[30] S. Chen and G. Doolen, Ann. Rev. Fluid Mech. **30**, 329 (1998).

[31] D. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models* (Springer, New York, 2000).

[32] P. L. Bhatnagar, E. P. Gross, and M. Krook, Phys. Rev. **94**, 511 (1954).

[33] X. He and L. S. Luo, Phys. Rev. E **56**, 6811 (1997).

[34] N. Martys, X. Shan, and H. Chen, Phys. Rev. E **58**, 6855 (1998).

[35] X. Shan, G. Yuan, and H. Chen, J. Fluid Mech. **550**, 413 (2006).

[36] L. Biferale *et al.*, Proc. Comput. Sci. **1**, 1075 (2010).

[37] L. Biferale *et al.*, Phil. Trans. R. Soc. A **369**, 2448 (2011).

[38] H. Baier *et al.*, Comput. Sci. Res. Dev. **25**, 149 (2010).

[39] F. Mantovani, M. Pivanti, S. F. Schifano, and R. Tripiccione, Comput. Fluids **88**, 743 (2013).

[40] L. Biferale *et al.*, Comput. Fluids **80**, 55 (2013).

[41] J. Kraus, M. Pivanti, S. F. Schifano, R. Tripiccione, and M. Zanella, in *Proceedings of the 25th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2013)* (IEEE, Piscataway, NJ, 2013), pp. 160–167.

[42] K. I. Read, Physica D **12**, 45 (1984).

[43] D. L. Youngs, Physica D **12**, 32 (1984).

[44] J. R. Ristorcelli and T. T. Clark, J. Fluid Mech. **507**, 213 (2004).

[45] P. Ramaprabhu and G. Dimonte, Phys. Rev. E **71**, 036314 (2005).

[46] A. Oparin and S. Abarzhi, Phys. Fluids **11**, 3306 (1999).

[47] V. Bychkov, M. Modestov, V. Akkerman, and L. E. Eriksson, Plasma Phys. Contr. Fusion **49**, B513 (2007).