An efficient local time stepping-Discontinuous Galerkin scheme for adaptive transient computations [1]

Jean-François Remacle [*,a], Katia Pinchedez [a],
Joseph E. Flaherty [a], Mark S. Shephard [a]

[a] *Scientific Computation Research Center, CII-7011, 110 8th Street, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, U.S.A.*

Abstract

We present an efficient local time stepping procedure for the adaptive transient solution of hyperbolic conservation laws. The spatial operator is discretized using the discontinuous Galerkin method while a new local time stepping procedure is used for time integration. The time discretization depends on local Courant stability conditions so that elements are advanced in time asynchronously but the scheme includes appropriate step coordination to remain consistent in time. Some results of adaptive mesh refinement calculations are presented for supersonic two and three-dimensional problems and a three-dimensional Rayleigh-Taylor flow instability.

Key words: Discontinuous Galerkin, local time stepping, parallel, CFD

1 Introduction

Transient inviscid flows, such as Rayleigh-Taylor instabilities [1–3] or blast wave propagations, are complex phenomena featuring sharp moving fronts. Explicit time integration is an efficient technique to capture these transient features. With such a strategy, the time-step must be selected to satisfy a Courant-Friedrichs-Lewy (CFL) stability condition [4]. In a method of lines

* Corresponding author: remacle@scorec.rpi.edu

[1] This work was supported by the ASCI Flash Center at the University of Chicago, through contract B341495, by the U.S. Army Research Office through grant DAAG55-98-1-0200, and by the National Science Foundation through grant DMS-0074174.

(MOL) approach, the allowable time step of an explicit time stepping scheme depends on the size of the smallest element of the mesh as well as on the maximum signal speed.

A large disparity in element sizes and signal speeds may then result in an excessive number of time-steps being taken in regions of low activity for the following reasons:

- adaptive h-type mesh refinement is often used to resolve fronts or shocks;
- it is often impossible to control the minimum element size when complex geometry contains small features;
- and the maximum signal speed of multiple fluid problems may vary widely from one fluid to another.

As an illustration, consider the mesh shown in Figure 1 which is used for a computation in §5. This mesh contains 352, 147 tetrahedra and was generated using a commercial mesh generation package [5]. The worst element has a shape-quality measure [6] of 0.2, which is considered very good. If δt is the smallest time step that satisfies the CFL condition for the whole mesh, Table 1 shows the number of elements N_p that would be able to take a time step $2^p \delta t$, $p = 0, 1, \cdots, 13$. The CFL condition has been based on the radius of a sphere inscribed in a tetrahedron.

2^p	1	2	4	8	16	32	64
N_p	45	839	18409	113877	73544	39544	52661
2^p	128	256	512	1024	2048	4096	8192
N_p	23539	13842	7253	4151	2442	1896	105

Table 1
Number of elements in the mesh of Figure 1 that could take a time step of $2^p \delta t$, $p = 0, 1, \cdots, 13$, and satisfy the CFL condition.

Only 45 elements are in the first category ($p = 0$) with the strongest stability constraint, while half of the elements could support a time step that is 32 times larger than this minimum.

Some investigations [7–11] have sought to overcome this inefficiency by using a *local refinement* strategy where spatially-dependent time steps are chosen as a function of the local Courant condition on an element.

Herein, we describe a local refinement procedure based on computing a sequence of integers that represent fractions of a larger (goal) time step. Elements are queued with respect to this sequence and advanced in time for a portion of the goal time step. The scheduling ensures that elements are advanced sequentially without any computational overhead due to searching to see if proper boundary data between elements is available to proceed.

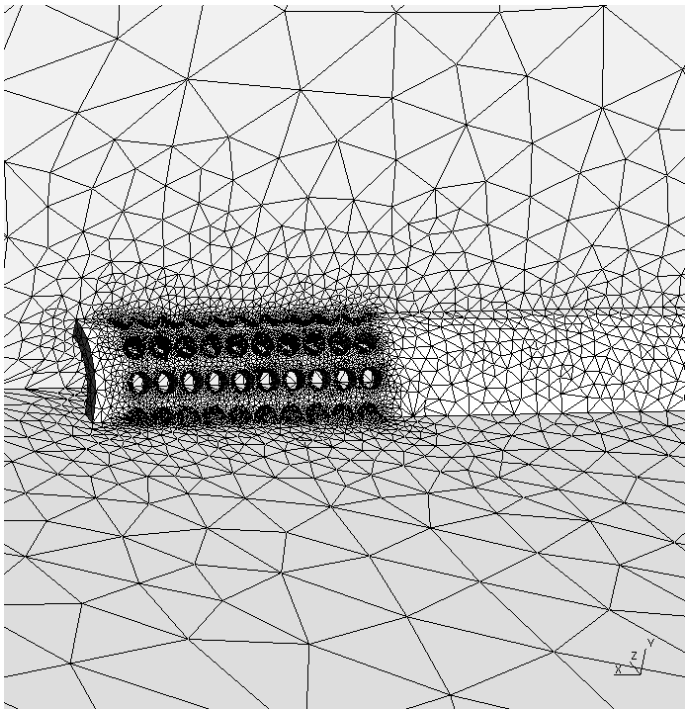

Fig. 1. Surface mesh of portion of a cylindrical tube containing a series of vent holes.

The aim is to propose a workable approach that is applicable to real problems with complex geometries, parallel computing and adaptivity. The local time-stepping procedure is applied to the adaptive (h-refinement) solution of a two-dimensional shock propagation problem and three-dimensional problems involving Rayleigh-Taylor instabilities and flows in vented tubes (§5). The Euler equations of compressible inviscid flow [4] are discretized in space by a discontinuous Galerkin (DG) method [12, 13] and in time by the new local time stepping algorithm with an explicit Runge-Kutta method.

In what follows, variables with a superimposed arrow, such as $\vec{n}$, refer to vectors in $\mathbb{R}^3$ and variables in bold type, such as $\mathbf{u}$, to a field of $(\mathsf{L}^2)^m$.

2 Discontinuous Galerkin Method

Consider an open set $\Omega \subset \mathbb{R}^3$ whose boundary $\partial\Omega$ is Lipschitz continuous with a normal $\vec{n}$ that is defined everywhere. We seek to determine $\mathbf{u}(\Omega, t) : \mathbb{R}^3 \times \mathbb{R} \to (\mathrm{L}^2(\Omega))^m$ as the solution of

$$\partial_t \mathbf{u} \; + \; \mathbf{div}\, \vec{\mathbf{F}}(\mathbf{u}) \; = \; \mathbf{r}. \tag{1}$$

Here $\mathbf{div} \; = (\mathrm{div}\,, \ldots, \mathrm{div}\,)$ is the vector valued divergence operator and $\vec{\mathbf{F}}(\mathbf{u}) = (\vec{F}_1(\mathbf{u}), \ldots, \vec{F}_m(\mathbf{u}))$ is the flux vector with the ith component $\vec{F}_i(\mathbf{u}) : (\mathrm{H}^1(\Omega))^m \to \mathrm{H}(\mathrm{div}\,, \Omega)$.

The functional space $\mathrm{H}(\mathrm{div}\,,\Omega)$ consists of square integrable vector valued functions whose divergence is also square integrable:

$$\mathrm{H}(\mathrm{div}\,,\Omega) \;=\; \left\{ \;\vec{v} \;\in\; \left(\mathsf{L}^2(\Omega)\right)^3 \quad \text{and} \quad \mathrm{div}\,\vec{v} \in \mathsf{L}^2(\Omega) \;\right\}.$$

With the aim of constructing a Galerkin form of (1), let $(\cdot,\cdot)_\Omega$ and $\langle\cdot,\cdot\rangle_{\partial\Omega}$ respectively denote the standard $\mathrm{L}^2(\Omega)$ and $\mathrm{L}^2(\partial\Omega)$ scalar products. Multiply equation (1) by a test function $\mathbf{w} \in V(\Omega)$, integrate over Ω and use the divergence theorem to obtain the following variational formulation:

$$(\partial_t \mathbf{u}, \mathbf{w})_\Omega - (\vec{\mathbf{F}}(\mathbf{u}), \mathbf{grad}\,\mathbf{w})_\Omega + \langle \vec{\mathbf{F}}(\mathbf{u})\cdot\vec{n}, \mathbf{w}\rangle_{\partial\Omega} = (\mathbf{r},\mathbf{w})_\Omega\;, \quad \forall \mathbf{w} \in \mathrm{H}(\mathrm{div}\,,\Omega). \tag{2}$$

Finite element methods (FEMs) involve a double discretization. First, the physical domain Ω is discretized into a collection of N_h elements

$$\mathcal{T}_h = \bigcup_{e=1}^{N_h} e \tag{3}$$

called a mesh. The continuous function space $V(\Omega)$ containing the solution of (2) is approximated on each element e of the mesh to define a finite-dimensional space $V_h(\mathcal{T}_h)$. On each element e of $\mathcal{T}_h$, we obtain:

$$(\partial_t \mathbf{u}_e, \mathbf{w})_e - (\vec{\mathbf{F}}(\mathbf{u}_e), \mathbf{grad}\,\mathbf{w})_e + \langle \mathbf{F}_n, \mathbf{w}\rangle_{\partial e} = (\mathbf{r},\mathbf{w})_e\;, \quad \forall \mathbf{w} \in V(e), \tag{4}$$

where $\mathbf{u}_e$ is a spatial approximation (usually polynomial) of $\mathbf{u}$ on element e.

Now, a discontinuous basis implies that the normal trace $\mathbf{F}_n = \vec{\mathbf{F}}(\mathbf{u})\cdot\vec{n}$ is not defined on ∂e. In this situation, a *n*umerical flux $\mathbf{F}_n(\mathbf{u}_e, \mathbf{u}_{e_k})$ is usually used on each portion ∂_{e_k} of ∂e shared by element e and neighboring element e_k. Here, $\mathbf{u}_e$ and $\mathbf{u}_{e_k}$ are the restrictions of solution $\mathbf{u}$, respectively, to element e and element e_k. This numerical flux must be continuous so $\vec{\mathbf{F}} \in \mathrm{H}(\mathrm{div}\,,\Omega)^m$ and be consistent so $\mathbf{F}_n(\mathbf{u},\mathbf{u}) = \vec{\mathbf{F}}(\mathbf{u})\cdot\vec{n}$. With such a numerical flux, equation (4) becomes

$$(\partial_t \mathbf{u}_e, \mathbf{w})_e - (\vec{\mathbf{F}}(\mathbf{u}_e), \mathbf{grad}\,\mathbf{w})_e + \sum_{k=1}^{n_e} \langle \mathbf{F}_n(\mathbf{u}_e, \mathbf{u}_{e_k}), \mathbf{w}\rangle_{\partial e_k} = (\mathbf{r},\mathbf{w})_e\;, \quad \forall \mathbf{w} \in V(e), \tag{5}$$

where n_e is the number of faces of element e. Only the normal traces have to be defined on ∂e_k and several operators are possible [14, 15]. It is usual to define the trace as the solution of a Riemann problem across ∂e_k. Herein, when we consider problems with strong shocks [15, 16], an exact Riemann solver is used to compute the numerical fluxes and a slope limiter [17] is used to produce monotonic solutions when polynomial degrees $p > 0$.

3 Time Discretization

In order to focus on time integration, let us write (5) in a more abstract form as

$$(\partial_t \mathbf{u}_e \mathbf{w})_e \;=\; \mathcal{S}(\mathbf{u}_e, \mathbf{u}_{e_1}, \ldots, \mathbf{u}_{e_{n_e}}) \tag{6}$$

where $\mathcal{S}$ is the spatial operator defined by (5). Equation (6) forms a system of ordinary differential equations (ODEs) in time.

Consider a discretization of time $t_e = \{t_e^0, t_e^1, \ldots\}$ for an element e, which may vary from element to element, and define

$$\Delta t_e^i \;=\; t_e^{i+1} \;-\; t_e^i. \tag{7}$$

Assuming that $\mathbf{u}_e(t)$ is the piecewise-linear function of time relative to t, see Figure 2, we have

$$\mathbf{u}_e(t) \;=\; \mathbf{u}_e(t_e^i) \;+\; \partial_t \mathbf{u}_e(t)\,(t - t_e^i) \quad , \qquad \text{for } t \in [t_e^i, t_e^{i+1}[, \tag{8}$$

where

$$\partial_t \mathbf{u}_e(t) \;=\; \frac{\mathbf{u}_e(t_e^{i+1}) - \mathbf{u}_e(t_e^i)}{\Delta t_e^i} \quad , \qquad \text{for } t \in [t_e^i, t_e^{i+1}[. \tag{9}$$

We have suppressed the spatial variation of $\mathbf{u}_e$ to emphasize the integration in time.

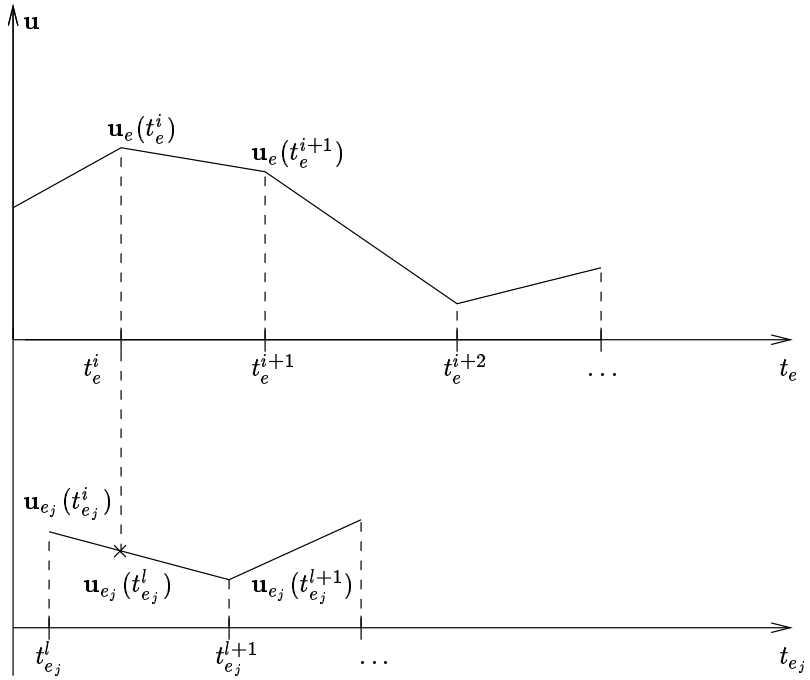

Fig. 2. Time discretization for elements e and e_j, where variables are approximated by piecewise-linear functions in time.

With this approximation, an explicit technique for solving problem (6) would have the following formulation:

$$\left(\frac{\mathbf{u}_e(t_e^{i+1}) - \mathbf{u}_e(t_e^i)}{\Delta t_e^i}, \mathbf{w}\right)_e = \mathcal{S}(\mathbf{u}_e(t_e^i), \mathbf{u}_{e_1}(t_e^i), \ldots, \mathbf{u}_{e_{n_e}}(t_e^i)). \tag{10}$$

Thus, in order to advance $\mathbf{u}_e$ from t_e^i to t_e^{i+1}, we must know the values of $\mathbf{u}_{e_j}(t_e^i)$ on all neighboring elements e_j, $j = 1, 2, \ldots, n_e$, at time t_e^i, see Figure 2. The selection of arbitrary time steps on different elements may, therefore, be computationally inefficient. Furthermore, since an explicit time integration typically requires thousands of time-steps, arbitrary time stepping could require excessive storage of $\mathbf{u}_e$ at several time steps and elements in order to guarantee that $\mathbf{u}_e$ may be evaluated at an arbitrary time t. A *method of lines* (MOL) approach, which uses the same time steps $\{t^0, t^1, \ldots\}$ on every element and which leads to

$$\left(\frac{\mathbf{u}_e(t^{i+1}) - \mathbf{u}_e(t^i)}{\Delta t^i}\mathbf{w}\right)_e = \mathcal{S}(\mathbf{u}_e(t^i), \mathbf{u}_{e_1}(t^i), \ldots, \mathbf{u}_{e_{n_e}}(t^i)), \tag{11}$$

overcomes these difficulties but may possibly be more expensive, as we shall now explain.

Explicit integration schemes are stable only when the time step is chosen sufficiently small [4]. Let $v_e(t^i)$ be the maximum signal speed on e at time t^i, and r_e be a measure of the size of e. An analysis of (11) then shows this method to be linearly stable when the Courant condition

$$\Delta t_e^i \leq \alpha \frac{r_e}{v_e(t^i)} \tag{12}$$

is satisfied. The above constant $\alpha > 0$ depends on the particular numerical method. In order to stabilize a MOL, we would have to choose a time step satisfying

$$\Delta t^i \leq \min_{e \in \mathcal{T}_h} \Delta t_e^i. \tag{13}$$

This inequality could imply a severe limitation when elementary time steps (12) vary widely due to mesh size differences. We have presented in Table 1 some statistics about allowable time steps where element sizes, hence, r_e, vary by a factor of 8000. Assuming, as likely, that accuracy is acceptable, the stability restriction (13) would then be a major limitation.

If the steady state, $\partial_t \mathbf{u} = 0$, is the only solution of interest, it is possible to use an inconsistent but stable local time-stepping scheme. Thus, each element may advance at its own time step according to its local Courant condition (12). The resulting scheme formally looks like equation (10). However, since the steady state is the only solution of interest, each element may use the latest available solution from its neighbors. No coordination is necessary. Hence,

there is no additional storage requirements. The temporal integration is no longer consistent, although, since the steady state is independent of time, consistency is regained as $t_e^i \to \infty$.

4 Consistent Local Time Stepping

As a compromise between the excessive storage requirements of an arbitrary local time-stepping scheme and the inefficiency of a MOL, we consider a method using a large (goal) time step Δt to synchronize the solution. Thus, all elements will have a set of common times, $t_e^i = t$, $\forall e \in \mathcal{T}_h$, where solutions are synchronized. The initial time $t = 0$ will be one such time. Flaherty et al. [9] have used this notion of goal times but were slightly more general in the sense that they did not require time-step synchronization. The present process of advancing the solution from goal time to goal time in large steps of size Δt is slightly simpler and presents additional scheduling advantages.

Consider an element e. Focusing on explicit time integration, we ensure that e has the necessary boundary data to advance in time from t_e^i without checking its neighbors e_j by requiring

$$t_{e_j}^i \;\le\; t_e^i \;\le\; t_{e_j}^{i+i}, \quad \text{for any } j \in [1, n_e]. \tag{14}$$

Consider a positive integer m and a set of its factors $\mathcal{F}_m = \{m_1, m_2, \ldots, m_k\}$ with $1 = m_1 < m_2 < \cdots < m_k$ and m/m_j, $j = 1, 2, \ldots, k$, integers. Let δt^i be a time-step that satisfies the global stability condition (13). We choose

$$\Delta t_e^i \;=\; m_j \, \delta t \tag{15}$$

where $j \in [1, k]$ is the maximum integer that satisfies the local Courant condition

$$m_j \, \delta_t \;<\; \alpha \, \frac{r_e}{v_e(t^i)}. \tag{16}$$

Valid local time steps are limited to the set $\mathcal{F}_m \cdot \delta t$. With this restriction, it is possible to develop an "efficient" local time-stepping algorithm such that:

- $\mathbf{u}_e$ needs only be stored at the current and previous local times;
- elements can be queued once and processed sequentially without restarting the whole process.

These conditions may be satisfied by clustering elements into k categories that depend on the chosen local time step. Let $\mathcal{G}(m_j)$ be the set of elements with local time-step $m_j \delta_t$. At the goal time t, any group $\mathcal{G}(n_j)$, $j = 1, 2, \ldots, k$, can be advanced in time. However, at subsequent local times, only those sets at a time that are in the range of the times of all other groups may advance.

As an example, suppose $\mathcal{F}_m = \{1, 2, 3, 6\}$ and the goal time step is $\Delta t = 6\delta t$. As shown in Figure 3, all sets of elements may take one local time step to advance from the goal time t. Subsequent to this, the elements in set $\mathcal{G}(1)$ may advance from time $t + \delta t$ to time $t + 2\delta t$ since boundary data is available from every set of elements. Subsequently, either set $\mathcal{G}(1)$ or $\mathcal{G}(2)$ may advance. This process continues with the final sequence order for processing the sets from t to $t + \Delta t$ being $\mathrm{S} = \{1, 2, 3, 6, 1, 2, 1, 1, 3, 2, 1, 1\}$. This method is efficient since the members of the sets $\mathcal{G}(m_j)$, $j = 1, 2, \ldots, k$, and the processing schedule S may be calculated before the integration in time.

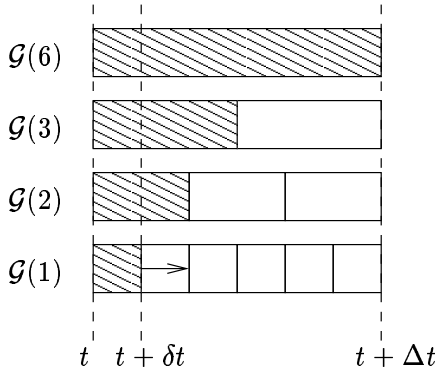

Fig. 3. Local time-stepping sequence for sets $\mathcal{F}_m = \{1, 2, 3, 6\}$. Shaded areas indicate the sizes of the local time steps for each set. At this stage, only set $\mathcal{G}(1)$ may advance in time since $\mathbf{u}$ is known at $t + \delta t$ by every set.

More generally, we construct a sequence $\mathrm{S}_i(n, \mathcal{F}_n)$ of integers in the following way:

$$\mathrm{S}_1 \;=\; \{m_1\},\; \mathrm{S}_2 \;=\; \{m_1, m_2\},\; \ldots\;,\; \mathrm{S}_i \;=\; \{\mathrm{S}_{i-1}, m_i\},\; \ldots,\; \mathrm{S}_k \;=\; \mathcal{F}_m. \quad (17)$$

Let $c(i, j)$ be the number of times that m_j is present in S_i, then

$$m_j\; c(i, j) \;\leq\; m_l\; c(i, l), \quad \text{for } l = 1, 2, \ldots, k. \quad (18)$$

The sequence terminates when

$$m_j\; c(i, j) \;=\; m, \quad \text{for } j \in [1, k]. \quad (19)$$

The number of steps to reach convergence is

$$Q(m, \mathcal{F}_m) \;= \sum_{i=1}^{k} \frac{m}{m_i}. \quad (20)$$

The converged sequence is $\mathrm{S} = \mathrm{S}_Q(m, \mathcal{F}_m)$. The following proposition presents an interesting property of S_Q.

Proposition 1 $\mathrm{S}_Q(2m, \{\mathcal{F}_m, 2m\})$ *may be deduced from* $\mathrm{S}_Q(m, \mathcal{F}_m)$ *as*

$$\mathrm{S}_Q(2m, \{\mathcal{F}_m, 2m\}) \;=\; \{\; \mathrm{S}_Q(m, \mathcal{F}_m)\;,\; 2m\;,\; \mathrm{S}_Q(2, \mathcal{F}_m\;)\}. \quad (21)$$

proof: If $\mathrm{S}_Q(2,\{1,2\}) = \{1,2,1\}$ then $\mathrm{S}_Q(n,\mathcal{F}_n)$ is a converged sequence. At convergence, all groups have reached n so that the whole sequence can be seen as a unique group advancing once to a time n. Sequence $\mathrm{S}_Q(n,\mathcal{F}_n)$ is then equivalent to the other sequence $\mathrm{S}_Q(n,\{n\})$ and $\mathrm{S}_Q(2n,\{\mathcal{F}_n,2n\}) = \mathrm{S}_Q(2n,\{n,2n\}) = \{n,2n,n\}$.

Proposition 1 presents an easy inductive approach for finding S_Q when $m = 2^p$ for p a non-negative integer and $\mathcal{F}_n^p = \{1,2,2^2,\ldots,2^p\}$. In this case, $\mathrm{S}_1(1,\{1\}) = \{1\}$, $\mathrm{S}_3(2,\{1,2\}) = \{1,2,1\}$, $\mathrm{S}_Q(4,\{1,2,4\}) = \{1,2,1,4,1,2,1\}$, ... Such sequences shall be important for adaptive mesh refinement when elements and time steps are refined by bisection.

We can calculate the average speedup of the local refinement method relative to a MOL by assuming that elements have been equidistributed between local time steps, i.e., by assuming that all groups $\mathcal{G}(m_j)$, $j = 1,2,\ldots,k$, contain the same number of elements. In this case, each advancement in time of a group has the same computational cost. The number of local time steps is the dimension Q of S_Q. With a MOL approach, the global time step would have to be restricted to δt in order to satisfy the stability condition (13). Thus, the number of time steps is km. The number of time steps required by the LRM is given by (20), and we may calculate an average speedup factor s_{avg}

$$s_{avg}(\mathcal{F}_m) \;=\; \frac{km}{Q(m,\mathcal{F}_m)} \;=\; \frac{k}{\displaystyle\sum_{i=1}^{k} 1/m_i}. \tag{22}$$

For the example presented in Figure 3, this speedup factor would be equal to $s_{avg} = 4/(1+1/2+1/3+1/6) = 2$.

The maximum speedup s_{max} occurs when a single element requires the smallest time step while the rest use the largest. Then, the speedup factor is

$$s_{max} \;=\; m_k. \tag{23}$$

For the example of Figure 3, the maximum speedup factor is then $s_{max} = 6$.

5 Numerical Examples

We present the results of three compressible inviscid flow problems involving the solution of the Euler equations [4] by a DG method. The three-dimensional Euler equations have the form (1) with

$$\mathbf{u} \;=\; \Big(\, \rho, \rho v_x, \rho v_y, \rho v_z, E \Big)^t, \tag{24}$$

$$\vec{\mathbf{F}}(\mathbf{u}) \;=\; \Big(\, \rho\,\vec{v}\,,\; \rho\, v_x\,\vec{v} \,+\, P\,\vec{e}_x\,,\; \rho\, v_y\,\vec{v} \,+\, P\,\vec{e}_y\,,\; \rho\, v_z\,\vec{v} \,+\, P\,\vec{e}_z\,,\; (\rho\, E \,+\, P)\,\vec{v}\,\Big), \tag{25}$$

$$\mathbf{r} \;=\; \mathbf{0}. \tag{26}$$

Here ρ is the fluid density, $\vec{v}$ the velocity, E the internal energy, P the pressure and $\vec{e}_x$, $\vec{e}_y$ and $\vec{e}_z$ are the unit vectors in the x, y and z directions, respectively. An equation of state of the form $P = P(\rho, E)$ is also necessary to close the system. The DG method and the associated software [12] may be used for any equation of state which only enters the numerical method through the calculation of the numerical flux. Here, we have chosen the perfect gas equation of state

$$P \;=\; (\gamma - 1)\,\rho\,\left[E - \frac{\|\vec{v}\|^2}{2}\right] \tag{27}$$

with the gas constant $\gamma = 1.4$.

5.1 Backward Facing Step

Consider the parallel Mach 3 flow of a gas in a channel where a step is impulsively inserted [15, 18]. The channel is of length 3, height 1 and the step is situated at $x = 0.6$ and of height 0.2. The initial conditions are $P = 1$, $\rho = 1$ and $\vec{v} = (M_s\sqrt{\gamma}, 0)$.

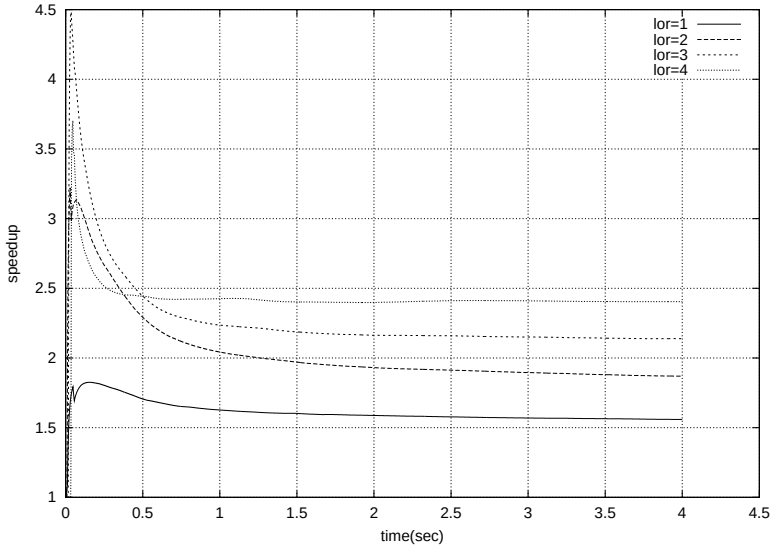

Fig. 4. Speedups as a function of t for the backward facing step computation using refinement strategies as shown in Table 2.

l_r	m	$\mathcal{F}_m$	$Q(n, \mathcal{F}_m)$	average speedup s_{avg}	actual Speedup
0	1	$\{1\}$	1	1	1.00
1	2	$\{1, 2\}$	3	$4/3 = 1.33$	1.56
2	4	$\{1, 2, 4\}$	7	$12/7 = 1.71$	1.82
3	8	$\{1, 2, 4, 8\}$	15	$32/15 = 2.13$	2.14
4	16	$\{1, 2, 4, 8, 16\}$	31	$80/31 = 2.58$	2.40

Table 2
Speedup factors corresponding to various levels of refinement for the backward facing step computation.

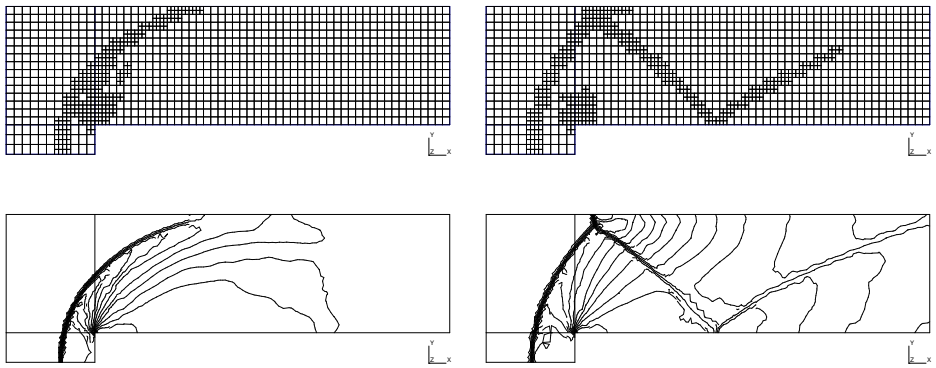

Fig. 5. Backward facing step computation using one level of refinement for the Mach $M_s = 3$ flow coming from the left. Mesh refinements (top) and density profiles (bottom) at $t = 0.25$ (left) and $t = 1.0$ (right).

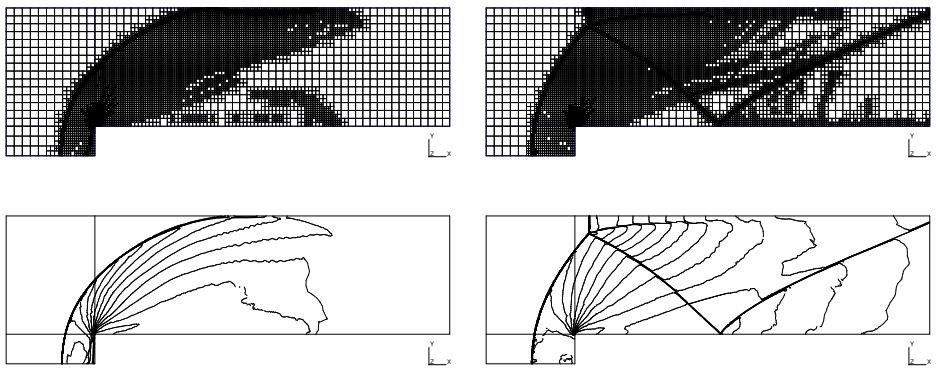

Fig. 6. Backward facing step computation using four level of refinement for the Mach $M_s = 3$ flow coming from the left. Mesh refinements (top) and density profiles (bottom) at $t = 0.25$ (left) and $t = 1.0$ (right).

The transient flow features shock propagation, shock reflections and stagnation

flow. In order to accurately capture these features, we use the DG method with piecewise-linear polynomial approximation and adaptive h-refinement. The non-conforming spatial refinement procedure divides elements into four smaller ones by bisecting their edges. The mesh is refined every ten time steps, so that the total number of mesh refinements may exceed 1000, see Figures 5 and 6. A more thorough description of the refinement procedure can be found in Remacle et al. [12].

We solve this problem using l_r levels of local refinement strategies as shown in Table 2. The set $\mathcal{F}_m = \{1, 2, 2^2, \ldots, 2^{l_r+1}\}$ seems to be the obvious choice. We have also tried other local refinement strategies but found no improvement. In Table 2, we present the average and actual speedups of the method. The actual speedup is the ratio of the c.p.u. time without local time stepping to the c.p.u. time with local time stepping. For this problem, we see that the average speedup s_a defined by (22) is a good approximation of the actual speedup.

Figure 4 presents the behavior of the speedup of local time stepping with respect to time, $0 \leq t \leq 4$. In the beginning of the simulation, a shock forms at the facing step and propagates backwards, grows, hits the top wall and, then hits the bottom wall. The refined region is, thus, bigger at the end of the calculation that at the beginning. This phenomenon is reflected in Figure 4. The speedup starts at 1.0 (no refinement), increases rapidly, and then decreases as the flow becomes steady. With $l_r = 4$, the actual speedup is approximatively 2.4.

Shocks are finely resolved for the case $l_r = 4$ as well as the contact discontinuity near the upper wall: a region of maximum discretization captures those discontinuities as we can see in Figures 5 and 6.

5.2 Double Mach reflection of a strong shock

The second test problem is the classical problem of the reflection of a planar shock moving at Mach 10 by a wedge with a half angle of 30^o [15, 16], see Figure 7.

The computational domain as shown with dashed lines in Figure 7, is a 4×1 unit rectangle oriented along the oblique surface. The reflecting wall lies on the bottom of the computational domain, starting at $x = 1/6, y = 0$. Boundary conditions at the top ($y = 1$) are set to correspond to the boundary conditions associated with the exact motion of a Mach 10 shock. The physical parameters for the gas ahead of the shock are $P_1 = 1$ and $\rho_1 = 1.4$. The Rankine-Hugoniot

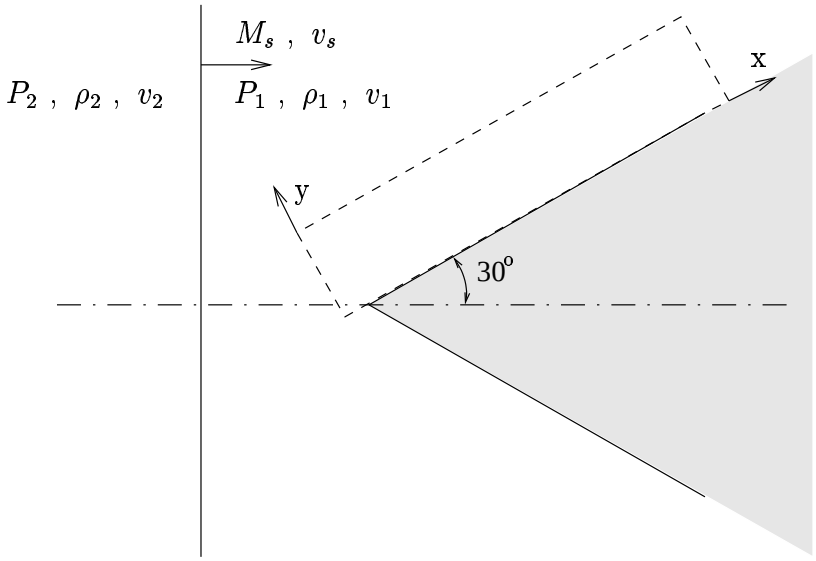

Fig. 7. Double Mach reflection of a strong shock traveling from left to right. Conditions ahead of the shock are identified with a subscript 1 and those behind the shock with a subscript 2.

relations

$$\begin{cases} v_s \;=\; M_s\,\sqrt{\gamma\,P_1/\rho_1} \;=\; 10, \\ p_2/p_1 \;=\; (2\,\gamma\,M_s^2 \;-\; (\gamma-1))/(\gamma+1), \\ \rho_2/\rho_1 \;=\; (\gamma+1)\;M_s^2/((\gamma-1)\;M_s^2+2), \\ \rho_1\,v_s \;=\; \rho_2(v_s-v_2), \end{cases} \tag{28}$$

are used to compute post shock conditions. Results of computations for $l_r = 1$ and $lr = 4$ are presented in Figure 8.

Unlike the backward facing step, local sound and fluid velocities are very different. This makes the LRM more useful since the local Courant numbers depend on both mesh sizes and fluid parameters. In the Euler equations, the maximum signal speed v_e (maximum eigenvalue of the Jacobian of the flux) is $v_e = c + ||\vec{v}||$. This value is attained in the jet formed by the double Mach reflection as illustrated in Figure 9. Since the solution of this problem is self similar, the area of the jet is increasing and takes a larger portion of the domain. Hence, we expect the efficiency of the LRM to decrease with time.

As an illustration, we have computed the speedups to time t for the sets $\mathcal{F}_k = \{1, 2, 4, \ldots, 2^k\}$ given in Figure 10. Speedup decreases as expected and does not improve when $k > 6$ since $\mathcal{G}(2^k)$ is always empty. Figure 10 shows that the optimum LRM strategy must account for more than the maximum level of refinement l_r. Since our local time stepping procedure involves very

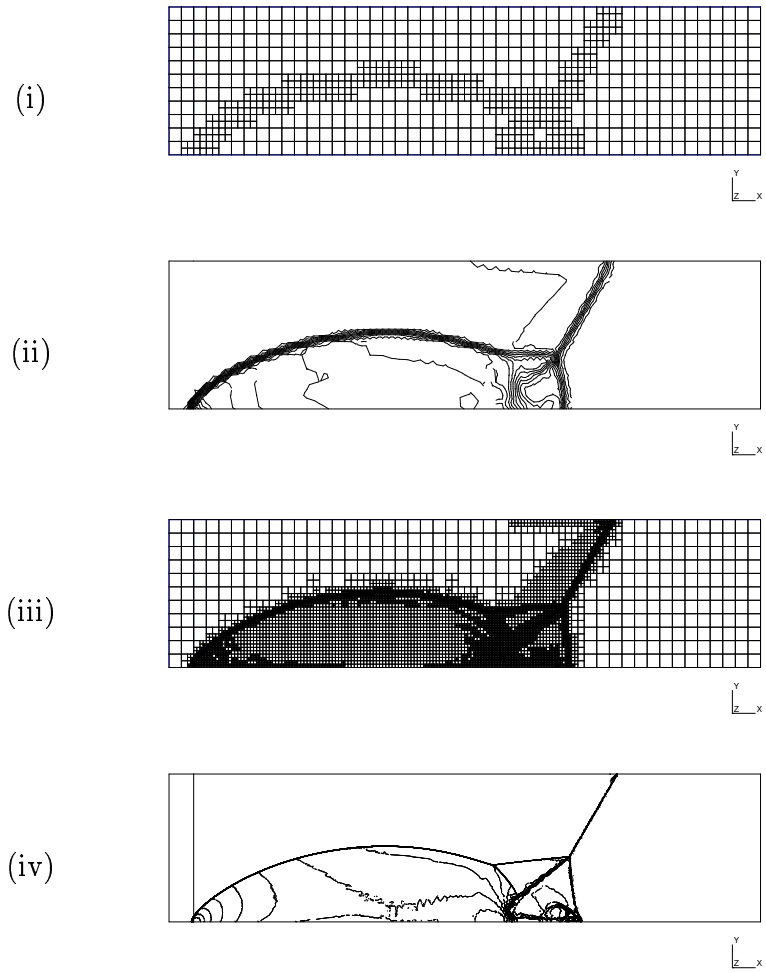

Fig. 8. Grids and density contours for the double Mach reflection at $t = 0.2$ with $l_r = 1$, images (i) and (ii), and $l_r = 4$, images (iii) and (iv).

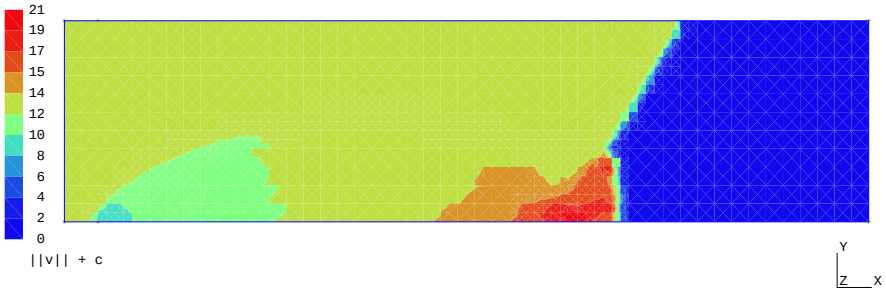

Fig. 9. Maximum signal speed at $t = 0.2$ for the double Mach reflection.

small computational overhead, we have found that the best strategy is to take k large enough so that no element has a time step greater than half its stability limit. With four levels of refinement, the jet formed by the double

Mach reflection, which is usually difficult to capture, is clearly resolved.

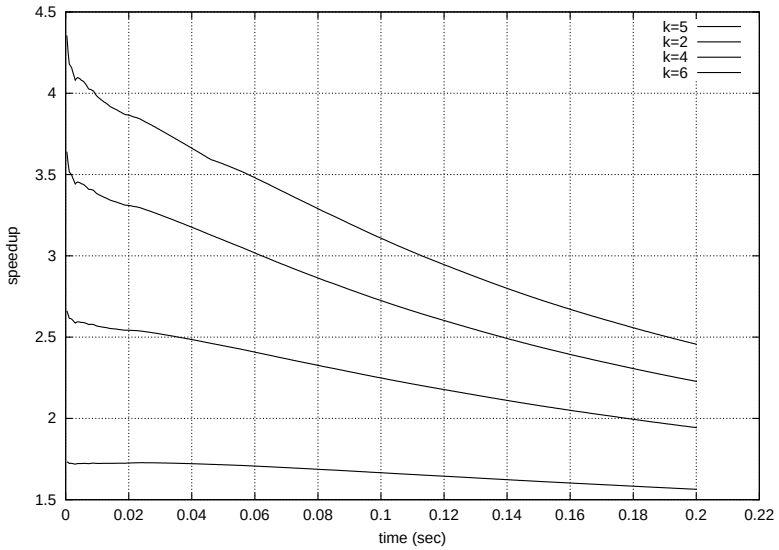

Fig. 10. Speedups for the double Mach reflection, as a function of time corresponding to different sets $\mathcal{F}_k = \{1, 2, 4, \ldots, 2^k\}$, $k = 2, 4, 5, 6$.

When shocks move rapidly as in this problem, the groups $\mathcal{G}(n_j)$ must be updated after each local time step. The local Courant condition (12) may indeed vary dramatically during a goal time step, especially when k is large.

5.3 Three-Dimensional Rayleigh-Taylor Instability

A Rayleigh-Taylor instability involves a heavy (cold) fluid overlying a light (warm) fluid [2, 3]. We consider two inviscid fluids initially in hydrostatic (unstable) equilibrium in a cavity. The upper half of the cavity is filled with a fluid of density two while the lower part is filled with a fluid of unit density. The initial pressure corresponds to hydrostatic equilibrium. An initial perturbation of the velocity initiates the instability. The flow is again governed by the Euler equations (24)-(25), with a body force corresponding to the gravity $\vec{g} = \{0, -1, 0\}$:

$$\mathbf{r} \;=\; (0, 0, -\rho, 0, 0)^t. \tag{29}$$

We solved this problem using $l_r = 3$ and $p = 1$ on a four-processor computer (Intel Pentium 3 at 700 MHz.). At the beginning of the computation, the number of degrees of freedom is less than one million and reaches ten million after 1.6 seconds of effective computation. Dynamic load balancing is used to preserve scalability [19]. Figure 12 shows the mesh configuration near the interface and Figure 13 shows density contours. The LRM may be applied in parallel with a global reduction used to compute Δt. In order to preserve the efficiency of the method, rounds of communication at stage j of the LRM

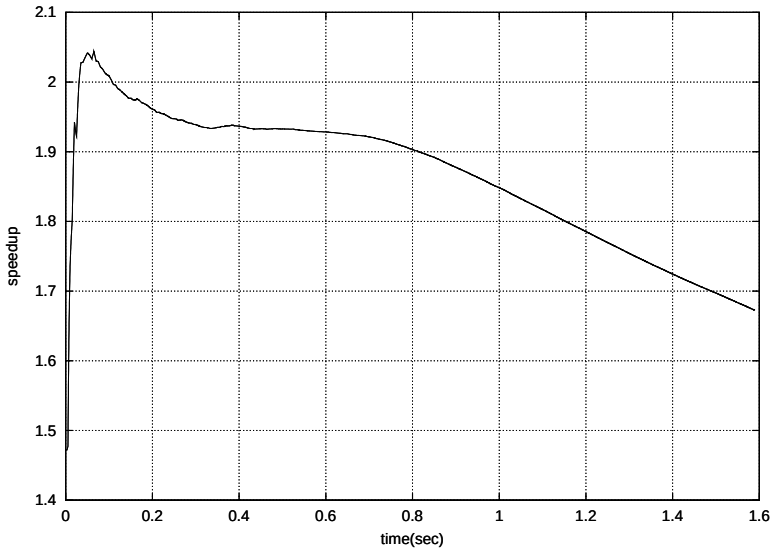

Fig. 11. Speedup for the Rayleigh-Taylor problem.

should remain local to the group $\mathcal{G}(m_j)$. Thus, only parts of the solution should be communicated at each small time step. Speedup due to the LRM is presented in Figure 11. In such unstable problems, the portion of the domain that needs refinement grows as the instability progresses. As we expected, the speedup decreases in time to reach a value of 1.7, see Figure 11.

5.4 Vented Tube

In this last example, we compute the flow in and about a quadrant of a cylindrical tube containing a series of vent holes, see Figure 1. This problem simulates the flow in a cannon having a "pepper pot" muzzle brake [20].

In the tube, the initial pressure P and velocity $\vec{v}$ vary linearly, with $P = 53.556 \cdot 10^6$ Pa and $v = 0$ at the closed end (breech) of the tube, and $P = 60.133 \cdot 10^6$ Pa and $v = 934.82$ m/s at the open end (muzzle) of the tube, whereas the temperature $T = 1500$ K is constant. Moreover, the flow is hypersonic in the jet. The exterior pressure and temperature have the values $P = 10^5$ Pa and $T = 288$ K. No penetration boundary conditions are applied on all parts of the tube.

Figures 14 and 15 show pressure and density distributions at $t = 0.24$ and 1 ms. We were not able to compare the speedup of the LRM since the MOL required time steps of the order 10^{-9} s. For the LRM, we used a time-step sequence in powers of two with a goal time step of $\Delta t = 1024\ \delta t = 10^{-6}$ s. We needed one thousand of these time steps in order to reach $t = 10^{-3}$ s. We observed a theoretical speedup of 25; however, this does not include the computational overhead of our scheduling strategy for the LRM. This arises

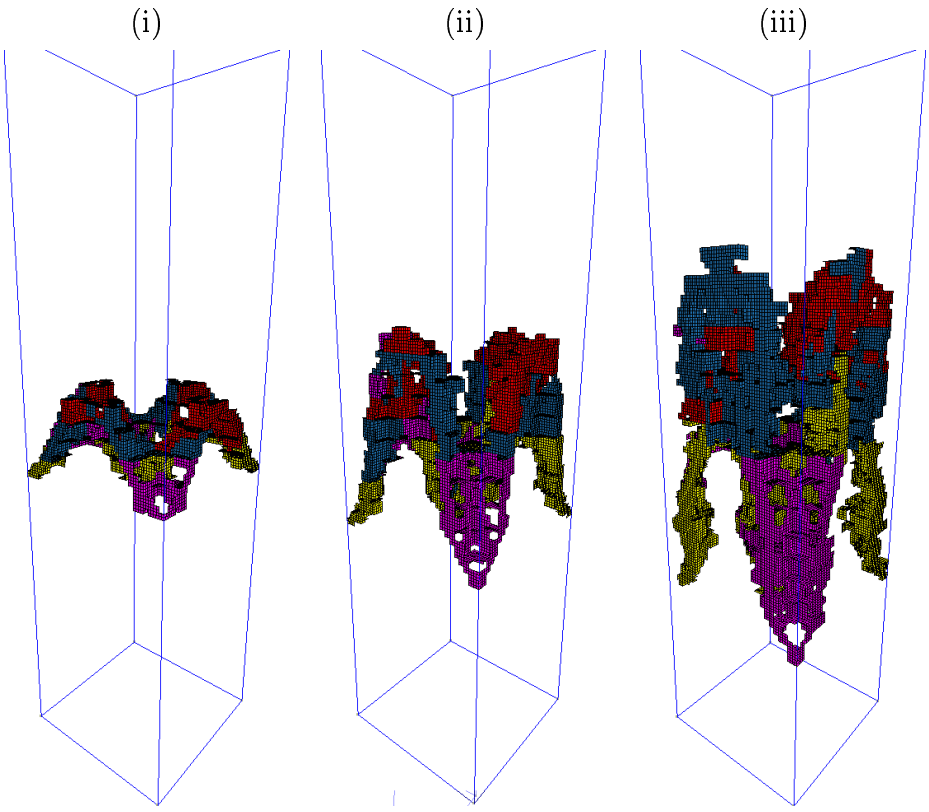

Fig. 12. The most refined elements in the mesh for a Rayleigh-Taylor instability. Image (i) shows the mesh after 24 refinement steps, image (ii) after 72 refinements, and image (iii) after 104 refinements. Element colors correspond to processor `ID`'s.

from the extra numerical fluxes that are computed for faces shared by elements in different groups $\mathcal{G}(m_j)$. If the groups of elements are compact regions in the mesh, most of the numerical fluxes are computed for elements belonging to the same group $\mathcal{G}(m_j)$. This is generally the case, smooth gradation in mesh density is usually preferred and regions with very different Mach numbers are also nested and separated by shocks. We did not observe a significant c.p.u. time penalty due to these extra calculations in previous examples. Therefore, we conjecture that 25 is an accurate measure for the speedup.

6 Conclusions

We have presented an efficient local time stepping method that can provide a substantial speedup with explicit time stepping. The main advantages of the new method are its simplicity of implementation and its computational efficiency. We have shown the efficiency of the method on four compressible flow problems which involve important considerations such as limiting and parallel computation. The method has been proven suitable for solving problems

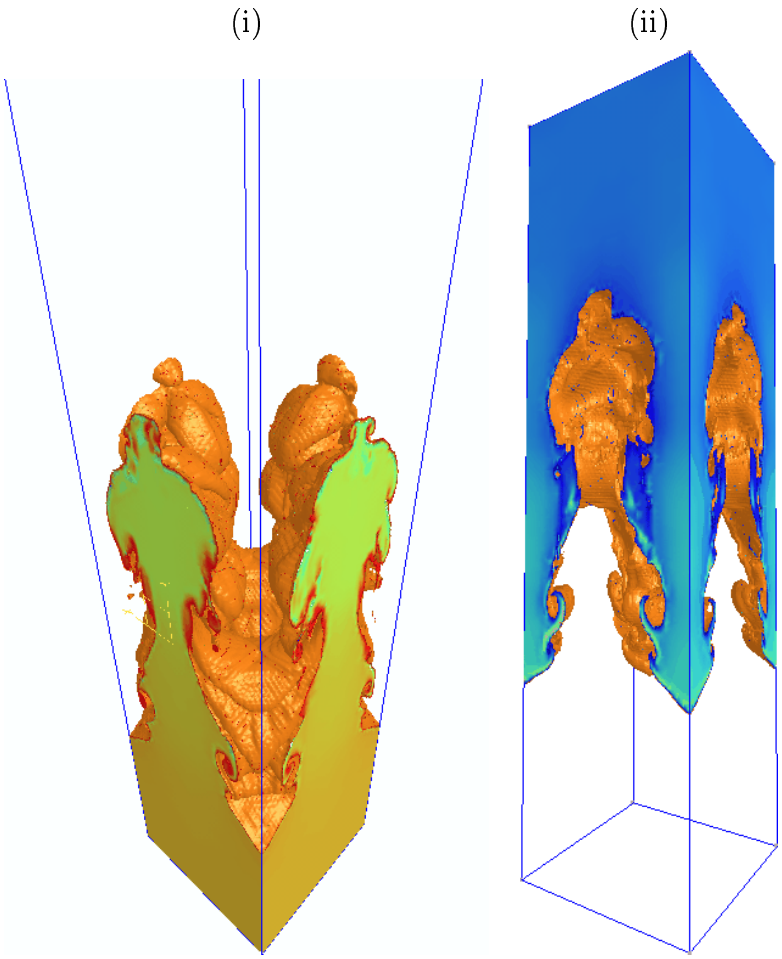

Fig. 13. Fluid density ρ for the Rayleigh Taylor instability. Image (i) shows the subdomain where $\rho < 1.5$ and image (ii) the subdomain where $\rho > 1.5$

involving realistic three-dimensional geometries and complex flows.

References

[1] P. G. Drazin, W. H. Reid, Hydrodynamic Stability, Cambridge University Press, 1982.

[2] Y.-N. Young, H. Tufo, A. Dubey, R. Rosner, On the miscible Rayleigh-Taylor instability, submitted to J. Fluid Mech. .

[3] J. Glimm, J. Grove, X. Li, W. Oh, D. C. Tan, The dynamics of bubble growth for Rayleigh-Taylor unstabe interfaces, Physics of Fluids 31 (1988) 447–465.

[4] R. LeVeque, Numerical Methods for Conservation Laws, Birkhäuser-Verlag, 1992.

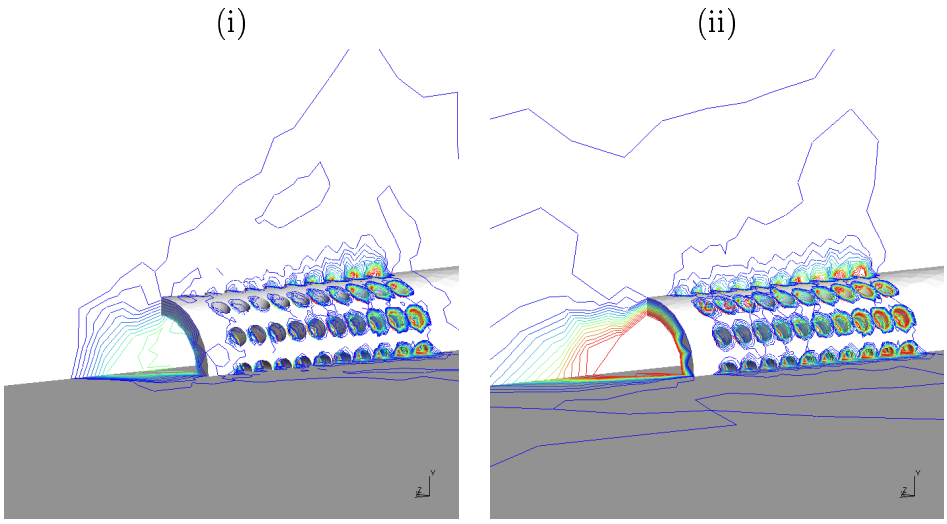

Fig. 14. Isobars on surfaces for the vented tube at $t = 0.24$ ms (i) and $t = 1.0$ ms (ii).

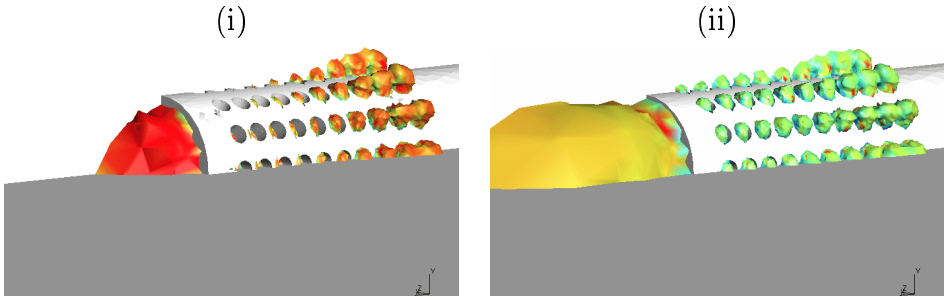

Fig. 15. Density isosurfaces $\rho = 5$ for the vented tube at t=0.24 ms (i) and t=1.0 ms (ii). Colors indicate values of $\|\vec{v}\|$.

[5] Simmetrix Inc., MeshSim, http://www.symmetrix.com, clifton Park, NY (2001).

[6] P. George, H. Borouchaki, Delaunay Triangulation and Meshing: Application to Finite Elements, Hermes, Paris, 1998, Ch. 1.

[7] C. Dawson, R. Kirby, High resolution schemes for conservation laws with locally varying time steps, SIAM Journal on Scientific Computing 22 (6) (2001) 2256–2281.

[8] M. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, Journal of Computational Physics 53 (1984) 484–512.

[9] J. Flaherty, R. Loy, M. Shephard, B. Szymanski, J. Teresco, L. Ziantz, Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws, Journal of Parallel and Distributed Computing 47 (1997) 139–152.

[10] P. Moore, J. Flaherty, A local refinement finite element method for one-dimensional parabolic systems, SIAM Journal on Numerical Analysis 27 (1990)

1422–1444.

[11] P. Moore, J. Flaherty, Adaptive local overlapping grid methods for parabolic systems in two space dimensions, Journal of Computational Physics 98 (1992) 54–63.

[12] J.-F. Remacle, J. Flaherty, M. Shephard, An adaptive discontinuous galerkin technique with an orthogonal basis applied to compressible flow problems, submitted to SIAM Review .

[13] B. Cockburn, G. Karniadakis, C.-W. Shu (Eds.), Discontinuous Galerkin Methods, Vol. 11 of Lecture Notes in Computational Science and Engineering, Springer, Berlin, 2000.

[14] B. V. Leer, Flux vector splitting for the Euler equations, ICASE report, NASA Langley Research Center (1995).

[15] P. Woodward, P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, Journal of Computational Physics 54 (1984) 115–173.

[16] P. Colella, H. M. Glaz, Efficient solution algorithms for the Riemann problem for real gases, Journal of Computational Physics 59 (1985) 264–289.

[17] R. Biswas, K. Devine, J. Flaherty, Parallel adaptive finite element method for conservation laws, Applied Numerical Mathematics 14 (1984) 255–283.

[18] B. Cockburn, C. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for the conservation laws II: General framework, Mathematics of Computations 52 (1989) 411–435.

[19] J.-F. Remacle, J. Flaherty, M. Shephard, Parallel algorithm oriented mesh database, in: Tenth International Meshing Roundtable, Newport Beach, California, 2001, pp. 197–206.

[20] J. E. Flaherty, X.-R. Li, K. Pinchedez, J.-F. Remacle, M. S. Shephard, Muzzle brakes evaluation (2001).