Lagrange-flux schemes: reformulating second-order accurate Lagrange-remap schemes for better node-based HPC performance

Florian de Vuyst, Thibault Gasc, Renaud Motte, Mathieu Peybernes, Raphaël Poncet

D o s s i e r

SimRace 2015: Numerical Methods and High Performance Computing for Industrial Fluid Flows
SimRace 2015 : Méthodes numériques et calcul haute performance pour la simulation d'écoulements complexes

Lagrange-Flux Schemes: Reformulating Second-Order Accurate Lagrange-Remap Schemes for Better Node-Based HPC Performance

Florian De Vuyst[1]*, Thibault Gasc[1,2,3], Renaud Motte[3], Mathieu Peybernes[4] and Raphaël Poncet[5]

[1] CMLA, ENS Cachan, Université Paris-Saclay, CNRS, 94235 Cachan - France
[2] Maison de la Simulation, USR 3441, CEA Saclay, 91191 Gif-sur-Yvette - France
[3] CEA DAM DIF, 91297 Arpajon - France
[4] CEA Saclay, DEN, 91191 Gif-sur-Yvette - France
[5] CGG, 27 avenue Carnot, 91300 Massy - France
e-mail: devuyst@cmla.ens-cachan.fr - thibault.gasc@cea.fr - renaud.motte@cea.fr - mathieu.peybernes@cea.fr - raphael.poncet@cgg.com

* Corresponding author

Abstract — *In a recent paper [Poncet R., Peybernes M., Gasc T., De Vuyst F. (2016) Performance modeling of a compressible hydrodynamics solver on multicore CPUs, in "Parallel Computing: on the road to Exascale"], we have achieved the performance analysis of staggered Lagrange-remap schemes, a class of solvers widely used for hydrodynamics applications. This paper is devoted to the rethinking and redesign of the Lagrange-remap process for achieving better performance using today's computing architectures. As an unintended outcome, the analysis has lead us to the discovery of a new family of solvers – the so-called Lagrange-flux schemes – that appear to be promising for the CFD community.*

Résumé — **Schémas Lagrange-flux : reformuler les schémas Lagrange-Projection d'ordre deux pour améliorer la performance HPC au niveau nœud de calcul** — Dans un article récent [Poncet R., Peybernes M., Gasc T., De Vuyst F. (2016) Performance modeling of a compressible hydrodynamics solver on multicore CPUs, in "Parallel Computing: on the road to Exascale"], nous avons effectué l'analyse de la performance d'un schéma de type Lagrange+projection à variables décalées ; cette classe de solveurs est très utilisée pour les applications d'hydrodynamique. Dans cet article, on s'intéresse à la reformulation des solveurs Lagrange-projection afin d'améliorer leur performance globale sur architectures de calculs standards. De manière inattendue, l'analyse nous a conduit vers la découverte d'une nouvelle famille de solveurs – appelés schémas Lagrange-flux – qui apparaissent comme très prometteurs dans la communauté CFD.

MOTIVATION AND INTRODUCTION

For complex compressible flows involving multiphysics phenomena like *e.g.* high-speed elastoplasticity, multimaterial interaction, plasma, gas-particles, etc., a Lagrangian description of the flow is generally preferred. To achieve robustness, some spatial remapping on a regular mesh may be added. A particular case is the family of the so-called Lagrange+remap schemes [1–3], also referred to as remapped Lagrange solvers that apply a remap step on a reference (Eulerian) mesh after each Lagrangian time advance. Legacy codes implementing remapped Lagrange solvers
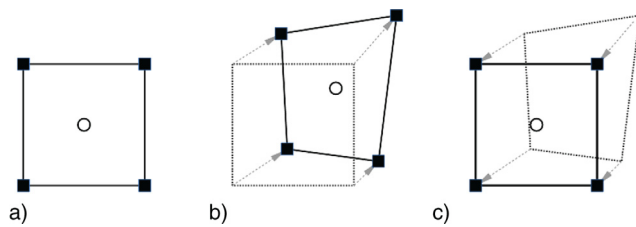
Figure 1
"Legacy" staggered Lagrange-remap scheme: thermodynamical variables are located at cell centers (circles) whereas velocity variables are located at cell nodes (squares). a) Eulerian cell, b) Lagrange step, c) Remapping step.

usually define thermodynamical variables at cell centers and velocity at cell nodes (Fig. 1). In Poncet *et al.* [4], we have achieved a multicore node-based performance analysis of a reference Lagrange-remap hydrodynamics solver used in industry. By analyzing each kernel (*i.e.* elementary computing function) of the whole algorithm, using roofline-type models [5] on one side and refined Execution Cache Memory (ECM) models [6, 7] on the other side, we have been able not only to quantitatively predict the performance of the whole algorithm – with relative errors in the single digit range – but also to identify a set of features that limit the whole performance. This can be roughly summarized into three points:

1. For typical mesh sizes of real applications, spatially staggered variables involve a rather big amount of communication to/from CPU caches and memory with low arithmetic intensity, thus lowering the whole performance;
2. Usual Alternating Direction (AD) strategies (see the appendix in [8]) or AD remapping procedures also generate too much communication with a loss of CPU occupancy;
3. For multimaterial flows using VOF-based interface reconstruction methods, there is a strong loss of performance due to some array indirections and noncoalescent data in memory. Vectorization of such algorithms is also not trivial.

From these observations and as a result of the analysis, we decided to "rethink" Lagrange-remap schemes, with possibly modifying some aspects of the solver in order to improve node-based performance of the hydrocode. We have looked for alternative formulations that reduce communication and improve both arithmetic intensity and SIMD (Single Instruction – Multiple Data) property of the algorithm. In this paper, we describe the process of redesign of Lagrange-remap schemes leading to higher performance solvers. Actually, this redesign methodology also gave us ideas of innovative Eulerian solvers. The emerging methods, named Lagrange-flux schemes, appear to be promising in the extended computational fluid dynamics community.

The paper is organized as follows. In Section 1, we first formulate the requirements for the design of better Lagrange-remap schemes. In Section 2 we give a description of the Lagrange step and formulate it under a finite volume form. In Section 3 we focus on the remap step which is reformulated as a finite volume scheme with pure convective fluxes. This interpretation is applied in Section 4 to build the so-called Lagrange-flux schemes. We also discuss the important issue of achieving second order accuracy (in both space and time). Performance comparisons between the reference Lagrange-remap and the proposed Lagrange-flux scheme are presented in Section 5. In Section 6 we comment the possible extension to multimaterial flow with the use of low-diffusive and accurate interface-capturing methods. We will close the paper by some concluding remarks, work in progress and perspectives.

1 REQUIREMENTS

Starting from "legacy" Lagrange-remap solvers and related observed performance measurements, we want to improve the performance of these solvers by modifying some of their features but under some constraints and requirements:

1. A Lagrangian solver (or description) must be used (allowing for multiphysics coupling);
2. To reduce communication, we prefer using collocated cell-centered variables rather than a staggered scheme;
3. To reduce communication, we prefer using a direct multidimensional remap solver rather than split alternating direction projections;
4. The method can be simply extended to second-order accuracy (in space and time);
5. The solver must be able to be naturally extended to multimaterial flows.

Before going further, let us comment the above requirements. The second requirement should imply the use of a cell-centered Lagrange solver. Fairly recently, Després and Mazeran in [9] and Maire *et al.* [10] (with high-order extension in [11]) have proposed pure cell-centered Lagrangian solvers based on the reconstruction of nodal velocities. In our study, we will examine if it is possible to use approximate and simpler Lagrangian solvers in the Lagrange+remap context, in particular for the sake of performance. The fourth assertion requires a full multidimensional remapping step, probably taking into account geometric elements (deformation of cells and edges) if we want to ensure high-order accuracy remapping. To summarize, our requirements are somewhat contradictory, and we have to find a good compromise between some simplifications-approximations and a loss of accuracy (or properties) of the numerical solver.

2 LAGRANGE STEP

As examples, let us consider the compressible Euler equations for two-dimensional problems. Denoting ρ, $\boldsymbol{u} = (u_i)_i$, $i \in \{1, 2\}$, p and E the density, velocity, pressure and specific total energy respectively, the mass, momentum and energy conservation equations read

$$\partial_t U_\ell + \nabla \cdot (\boldsymbol{u} U_\ell) + \nabla \cdot \boldsymbol{\pi}_\ell = 0, \quad \ell = 1, \ldots, 4 \quad (1)$$

where $U = (\rho, (\rho u_i)_i, \rho E)$, $\boldsymbol{\pi}_1 = \vec{0}$, $\boldsymbol{\pi}_2 = (p, 0)^T$, $\boldsymbol{\pi}_3 = (0, p)^T$ and $\boldsymbol{\pi}_4 = p\boldsymbol{u}$. For the sake of simplicity, we will use a perfect gas equation of state $p = (\gamma - 1)\rho(E - \frac{1}{2}|\boldsymbol{u}|^2)$, $\gamma \in (1, 3]$. The speed of sound c is given by $c = \sqrt{\gamma p / \rho}$.

For any volume V_t that moves with the fluid, from the Reynolds transport theorem we have

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\mathcal{V}_t} U_\ell \mathrm{d}\boldsymbol{x} = \int_{\partial \mathcal{V}_t} \{\partial_t U_\ell + \nabla \cdot (\boldsymbol{u}\, U_\ell)\} \mathrm{d}\boldsymbol{x}$$

$$= -\int_{\partial \mathcal{V}_t} \boldsymbol{\pi}_\ell \cdot \boldsymbol{v} \mathrm{d}\sigma$$

where $\boldsymbol{v}$ is the normal unit vector exterior to $\mathcal{V}_t$. This leads to a natural explicit finite volume scheme in the form

$$|K^{n+1,L}|(U_\ell)_K^{n+1,L} = |K|(U_\ell)_K^n$$

$$- \Delta t^n \sum_{A^{n+\frac{1}{2},L} \subset \partial K^{n+\frac{1}{2},L}} |A^{n+\frac{1}{2},L}| \boldsymbol{\pi}_A^{n+\frac{1}{2},L} \cdot v_A^{n+\frac{1}{2},L} \quad (2)$$

In (2), the superscript "L" indicates the Lagrange evolution of the quantity. Any Eulerian cell K is deformed into the Lagrangian volume $K^{n+\frac{1}{2},L}$ at time $t^{n+\frac{1}{2}}$, and into the Lagrangian volume $K^{n+1,L}$ at time t^{n+1}. The pressure flux terms through the edges A are evaluated at time $t^{n+\frac{1}{2}}$ in order to get second-order accuracy in time. Of course, that means that we need a predictor step for the velocity field $\boldsymbol{u}^{n+\frac{1}{2},L}$ at time $t^{n+\frac{1}{2}}$ (not written here for simplicity).

Notations

To simplify, in all what follows we will use the notation $\boldsymbol{v}^{n+\frac{1}{2}} = \boldsymbol{u}^{n+\frac{1}{2},L}$.

3 RETHINKING THE REMAPPING STEP

The remapping step considers the remapping fields U_ℓ defined at cell centers $K^{n+1,L}$ on the initial (reference) Eulerian mesh with cells K. Let us denote $\mathcal{R}^{n+1,L}$ a linear operator that reconstructs piecewise polynomial functions from a discrete field $U^{n+1,L}$ defined at cell centers of the Lagrangian mesh $\mathcal{M}^{N+1,L}$ at time t^{n+1}. The remapping process consists in projecting the reconstructed field on piecewise-constant functions on the Eulerian mesh, according to the integral formula

$$U_K^{n+1} = \frac{1}{|K|} \int_K \mathcal{R}^{n+1,L} U^{n+1,L}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \quad (3)$$

Practically, there are many ways to consider the projection operation (3). One can assemble elementary projection contributions by computing the volume intersections between the reference mesh and the deformed mesh. But this procedure requires the computation of all the geometrical elements. Moreover, the projection needs local tests of projection with conditional branching (think about the very different cases of a compression and expansion). Thus the procedure is not SIMD and potentially leads to a loss of performance. The incremental remapping can also be interpreted as a transport/advection algorithm, as emphasized by Dukowicz and Baumgardner [12] that appears to be better suited for SIMD treatments.

Let us now write a different original formulation of the remapping step. In this step, there is no time evolution of any quantity, and in some sense we have $\partial_t U = 0$, that we write

$$\partial_t U = \partial_t U + \nabla \cdot (-\boldsymbol{v}^{n+\frac{1}{2}} U) + \nabla \cdot (\boldsymbol{v}^{n+\frac{1}{2}} U) = 0$$

We decide to split up this equation into two substeps:
(i) Backward convection:

$$\partial_t U + \nabla \cdot (-\boldsymbol{v}^{n+\frac{1}{2}} U) = 0 \quad (4)$$

(ii) Forward convection:

$$\partial_t U + \nabla \cdot (\boldsymbol{v}^{n+\frac{1}{2}} U) = 0 \quad (5)$$

Each convection problem is well-posed on the time interval $[0, \Delta t^n]$. Let us now focus into these two steps and the way to solve them.

3.1 Backward Convection in Lagrangian Description

After the Lagrange step, if we solve the backward convection problem (4) over a time interval Δt^n using a Lagrangian description, we have

$$|K|(U_\ell)_K^{n,\star} = |K^{n+1,L}|(U_\ell)_K^{n+1,L} \quad (6)$$

Actually, from the cell $K^{n+1,L}$ we go back to the original cell K with conservation of the conservative quantities. For $\ell = 1$ (conservation of mass), we have

$$|K| \rho_K^{n,\star} = |K^{n+1,L}| \rho_K^{n+1,L}$$

showing the variation of density by volume variation. For $\ell = 2, 3, 4$, it is easy to see that both velocity and specific total energy are kept unchanged in this step:

$$\boldsymbol{u}^{n,\star} = \boldsymbol{u}^{n+1,L}, \quad E^{n,\star} = E^{n+1,L}$$

Thus, this step is clearly computationally inexpensive.

3.2 Forward Convection in Eulerian Description

From the discrete field $(U_K^{n,\star})_K$ defined on the Eulerian cells K, we then solve the forward convection problem over a time step Δt^n under an Eulerian description. A standard finite volume discretization of the problem will lead to the classical time advance scheme

$$U_K^{n+1} = U_K^{n,\star} - \frac{\Delta t^n}{|K|} \sum_{A \subset \partial_K} |A| \, U_A^{n+\frac{1}{2},\star} \, (\boldsymbol{v}_A^{n+\frac{1}{2}} \cdot v_A) \qquad (7)$$

for some interface values $U_A^{n+\frac{1}{2},\star}$ defined from the local neighbor values $U_K^{n,\star}$. We finally get the expected Eulerian values U_K^{n+1} at time t^{n+1}.

Notice that from (6) and (7) we have also

$$|K| \, U_K^{n+1} = |K^{n+1,L}| \, U_K^{n+1,L} - \Delta t^n \sum_{A \subset \partial K} |A| \, U_A^{n+\frac{1}{2},\star} \, (\boldsymbol{v}_A^{n+\frac{1}{2}} \cdot v_A) \tag{8}$$

thus completely defining the remap step under the finite volume scheme form (8). One can notice that neither mesh intersections nor geometric considerations are required to achieve this remapping process. The finite volume form (8) is now suitable for a straightforward vectorized SIMD treatment. From (8) it is easy to achieve second-order accuracy for the remapping step by usual finite volume tools (MUSCL reconstruction + second-order accurate time advance scheme for example).

3.3 Full Lagrange+Remap Time Advance

Let us note that the Lagrange+remap scheme is actually a conservative finite volume scheme: putting (2) into (8) gives for all ℓ:

$$(U_\ell)_K^{n+1} = (U_\ell)_K^n - \frac{\Delta t^n}{|K|} \sum_{A^{n+\frac{1}{2},L} \subset \partial K^{n+\frac{1}{2},L}} |A^{n+\frac{1}{2},L}| \, (\boldsymbol{\pi}_\ell)_A^{n+\frac{1}{2},L} \cdot v_A^{n+\frac{1}{2},L}$$
$$- \frac{\Delta t^n}{|K|} \sum_{A \subset \partial K} |A| \, (U_\ell)_A^{n+\frac{1}{2},\star} \left(\boldsymbol{v}_A^{n+\frac{1}{2}} \cdot v_A \right) \tag{9}$$

that can also be written

$$(U_\ell)_K^{n+1} = (U_\ell)_K^n - \frac{\Delta t^n}{|K|} \sum_{A \subset \partial K} |A| \left(\frac{|A^{n+\frac{1}{2},L}|}{|A|} \, (\boldsymbol{\pi}_\ell)_A^{n+\frac{1}{2},L} \cdot v_A^{n+\frac{1}{2},L} \right)$$
$$- \frac{\Delta t^n}{|K|} \sum_{A \subset \partial K} |A| \left((U_\ell)_A^{n+\frac{1}{2},\star} \, (\boldsymbol{v}_A^{n+\frac{1}{2}} \cdot v_A) \right) \tag{10}$$

We recognize in (10) pressure-related fluxes and convective fluxes that define the whole numerical flux.

3.4 Comments

The finite volume formulation (10) is attractive and seems rather simple at first sight. But we should not forget that we have to compute a Lagrangian velocity vector field $\boldsymbol{v}^{n+\frac{1}{2}} = \boldsymbol{u}^{n+\frac{1}{2},L}$ where the variables should be located at cell nodes to return a well-posed deformation. Moreover, expression (10) involves geometric elements like the length of the deformed edges $A^{n+\frac{1}{2},L}$. Among the rigorous collocated Lagrangian solvers, let us mention the GLACE scheme by Després-Mazeran [9] and the cell-centered EUCCLHYD solver by Maire *et al.* [10]. Both are rather computationally expensive and their second-order accurate extension is not easy to achieve.

Although it is possible to couple these Lagrangian solvers with the flux-balanced remapping formulation, it is also of interest to think about ways to simplify or approximate the Lagrange step without losing second-order accuracy. One of the difficulty in the analysis of Lagrange-remap schemes is that, in some sense, space and time are coupled by the deformation process.

Below, we derive a formulation that leads to a clear separation between space and time, in order to simply control the order of accuracy. The idea is to make the time step tend to zero in the Lagrange-remap scheme (method of lines [13]), then exhibit the instantaneous spatial numerical fluxes through the Eulerian cell edges that will serve for the construction of an explicit finite volume scheme. Because the method needs an approximate Riemann solver in Lagrangian form, we will call it a Lagrange-flux scheme.

4 DERIVATION OF A SECOND-ORDER ACCURATE LAGRANGE-FLUX SCHEME

From the intermediate conclusions of the discussion 3.4 above, we would like to be free from any rather expensive collocated Lagrangian solver. However, such a Lagrangian solver seems necessary to correctly and accurately define the deformation velocity field $\boldsymbol{v}^{n+\frac{1}{2}}$ at time $t^{n+\frac{1}{2}}$.

In what follows, we are trying to deal with time accuracy in a different manner. Let us come back to the Lagrange+remap formula (10). Let us consider a "small" time step t that fulfills the usual stability CFL condition. We have

$$(U_\ell)_K(t) = (U_\ell)_K^n - \frac{t}{|K|}$$

$$\times \sum_{A \subset \partial K} |A| \left(\frac{|A^L(t/2)|}{|A|} (\boldsymbol{\pi}_\ell)_A^L(t/2) \cdot v_A^L(t/2) \right)$$

$$- \frac{t}{|K|} \sum_{A \subset \partial K} |A| (U_\ell)_A^\star(t/2) \, \boldsymbol{v}_A(t/2) \cdot v_A$$

By making t tend to zero, $(t > 0)$, we have $A^L(t/2) \to A$, $(\boldsymbol{\pi}_\ell)^L(t/2) \to \boldsymbol{\pi}_\ell$, $\boldsymbol{v}(t/2) \to \boldsymbol{u}$, $(U_\ell)^\star \to U_\ell$, then we get a semi-discretization in space of the conservation laws. That can be seen as a particular method of lines [13]:

$$\frac{\mathrm{d}(U_\ell)_K}{\mathrm{d}t} = -\frac{1}{|K|} \sum_{A \subset \partial K} |A| \left((\boldsymbol{\pi}_\ell)_A \cdot v_A \right)$$

$$- \frac{1}{|K|} \sum_{A \subset \partial K} |A| (U_\ell)_A (\boldsymbol{u}_A \cdot v_A) \quad (11)$$

We get a classical finite volume method

$$\frac{\mathrm{d}U_K}{\mathrm{d}t} = -\frac{1}{|K|} \sum_{A \subset \partial K} |A| \, \Phi_A$$

with a numerical flux Φ_A whose components are

$$(\Phi_\ell)_A = (U_\ell)_A (\boldsymbol{u}_A \cdot v_A) + (\boldsymbol{\pi}_\ell)_A \cdot v_A \quad (12)$$

In (11), pressure fluxes $(p_\ell)_A$ and interface normal velocities $(\boldsymbol{u}_A \cdot v_A)$ can be computed from an approximate Riemann solver in Lagrangian coordinates (for example the Lagrangian HLL solver, [14]). Then, the interface states $(U_\ell)_A$ should be computed from an upwind process according to the sign of the normal velocity $(\boldsymbol{u}_A \cdot v_A)$. This is interesting because the resulting flux has similarities with the so-called Advection Upstream Splitting Method (AUSM) flux family proposed by Liou [15], but the construction here is different and, in some sense, justifies the AUSM splitting.

To get higher-order accuracy in space, one can use a standard MUSCL reconstruction + slope limiting process involving classical slope limiters like for example Sweby's limiter function [16]:

$$\phi(a, b) = (ab > 0) \, \mathrm{sign}(a) \, \max(\min(|a|, \beta|b|),$$

$$\min(\beta|a|, |b|)) \quad (13)$$

with $\beta \in [1, 2]$ for achieving second order accuracy. At this stage, because there is no time discretization, everything is defined on the Eulerian mesh and fluxes are located at the edges of the Eulerian cells. This is one originality of this scheme compared to the legacy staggered Lagrange-remap scheme that has to use variables defined on the Lagrangian cells.

To get high-order accuracy in time, one can then apply a standard high-order time advance scheme (Runge-Kutta 2 (RK2), etc.). For the second-order Heun scheme for example, we have the following algorithm:

1. Compute the time step Δt^n subject to some CFL condition;
2. *Predictor step*. MUSCL reconstruction + slope limitation on primitive variables ρ, $\boldsymbol{u}$ and p. From the discrete states U_K^n, compute a discrete gradient for each cell K and interpolated values at cell interfaces;
3. Use a Lagrangian approximate Riemann solver to compute pressure fluxes $\boldsymbol{\pi}_A^n$ and interface velocities $\boldsymbol{u}_A^n$ from the MUSCL reconstructed values at the interface;
4. Compute the upwind edge values $(U_\ell)_A^n$ according to the sign of $(\boldsymbol{u}_A^n \cdot v_A)$;
5. Compute the numerical flux Φ_A^n as defined in (12);
6. Compute the first order predicted states $U_K^{\star, n+1}$:

$$U_K^{\star, n+1} = U_K^n - \frac{\Delta t^n}{|K|} \sum_{A \subset \partial K} |A| \Phi_A^n$$

7. *Corrector step*. MUSCL reconstruction + slope limitation: from the discrete values $U_K^{\star, n+1}$, compute a discrete gradient for each cell K and interpolated values at cell interfaces;
8. Use a Lagrangian approximate Riemann solver to compute pressure fluxes $\boldsymbol{\pi}_A^{\star, n+1}$ and interface velocities $\boldsymbol{u}_A^{\star, n+1}$ from the MUSCL reconstructed values at the interface;
9. Compute the upwind edge values $(U_\ell)_A^{\star, n+1}$ according to the sign of $(\boldsymbol{u}_A^{\star, n+1} \cdot v_A)$;
10. Compute the numerical flux $\Phi_A^{\star, n+1}$ as defined in (12);
11. Compute the second-order accurate states U_K^{n+1} at time t^{n+1}:

$$U_K^{n+1} = U_K^n - \frac{\Delta t^n}{|K|} \sum_{A \subset \partial K} |A| \frac{\Phi_A^n + \Phi_A^{\star, n+1}}{2}$$

One can appreciate the simplicity of the numerical solver compared to the legacy staggered Lagrange-remap algorithm. The complexity of the latter mainly due to various kernel (function) calls and too much communications is detailed in [4]. Here the predictor and corrector kernel functions have similar programming codes and there is no intermediate variables to save in memory.
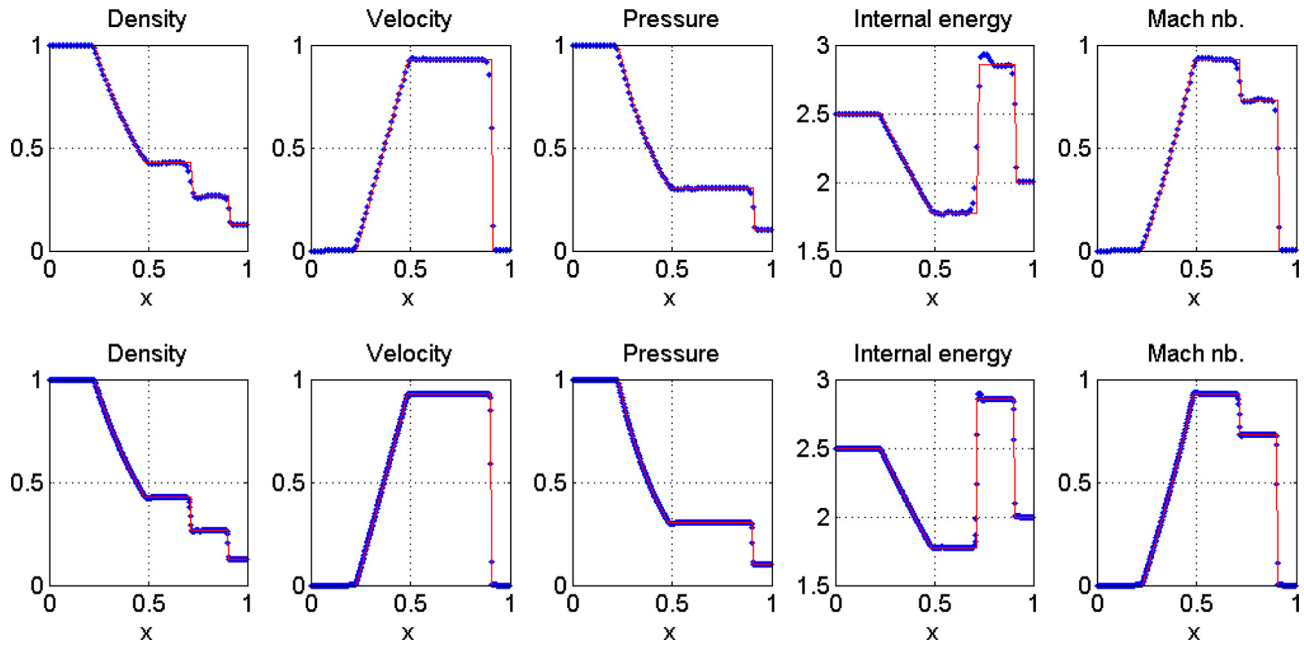
Figure 2
Second-order Lagrange-flux scheme on reference Sod's shock tube problem on two different mesh grids, 100 and 400 respectively. The solid red line is the analytic solution and blue points are the numerical solution.

4.1 Lagrangian HLL Approximate Solver

A HLL approximate Riemann solver [14] in Lagrangian coordinates can be used to easily compute interface pressure and velocity. For a local Riemann problem made of a left state U_L and a right state U_R, the contact pressure $p^\star$ is given by the formula

$$p^\star = \frac{\rho_R p_L + \rho_L p_R}{\rho_L + \rho_R} - \frac{\rho_L \rho_R}{\rho_L + \rho_R} \max(c_L, c_R)(u_R - u_L)$$

(14)

and the normal contact velocity $u^\star$ by

$$u^\star = \frac{\rho_L u_L + \rho_R u_R}{\rho_L + \rho_R} - \frac{1}{\rho_L + \rho_R} \frac{p_R - p_L}{\max(c_L, c_R)}$$

(15)

leading to simple formulas easily implementable in the second-order Heun time integration scheme.

4.2 Numerical Experiments

As example, we test the Lagrange-flux scheme presented in Section 4 on few one-dimensional shock tube problems. We use a RK2 time integrator and a MUSCL reconstruction with the Sweby slope limiter given in (13).

4.2.1 Sod's Shock Tube [17]

The initial data defined on space interval $[0, 1]$ is made of two constant states $(\rho, u, p)_L = (1, 0, 1)$ and $(\rho, u, p)_R = (0.125, 0, 0.1)$ with initial discontinuity at $x = 0.5$. We successively test the method on two uniform mesh grids made of 100 and 400 cells, respectively. The final computational time is $T = 0.23$ and we use a CFL number equal to 0.25 and a Sweby limiter with coefficient $\beta = 1.5$. On Figure 2, one can observe a nice behavior of the Euler solver, with sharp discontinuities and a low numerical diffusion into the rarefaction fan even for the coarse grid.

4.2.2 Two-Rarefaction Shock Tube

The second reference example is a case of two moving-away rarefaction fans under near-vacuum conditions [14]. It is known that the Roe scheme breaks down for this case. The related Riemann problem is made of the left state $(\rho, u, p)_L = (1, -2, 0.4)$ and right state $(\rho, u, p)_R = (1, 2, 0.4)$. The final time of $T = 0.16$. We again test the method on a coarse mesh (200 points) and a fine mesh (2000 points). Numerical results are given in Figure 3. The numerical scheme appears to be robust especially in near-vacuum zones where both density and pressure are close to zero.
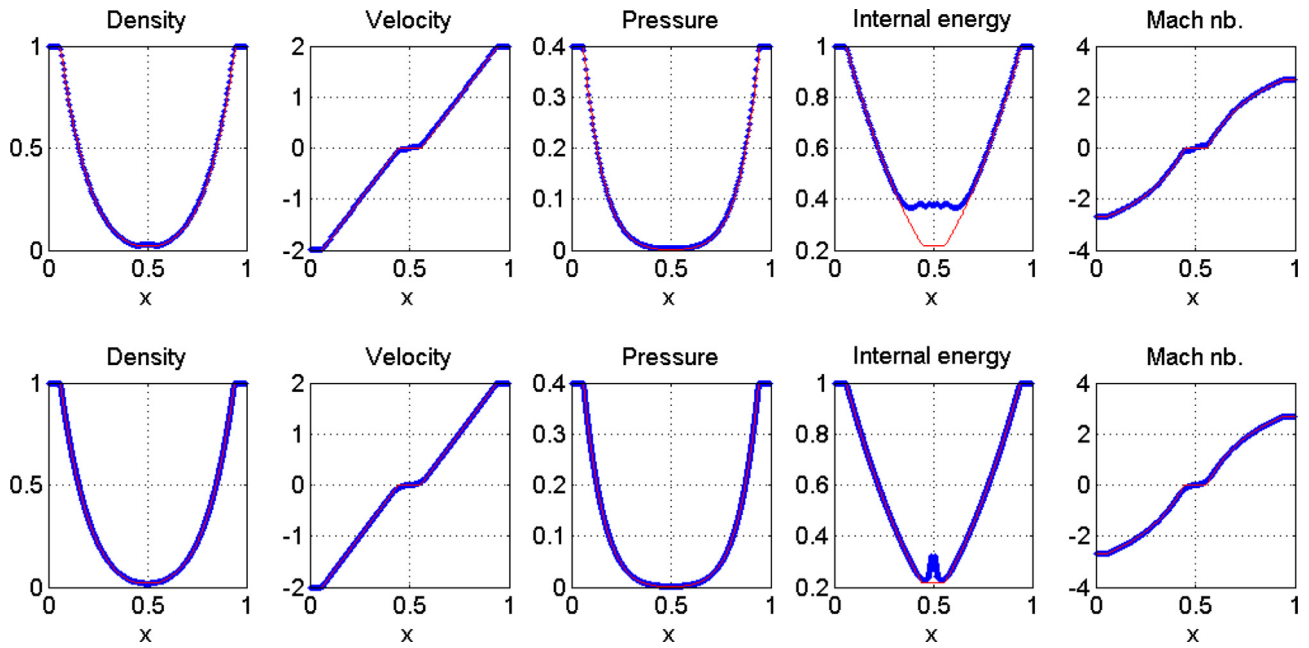
Figure 3
Second-order Lagrange-flux scheme on a double rarefaction near-vacuum case on two different mesh grids, 200 and 2000 points respectively. The solid red line is the analytic solution.

4.2.3 Case With Sonic Rarefaction and Supersonic Contact

The following shock tube case with initial data $(\rho, u, p)_L = (5, 0, 5)$ and $(\rho, u, p)_R = (0.125, 0, 0.1)$ generates a sonic 1-rarefaction, a supersonic 2-contact discontinuity and a 3-shock wave. The final time is $T = 0.16$ and we use 400 mesh points, CFL $= 0.25$. Numerical results show a good capture of the rarefaction wave, without any non-entropic expansion-shock (Fig. 4).

4.2.4 Case of Shock-Shock Hypersonic Shock Tube

This last shock tube problem is a violent flow case made of two hitting fluids with $(\rho, u, p)_L = (1, 5, 1)$ and $(\rho, u, p)_R = (1, -5, 0.01)$. Both 1-wave and 3-wave are shock waves, and the right state has a Mach number of order 40. Final time is $T = 0.16$, we use 400 grid points and the limiter coefficient β is here 1 (equivalent to the minmod limiter). One can observe a nice behavior of the solver: there is no pressure or velocity oscillations at the contact discontinuity, and the numerical scheme preserves the positivity of density, pressure and internal energy (Fig. 5).

5 PERFORMANCE RESULTS

In this section, we compare the new Lagrange-flux scheme to the reference (staggered) Lagrange-remap scheme in terms of Millions of Cell Updates Per second (denoted hereafter as MCUPs). Tests are performed on a standard 2×8 cores Intel Sandy Bridge server E5-2670. Each core has a frequency of 2.6 GHz, and supports Intel's AVX (Advanced Vector Extension) vector instructions. For multicore support, we use the multithreading programming interface `OpenMP`.

In the reference staggered Lagrange-remap solver [4], thermodynamic variables are defined at grid cell centers while velocity variables are defined at mesh nodes. Due to this staggered discretization and the Alternating Direction (AD) remapping procedures, this solver is decomposed into nine kernels. This decomposition mechanically decreases the mean Arithmetic Intensity (AI) of the solver.

On the other hand, the Lagrange-flux algorithm consists in only two kernels with a relative high arithmetic intensity which leads to two Compute-Bound (CB) kernels. In the first kernel, named `PredictionLagrangeFlux()`, an appropriate Riemann solver is called, face fluxes are computed and variables are updated for the prediction. The second kernel, named `CorrectionLagrangeFlux()`, is close in terms of algorithmic steps, since it also uses a Riemann solver, computes fluxes and updates the variables for the correction part of the solver.

In order to assess the scalability and absolute performance of both schemes, we present in Table 1 a performance
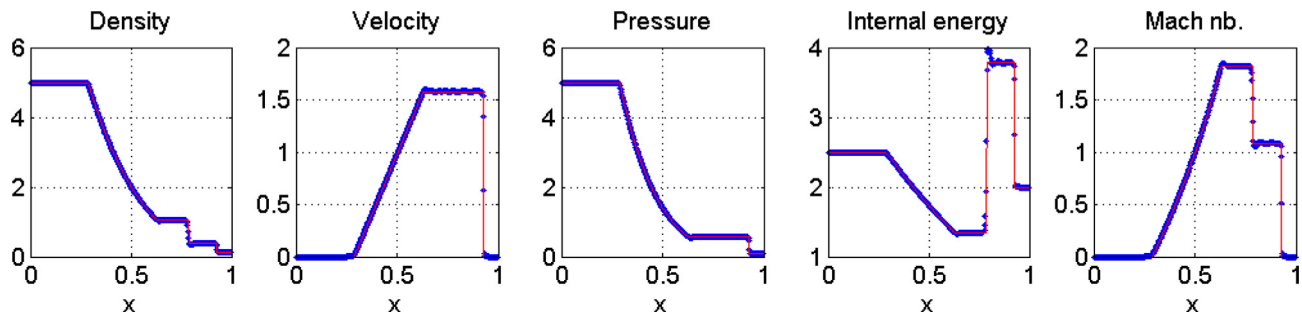
Figure 4

Shock tube problem with sonic rarefaction fan, 400 mesh points, final time T is $T = 0.18$, the solid red line is the analytic solution.
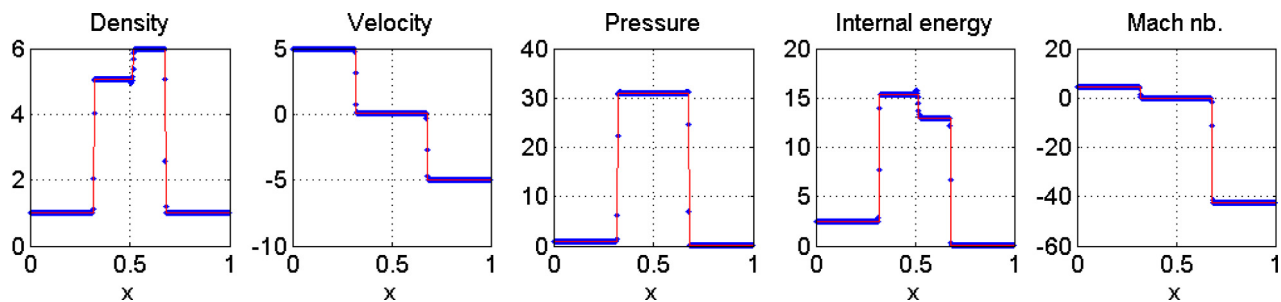

Figure 5

Shock-shock case with subsonic-hypersonic shock, 400 mesh points, final time T is $T = 0.16$, the solid red line is the analytic solution.

TABLE 1

Performance comparison between the reference Lagrange-remap solver and the Lagrange-flux solver in MCUPs, using different machine configurations. Scalability (last column) is computed as the speedup of the multithreaded vectorized version compared to the baseline purely sequential version. Tests are performed for fine meshes, such that kernel data lies in DRAM memory. The Lagrange-flux solver exhibits superior scalability, because it has — by design — better arithmetic intensity.

Scheme	1 core	1 core AVX	16 cores AVX	Scalability
Lagrange-flux	2.6	5.8	81.0	31.1
Reference	2.5	3.8	37.0	14.8

comparison study. First, we notice that the baseline performance – *i.e.* the single core absolute performance without vectorization – is quite similar for the two schemes, as can be seen in the first column. However, we the

Lagrange-flux scheme has a better scalability, due to both vectorization and multithreading: our Lagrange-flux implementation achieves a speedup of 31.1X with 16 cores and AVX vectorization (a 2X speedup from AVX vectorization, which is ideal for this solver with many divisions instructions, and an almost 16X perfect speedup from the multicore usage) whereas the reference Lagrange-remap algorithm reaches a speed-up of only 14.8X. This difference is mainly due to the memory-bound kernels composing the reference Lagrange-remap scheme. Indeed, speedups due to AVX vectorization and multithreading are not ideal for kernels with relatively low intensity since memory bandwidth is shared between cores.

6 DEALING WITH MULTIMATERIAL FLOWS

Although this is not the aim and the scope of the present paper, we would like to give an outline of the possible

extension of Lagrange-flux schemes to compressible multimaterial/multifluid flows, *i.e.* flows that are composed of different immiscible fluids and separated by free boundaries.

For pure Lagrange+remap schemes, usually VOF-based Interface Reconstruction (IR) algorithms are used (Young's PLIC, etc.). After the Lagrangian evolution, for the cells that host more than one fluid, fluid interfaces are reconstructed. During the remapping step, one has to evaluate the mass fluxes per material. From the computational point of view and computing performance, this process generally slows down the whole performance because of many array indirections in memory and specific treatment into mixed cells along with the material interfaces.

If the geometry of the Lagrangian cells is not completely known (as in the case of Lagrange-flux schemes), we have to proceed differently. A possibility is to use Interface Capturing (IC) schemes, *e.g.* conservative Eulerian schemes that evaluate the convected mass fluxes through Eulerian cell edges. This can be achieved by the use of antidiffusive/low-diffusive advection solvers in the spirit of Després-Lagoutière's limited-downwind scheme [18] of VoFire [19]. In a recent work [20], we have analyzed the origin of known artifacts and numerical interface instabilities for this type of solvers and concluded that the reconstruction of fully multidimensional gradients with multidimensional gradient limiters was necessary. Thus, we decided to use low-diffusive advection schemes with a Multidimensional Limiting Process (MLP) in the spirit of [21]. The resulting method is quite accurate, shape-preserving and free from any artifact. We show some numerical results in the next two subsections. Let us emphasize that the interface capturing strategy perfectly fits with the Lagrange-flux flow description, and the resulting schemes are really suitable for vectorization (SIMD feature) with data coalescence into memory.

6.1 Interface Capturing for Pure Advection Problems

Let us first present numerical results on a pure scalar linear advection problem. The forward-backward advection case proposed by Rider and Kothe [22] is a hat-shaped function which is advected and stretched into a rotating velocity field, leading to a filament structure. Then by applying the opposite velocity field, one have to retrieve the initial disk shape. In Figure 6, we show the numerical solutions obtained on a grid 500^2 for both the passive scalar field of variable $z \in [0, 1]$ and the quantity $z(1 - z)$ that indicates the numerical spread rate of the diffuse interface. One can conclude the good behavior of the method, providing both stability, accuracy and shape preservation.
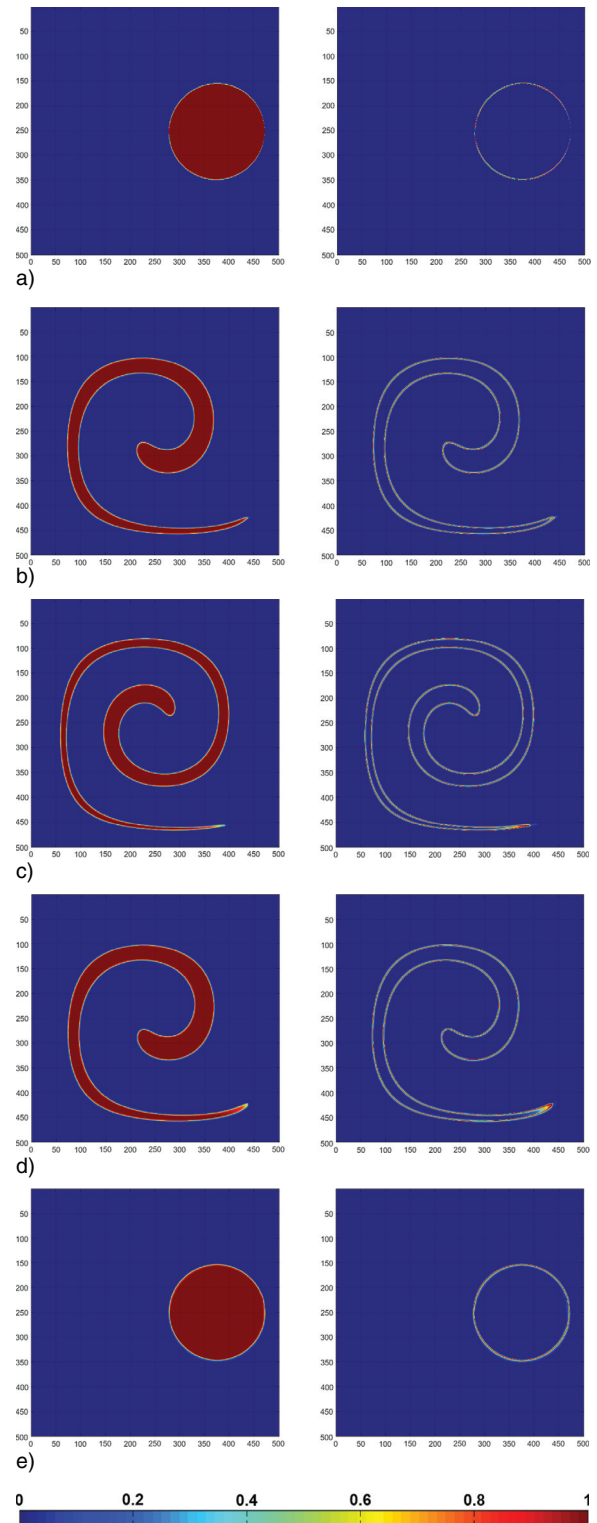

Figure 6

Validation of the low-diffusive interface capturing scheme on the Kothe-Rider advection case, mesh 500^2. Plot of the passive variable $z \in [0, 1]$ on the left column, and interface smearing indicator $\mu = z(1 - z)$ in the right column. a) At initial time $t = 0$, b) at time $t = 3$, c) at time $t = 6$, d) at time $t = 9$, e) at final time $t = 12$.
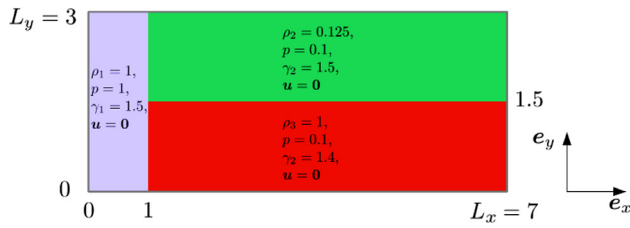
Figure 7

Geometry and initial configuration for the reference triple-point case.
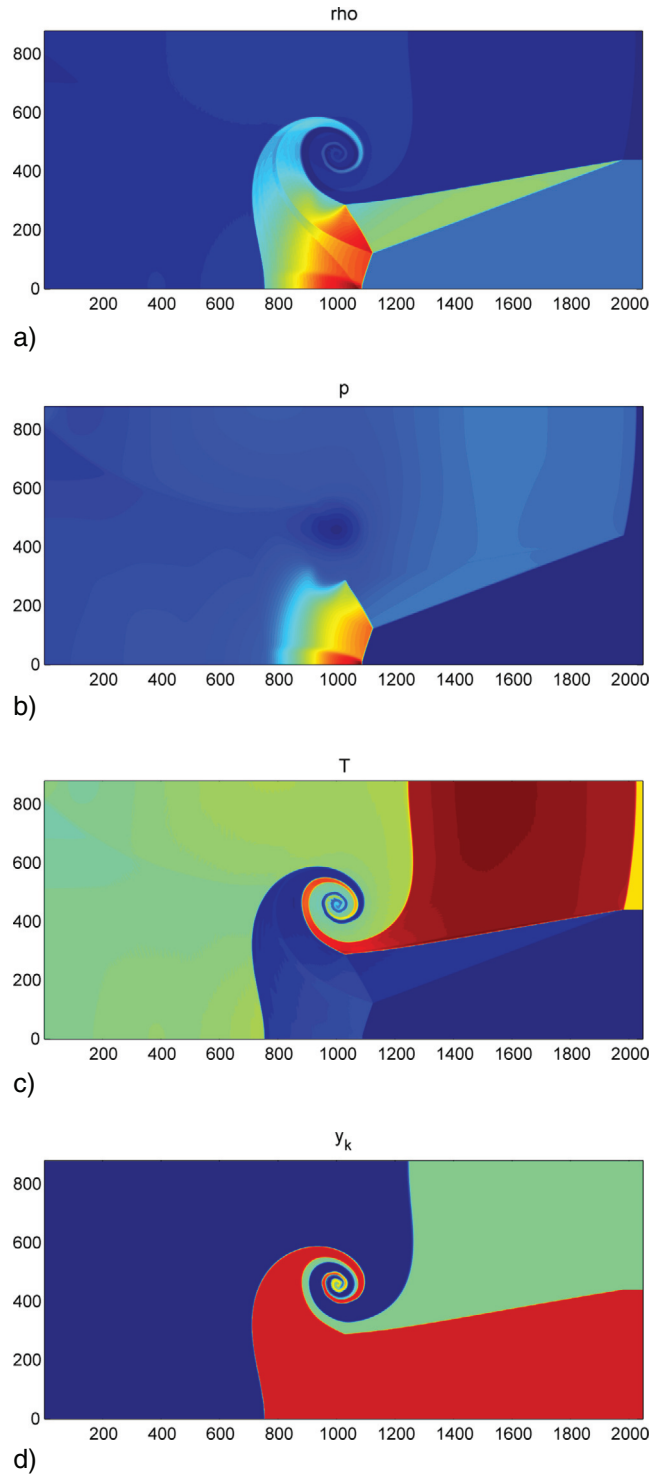

Figure 8

Results on the multimaterial "triple point" case (perfect gases) using a collocated Lagrange+remap solver + low-diffusive interface capturing advection scheme, mesh made of 2048×878 points. Final time is $T = 3.3530$. a) Density field, b) pressure field, c) temperature field, d) colored representation of material indicators.

6.2 Three-Material Hydrodynamics Problem

We then consider our interface capturing method for multifluid hydrodynamics. Because material mass fractions are advected, that is

$$\partial_t y_k + \boldsymbol{u} \cdot \nabla y_k = 0, \quad k = 1, \ldots, N$$

one can use the advection solver of these variables but we prefer dealing with the conservative form of the equations

$$\partial_t (\rho y_k) + \nabla \cdot (\rho y_k \boldsymbol{u}) = 0$$

in order to enforce mass conservation [23]. It is known that Eulerian interface-capturing schemes generally produce spurious pressure oscillations at material interfaces [24, 25]. Some authors propose locally non conservative approaches [26, 27] to prevent from any pressure oscillations. Here we have a full conservative Eulerian strategy involving a specific limiting process which is free from any pressure oscillation at interfaces, providing strong robustness. This will be explained in a next paper.

The multimaterial Lagrange-flux scheme is tested on the reference "triple point" test case, found *e.g.* in Loubère *et al.* [28]. This problem is a three-state two-material 2D Riemann problem in a rectangular vessel. The simulation domain is $\Omega = (0, 7) \times (0, 3)$ as described in Figure 7. The domain is split up into three regions Ω_i, $i = 1, 2, 3$ filled with two perfect gases leading to a two-material problem. Perfect gas equations of state are used with $\gamma_1 = \gamma_3 = 1.5$ and $\gamma_2 = 1.4$. Due to the density differences, two shocks in sub-domains Ω_2 and Ω_3 propagate with different speeds. This creates a shear along the initial contact discontinuity and the formation of a vorticity. Capturing the vorticity is of course the difficult part to compute. We use a rather fine mesh made of 2048×878 points (about 1.8 M cells).

On Figure 8, we plot the density, pressure, temperature fields respectively and indicate the location of the three material zones. One can observe a nice capture of both shocks and contact discontinuities. The vortex is also captured accurately.

CONCLUDING REMARKS AND PERSPECTIVES

This paper is primarily focused on the redesign of Lagrange-remap hydrodynamics solvers in order to achieve better HPC node-based performance. We have reformulated the remapping step under a finite volume flux balance, allowing for a full SIMD algorithm. As an unintended outcome, the analysis has lead us to the discovery of a new promising family of Eulerian solvers – the so-called Lagrange-flux solvers – that show simplicity of implementation, accuracy, and flexibility with a high-performance capability compared to the legacy staggered Lagrange-remap scheme. Interface capturing methods can be easily plugged for solving multimaterial flow problems. Ongoing work is focused of the effective performance modeling, analysis and measurement of Lagrange-flux schemes with comparison of reference "legacy" Lagrange-remap solvers including multimaterial interface capturing on different multicore processor architectures. Because of the multicore+vectorization scalability of Lagrange-flux schemes, one can also expect high-performance on manycore co-processors like Graphics Processing Units (GPU) or Intel MIC. This will be the aim of next developments.

REFERENCES

1 Hirt C.W., Amsden A.A., Cook J.L. (1974) An arbitrary Lagrangian–Eulerian computing method for all flow speeds, *J. Comput. Phys.* **14**, 227–253.

2 Benson D. (1992) Computational methods in Lagrangian and Eulerian hydrocodes, *Comput. Methods Appl. Mech. Eng.* **99**, 235–394.

3 Youngs D. (2007) The Lagrange-remap method, in F.F. Grinstein, L.G. Margolin, W.J. Rider (eds), *Implicit large Eddy simulation: Computing turbulent flow dynamics*, Cambridge University Press.

4 Poncet R., Peybernes M., Gasc T., De Vuyst F. (2016) Performance modeling of a compressible hydrodynamics solver on multicore CPUs, in *IOS Ebook: Parallel Computing: on the road to Exascale*, Series "Advances in parallel computing", Joubert G.R. et al. (ed.), pp. 449-458, DOI: 10.3233/978-1-61499-621-7-449.

5 Williams S., Waterman A., Patterson Roofline D. (2009) An insightful visual performance model for multicore architectures, *Commun. ACM* **52**, 65–76.

6 Treibig J., Hager G. (2010) Introducing a performance model for bandwidth-limited loop kernels, *Proceedings of the Workshop "Memory issues on Multi- and Manycore Platform" at PPAM 2009*, Lecture Notes in Computer Science **6067**, 615–624.

7 Stengel H., Treibig J., Hager G., Wellein G. (2015) Quantifying performance bottlenecks of stencil computations using the Execution-Cache-Memory model, *Proc. ICS'15, Proc. of the 29th ACM on Int. Conf. on Supercomputing*, pp. 207-2016, ACM, New York, ISBN: 978-1-4503-3559-1, DOI: 10.1145/2751205.2751240.

8 Colella P., Woodward P.R. (1984) The numerical simulation of two-dimensional fluid flow with strong shocks, *J. Comput. Phys.* **54**, 115–173.

9 Després B., Mazeran C. (2005) Lagrangian gas dynamics in two dimensions and Lagrangian systems, *Arch. Rational Mech. Anal.* **178**, 327–372.

10 Maire P.-H., Abgrall R., Breil J., Ovadia J. (2007) A cell-centered Lagrangian scheme for compressible flow problems, *SIAM J. Sci. Comput.* **29**, 1781–1824.

11 Maire P.-H. (2009) A high-order cell-centered Lagrangian scheme for two-dimensional compressible fluid flows on unstructured meshes, *J. Comput. Phys.* **228**, 2391–2425.

12 Dukowicz J.K., Baumgardner J.R. (2000) Incremental remapping as a transport/advection algorithm, *J. Comput. Phys.* **160**, 318–335.

13 Schiesser W.E. (1991) *The Numerical Method of Lines*, Academic Press, ISBN 0-12-624130-9.

14 Toro E.F. (2009) *Riemann solvers and numerical methods for fluid dynamics. A practical introduction*, 3rd edn., Springer, ISBN 978-3-540-25202-3, DOI: 10.1007/b79761.

15 Liou M.S. (1996) A sequel to AUSM: AUSM+, *J. Comput. Phys.* **129**, 2, 364–382.

16 Sweby P.K. (1984) High resolution schemes using flux-limiters for hyperbolic conservation laws, *SIAM J. Numer. Anal.* **21**, 995–1011.

17 Sod G.A. (1971) A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws, *J. Comput. Phys.* **27**, 1–31.

18 Després B., Lagoutière F. (2001) Contact discontinuity capturing schemes for linear advection and compressible gas dynamics, *J. Sci. Comput.* **16**, 479–524.

19 Després B., Lagoutière F., Labourasse E., Marmajou I. (2010) An antidissipative transport scheme on unstructured meshes for multicomponent flows, *IJFV* **7**, 30–65.

20 De Vuyst F., Béchereau M., Gasc T., Motte R., Peybernes M., Poncet R. (2016) Stable and accurate low-diffusive interface capturing advection schemes, *Proc. of the MULTIMAT 2015 Conference Würsburg*, arXiv:1605.07091 (preprint).

21 Park J.S., Kim C. (2011) Multi-dimensional limiting process for discontinuous Galerkin methods on unstructured grids, in *Computational Fluid Dynamics 2010*, Kuzmin A. (ed.), Springer, pp. 179–184, ISBN 978-3-642-17883-2.

22 Rider A.J., Kothe D.B. (1998) Reconstructing volume tracking, *J. Comput. Phys.* **141**, 112–152.

23 Bernard-Champmartin A., De Vuyst F. (2014) A low diffusive Lagrange-remap scheme for the simulation of violent air-water free-surface flows, *J. Comput. Phys.* **274**, 19–49.

24 Abgrall R. (1996) How to prevent pressure oscillations in multicomponent flow calculations: A quasi conservative approach, *J. Comput. Phys.* **125**, 150–160.

25 Saurel R., Abgrall R. (1999) A simple method for compressible multifluid flows, *SIAM J. Sci. Comput.* **21**, 1115–1145.

26 Farhat C., Rallu A., Shankaran S. (2008) A higher-order generalized ghost fluid method for the poor for the three-dimensional two-phase flow computation of underwater implosions, *J. Comput. Phys.* **227**, 7640–7700.

27 Bachmann M., Helluy P., Jung J., Mathis H., Müller S. (2013) Random sampling remap for compressible two-phase flows, *Comput. Fluids* **86**, 275–283.

28 Loubère R., Maire P.-H., Shashkov M., Breil J., Galera S. (2010) ReALE: A reconnection-based arbitrary-Lagrangian-Eulerian method, *J. Comput. Phys.* **229**, 4724–4761.