ESAIM: PROCEEDINGS AND SURVEYS, September 2014, Vol. 45, p. 290-299
J.-S. Dhersin, Editor

TIME-IMPLICIT HYDRODYNAMICS FOR EULER FLOWS *

Serge Van Criekingen[1], Edouard Audit[1], Jeaniffer Vides[2] and Benjamin Braconnier[3]

Abstract. We consider the Euler equations with gravity source terms, and derive a time-implicit resolution scheme from the explicit one developed by Vides et al. [8]. This requires computing a Jacobian matrix, which is done symbolically using the automatic differentiation tool TAPENADE developed at Inria. The resulting sparse linear system is solved using the PETSc library. We present parallel numerical results for a 2-D Rayleigh-Taylor instability on up to 4096 CPU cores.

Résumé. Nous considérons les équations d'Euler avec des termes sources décrivant la gravitation, et nous construisons un schéma de résolution implicite en temps sur base de celui, explicite, développé par Vides et al. [8]. Le calcul d'une matrice jacobienne est requis, et il est réalisé symboliquement en utilisant l'outil de différentiation automatique TAPENADE developpé à Inria. Le système linéaire obtenu est résolu via la bibliothèque parallèle PETSc. Nous présentons des résultats numériques pour l'instabilité de Rayleigh-Taylor en deux dimensions, obtenus en utilisant jusqu'à 4096 cœurs CPU en parallèle.

Introduction

The starting point of this study is the explicit scheme developed by Vides et al. [8] for the Euler-Poisson model, i.e., the Euler equations supplemented by gravitational effects. This development was done in view of numerically simulating gravitational flows in astrophysics, and implemented within the HERACLES [4, 7] astrophysics code. The scheme in [8] is based on a finite volume discretization with a Godunov-type solver for the Euler equations, and a finite difference discretization with standard Krylov-type solver for the Poisson equation. The specificity of this scheme is that the Godunov-type solver for the Euler equations takes into account the gravitational effects. Moreover, it uses a relaxation procedure for the thermodynamic pressure and the gravitational potential.

We here aim at developing an implicit version of the explicit scheme developed by Vides et al. However, in this first study, we do not consider the full Euler-Poisson model, but rather focus on the Euler equations with gravity source terms. The interest of developing an implicit scheme is to overcome the restriction due to the CFL condition. This condition can be very restrictive in systems in gravitational equilibrium, e.g. stars, where movements are typically very slow. Similarly to [8], our developments are made within the HERACLES [4, 7] code.

* *The authors wish to thank Mikolaj Szydlarski for fruitful discussions.*

[1] CEA, Maison de la Simulation, USR 3441, CEA - CNRS - INRIA - Univ. Paris Sud - UVSQ, Gif-sur-Yvette, France
[2] INRIA, Maison de la Simulation, USR 3441, CEA - CNRS - INRIA - Univ. Paris Sud - UVSQ, Gif-sur-Yvette, France
[3] IFP Energies Nouvelles, Rueil-Malmaison, France

Article published online by EDP Sciences and available at http://www.esaim-proc.org or http://dx.doi.org/10.1051/proc/201445030

In our study, the Jacobian matrix, inherent to any implicit scheme, is computed symbolically by automatic differentiation using the TAPENADE code [5,6]. As for the resulting linear system, it is solved using PETSc, a suite of data structures and routines for the scalable parallel solution of scientific applications modeled by partial differential equations, which has been used in many large-scale application projects [1–3].

The paper is organized as follows. In Section 1, the Euler equations with gravity source terms are introduced, as well as most of our notations. Then, Section 2 briefly describes the explicit setting of the scheme in [8] for which we propose an implicit version, described in Section 3. Numerical results are presented in Section 4, for a classical Rayleigh-Taylor instability test case in two dimensions. Note that we here do not perform a full performance comparison between the implicit and explicit schemes. As explained in Section 4, such a comparison would be fair only given a clear target result of interest, which is not the case in this general presentation of first implicit results. However, we investigate the weak and strong scalability of both explicit and implicit schemes on up to 4096 CPU cores.

1. Mathematical Model

The Euler equations with gravity source terms can be written as

$$\left\{\begin{array}{lcll} \partial_t \rho & + & \nabla\cdot(\rho\mathbf{u}) & = & 0 \\ \partial_t \rho\mathbf{u} & + & \nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u}+p) & = & -\rho\nabla\phi \\ \partial_t \rho E & + & \nabla\cdot((\rho E+p)\mathbf{u}) & = & -\rho\mathbf{u}\cdot\nabla\phi \end{array}\right. \tag{1}$$

where $\rho > 0$ is the density, $\mathbf{u} \in \mathbb{R}^d$ the velocity (with d the space dimension), E the specific energy and ϕ a smooth gravity potential. The thermodynamic pressure p is modeled as a function of the density ρ and the internal specific energy $\epsilon = E - |\mathbf{u}|^2/2$. The pressure law can typically be the perfect gas equation of state $p(\rho,\epsilon) = (\gamma-1)\rho\epsilon$ with γ a given constant.

For notational simplicity, we group the hydrodynamic unknowns into the state vector $\mathbf{W} = (\rho, \rho\mathbf{u}, \rho E)^T$, such as to rewrite (1) as

$$\partial_t \mathbf{W} + \nabla\cdot\mathbf{F}(\mathbf{W}) + \mathbf{B}(\mathbf{W})\nabla\phi = 0, \tag{2}$$

where

$$\begin{aligned} \mathbf{F}(\mathbf{W}) &= \left(\rho\mathbf{u}, \rho\mathbf{u}\otimes\mathbf{u}+p, (\rho E+p)\mathbf{u}\right)^T, \\ \mathbf{B}(\mathbf{W}) &= \rho\left(\mathbf{0}_d, \mathbf{e}_1, ..., \mathbf{e}_d, \mathbf{u}\right)^T, \end{aligned} \tag{3}$$

with $\mathbf{0}_d$ and $\mathbf{e}_i$ being respectively the null and i$^{\text{th}}$ canonical vector in $\mathbb{R}^d$.

Lastly, the admissible state vectors form the convex set Ω_d:

$$\Omega_d = \left\{\mathbf{W}\in\mathbb{R}^{2+d};\ \rho>0,\ \mathbf{u}\in\mathbb{R}^d,\ E - \frac{|\mathbf{u}|^2}{2} > 0\right\}. \tag{4}$$

2. Reminder of the Explicit Scheme

In [8], the Euler equations with gravity (1) are handled with a finite volume discretization and a Godunov-type solver which takes into account the gravitational effects. For notational simplicity, we here adopt a 1-D setting. Assuming a uniform mesh with cells of size Δx, the state vector in the mesh cell of index i evolves from time t^n to time $t^{n+1} = t^n + \Delta t$ as

$$\mathbf{W}_i^{n+1} = \mathbf{W}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+\frac{1}{2}}^{L,n} - \mathbf{F}_{i-\frac{1}{2}}^{R,n}\right) \tag{5}$$

where $\mathbf{F}_{i+\frac{1}{2}}^{L,n}$ and $\mathbf{F}_{i-\frac{1}{2}}^{R,n}$ are the left and right numerical fluxes, as in the traditional finite volume framework. These numerical fluxes are obtained by solving Riemann problems at the cell interface $x_{i\pm\frac{1}{2}}$, such that $\mathbf{F}_{i+\frac{1}{2}}^{L,n}$

depends on $\mathbf{W}_i^n$ and $\mathbf{W}_{i+1}^n$, and $\mathbf{F}_{i-\frac{1}{2}}^{L,n}$ depends on $\mathbf{W}_i^n$ and $\mathbf{W}_{i-1}^n$. Note that the discretized gravity source term is introduced into the approximate Riemann solver, as described in [8]. Note furthermore that the resulting numerical fluxes are not conservative (i.e., $\mathbf{F}_{i\pm\frac{1}{2}}^{L,n} \neq \mathbf{F}_{i\pm\frac{1}{2}}^{R,n}$) due to the gravity term.

3. Implicit Scheme

Formally, changing from an explicit to an implicit scheme consists only in considering the fluxes on the right-hand side of (5) at time step $n+1$ instead of n:

$$\mathbf{W}_i^{n+1} = \mathbf{W}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+\frac{1}{2}}^{L,n+1} - \mathbf{F}_{i-\frac{1}{2}}^{R,n+1}\right). \tag{6}$$

We follow a linearly implicit approach leading to a Jacobian linear system, as we now describe. First we define

$$\mathcal{F}_i = \frac{1}{\Delta x}\left(\mathbf{F}_{i+\frac{1}{2}}^{L,n+1} - \mathbf{F}_{i-\frac{1}{2}}^{R,n+1}\right).$$

Note that, through the Riemann solvers, $\mathcal{F}_i$ can be seen as a function of the state vector values at time step $n+1$ such that from equation (6) we may write

$$\frac{\mathbf{W}_i^{n+1} - \mathbf{W}_i^n}{\Delta t} = -\mathcal{F}_i(\mathbf{W}_i^{n+1}, \mathbf{W}_{i\pm1}^{n+1})$$

for each mesh cell i. The equivalent for the whole mesh of this last expression reads

$$\frac{\mathbf{W}^{n+1} - \mathbf{W}^n}{\Delta t} = -\mathcal{F}\left(\mathbf{W}^{n+1}\right).$$

where $\mathcal{F}$ is the global equivalent to $\mathcal{F}_i$, including the appropriate boundary conditions. A first-order Taylor approximation applied to $\mathcal{F}$ then yields

$$\frac{\mathbf{W}^{n+1} - \mathbf{W}^n}{\Delta t} \approx -\mathcal{F}(\mathbf{W}^n) - \frac{\partial \mathcal{F}}{\partial \mathbf{W}}\left(\mathbf{W}^{n+1} - \mathbf{W}^n\right)$$

where $\frac{\partial \mathcal{F}}{\partial \mathbf{W}}$ is the Jacobian matrix. We then have

$$\mathcal{J}\left(\mathbf{W}^{n+1} - \mathbf{W}^n\right) \approx -\mathcal{F}(\mathbf{W}^n) \tag{7}$$

where $\mathcal{J}$ is defined as

$$\mathcal{J} = \frac{\mathcal{I}}{\Delta t} + \frac{\partial \mathcal{F}}{\partial \mathbf{W}}$$

with $\mathcal{I}$ the identity operator. Note that $\mathcal{J}$ is not symmetric, but block symmetry is achieved.

At each time step, $\mathcal{J}$ needs to be evaluated and the Jacobian linear system (7) needs to be solved. To evaluate $\frac{\partial \mathcal{F}}{\partial \mathbf{W}}$, we utilize the automatic differentiation engine TAPENADE developed at Inria [5, 6]: given a Fortran or C code that computes a function, TAPENADE creates a new code that computes its derivatives. As for solving the Jacobian linear systems, the PETSc library [1–3] is used, this tool being particularly well adapted to handle such large sparse systems in parallel. In the numerical results below, the BiCGStab method of PETSc was employed, with global block-Jacobi preconditioning and one sweep of Incomplete Lower-Upper with no fill-in (ILU(0)) within each block.

4. Numerical Tests

4.1. Benchmark Presentation

We consider the Rayleigh-Taylor instability in 2-D. This instability consists in the fall of a dense fluid into a lighter one due to gravitation. Initially, both fluids are totally segregated, with the dense one on top. We here consider a 2-D model of this experiment, in the $x-y$ plane, with gravity acting opposite to the x-direction. The computational domain is $(0, L_x) \times (0, L_y)$ with $L_x = 4\,m$, $L_y = 1\,m$, and the uniform gravity potential is given by $\phi = g\,x$ with $g = (-10, 0)\,m.s^{-2}$. At time $t = 0$, the interface between the two fluids is perturbed as follows: the dense fluid fills all the points in the domain for which $x > \frac{1}{2}L_x\left(1 - \frac{1}{10}cos((\frac{y}{L_y} - \frac{1}{2})\pi)\right)$. Both fluids are assumed to be governed by a perfect gas equation of state: $p = (\gamma - 1)\rho\epsilon$ where $\gamma = 1.4$. The dense fluid's density is set to $\rho = 2\ kg.m^{-3}$, while the light fluid's one is set to $\rho = 1\ kg.m^{-3}$. As for boundary conditions, we take reflective conditions in both directions.

With such settings, the experiment runs as displayed on Figures 1 to 3. These figures show the evolution of the instability from $t = 0s$ to $t = 7s$, as modeled by the explicit (on the left) and implicit (on the right) schemes, for different spatial refinements. We will come back to these figures after giving some remarks on the discretization used.

4.2. Discretization

For the spatial discretization, we employ a uniform mesh with square cells (edge length $\Delta x = \Delta y$) indexed with the subscripts (i, j). For the time discretization, the time step length is re-evaluated at the end of each time iteration, proceeding as follows: first the CFL limit is computed classically according to

$$\text{cfl\_limit} = \min_{(i,j)} \left(\frac{c_{s_{(i,j)}} + |u_x|_{(i,j)}}{\Delta x} + \frac{c_{s_{(i,j)}} + \left|u_y\right|_{(i,j)}}{\Delta y} \right)^{-1}$$

with the sound speed $c_s^2 = \gamma(\gamma - 1)\epsilon$, $|u_x|$ and $\left|u_y\right|$ the absolute value of the speed in the x- and y-directions, respectively, and the minimum taken on all the mesh cells. Then, in the explicit case, the time step is taken as half the CFL limit:

$$dt_{\text{expl}} = \frac{1}{2}\,\text{cfl\_limit}.$$

For the implicit case, the CFL limit can be overcome. In our study, the time step length is determined as a function of the relative variation of ρ and E from one step to the next. More precisely, we define

$$dt_{\text{impl}} = \min \left(\frac{\delta_{dt}\ \delta_{rel}}{\max\left(\delta_{dt} \max_{(i,j)} \left|\frac{\Delta\rho}{\rho}\right|_{(i,j)},\ \delta_{rel}\right)}, \frac{\delta_{dt}\ \delta_{rel}}{\max\left(\delta_{dt} \max_{(i,j)} \left|\frac{\Delta E}{E}\right|_{(i,j)},\ \delta_{rel}\right)} \right) \text{cfl\_limit} \qquad (8)$$

where, at time step n, $\frac{\Delta\rho}{\rho} = \frac{\rho^n}{\rho^{n-1}} - 1$, and similarly for E. To explain this formula, let us take two cases into consideration:

- For small variations of ρ and E, formula (8) reduces to $dt_{\text{impl}} = \delta_{dt}$ cfl_limit, so that δ_{dt} determines the time step length increase allowed in the implicit case. In the numerical tests reported here, δ_{dt} was set to 1.05.

- For large variations of ρ and E, formula (8) becomes

$$dt_{\text{impl}} = \min \left(\frac{\delta_{rel}}{\left(\max_{(i,j)} \left| \frac{\Delta \rho}{\rho} \right|_{(i,j)} \right)}, \frac{\delta_{rel}}{\left(\max_{(i,j)} \left| \frac{\Delta E}{E} \right|_{(i,j)} \right)} \right) \text{cfl\_limit} \tag{9}$$

so that δ_{rel} can be interpreted as the desired relative variations of the considered quantities (ρ or E) per time step: compared to the CFL limit, the time step will be reduced if the actual variations are larger than δ_{rel}, and increased in the opposite case. In the numerical tests reported here, δ_{rel} was set to 0.05.

4.3. **Qualitative results**

We now look again at Figures 1 to 3, on each of which the same spatial mesh is used for both explicit and implicit schemes. We stopped the calculations after an evolution of $7s$ because, for now, we want to avoid (at least in the implicit scheme) the peculiarities appearing after the dense fluid has hit the left face ($x = 0$).

We first concentrate on the effect of mesh refinement for both schemes. Mesh refinement affects the shape of the curls appearing on the plots, this effect becoming more visible as the system evolves. Also, starting at around $4s$ of evolution, we notice the effect of mesh refinement on the distance achieved by the dense fluid in the lighter one at a given time of the evolution.

Then, we compare the explicit and implicit solutions for a given mesh refinement. The general evolution trend is satisfyingly similar, even though the implicit results appear more diffusive than the explicit ones. Also, discrepancies between the explicit and implicit solutions become prominent after $7s$ of evolution.

We conclude that comparing performance results between the explicit and implicit schemes is a difficult task in absolute terms: determining the appropriate mesh refinement for each scheme, as well as the appropriate time step length, should be done in view of the target result(s) of interest. Therefore, in the quantitative results below, we restrict our attention to the costs per finite volume cell and per time step for both schemes.

Finally, note that unphysical dissymmetries appear at evolution times $t = 6s$ and $t = 7s$ in the implicit case, when the 4096×1024 mesh is used. This might probably be due to round-off errors affecting the symmetry of the Jacobian system, and therefore not inherent to the implicit method itself.

4.4. **Quantitative Results**

Figures 4 and 5 respectively present weak and strong scaling results obtained using 64 to 4096 CPU cores, each core being assigned one MPI task.

We ran our calculation on the *Jade* supercomputer hosted by the French CINES computing center. The CPU nodes used were 3 GHz Intel Xeon Quad-Core E5472 ("Harpertown"). In terms of running time, we were limited to 24 hours. In order to obtain comparable results, the same evolution time was used for all runs: we limited our calculations to $2.844s$ of evolution, which corresponds to the evolution time reached after a 24 hour run on *Jade* for the most time-consuming implicit run (16384×4096 mesh on 64 cores). Note that our explicit runs were typically longer than the implicit ones for the same grids, which means that all of the explicit runs did not reach the $2.844s$ evolution time. However, this does not affect our results in terms of cost per cell and per time step because, while in the implicit case this cost takes some evolution time before reaching a relatively constant value, it is not the case in the explicit case, where this cost appears constant from the start of the evolution. It appeared therefore more appropriate to use the full extent of our 24 hour runtime on *Jade* in the implicit case.

In terms of memory footprint, we observed in our 2-D tests that the implicit scheme requires about 3 times more memory than the explicit one.

4.4.1. *Weak Scaling*

Figure 4 displays the cost per cell and per time step, employing 64, 256, 1024 and 4096 cores, each of these cores dealing with a 128×128 square sub-domain. In turn, this corresponds to 4 different meshes for the whole domain, from 2048×512 to 16384×4096, all of them satisfying the 4-to-1 aspect ratio of the physical problem.

We observe that weak scalability is well achieved by the explicit scheme, but not so well by the implicit one. This can be blamed on the loss of performance of the ILU(0) preconditioner: the average number of BiCGStab inner iterations per time step increases from 8 with 64 cores to 19 with 4096 cores.

As expected, the cost per cell and per time step is higher using the implicit scheme rather than the explicit one: from 16 times on 64 cores to 27 times on 4096 cores. This extra cost per time step is clearly compensated in our tests by the longer implicit time step, but, as stated above, we prefer to avoid comparing implicit and explicit schemes in absolute terms, i.e., without a clear target result.

It is important to mention here that results on 4096 cores (and to a lesser extent those on 1024 cores) highly depend on the workload of the *Jade* supercomputer: results shown here are the best ones (i.e., were obtained on Sundays), and could be up to 2 to 3 times worse in terms of cost on heavy workload days. Results on 64 and 256 cores did not exhibit such dependency on the workload.

4.4.2. *Strong Scaling*

Figure 5 displays cost per cell and per time step for a 16384×4096 mesh, using 64 to 4096 cores. Strong scalability is well achieved by both explicit and implicit schemes. For the latter, the average number of BiCGStab inner iterations per time step remains nearly constant: from 17 with 64 cores to 19 with 4096 cores. Similarly the extra cost per cell and per time step for using the implicit scheme remains nearly constant (factor 23 to 27).

Note that here again dependency on the workload was observed for large number of nodes, and only the best results are displayed.

5. Conclusions and Perspectives

An implicit version of the explicit scheme of Vides et al. [8] was presented. A good qualitative agreement between the explicit and implicit solutions was observed, even if discrepancies between them, very limited at start, appeared to grow along the evolution. Using the implicit scheme rather than the explicit one implies an increase in cost per cell and per time step by a factor that turned out to range from 16 to 27 in our calculations, depending on the number of CPU cores employed. Nevertheless, this extra cost could be, at least in our tests, compensated by a longer time step. However, performance comparisons between both schemes should not be done in absolute terms, the appropriate space and time refinement for each scheme being highly dependent on the target result(s) of interest in production runs.

Our numerical results show that, while weak and strong scalabilities are both achieved by the explicit scheme, the proposed implicit scheme achieves strong but not weak scalability. This can be blamed on the lack of scalability of the local ILU(0) preconditioner used to solve the Jacobian system. Therefore, other local preconditioners available in PETSc, as well as direct solvers interfaced by it, like MUMPS, SuperLU and PaStiX, should be considered. Moreover, global preconditionings other than block Jacobi need to be tested (e.g., Algebraic Schwartz Method with various overlaps) as well as Krylov accelerations other than BiCGStab (e.g. GMRES). Using the full potential of the PETSc library will then hopefully provide full scalability. In turn we would have demonstrated the possibility to derive an implicit version of a given explicit scheme in a relatively easy way, given the ease of use of the TAPENADE tool used to derive the Jacobian matrix.

References

[1] Satish Balay, Jed Brown, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.3, Argonne National Laboratory, 2012.

[2] Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, `http://www.mcs.anl.gov/petsc`, 2012.

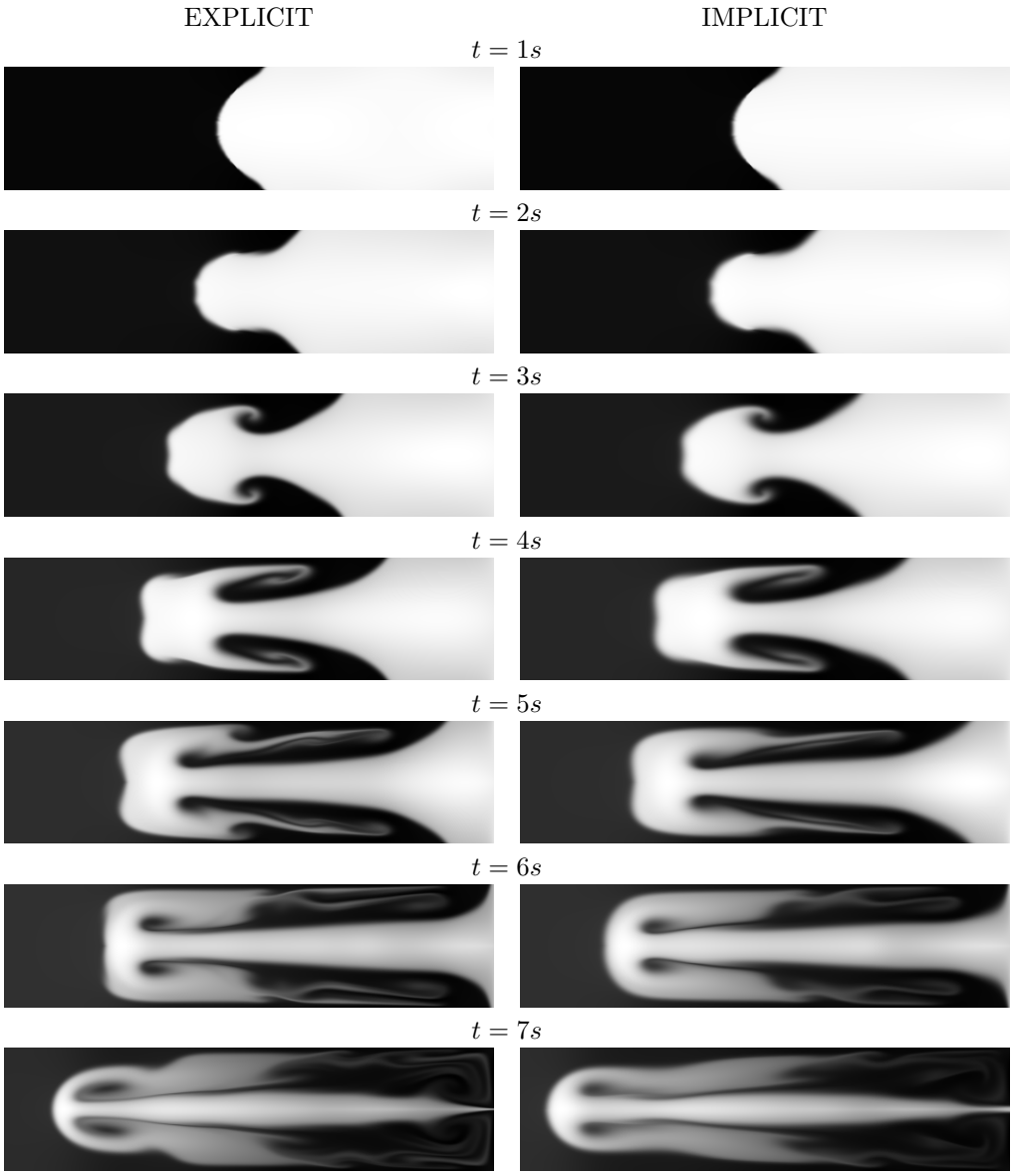

FIGURE 1. Rayleigh-Taylor instability computed with the explicit and implicit schemes with a 1024×256 mesh. Computations performed on 16 processors.

[3] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[4] M. González, E. Audit, and P. Huynh. HERACLES: a three-dimensional radiation hydrodynamics code. *Astronomy & Astrophysics*, 464:429–435, 2007.

[5] L. Hascoët. TAPENADE: a tool for automatic differentiation of programs. In *Proceedings of 4^{th} European Congress on Computational Methods, ECCOMAS'2004, Jyvaskyla, Finland*, 2004.

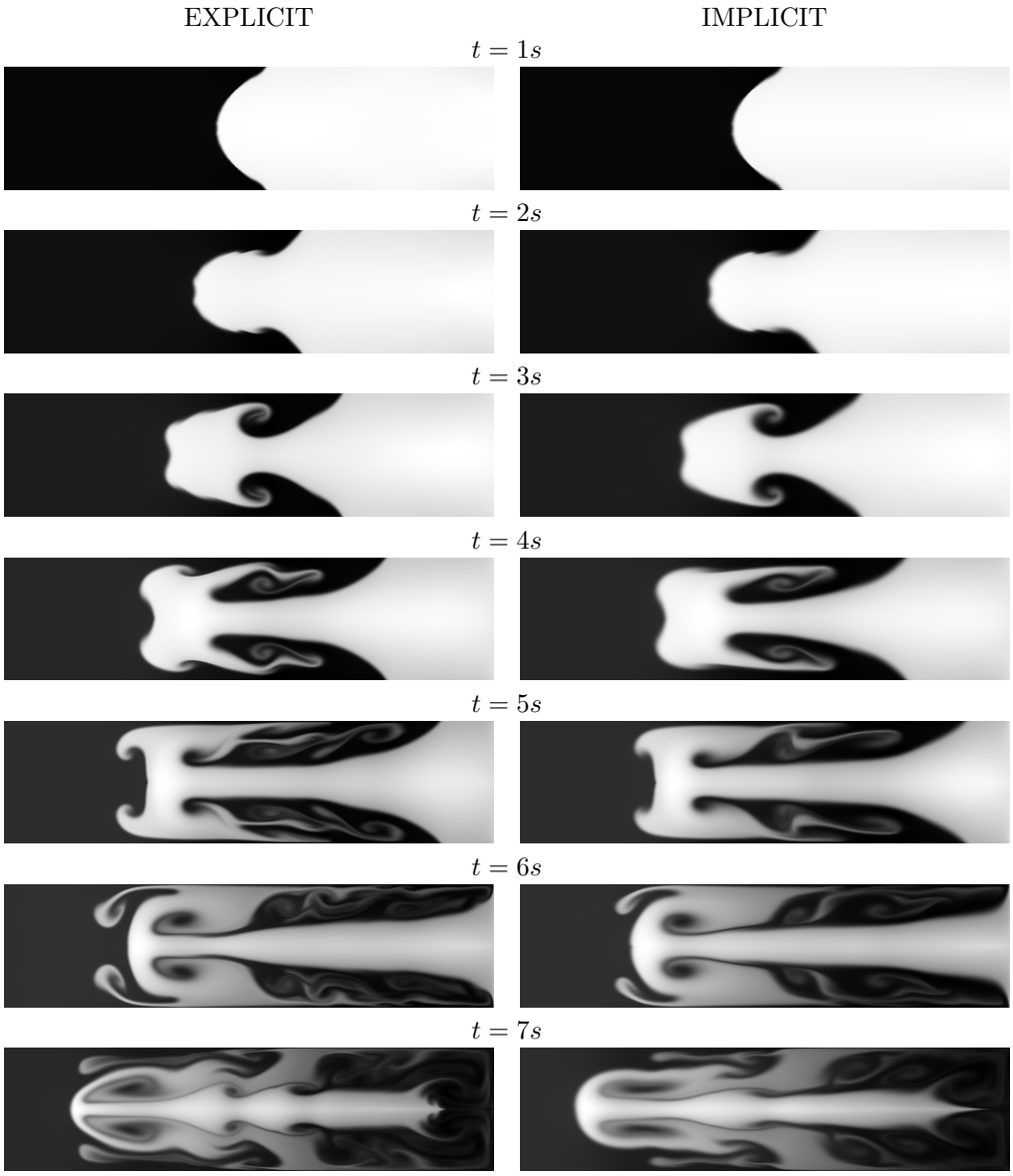

FIGURE 2. Rayleigh-Taylor instability computed with the explicit and implicit schemes with a 2048×512 mesh. Computations performed on 64 processors.

[6] Inria Tropics team. TAPENADE Web page, 2012. http://www-tapenade.inria.fr:8080/tapenade/index.jsp and http://www-sop.inria.fr/tropics.

[7] N. Vaytet. HERACLES Web page. http://irfu.cea.fr/Projets/Site_heracles/index.html.

[8] J. Vides, B. Braconnier, E. Audit, C. Berthon, and B. Nkonga. A Godunov-type solver for the numerical approximation of gravitational flows. *Communications in Computational Physics*, 15(1):46–75, 2014.

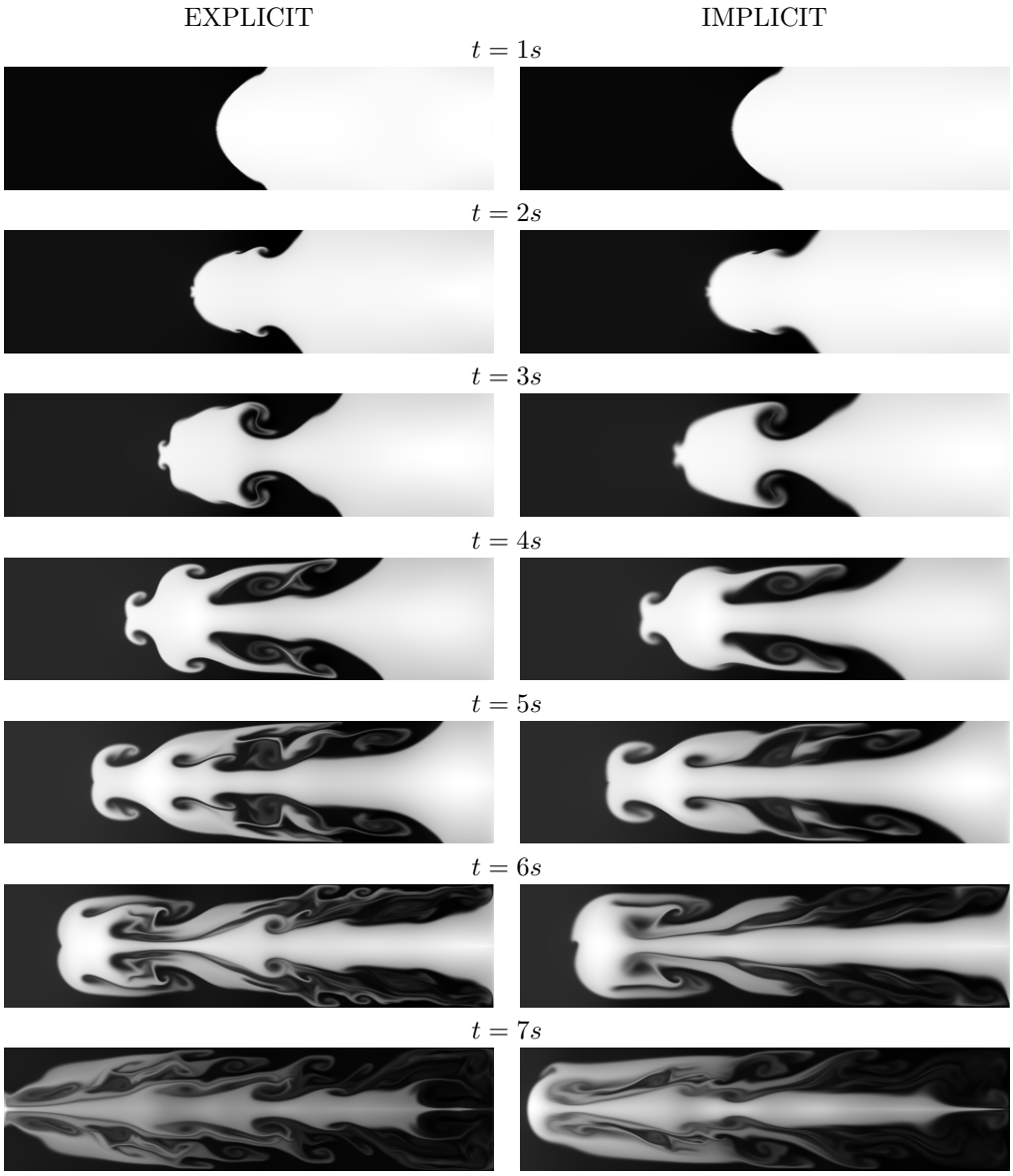

FIGURE 3. Rayleigh-Taylor instability computed with the explicit and implicit schemes with a 4096×1024 mesh. Computations performed on 256 processors.

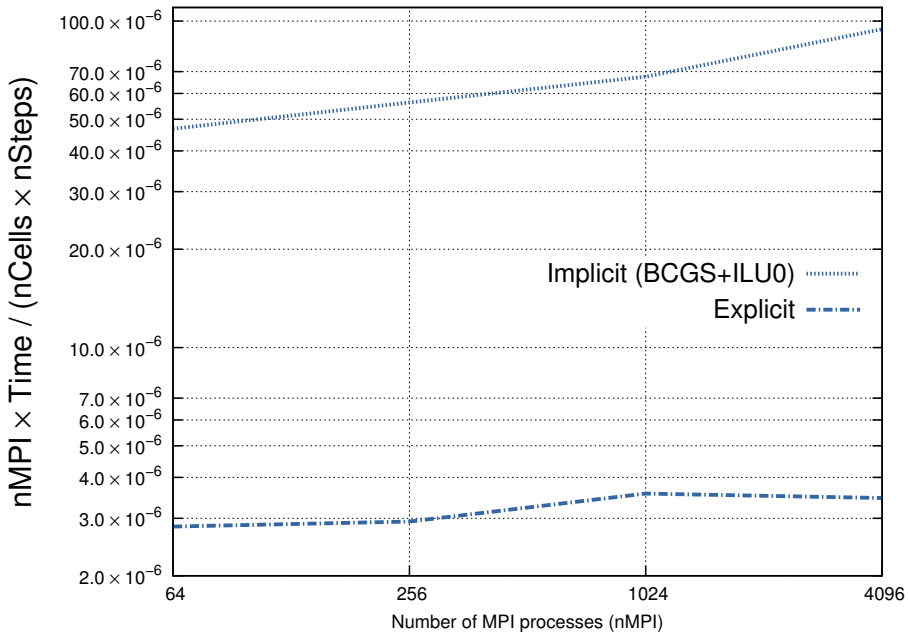

FIGURE 4. Weak scaling for the Rayleigh-Taylor instability: each core deals with a 128×128 square sub-domain. Therefore meshes range from 2048×512 (for nMPI=64) to 16384×4096 (for nMPI=4096).

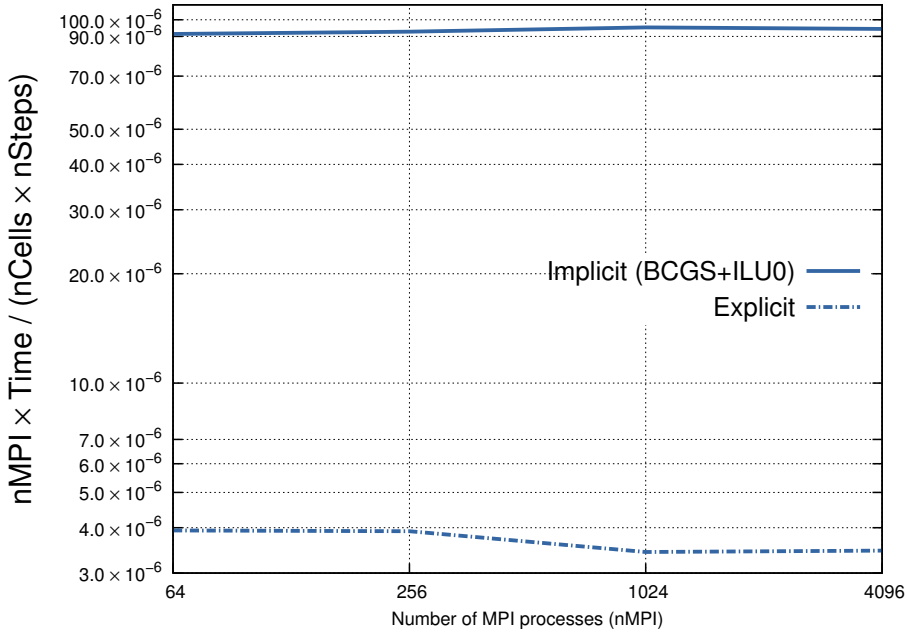

FIGURE 5. Strong scaling for the Rayleigh-Taylor instability, for a 16384×4096 mesh.