PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A
MATHEMATICAL, PHYSICAL & ENGINEERING SCIENCES

Numerical simulations of Rayleigh–Taylor front evolution in turbulent stratified fluids

L. Biferale, F. Mantovani, F. Pozzati, M. Sbragaglia, A. Scagliarini, F. Schifano, F. Toschi and R. Tripiccione

Phil. Trans. R. Soc. A 2011 **369**, 2448-2455
doi: 10.1098/rsta.2011.0048

PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

Phil. Trans. R. Soc. A (2011) **369**, 2448–2455
doi:10.1098/rsta.2011.0048

Numerical simulations of Rayleigh–Taylor front evolution in turbulent stratified fluids

By L. Biferale[1], F. Mantovani[2], F. Pozzati[3], M. Sbragaglia[1,*], A. Scagliarini[1], F. Schifano[3], F. Toschi[4,5] and R. Tripiccione[3]

[1]*Department of Physics, University of Tor Vergata and INFN, Rome, Italy*
[2]*Deutsches Elektronen Synchrotron (DESY), 15738 Zeuthen, Germany*
[3]*Department of Physics, University of Ferrara and INFN, Ferrara, Italy*
[4]*Department of Physics, Department of Mathematics and Computer Science and J.M. Burgerscentrum, Eindhoven University of Technology, The Netherlands*
[5]*CNR-IAC, Via Dei Taurini 19, 00185 Rome, Italy*

We present state-of-the-art numerical simulations of a two-dimensional Rayleigh–Taylor instability for a compressible stratified fluid. We describe the computational algorithm and its implementation on the QPACE supercomputer. High resolution enables the statistical properties of the evolving interface that we characterize in terms of its fractal dimension to be studied.

Keywords: Rayleigh–Taylor turbulence; lattice Boltzmann models; stratified fluids

1. Introduction

The lattice Boltzmann method (LBM) is a discrete (in both position and momentum spaces) computational scheme that describes the dynamics of fluids. Key advantages are the relative ease with which complex physics can be implemented in the model, as well as computational efficiency on massively parallel computers. Here, we focus on both aspects; we recall how LBMs are able to describe thermal fluid problems and then describe an efficient implementation on a massively parallel supercomputer based on the IBM PowerXCell-8i processor. The combination of computational accuracy and efficiency allows us to explore with very high resolution the dynamics of the interface during the growth of the mixing layer in the Rayleigh–Taylor problem.

2. The lattice Boltzmann model

In this section, we briefly recall the computational method and the corresponding simulated equations. More details and validation can be found in Scagliarini *et al.* [1] and Biferale *et al.* [2]. The lattice description is given in terms of

*Author for correspondence (sbragaglia@roma2.infn.it).

One contribution of 25 to a Theme Issue ‘Discrete simulation of fluid dynamics: applications’.

lattice populations, $f_l(x,t)$, each characterized by its velocity, $\boldsymbol{c}_l$; the discrete-time evolution of the populations is described by

$$f_l(\boldsymbol{x}+\boldsymbol{c}_l\Delta t, t+\Delta t) - f_l(\boldsymbol{x},t) = -\frac{\Delta t}{\tau}(f_l(\boldsymbol{x},t) - f_l^{(\mathrm{eq})}).$$

Macroscopic density, momentum and temperature are defined in terms of the $f_l(x,t)$: $\rho = \sum_l f_l$, $\rho\boldsymbol{u} = \sum_l \boldsymbol{c}_l f_l$, $D\rho T = \sum_l |\boldsymbol{c}_l - \boldsymbol{u}|^2 f_l$. The kinetic and thermal description of a compressible gas of variable density ρ, velocity $\boldsymbol{u}$, internal energy $\mathcal{K}$ and subject to a local body force $\boldsymbol{g}$ is given by: $\partial_t\rho + \partial_i(\rho u_i) = 0$; $\partial_t(\rho u_k) + \partial_i(P_{ik}) = \rho g_k$; $\partial_t\mathcal{K} + 1/2\partial_i q_i = \rho g_i u_i$, where P_{ik} and q_i are the momentum and energy fluxes. It can be shown that these equations are recovered starting from a continuum Boltzmann equation by introducing an appropriate 'shift' of the velocity and temperature fields in the local equilibrium [3]. Also, lattice discretization induces non-trivial correction terms in the hydrodynamical evolution equations. In particular, both momentum and temperature must be 'renormalized' in order to recover the correct description from the discretized LBM variables: the first correction to momentum is given by the pre- and post-collisional average [4], $\boldsymbol{u}^{(H)} = \boldsymbol{u} + (\Delta t/2)\boldsymbol{g}$, and the first non-trivial correction to temperature by $T^{(H)} = T + (\Delta t)^2 g^2/4D$ ($D = 2$ is the dimensionality of the system; see [3]). Using these 'renormalized' hydrodynamical fields it is possible to recover, through a Taylor expansion in Δt, the thermo-hydrodynamical equations [1,3],

$$D_t\rho = -\rho\partial_i u_i^{(H)}, \tag{2.1}$$

$$\rho D_t u_i^{(H)} = -\partial_i p - \rho g\delta_{i,z} + \nu\partial_{jj}u_i^{(H)} \tag{2.2}$$

and

$$\rho c_{\mathrm{v}} D_t T^{(H)} + p\partial_i u_i^{(H)} = k\partial_{ii}T^{(H)}, \tag{2.3}$$

where we use the material derivative, $D_t = \partial_t + u_j^{(H)}\partial_j$, and neglect viscous heating. In the above, c_{v} is the specific heat at constant volume for an ideal gas, $p = \rho T^{(H)}$, and ν and k are the transport coefficients. We note that similar hydrodynamical equations can be approached with hybrid schemes (see [5] for a review and [6]). In such cases, one has to deal with a mixed lattice Boltzmann finite difference scheme whose competitiveness with respect to our single population lattice Boltzmann scheme settles interesting points for computational research in the future: in the former method the computational advantage of using a smaller number of population fields has to be balanced against the shortcoming of a smaller integration time step (compared with the single population method here used).

3. Lattice Boltzmann method implementation on QPACE

In this section, we describe our parallel implementation of the LBM algorithm described above, which we tuned for the QPACE computer and used for a large simulation campaign in spring 2010. QPACE [7,8]—currently ranking as the top entry of the Green500 list (http://www.green500.org/)—is a massively parallel computer optimized for compute-intensive computational physics applications. Its processing nodes use the IBM PowerXCell-8i processor, a multi-core

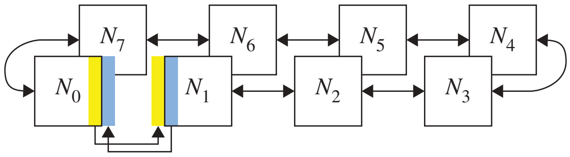

Figure 1. Graphical representation of a ring of QPACE nodes. Each node exchanges data with its nearest neighbours. The `comm()` procedure exchanges sites sitting at the borders of the sub-lattice, as shown for nodes N_0 and N_1. (Online version in colour.)

heterogeneous processor with a peak performance of about 100 Gflops. Nodes are interconnected by a three-dimensional toroidal network. The processor choice and the interconnection topology make QPACE a somewhat unusual architecture; however, our work is relevant in a more general framework, since (i) the pattern of data traffic associated with our parallel implementation has a simple structure so it can be quickly re-coded for virtually any other network structure and (ii) the multi-core architecture of the cell processor is becoming more and more common to all high-performance engines, so our results are a valuable starting point for other implementations, e.g. for the multi-core processors recently introduced by Intel.

Our code belongs to the D2Q37 class, so each lattice site is described by a structure with several members, containing the double precision floating array `p[]` of 37 LBM populations and just a few additional variables. In order to use all the performance of the QPACE system, parallelism has to be exploited at several levels. First, the lattice has to be split over the processing nodes. We split our lattice of size $L_x \times L_y$ on N_p nodes along the x dimension, that is, we allocate on each node a sub-lattice of size $L_x/N_\mathrm{p} \times L_y$. This splitting makes the parallelization of the code easier: the overhead introduced with respect to a different splitting is negligible.

The LB code evolves over the system for many time steps. At each step, each site is processed in several computational phases. We allocate two copies of the lattice in the memory: each phase reads operands from one copy and writes results to the other. We have four processing phases within the loop over the time steps, as follows.

1. `comm()`: this step moves data between the nodes on the machine. Indeed, processing sites at the borders of each sub-lattice require that data are stored on neighbour nodes. We maintain an updated copy of the required sites sitting at the border of each sub-lattice (figure 1). Routine `comm()` copies data associated with these border sites using QPACE communication routines. As discussed above, rewriting this routine for a different communication package requires little effort.

2. `stream()`: for each lattice point, this phase gathers the population elements of neighbour points (up to three lattice sites away) whose velocities cause them to drift to the current site. This phase accesses non-contiguous memory locations according to a fairly complex addressing scheme.

3. `bc()`: this phase enforces boundary conditions (e.g. constant temperature) at the top and bottom of the lattice.

4. `coll()`: for each site, this phase performs the collision among the populations gathered by the `stream()` phase. This step is computationally intensive and completely local (i.e. arithmetic operations use only those

populations in the site). A further level of parallelism is within each node: we further split the sub-lattice among the processor cores (called synergistic processing units (SPUs) in CELL jargon), so that each SPU handles a strip within the sub-lattice assigned to the node. `coll()` uses a double-buffer scheme to overlap memory access and computation: we move fresh data from the main memory to each local store inside the SPUs (and vice versa) concurrently with the computation. This approach is used also by `stream()`; here, we exploit the parallelism owing to the SPUs and a multiple-buffer scheme to implement a pipeline of direct memory access operations that allow us to perform the time-consuming scattered accesses to memory in the local store (as opposed to the main memory).

The CELL processor offers yet a third level of parallelism, namely *vector operations*, i.e. the possibility to perform Single Instruction, Multiple Data (SIMD) operations on data types of 128 bits (a pair of double precision float values). We exploit this feature by pairing all components of the data structure of two adjacent sites in one vector (`vector double`), so all independent arithmetic operations are performed concurrently in SIMD mode.

4. Performance

We summarize the performance of our code in table 1; we list the time spent in each phase of the program for one time step (i.e. one full sweep of the lattice) for several lattice sizes and for a QPACE configuration of 32 processors (the typical configuration used in our extensive runs, even if, occasionally, we used larger systems with up to 128 nodes). Some comments are in order. (i) `coll()` is the most time-consuming part of our code. This is the floating point-intensive part of the program. (ii) The performance bottleneck is mostly in routine `stream()`, which accesses memory at sparse addresses and performs a negligible amount of computation. (iii) Establishing the appropriate boundary conditions (routine `bc()`) has a negligible impact on performance. (iv) Moving data among processors (routine `comm()`) has a limited impact on performance, not larger than approximately 5 per cent. (v) The last column in table 1 lists values of $T_{\mathrm{p}} = T(N_{\mathrm{p}}/L_x L_y)$, that is, the total time spent by the program for each site of the lattice, independently of the number of processors. For perfect scaling, this number should remain constant: fluctuations less than or equal to 10 per cent show very good scalability for physically interesting configurations. (vi) We can make the previous point more accurate with a rough performance model that takes into account the fact that the time spent in communication should depend only on L_y (and not on the number of processors), while all the other phases are in principle fully parallelizable. One writes then $T = c_1 L_y + c_2 L_x L_y / N_{\mathrm{p}}$, where T is the total execution time, N_{p} is the number of processors and c_1 and c_2 are fitted constants. We find a good fit with $c_1 \approx 0.003\,\mathrm{ms}$ and $c_2 \approx 0.00049\,\mathrm{ms}$. Rewriting $T/L_y = c_1 + c_2 L_x / N_{\mathrm{p}}$, we find that the communication phase is the limiting factor (in the spirit of Amdahl's Law) to scaling; however, inserting the actual values for c_1 and c_2, we see that serious violations to scaling occur only when the second term becomes comparable to the first, e.g. for $L_x/N_{\mathrm{p}} \leq 6$, a very unrealistic case indeed. (vii) Our sustained performance is 14–17% of

Table 1. Breakdown of the execution time for each processing phase. All numbers are in milliseconds (microsecond for the last line). T_p, defined as $T(N_\mathrm{p}/L_x L_y)$, should be a constant for perfect scaling; it varies by approximately 10%.

	2048 × 3600	4096 × 6000	4096 × 16000
`comm()`	11.6	19.4	51.4
`stream()`	37.2	135.0	360.2
`bc()`	0.8	1.7	1.7
`coll()`	71.8	239.3	637.9
total	121.4	451.1	1051.2
T_p	527	489	513

peak; our implementation has reached a reasonable level of performance for a real-life production code. Further improvements—mainly merging stream and collision, to avoid unnecessary memory access—may improve performance by up to approximately 20 per cent.

5. Simulating the Rayleigh–Taylor system

We have used the LBM code described here for an extensive series of runs on QPACE, characterizing the Rayleigh–Taylor instability that develops when a heavy fluid is superposed above a lighter one in a constant gravity field. The Rayleigh–Taylor instability has applications in different fields, while the physics beyond the instability still has several open problems. Small-scale fluctuations may differ in two- or three-dimensional geometries, but the large-scale mixing layer growth is expected not to change its qualitative evolution [9,10]. In our simulations, the starting configuration is a single component compressible flow in a two-dimensional tank of size $L_x \times L_z$, with adiabatic and no-slip boundary conditions at the top and bottom walls and periodic boundary conditions on the vertical boundaries. The initial interface is at height $z=0$, in a box extending from $z=-L_z/2$ to $z=+L_z/2$. In the two half volumes the temperature is initially constant, with the corresponding hydrostatic density profiles, ρ_0, verifying $\partial_z p_0(z) = -g\rho_0(z)$. The full solution then has an exponentially decaying behaviour in the two half volumes, each one driven by its own temperature value. The unstable initial hydrostatic configuration is given by

$$\left.\begin{aligned} &T_0(z) = T_\mathrm{u}; \quad \rho_0(z) = \rho_\mathrm{u} \exp\left(\frac{-g(z-z_c)}{T_\mathrm{u}}\right); \quad z>0 \\ \text{and} \quad &T_0(z) = T_\mathrm{d}; \quad \rho_0(z) = \rho_\mathrm{b} \exp\left(\frac{-g(z-z_c)}{T_\mathrm{d}}\right); \quad z<0. \end{aligned}\right\} \tag{5.1}$$

Equilibrium requires the same pressure at the interface, $z=z_c=0$, which translates into a simple condition on the pre-factors: $\rho_\mathrm{u} T_\mathrm{u} = \rho_\mathrm{b} T_\mathrm{d}$. As $T_\mathrm{u} < T_\mathrm{d}$, at the interface $\rho_\mathrm{u} > \rho_\mathrm{b}$. We have performed three sets of simulations (parameters

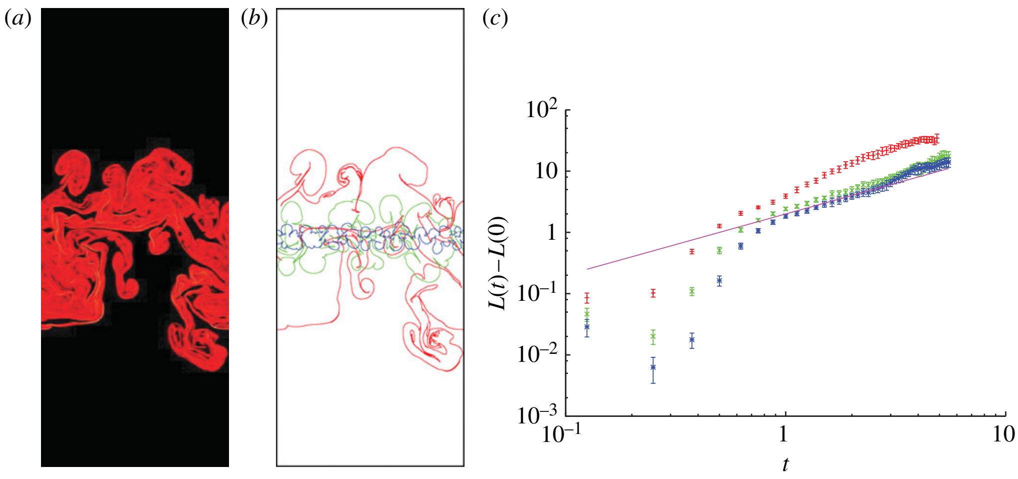

Figure 2. (a) A typical Rayleigh–Taylor configuration shaded according to the local values of the temperature gradient squared. Pixels are shaded in grey when above threshold, in black otherwise. (b) Starting from shaded configurations the algorithm for the interface reconstruction is applied in order to extract the external hull. Represented here are three different times, $t/\tau = 1.25, 2.5$ and 5 (the last corresponds to the same configuration as in (a)). (c) Averaged hull length minus its initial value, $L(t) - L(0)$. We show the hull growth for different values of the threshold, respectively $K = 10^{-7}$ (plus symbols), 10^{-8} (crosses) and 10^{-9} (asterisks), and x is represented by the continuous line (see text). The overall behaviour is insensitive to the threshold chosen and the interface *length* grows linearly in time except for in its very early stages. (Online version in colour.)

Table 2. Parameters for the Rayleigh–Taylor runs. Atwood number, $At = (T_\mathrm{d} - T_\mathrm{u})/(T_\mathrm{d} + T_\mathrm{u})$; viscosity, ν; gravity, g; temperature in the upper half region, T_u; temperature in the lower half region, T_d; normalization time, $\tilde{\tau} = \sqrt{L_x/(g\,At)}$.

	At	L_x	L_z	ν	g	T_u	T_d	$\tilde{\tau}$
run A	0.05	1024	2400	0.001	2.5×10^{-4}	0.95	1.05	9×10^3
run B	0.4	1664	4400	0.1	1×10^{-4}	0.6	1.4	6.5×10^3
run C	0.05	1024	2400	0.005	5×10^{-5}	0.95	1.05	2×10^4

are summarized in table 2): (i) with a large enough adiabatic gradient (but small Atwood number), in order to address stratification effects on the mixing layer growth, while still being very close to the Boussinesq approximation; (ii) with large Atwood number, in order to describe compressibility effects, outside the Boussinesq regime but far from the adiabatic profile; and (iii) with small adiabatic gradient and small Atwood number, to compare with (ii). In figure 2, we show snapshots of the typical interface evolution during the Rayleigh–Taylor instability, displaying all the complexity of the phenomena. For further details on the simulations see Scagliarini *et al.* [1].

6. Mixing layer hull

During the evolution of the Rayleigh–Taylor instability, the mixing layer grows into a complex geometrical object characterized by plumes and entrainment regions (figure 2). Here, we characterize the statistical properties of the hull embedding the mixing layer, and try to quantify its time evolution and fractal dimension. It is evident that the mixing layer itself is non-homogeneous, making the definition of a mixing layer thickness somehow questionable. In order to perform any analysis, the first issue to solve is how to properly identify the external hull of the mixing layer. We proceed in a simple way, as detailed in the following. First, we identify (flag) the regions where the gradient of the temperature field is above some prescribed threshold: $|\nabla T(\boldsymbol{x})|^2 > K$. This produces a rough indication of the interface position (figure 2*a*). From the resulting structure, we extract the top and bottom hull by means of a refined biased walk algorithm [11]. This algorithm has two steps, as follows. A walker moves vertically until the hull is hit (one walker is released from below and one from above). Once the hull is hit (i.e. the next pixel is a flagged one) the walker proceeds by performing a biased walk to explore the hull moving to the right, while another walker moves to the left. The whole procedure is then repeated for all possible horizontal starting positions of the vertically landing walker. This simple algorithm is able to identify the hull even when the latter has non-connected regions. Typical results of the algorithm are shown in figure 2*b*. We measure two different quantities: the average length of the interface and its fractal dimensions. The total length of the hull (including both top and bottom hulls) is measured as a function of time. As the length of the interface fluctuates, we take the ensemble average over all our independent runs. At the start of the simulation the length of the hull is close to $2 \cdot L_x$, so it is more interesting to consider $L(t)$ minus its initial value, $\Delta L(t) = L(t) - 2L_x$. We find that the quantity $\Delta L(t)$, except for a short early stage, grows linearly with time t and its growth is independent of the threshold chosen, as shown in figure 2*c*. To go beyond the average length, one can quantify the geometrical measure of the hull by means of its fractal dimension. To this aim, we measure the mass density $M_q(r) = 1/r^2 \int m(x)^q \mathrm{d}^2 x$ over tiles of different size r, and extract a scaling behaviour $M_q(r) \sim (r/L_x)^{-\mu_q}$ with $\mu_q = (2 - D_q)(q - 1)$, where D_q is the generalized dimensions of order q. We find that the interface is not fractal, as all fractal measures coincide with unity within errors.

7. Conclusions

We have presented state-of-the-art numerical investigations of two-dimensional compressible and stratified Rayleigh–Taylor turbulence. We have provided details of an efficient implementation on the QPACE supercomputer and preliminary analysis of the time behaviour of the statistical properties of the evolving interface. Future development will consider extending the algorithm to three dimensions. This step will require a considerable increase in the number of lattice speeds in order to maintain the desired degree of isotropy for the heat flux (this number may increase up to around 100 speeds to maintain the on-lattice constraint). If one removes the stipulation that lattice velocities are defined only on grid points and one also allows for off-lattice discretized velocity sets, the number of vectors needed to recover isotropy for moments up to order eight can be reduced [12].

We thank the QPACE development team for support during the implementation of our code and execution of the simulations. We furthermore acknowledge access to QPACE and eQPACE.

References

1 Scagliarini, A., Biferale, L., Sbragaglia, M., Sugiyama, K. & Toschi, F. 2010 Lattice Boltzmann methods for thermal flows: continuum limit and applications to compressible Rayleigh–Taylor systems. *Phys. Fluids* **22**, 055101. (doi:10.1063/1.3392774)

2 Biferale, L., Mantovani, F., Sbragaglia, M., Scagliarini, A., Toschi, F. & Tripiccione, R. 2010 High resolution numerical study of Rayleigh–Taylor turbulence using a thermal lattice Boltzmann scheme. *Phys. Fluids* **22**, 115112. (doi:10.1063/1.3517295)

3 Sbragaglia, M., Benzi, R., Biferale, L., Chen, H., Shan, X. & Succi Journ, S. 2009 Lattice Boltzmann method with self-consistent thermo-hydrodynamic equilibria. *J. Fluid Mech.* **628**, 299–309. (doi:10.1017/S002211200900665X)

4 Guo, Z., Zheng, C. & Shi, B. 2002 Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Phys. Rev. E* **65**, 046308. (doi:10.1103/PhysRevE.65.046308)

5 Cates, M. E., Henrich, O., Marenduzzo, D. & Stratford, K. 2009 Lattice Boltzmann simulations of liquid crystalline fluids: active gels and blue phases. *Soft Matter* **5**, 3791–3800. (doi:10.1039/b908659p)

6 Gonnella, G., Lamura, A., Piscitelli, A. & Tiribocchi, A. 2010 Phase separation of binary fluids with dynamic temperature. *Phys. Rev. E* **82**, 046302. (doi:10.1103/PhysRevE.82.046302)

7 Goldrian, G. *et al.* 2008 Quantum chromodynamics parallel computing on the cell broadband engine. *Comput. Sci. Eng.* **10**, 46–54. (doi:10.1109/MCSE.2008.153)

8 Baier, H. *et al.* 2009 QPACE: a QCD parallel computer based on cell processors. In *Proc. XXVII Int. Symp. on Lattice Field Theory, Beijing, China, 25–31 July 2009*.

9 Chertkov, M. 2003 Phenomenology of Rayleigh–Taylor turbulence. *Phys. Rev. Lett.* **91**, 115001. (doi:10.1103/PhysRevLett.91.115001)

10 Cabot, W. 2006 Comparison of two- and three-dimensional simulations of miscible Rayleigh–Taylor instability. *Phys. Fluids* **18**, 045101. (doi:10.1063/1.2191856)

11 Stauffer, D. & Aharony, A. 1994 *Introduction to percolation theory*, 2nd edn. London, UK: Taylor and Francis.

12 Surmas, R., Pico Ortiz, C. E. & Philippi, P. C. 2009 Simulating thermohydrodynamics by finite difference solutions of the Boltzmann equations. *Eur. Phys. J. Special Topics* **171**, 81–90. (doi:10.1140/epjst/e2009-01014-x)