Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 209 (2005) 105–138
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A subcell remapping method on staggered polygonal grids for arbitrary-Lagrangian–Eulerian methods

Raphaël Loubère, Mikhail J. Shashkov *

Los Alamos National Laboratory, T-7, MS B284 Los Alamos, NM, USA

Received 27 September 2004; received in revised form 3 March 2005; accepted 17 March 2005
Available online 4 May 2005

Abstract

We describe a new remapping algorithm for use in arbitrary Lagrangian–Eulerian (ALE) simulations. The new features of this remapper are designed to complement a staggered-mesh Lagrangian phase in which the cells may be general polygons (in two dimensions), and which uses subcell discretizations to control unphysical mesh distortion and hourglassing. Our new remapping algorithm consists of three stages. A *gathering stage*, in which we interpolate momentum, internal energy, and kinetic energy to the subcells in a conservative way. A *subcell remapping stage*, in which we conservatively remap mass, momentum, internal, and kinetic energy from the subcells of the Lagrangian mesh to the subcells of the new rezoned mesh. A *scattering stage*, in which we conservatively recover the primary variables: subcell density, nodal velocity, and cell-centered specific internal energy on the new rezoned mesh. We prove that our new remapping algorithm is conservative, reversible, and satisfies the DeBar consistency condition. We also demonstrate computationally that our new remapping method is robust and accurate for a series of test problems in one and two dimensions.

PACS: 65N05; 65N10; 65N15; 80A20

Keywords: Remapping; Conservative interpolation; ALE methods

1. Introduction and background

In numerical simulations of multidimensional fluid flow, the relationship between the motion of the computational grid and the motion of the fluid is an important issue. Two choices that are typically made

* Corresponding author. Tel.: +1 505 667 4400; fax: +1 505 665 5757.
E-mail addresses: loubere@lanl.gov (R. Loubère), misha@t7.lanl.gov, shashkov@lanl.gov (M.J. Shashkov).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.03.019

represent either a Lagrangian framework, in which the mesh moves with the local fluid velocity, or an Eulerian framework, in which the fluid flows through a grid fixed in space. More generally, however, the motion of the grid can be chosen arbitrarily. The philosophy of the arbitrary Lagrangian–Eulerian methodology (ALE; cf. [14,3,4,20,15,16,26]) is to exploit this degree of freedom to improve both the accuracy and the efficiency of the simulation. The main elements of most ALE algorithms are an explicit Lagrangian phase, a rezone phase in which a new grid is defined, and a remap phase in which the Lagrange solution is transferred to the new grid [20].

Most ALE codes use a grid of fixed connectivity that, in two spatial dimensions, is formed by quadrilaterals or by a mix of quadrilaterals and triangles, the latter being considered as degenerate quadrilaterals. Ultimately, we are interested in the development of ALE methods for meshes whose connectivity may change during the calculation. In such methods, the total number of cells remains fixed, but the number of edges bounding each cell may change with time, leading to the appearance of general polygonal cells. As a first step toward this goal, here we consider ALE methods on a mesh with fixed connectivity, but allow the mesh to contain general polygonal cells. Extending the ALE methodology to this more general mesh is valuable in itself as it simplifies the setup process for computational domains with complex geometrical shapes and helps to avoid artificial mesh imprinting due to the restrictions of a purely quadrilateral mesh, [6,7].

In the rest of this introductory section, we will present notation related to a general polygonal staggered mesh, will review algorithms for the Lagrangian phase and rezone phase as presented in [8,9,17,28,33,32], and finally will describe the main ideas of our new remap procedure, which is the main topic of this paper.

1.1. Polygonal mesh

We consider a two-dimensional computational domain Ω, assumed to be a general polygon. We assume we are given a mesh on Ω whose cells, $\{c\}$, cover the domain without gaps or overlaps. Each cell may be a general polygon, and is assigned an unique index that for simplicity will also be denoted by c. The set of vertices (nodes) of the polygons is denoted by $\{n\}$, where each node has an unique index n. Then each cell can be defined by an ordered set of vertices. We denote the set of vertices of a particular cell c by $N(c)$. Further, we denote the set of cells that share a particular vertex n by $C(n)$. Note that each vertex may be shared by an arbitrary number of cells. We will subdivide each cell into a set of quadrilaterals that we will term subcells. A pair of indexes c and n uniquely defines a quadrilateral, identified as subcell cn; this subcell is constructed by connecting the geometrical center of the cell c with the middle points of cell faces having the same node n as one end point and the node itself (see Fig. 1). Hence each cell can be divided uniquely into quadrilaterals (subcells or corners).

We denote the cell and subcell volumes (in 2D Cartesian geometry these are areas) by $V(c)$ and $V(cn)$, where by construction $V(c) = \sum_{n\in N(c)} V(cn)$. A nodal volume can be defined as the sum of the volumes of subcells shared by the node n, i.e., $V(n) = \sum_{c\in C(n)} V(cn)$.

1.2. Lagrangian phase

The equations of Lagrangian gas dynamics can be written as

$$\frac{1}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\mathbf{div}\,\mathbf{u}, \qquad \rho\frac{\mathrm{d}u}{\mathrm{d}t} = -\mathbf{grad}\,p, \qquad \rho\frac{\mathrm{d}\varepsilon}{\mathrm{d}t} = -p\ \mathbf{div}\,\mathbf{u}, \tag{1.1}$$

where ρ is the density, p is the pressure, ε is the specific internal energy, and $\mathbf{u} = (u, v)$ is the velocity. The pressure is linked to density and specific internal energy via an equation of state: $p = p(\rho, \varepsilon)$. This system of Eq. (1.1) is solved by the Lagrangian phase.

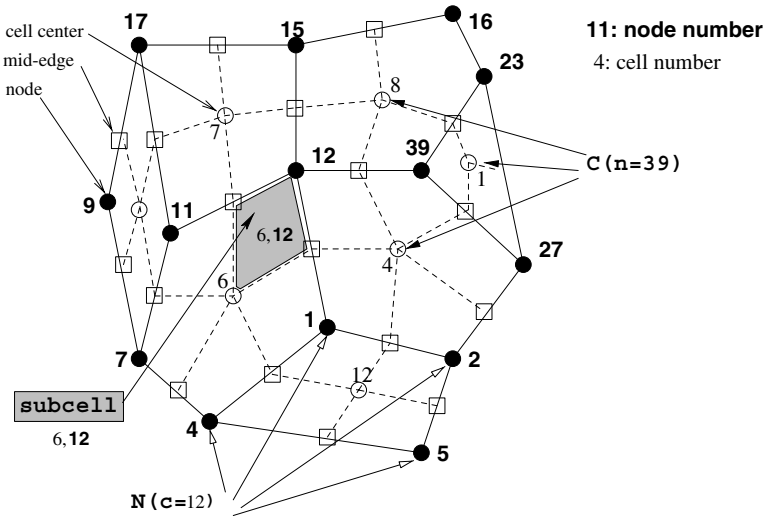

Fig. 1. Grid and notations. The ○ are the cell centers, the ● are the nodes (vertices), the □ are the mid-face (edge) points. The set of vertices for cell $c = 12$ is $N(c = 12) = \{\mathbf{5,2,1,4}\}$, and the set of cells sharing node, $n = 39$ is $C(n = \mathbf{39}) = \{1, 8, 4\}$. The gray subcell, $cn = 6, 12$ is the quadrilateral defined by connecting the geometrical center of the cell $c = 6$ with middle points of cell faces having the same node $n = \mathbf{12}$ as one end point and the node itself.

A discretization of the gas dynamic equations for the Lagrangian phase of the ALE method for a mesh consisting of general polygons is described in [8,9], based on the philosophy of compatible hydrodynamic discretization [12]. This discretization assumes a staggered grid, where the components of the velocity vector are defined at the nodes (vertices) of the cells, $\mathbf{u}(n) = (u(n), v(n))$, and where the thermodynamic variables density $\rho(c)$ and internal energy $\varepsilon(c)$ are defined at the cell centers. In addition to nodal and cell-centered quantities, this discretization employs as additional variables the densities of the subcells, $\rho(cn)$. The preservation of subcell mass during the Lagrangian phase of the calculation introduces new forces that prevent artificial grid distortion and hourglass patterns. This enhancement of the Lagrangian algorithm was shown to be effective both for quadrilateral meshes [10], as well as for polygonal meshes [8]. The Lagrangian phase including subcell forces, is conservative; i.e., discrete forms of mass, momentum, and total energy are conserved [12]. The use of subcell masses and corresponding densities places new requirements on the remap phase of an ALE method because these subcell densities have to be remapped in addition to the usual remapping of the primary variables—nodal velocities, cell-centered densities and internal energies.

We define the subcell mass in terms of the primary cell variables as follows:

$$m(cn) = \rho(cn)V(cn). \tag{1.2}$$

Then the mass of the cell and of the node are defined

$$m(c) = \sum_{n \in N(c)} m(cn), \quad m(n) = \sum_{c \in C(n)} m(cn). \tag{1.3}$$

All of these masses are employed in the Lagrangian phase of our ALE method. Since the subcell mass, $m(cn)$ is assumed to be Lagrangian and so does not change with time, it follows that:

$$\rho(cn) = m(cn)/V(cn), \tag{1.4}$$

which serves as a definition of the subcell density for a given subcell mass. The masses of the individual cells and nodes are also Lagrangian because they are sums of the masses of the associated subcells. The mass of

the cell is used in the equation for the internal energy, while the mass of the node is used in the momentum equation. Finally, by definition we have

$$\rho(c) = \frac{m(c)}{V(c)} = \frac{\sum_{n\in N(c)} m(cn)}{\sum_{n\in N(c)} V(cn)}.$$

The total mass, M, which is conserved in the Lagrangian phase is

$$M = \sum_{cn} m(cn) = \sum_{c} m(c) = \sum_{n} m(n). \tag{1.5}$$

On the staggered mesh, momentum is most naturally defined at the nodes

$$\mu(n) = m(n)u(n), \quad v(n) = m(n)v(n), \tag{1.6}$$

or equivalently

$$u(n) = \mu(n)/m(n), \quad v(n) = v(n)/m(n). \tag{1.7}$$

Note that as a result of the remap stage we will have new momenta and masses at the nodes, so to recover velocities we will use (1.7) as the definition of velocities for given momenta and nodal mass. The total momentum components, μ_u, μ_v, which are individually conserved in the Lagrangian phase, are

$$\mu_u = \sum_{n} m(n)u(n), \quad \mu_v = \sum_{n} m(n)v(n). \tag{1.8}$$

It will be useful to define a cell-centered momenta as

$$\mu(c) = \sum_{n\in N(c)} m(cn)u(n), \quad v(c) = \sum_{n\in N(c)} m(cn)v(n). \tag{1.9}$$

Using this definition and the definition of nodal mass, the total momentum components (μ_u, μ_v)—see Eq. (1.8)—can be expressed as

$$\mu_u = \sum_{c} \mu(c), \quad \mu_v = \sum_{c} v(c). \tag{1.10}$$

Kinetic energy is also most naturally defined at the nodes

$$K(n) = m(n)\frac{|\mathbf{u}(n)|^2}{2}. \tag{1.11}$$

The internal energy is naturally defined at the cells

$$\mathscr{E}(c) = m(c)\varepsilon(c). \tag{1.12}$$

In analogy to (1.7), Eq. (1.12) can be used after the remap phase to define $\varepsilon(c)$ given $\mathscr{E}(c)$ and $m(c)$

$$\varepsilon(c) = \mathscr{E}(c)/m(c). \tag{1.13}$$

The total energy, which is also conserved in the Lagrangian phase, is

$$E = \sum_{c} \mathscr{E}(c) + \sum_{n} K(n). \tag{1.14}$$

Later we will require the concept of a cell-centered kinetic energy, which we define as follows:

$$K(c) = \sum_{n\in N(c)} m(cn)\frac{|\mathbf{u}(n)|^2}{2}. \tag{1.15}$$

Using this definition and the definition of nodal mass, the total energy, E (see formula (1.14)), can be expressed as

$$E = \sum_c (\mathscr{E}(c) + K(c)). \tag{1.16}$$

By introducing total internal and kinetic energies as

$$\mathscr{E} = \sum_c \mathscr{E}(c), \quad K = \sum_c K(c), \tag{1.17}$$

we finally can express the total energy as

$$E = \mathscr{E} + K. \tag{1.18}$$

1.3. Rezone phase

In the rezone phase, we use the reference Jacobian matrix (RJM) strategy described in [17,28]. The RJM rezone algorithm is based on a nonlinear optimization procedure that requires a valid mesh as an initial guess, and so it may be necessary to untangle the mesh (see e.g., [33,32]) prior to rezoning. The RJM rezone strategy ensures the continuing geometric quality of the computational grid, while keeping the "rezoned" grid at each time step as close as possible to the Lagrangian grid. Sets of cells and nodes of rezoned mesh will be denoted by $\{\tilde{c}\}$ and $\{\tilde{n}\}$, respectively.

When the rezoned and Lagrangian grids are sufficiently close to each other, it is possible to use a local procedure on the remapping stage, meaning that mass, energy and momentum are exchanged only between neighboring cells. Local rezoning is conceptually simpler and computationally less expensive than global rezoning. For some of the tests presented in Section 7, we will use the ALE code in the Eulerian framework, so that the rezoned mesh will always coincide with the initial mesh (see e.g., [25]).

1.4. Summary of the new remapping algorithm

To guarantee conservation in the overall ALE simulation, the remapping phase must conservatively interpolate the Lagrange solution onto the rezoned grid. The main purpose of this paper is to describe a new algorithm for remapping on a general, polygonal, staggered grid, including treatment of the density defined in the subcells. Readers interested in the history of remapping methods on staggered meshes are referred to [4,5,26,25,19,21,1,13,23].

To the best of our knowledge, there is no existing remapping method that addresses all of our requirements—remapping on a general polygonal staggered mesh with subcell densities.

We have designed a new remapping strategy consisting of the three following stages:

- First: *Gathering stage*. We define momentum, internal energy, and kinetic energy in the subcells. Recall that the mass of subcell is already defined by (1.2). Mass, momentum, internal energy and kinetic energy in the subcells are defined in such a way that the corresponding total quantities (defined as the sums over subcells) are the same as those at the end of the Lagrangian phase, ensuring that the gathering stage is conservative.
- Second: *Subcell remapping stage*. We use the algorithm described in [18] to remap mass, momentum, internal, and kinetic energy from the subcells of the Lagrangian mesh to the subcells of the new rezoned mesh. This algorithm is linearity-preserving and computationally efficient. It consists of a piecewise linear reconstruction and an approximate integration based on the notion of swept regions. The algorithm does not require finding the intersections of the Lagrangian mesh with the rezoned mesh, which

contributes to its efficiency. The algorithm is conservative: total mass, momentum, internal and kinetic energy over subcells of the rezoned mesh are the same as mass, momentum, internal and kinetic energy over subcells of Lagrangian mesh. The total energy is also conserved, being the sum of (individually conserved) internal and kinetic energies. We suggest that remapping internal and kinetic energy separately is more accurate than remapping total energy, because we are not combining two quantities that can have very different magnitudes and behavior.

- Third: *Scattering stage*. We recover the primary variables—subcell density, nodal velocity, and cell-centered specific internal energy—on the new rezoned mesh.
  - Subcell density is recovered by using the remapped mass and volume of the subcell of the rezoned mesh in Eq. (1.4). The subcell masses and the corresponding densities are then adjusted using a conservative repair procedure [18,29,21] to enforce local bounds, which may be violated during the subcell remapping stage. This produces the final subcell density and the corresponding subcell mass that will be used in next time step. The new nodal masses and the cell-centered masses are defined using Eq. (1.2).
  - Next, we define the remapped nodal momenta using the remapped subcell momenta, in such a way that total momenta is conserved (see details in Sections 2 and 3). New velocity components are defined according to (1.7). Then nodal velocity is repaired, resulting in the final velocity that will be used to move the point during the Lagrangian phase in the next computational cycle.
  - To enforce the conservation of total energy, the discrepancy between the remapped kinetic energy in the cell and the kinetic energy that is computed from the remapped subcell masses and the final nodal velocities is contributed to the remapped internal energy in the cell. The new internal energy is recovered using (1.13). Finally, the internal energy and the corresponding specific internal energy are conservatively repaired.

The outline of the rest of this paper is as follows. In Section 2 we will give a precise statement of our goals for remapping on the staggered mesh and will list the desired properties of the remapping algorithm. In Section 3 we will define momentum, internal, and kinetic energy in the subcells of the Lagrangian mesh (gathering stage). The properties of the remapping of subcell quantities from the Lagrangian mesh to the rezoned mesh are briefly described in Section 4 (subcell remapping stage). The definition of the subcell density, nodal velocity and cell-centered specific internal energy on the rezoned mesh (scattering stage) is described in Section 5 and in Appendix A. In Section 6, we prove that our new remapping algorithm is conservative, reversible, and that the DeBar consistency condition for remapping of velocity [4] is satisfied. Numerical results that demonstrate the accuracy and convergence of the remapping algorithm are presented in Section 7. Finally, we conclude the paper in Section 8.

2. Statement of the remapping

As a result of the Lagrangian phase of a computational cycle, we have a mesh consisting of cells $\{c\}$, and nodes $\{n\}$. We will call this the Lagrangian or old mesh. We have values of density, $\rho(cn)$ in subcells, values of specific internal energy, $\varepsilon(c)$, in cells, and values of the components of velocity, $u(n)$, $v(n)$, at the nodes of the old mesh. As a result of the rezone phase, we have the rezoned or new mesh consisting of cells $\{\tilde{c}\}$, and nodes $\{\tilde{n}\}$. An example of old and new meshes is given in Fig. 2. The goal of the remapping phase is to find an accurate approximation to $\rho(\widetilde{cn})$, $\varepsilon(\tilde{c})$, $u(\tilde{n}), v(\tilde{n})$ on the new mesh. Using the primary variables we can define the total mass M, the momentum vector (μ_u, μ_v), the internal energy $\mathscr{E}$, the kinetic energy K, and the total energy E on the old mesh from Eqs. (1.5), (1.10), (1.17) and (1.18), respectively.

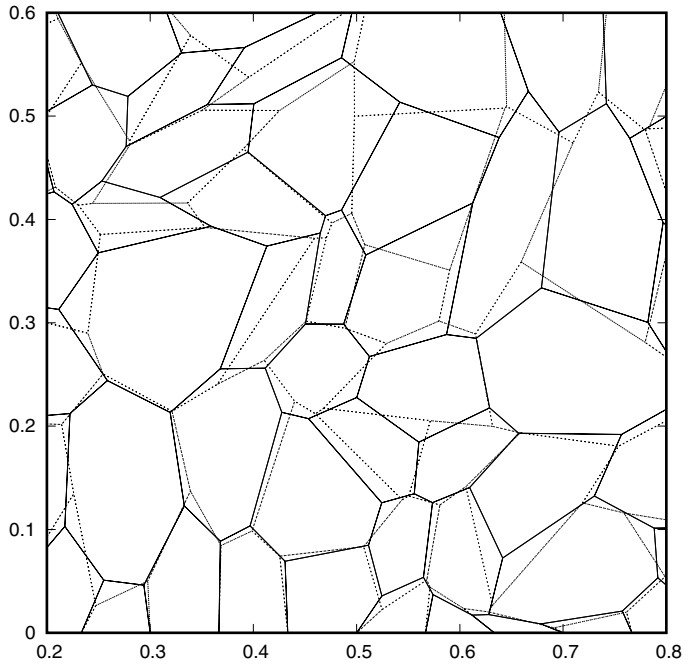

Fig. 2. Fragment of the Lagrangian (dotted lines) and the rezoned (solid lines) grids.

The remapping algorithm must satisfy the following requirements:

- *Conservation*. The total mass, momenta and energy of the new mesh must be the same as that of the old mesh

$$\tilde{M} = M,\ \tilde{\mu}_u = \mu_u, \quad \tilde{\mu}_v = \mu_v, \quad \tilde{E} = E.$$

  This property, combined with the same conservation properties of the Lagrangian phase, guarantees the conservation of the overall ALE method.
- *Bound-preservation*. The remapped density, velocity components and internal energy have to be contained within physically justified bounds, which are determined from the corresponding fields in the Lagrangian solution. For example, density and internal energy have to be positive. Moreover, because we assume that the new mesh is obtained from a small displacement of the old mesh, one can require that the new value lie between bounds determined by the values of its neighbors on the old mesh, [18].
- *Accuracy*. It is straightforward to define accuracy in the remap of density; we will require that the remap of density is linearity-preserving. That is, if the values on the old mesh are obtained from a global linear function, then the values on the new mesh have to coincide with the values of the same linear function on the new mesh. For the remap of velocity, there are several different notions related to accuracy. For example, one widely used test of consistency is the so-called DeBar condition (see for example [4]) which can be stated as follows: if a body has a uniform velocity and spatially varying density, then the remapping process should exactly reproduce a uniform velocity. For internal energy, the situation is more complicated. We will demonstrate the accuracy of our new algorithm through the practical expedient of well-chosen test problems.

- *Reversibility*. If the new and old meshes are identical, then the remapped primary variables should show no change. This property is closely related to the notion of being free of inversion error, see [4], where it is stated that if the new and old grids coincide, then the remapped velocity on new mesh should coincide with the velocity on the old mesh.

3. Gathering

In the *gathering stage*, we define mass (which is already known), momentum, internal energy, and kinetic energy in the subcells: $m(cn),\ \mu(cn),\ v(cn),\ \mathscr{E}(cn), K(cn)$ such that the corresponding total quantities maintain the same values as they have at the end of the Lagrangian phase

$$\begin{aligned} M^s &\stackrel{\text{def}}{=} \sum_{cn} m(cn) = M, \\ \mu_u^s &\stackrel{\text{def}}{=} \sum_{cn} \mu(cn) = \mu_u, \quad \mu_v^s \stackrel{\text{def}}{=} \sum_{cn} v(cn) = \mu_v, \\ \mathscr{E}^s &\stackrel{\text{def}}{=} \sum_{cn} \mathscr{E}(cn) = \mathscr{E}, \quad K^s \stackrel{\text{def}}{=} \sum_{cn} K(cn) = K. \end{aligned} \tag{3.1}$$

Here the superscript s emphasizes that the corresponding total quantities are defined by summation over subcells. Clearly, if we conserve the total kinetic and the total internal energy, then the total energy

$$E^s \stackrel{\text{def}}{=} \mathscr{E}^s + K^s$$

is also conserved, i.e.,

$$E^s = E. \tag{3.2}$$

As follows from Eqs. (1.10) and (1.17), the total momenta, kinetic energy and internal energy can be expressed by summation of the corresponding cell-centered quantities given by (1.9), (1.12) and (1.15). This suggests the following *design principle*: construct the subcell quantities in such a way that conservation is ensured on cell-by-cell basis. For example, the momentum components $\mu(cn)$ satisfy the following equation:

$$\sum_{n \in N(c)} \mu(cn) = \mu(c), \tag{3.3}$$

and similarly for the other quantities. Thus all requirements of conservation listed in (3.1) will be satisfied.

We note that there is no unique solution for such a construction, because there is one constraint whereas the number of unknowns is equal to the number of subcells in the given cell. For example, Eq. (1.9) suggests that the simplest way to satisfy (3.3) is to define $\mu(cn)$

$$\mu(cn) = m(cn)u(n). \tag{3.4}$$

However, this will not be accurate enough in general; e.g., in the case of a constant density, it will be exact only for a constant velocity field. In the next section we will describe a more accurate algorithm, which in the case of a constant density will be exact for any linear velocity field.

3.1. Definition of subcell momenta

We will present the procedure for defining the x-component of momentum $\mu(cn)$, noting that the definition of the other component $v(cn)$ is similar. Also, for brevity, we will refer to the velocity component u simply as velocity. The total number of nodes (and hence of subcells) of the cell c is denoted by $|N(c)|$.

We will use N instead of $|N(c)|$ as we will never need to use a local indexing for two different cells at the same time. The nodes of the cell under consideration are enumerated from 1 to N in counter-clockwise order.

The subcell momenta will be defined as

$$\mu(cn) = m(cn)u(cn),$$

where $u(cn)$—yet to be defined—has the meaning of a subcell velocity, see Fig. 3 for illustration.

The subcell velocities $u(cn)$ must be defined such that the total momentum of the cell, defined in (1.9), is conserved, i.e.,

$$\sum_{n\in N(c)} m(cn)u(cn) = \sum_{n\in N(c)} m(cn)u(n) = \mu(c). \tag{3.5}$$

As previously mentioned, a simple solution that satisfies (3.5) is to set $u(cn) = u(n)$. However because $u(cn)$ has the meaning of a velocity in subcell, setting it to the velocity in the corresponding node will not be accurate. Instead we seek a more accurate estimate of $u(cn)$ in the form

$$u(cn) = \frac{u(c) + u(n) + u_{n,n^+} + u_{n^-,n}}{4}. \tag{3.6}$$

Here $u(c)$ is not yet defined, and

$$u_{n^-,n} = \frac{1}{2}(u(n^-) + u(n)), \quad u_{n,n^+} = \frac{1}{2}(u(n) + u(n^+)) \tag{3.7}$$

are the approximations of the velocities at the mid-edge points based on the velocities of the corresponding nodes in the cell c (n^- and n^+ are the previous/next nodes with respect to n in the list of vertices of cell c, see Fig. 4). The velocity $u(c)$ has the meaning of a velocity at the cell center. Eq. (3.6) states that the velocity in the center of subcell is a simple average of velocities in the corners of the subcell. Three of these velocities, $u(n)$, u_{n,n^+}, $u_{n^-,n}$ are known quantities and one, $u(c)$, will be defined by conserving the momentum of the cell.

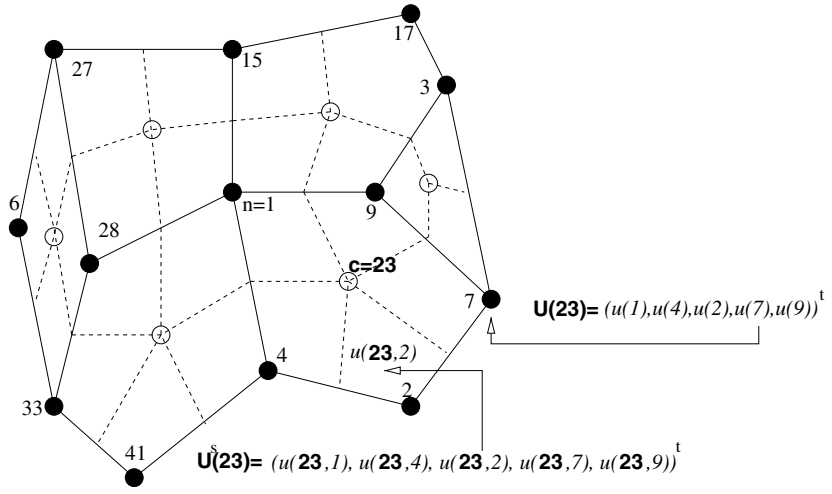

Fig. 3. Vectors $\mathbf{U}(\mathbf{c})$ and $\mathbf{U}^s(c)$ for a given mesh. The numbers are the global indexes of the nodes. $\mathbf{U}(\mathbf{23}) = (u(1), u(4), u(2), u(7), u(9))^t$ is the vector of nodal velocities, for example $u(7)$ is the velocity of node number 7. $\mathbf{U}^s(\mathbf{23}) = (u(\mathbf{23},1), u(\mathbf{23},4), u(\mathbf{23},2), u(\mathbf{23},7), u(\mathbf{23},9))^t$ is the vector of subcell velocities, for example $u(\mathbf{23},2)$ is the velocity of the subcell uniquely defined by cell **23** and node number 2. The local neighbor nodes of $n = 1$ in cell **23** are $n^- = 9$ and $n^+ = 4$, the mid-edge points being called n, $n^- = 1,9$ and $n,n^+ = 1,4$.

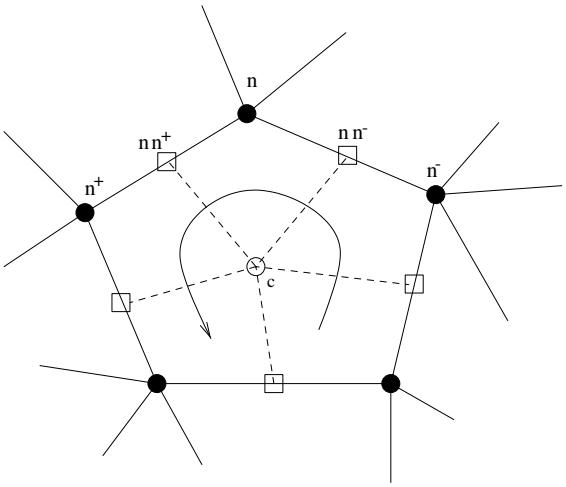

Fig. 4. Neighbor nodes of node n along the boundary of the cell c (using counterclockwise ordering) are n^- and n^+. The mid-edge points are denoted as nn^- and nn^+.

From (3.5) and (3.6) we obtain the following equation for $u(c)$:

$$\sum_{n\in N(c)} m(cn)\left(\frac{u(c)+u(n)+u_{n,n^+}+u_{n^-,n}}{4}\right)=\sum_{n\in N(c)} m(cn)u(n),$$

which is one equation with one unknown. Using the definitions of u_{n,n^+} and $u_{n^-,n}$ from (3.7), we derive a formula for $u(c)$:

$$u(c)=\frac{1}{m(c)}\sum_{n\in N(c)} m(cn)\left(2u(n)-\frac{1}{2}u(n^+)-\frac{1}{2}u(n^-)\right). \tag{3.8}$$

An equivalent form of this definition is

$$u(c)=\frac{1}{m(c)}\sum_{n\in N(c)} m(cn)u(n)-\frac{1}{m(c)}\sum_{n\in N(c)} m(cn)\frac{u(n^+)-2u(n)+u(n^-)}{2}. \tag{3.9}$$

From this equation it is clear that if $u(n)=\mathscr{C}$ then $u(c)=\mathscr{C}$; therefore in the case of constant velocity with any arbitrary distribution of masses, our definition is exact. It is also easy to verify that if all the subcell masses are the same, then

$$u(c)=\frac{1}{|N(c)|}\sum_{n\in N(c)} u(n),$$

meaning that this formula is exact for a linear velocity field.

Substituting (3.8) into (3.6) yields

$$\begin{aligned}u(cn)&=\frac{1}{4}\left(2u(n)+\frac{u(n^+)}{2}+\frac{u(n^-)}{2}\right)+\left[\frac{1}{4}\frac{1}{m(c)}\sum_{k\in N(c)} m(ck)\left(2u(k)-\frac{1}{2}u(k^+)-\frac{1}{2}u(k^-)\right)\right]\\&=\frac{1}{4}\left(2u(n)+\frac{u(n^+)}{2}+\frac{u(n^-)}{2}\right)+\sum_{k\in N(c)}\frac{m(ck)}{8m(c)}(4u(k)-u(k^+)-u(k^-)).\end{aligned} \tag{3.10}$$

The last term in (3.10) can be transformed as follows. First we split this term in three separate sums

$$\sum_{k\in N(c)}\frac{m(ck)}{8m(c)}(4u(k)-u(k^+)-u(k^-))=\sum_{k\in N(c)}\frac{m(ck)}{8m(c)}4u(k)-\sum_{k\in N(c)}\frac{m(ck)}{8m(c)}u(k^+)-\sum_{k\in N(c)}\frac{m(ck)}{8m(c)}u(k^-).$$

Now by shifting the index in second and third sum and combining the resulting expressions, we get

$$\sum_{k\in N(c)}\frac{m(ck)}{8m(c)}4u(k)-\sum_{k\in N(c)}\frac{m(ck^-)}{8m(c)}u(k)-\sum_{k\in N(c)}\frac{m(ck^+)}{8m(c)}u(k)=\sum_{k\in N(c)}u(k)\left(\frac{-m(ck^-)+4m(ck)-m(ck^+)}{8m(c)}\right). \tag{3.11}$$

Finally, using (3.11) and (3.10) we get

$$u(cn)=\frac{1}{4}\left(2u(n)+\frac{u(n^+)}{2}+\frac{u(n^-)}{2}\right)+\left[\sum_{k\in N(c)}u(k)\left(\frac{-m(ck^-)+4m(ck)-m(ck^+)}{8m(c)}\right)\right]. \tag{3.12}$$

(In Appendix A we present a 1D analog of the derivation of this formula for the subcell velocity.)

Eq. (3.12) defines the subcell velocities $u(cn)$ in terms of the nodal velocities $u(n)$. As a result of Eq. (3.8), and the previously discussed properties of $u(c)$, Eq. (3.12) is exact in the following cases:

- A constant velocity and an arbitrary mass distribution (this property will be used later to prove the DeBar condition).
- An equal subcell mass distribution and a linear velocity.

Let us rewrite (3.12) for all $n\in N(c)$ in matrix form. To do this, we represent the velocities $u(n)$ of the vertices of one particular cell c as the elements of a vector $\mathbf{U}(c)$

$$\mathbf{U}(c)=\{u(n),\quad n\in N(c)\}^t. \tag{3.13}$$

Similarly, we represent the subcell velocities as the elements of a vector $\mathbf{U}^s(c)$

$$\mathbf{U}^s(c)=\{u(cn),\quad n\in N(c)\}^t.$$

A graphical illustration of these definitions of $\mathbf{U}(c)$, $\mathbf{U}^s(c)$ is shown in Fig. 3. As a matter of notation, we will use bold letters to denote column vectors while matrices will be denoted with a capital bold over-lined letter as $\bar{\mathbf{I}}_c$.

Now (3.12) can be rewritten in matrix form as follows:

$$\mathbf{U}^s(c)=\bar{\mathbf{I}}_c\mathbf{U}(c), \tag{3.14}$$

where the matrix $\bar{\mathbf{I}}_c$ is

$$\bar{\mathbf{I}}_c=\frac{1}{4}\cdot\begin{pmatrix}2&\frac{1}{2}&0&0&\cdots&0&\frac{1}{2}\\ \frac{1}{2}&2&\frac{1}{2}&0&\cdots&0&0\\ 0&\frac{1}{2}&2&\frac{1}{2}&\cdots&0&0\\ \vdots&\vdots&\vdots&\vdots&\ddots&\vdots&\vdots\\ \frac{1}{2}&0&0&0&\cdots&\frac{1}{2}&2\end{pmatrix}+\begin{pmatrix}Q_{N,1,2}&Q_{1,2,3}&Q_{2,3,4}&\cdots&Q_{N-1,N,1}\\ Q_{N,1,2}&Q_{1,2,3}&Q_{2,3,4}&\cdots&Q_{N-1,N,1}\\ Q_{N,1,2}&Q_{1,2,3}&Q_{2,3,4}&\cdots&Q_{N-1,N,1}\\ \vdots&\vdots&\vdots&\ddots&\vdots\\ Q_{N,1,2}&Q_{1,2,3}&Q_{2,3,4}&\cdots&Q_{N-1,N,1}\end{pmatrix}. \tag{3.15}$$

Each element of the second matrix in (3.15) is formed from three consecutive subcell masses, and in terms of the global indexing

$$Q_{n^-,n,n^+}=\frac{-m(cn^-)+4m(cn)-m(cn^+)}{8m(c)}.$$

In Appendix A we prove by construction that the matrix $\bar{\mathbf{I}}_c$ is always invertible, meaning that Eq. (3.14) prescribes a one-to-one correspondence between the nodal and the subcell velocities in a given cell. As mentioned earlier, the fact that Eq. (3.12) is exact when $u(n) = \mathscr{C}$ implies the following relations for the matrix $\bar{\mathbf{I}}_c$:

$$\bar{\mathbf{I}}_c\mathbf{C} = \mathbf{C}, \quad (\bar{\mathbf{I}}_c)^{-1}\mathbf{C} = \mathbf{C}, \tag{3.16}$$

where $\mathbf{C}$ is constant vector of length N, each of whose components equals $\mathscr{C}$. Finally, by construction we ensure that momentum is conserved in each cell, and therefore is conserved for the entire domain.

3.2. Definition of subcell kinetic energy

The subcell specific kinetic energy is denoted $k(cn)$, and we will require that the total kinetic energy in the cell c is conserved

$$\sum_{n\in N(c)} m(cn)k(cn) = \sum_{n\in N(c)} m(cn)\frac{|\mathbf{u}(n)|^2}{2} = K(c). \tag{3.17}$$

If we represent the specific kinetic energies at the vertices and in the subcells of cell c as components of vectors

$$\mathbf{k}(c) = \left\{ k(n) = \frac{|\mathbf{u}(n)|^2}{2}, n \in N(c) \right\}^t, \quad \mathbf{k}^s(c) = \{k(cn),\ n \in N(c)\}^t, \tag{3.18}$$

then by definition

$$\mathbf{k}^{\mathbf{s}}(c) \overset{\text{def}}{=} \bar{\mathbf{I}}_c\mathbf{k}(c). \tag{3.19}$$

and the subcell kinetic energy is finally given by

$$K(cn) = m(cn)k(cn). \tag{3.20}$$

By construction the kinetic energy in each cell is preserved, ensuring that the kinetic energy of the entire domain is preserved as well. We emphasize that

$$k(cn) \neq \frac{|\mathbf{u}(cn)|^2}{2},$$

because $k(cn)$ and $\mathbf{u}(cn)$ are defined independently. Moreover we cannot set $k(cn) = \frac{|\mathbf{u}(cn)|^2}{2}$ because this definition would not conserve the kinetic energy in the cell.

3.3. Definition of subcell internal energy

The specific internal energy ε is a cell-centered quantity. Thus the construction we developed for the subcell velocity cannot be applied. Instead, we will define the subcell internal energy in the following two steps:

(1) We will prescribe a linear reconstruction of the internal energy per unit volume in the cell c, which is denoted $(\rho\varepsilon)_c(x, y)$. This reconstruction:
- must be conservative in the cell, that is,

$$\int_c (\rho\varepsilon)_c(x, y)\,\mathrm{d}x\,\mathrm{d}y = \mathscr{E}(c) = \rho(c)\varepsilon(c)V(c) = m(c)\varepsilon(c). \tag{3.21}$$

- must be exact when $\rho(x,y)\varepsilon(x,y)$ is a linear function.

(2) We will compute the subcell internal energy by integrating $(\rho\varepsilon)_c(x,y)$ over the corresponding subcell.

In each cell c, the function $(\rho\varepsilon)_c(x,y)$ has the form

$$(\rho\varepsilon)_c(x,y) = \rho(c)\varepsilon(c) + \delta_x^c(x - x_c) + \delta_y^c(y - y_c), \tag{3.22}$$

with (x_c, y_c) being the centroid of the cell

$$x_c = \frac{1}{V(c)}\int_c x\,\mathrm{d}x\,\mathrm{d}y, \quad y_c = \frac{1}{V(c)}\int_c y\,\mathrm{d}x\,\mathrm{d}y.$$

We define the slopes δ_x^c, δ_y^c, by the Barth–Jespersen (BJ) algorithm [2]. The BJ algorithm is an algorithm for piecewise linear reconstruction of a function $f(x,y)$ given by its means $\bar{f}_c = \frac{1}{V(c)}\int_c f(x,y)\,\mathrm{d}x\,\mathrm{d}y$ over mesh cells. In cell c, function $f(x,y)$ is represented by the linear function $f_c(x,y)$. The BJ algorithm has the following properties:

- The mean of $f_c(x,y)$ over cell c is equal to the given mean value $\bar{f}_c$, that is
$$\frac{1}{V(c)}\int_c f_c(x,y)\,\mathrm{d}x\,\mathrm{d}y = \bar{f}_c.$$
- It is exact if $f(x,y)$ is a global linear function, $f(x,y) = a + bx + cy$.
- In each cell c, the linear function $f_c(x,y)$ is constructed in a such a way that its values at the cell vertices are within the bounds defined by the maximum and the minimum of the mean values over the set $C(c)$, consisting of cell c itself and its nearest neighbors. That is,
$$\min_{k\in C(c)} \bar{f}_k \leqslant f_c(x_n, y_n) \leqslant \max_{k\in C(c)} \bar{f}_k, \quad n \in N(c).$$

Details of the BJ algorithm can be found in [2] and in Appendix A of [22].

It is easy to verify that this reconstruction (3.22) is conservative because

$$\begin{aligned}\int_c (\rho\varepsilon)(x,y)\,\mathrm{d}x\,\mathrm{d}y &= \int_c \Big(\rho(c)\varepsilon(c) + \delta_x^c(x - x_c) + \delta_y^c(y - y_c)\Big)\,\mathrm{d}x\,\mathrm{d}y \\ &= V(c)\rho(c)\varepsilon(c) + \delta_x^c\left\{\underbrace{\int_c x\,\mathrm{d}x\,\mathrm{d}y - V(c)x_c}_{=0}\right\} + \delta_y^c\left\{\underbrace{\int_c y\,\mathrm{d}x\,\mathrm{d}y - V(c)y_c}_{=0}\right\} = m(c)\varepsilon(c) \\ &= \mathscr{E}(c),\end{aligned}$$

where the expressions in curly brackets are zero because of the definition of the centroid.

The subcell internal energy $\mathscr{E}(cn)$ is defined as the integral of $(\rho\varepsilon)_c(x,y)$ over the subcell cn (step two of the algorithm)

$$\mathscr{E}(cn) = \int_{cn} (\rho\varepsilon)_c(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{3.23}$$

The internal energy over each cell is conserved because

$$\mathscr{E}(c) = \int_c (\rho\varepsilon)_c(x,y)\,\mathrm{d}x\,\mathrm{d}y = \sum_{n\in N(c)}\left(\int_{cn}(\rho\varepsilon)_c(x,y)\,\mathrm{d}x\,\mathrm{d}y\right) = \sum_{n\in N(c)}\mathscr{E}(cn),$$

and in consequence, the internal energy is conserved over the entire domain.

4. Subcell remapping

For the *subcell remapping stage*, we employ the algorithm described in [18] to remap mass, momentum, internal, and kinetic energy from the subcells of the Lagrangian mesh to the subcells of the new rezoned mesh. This algorithm produces values for the mass, momentum, internal energy, and kinetic energy in the each of the subcells of the new mesh: $m(\widetilde{cn}), \mu(\widetilde{cn}), v(\widetilde{cn}), \mathscr{E}(\widetilde{cn}), K(\widetilde{cn})$. It is conservative, i.e.,

$$\begin{aligned}
&\tilde{M}^s \overset{\text{def}}{=} \sum_{\widetilde{cn}} m(\widetilde{cn}) = M^s,\\
&\tilde{\mu}_u^s \overset{\text{def}}{=} \sum_{\widetilde{cn}} \mu(\widetilde{cn}) = \mu_u^s, \quad \tilde{\mu}_v^s \overset{\text{def}}{=} \sum_{\widetilde{cn}} v(\widetilde{cn}) = \mu_v^s,\\
&\tilde{\mathscr{E}}^s \overset{\text{def}}{=} \sum_{\tilde{cn}} \mathscr{E}(\widetilde{cn}) = \tilde{\mathscr{E}}^s, \quad \tilde{K}^s \overset{\text{def}}{=} \sum_{\tilde{cn}} K(\tilde{cn}) = \tilde{K}^s,
\end{aligned} \tag{4.1}$$

and linearity-preserving. Clearly, if the total kinetic and the total internal energy are conserved, then the total energy

$$\widetilde{E}^s \overset{\text{def}}{=} \tilde{\mathscr{E}}^s + \tilde{K}^s$$

is also conserved

$$\tilde{E}^s = E^s. \tag{4.2}$$

For future analysis we note that when the new mesh coincides with the old mesh, then the subcell remapping process does not change the subcell quantities. We want to emphasize that in this stage one could use any other accurate conservative remapping algorithm for cell-centered (cells being the subcells in this context) quantities.

5. Scattering

The third element of our algorithm is the *scattering stage*, in which we recover the primary variables—i.e., subcell density, $\rho(\widetilde{cn})$, nodal velocity, $u(\tilde{n}), v(\tilde{n})$, and cell-centered specific internal energy $\varepsilon(\tilde{c})$ – on the new mesh. At the beginning of the scattering stage, we have the following subcell quantities on the new mesh: mass $m(\widetilde{cn})$, momenta $\mu(\widetilde{cn}), v(\widetilde{cn})$, internal energy $\mathscr{E}(\widetilde{cn})$ and kinetic energy $K(\widetilde{cn})$.

The scattering stage has to maintain conservation, meaning that the primary variables on new mesh must satisfy the following conditions:

$$\tilde{M} = \sum_{\widetilde{cn}} m(\widetilde{cn}) = \sum_{\widetilde{cn}} \rho(\widetilde{cn}) V(\widetilde{cn}) = \tilde{M}^s, \tag{5.1}$$

$$\tilde{\mu}_u = \sum_{\tilde{n}} m(\tilde{n}) u(\tilde{n}) = \tilde{\mu}_u^s, \quad \tilde{\mu}_v = \sum_{\tilde{n}} m(\tilde{n}) v(\tilde{n}) = \tilde{\mu}_v^s, \tag{5.2}$$

$$\tilde{E} = \sum_{\tilde{c}} (\mathscr{E}(\tilde{c}) + K(\tilde{c})) = \sum_{\tilde{c}} \left[m(\tilde{c})\varepsilon(\tilde{c}) + \sum_{\tilde{n}\in N(\tilde{c})} m(\widetilde{cn}) \frac{|\mathbf{u}(\tilde{n})|^2}{2} \right] = \tilde{\mathscr{E}}^s + \tilde{K}^s = \tilde{E}^s, \tag{5.3}$$

where

$$m(\tilde{c}) = \sum_{\tilde{n}\in N(\tilde{c})} m(\widetilde{cn}), \quad m(\tilde{n}) = \sum_{\tilde{c}\in C(\tilde{n})} m(\widetilde{cn}). \tag{5.4}$$

5.1. Definition of subcell density

The subcell density is recovered using Eq. (1.4),

$$\rho(\widetilde{cn}) = m(\widetilde{cn})/V(\widetilde{cn}).$$

The subcell masses and densities are then corrected using a conservative repair procedure [18] to reinforce local bounds that may have been violated during the subcell remapping stage. Because we assume that the rezoned grid is close to the Lagrangian grid, we choose the bounds for $\rho(\widetilde{cn})$ as the minimal and maximal values of the subcell densities in the neighboring old subcells (i.e., before remapping). We will continue to use the same notation $\rho(\widetilde{cn}), m(\widetilde{cn})$ for the repaired quantities, and will employ the same convention for other remapped and repaired quantities later in the paper.

5.2. Definition of nodal velocity

First, we define the new subcell velocity from subcell momenta and masses

$$u(\widetilde{cn}) \stackrel{\text{def}}{=} \frac{\mu(\widetilde{cn})}{m(\widetilde{cn})}. \tag{5.5}$$

Next we define the nodal velocities, $u^{\tilde{c}}(\tilde{n})$, for $\tilde{n} \in N(\tilde{c})$ with respect to cell $\tilde{c}$, by inverting Eq. (3.14), applied to the new mesh

$$\mathbf{U}(\tilde{c}) = (\bar{\mathbf{I}}_{\tilde{c}})^{-1}\mathbf{U}^{\mathbf{s}}(\tilde{c}), \tag{5.6}$$

with the formal vector notation

$$\mathbf{U}(\tilde{c}) = \left\{u^{\tilde{c}}(\tilde{n}),\ \tilde{n} \in N(\tilde{c})\right\}^t, \quad \mathbf{U}^{\mathbf{s}}(\tilde{c}) = \left\{u(\widetilde{cn}),\ \tilde{n} \in N(\tilde{c})\right\}^t.$$

We have introduced a new notation, $u^{\tilde{c}}(\tilde{n})$, because in general, Eq. (5.6) will give different results for the same node $\tilde{n}$ for different cells $\tilde{c}$. Note that the matrix $\bar{\mathbf{I}}_{\tilde{c}}$ is constructed using the final subcell masses of the new mesh.

Finally, a unique nodal velocity at the node of the new cell can be defined

$$u(\tilde{n}) = \frac{1}{m(\tilde{n})} \sum_{\tilde{c} \in C(\tilde{n})} m(\widetilde{cn}) u^{\tilde{c}}(\tilde{n}). \tag{5.7}$$

It is easy to show that momentum is conserved, i.e., $\tilde{\mu}_u = \tilde{\mu}_u^s$. In fact,

$$\tilde{\mu}_u \stackrel{\text{def}}{=} \sum_{\tilde{n}} m(\tilde{n}) u(\tilde{n}), \tag{5.8}$$

and the definition of $u^{\tilde{c}}(\tilde{n})$ in (5.7), gives

$$\sum_{\tilde{n}} m(\tilde{n}) u(\tilde{n}) = \sum_{\tilde{n}} \sum_{\tilde{c} \in C(\tilde{n})} m(\widetilde{cn}) u^{\tilde{c}}(\tilde{n}). \tag{5.9}$$

By changing the order of summation in right-hand-side of the previous equation we derive

$$\sum_{\tilde{n}} \sum_{\tilde{c} \in C(\tilde{n})} m(\widetilde{cn}) u^{\tilde{c}}(\tilde{n}) = \sum_{\tilde{c}} \sum_{\tilde{n} \in N(\tilde{c})} m(\widetilde{cn}) u^{\tilde{c}}(\tilde{n}). \tag{5.10}$$

From the definition of $u^{\tilde{c}}(n)$ we have

$$\sum_{\tilde{n} \in N(\tilde{c})} u^{\tilde{c}}(\tilde{n}) m(\widetilde{cn}) = \sum_{\tilde{n} \in N(\tilde{c})} \mu(\widetilde{cn}) = \tilde{\mu}_u^s. \tag{5.11}$$

From (5.8)–(5.11) we conclude that momentum is conserved

$$\tilde{\mu}_u = \tilde{\mu}_u^s. \tag{5.12}$$

In the final step, velocity is repaired with respect to bounds chosen as the maximal and minimal values of $u(n)$ (i.e., values before remapping) over the following stencil (see Fig. 5):

$$n \in \cup_{c \in C(n)} \{k \in N(c)\}.$$

After the repair stage, we obtain the final velocity at the nodes of the new mesh. This velocity will be used in the Lagrangian phase in the next time step. The definition of the new nodal velocity by Eq. (5.7) introduces dissipation, that is, the kinetic energy decreases. The repair process by itself conserves the total momenta, but also can change the kinetic energy. The overall change in kinetic energy will be accounted for in the definition of cell-centered specific internal energy. The final kinetic energy in the new cell is given by

$$K(\tilde{c}) = \sum_{\tilde{n} \in N(\tilde{c})} m(\widetilde{cn}) \frac{|\mathbf{u}(\tilde{n})|^2}{2}. \tag{5.13}$$

5.3. Definition of cell-centered specific internal energy

The final specific internal energy has to be defined to ensure the conservation of total energy. At this stage of the scattering, we know the following quantities for each cell:

- the final kinetic energy $K(\tilde{c})$ in the cell evaluated from Eq. (5.13), in which the final velocities and the final subcell masses are used;
- the remapped subcell internal energy $\mathscr{E}(\widetilde{cn})$, and the remapped subcell kinetic energy, $K(\widetilde{cn})$.

By definition, the total energy in the cell is

$$E(\tilde{c}) = \mathscr{E}(\tilde{c}) + K(\tilde{c}), \tag{5.14}$$

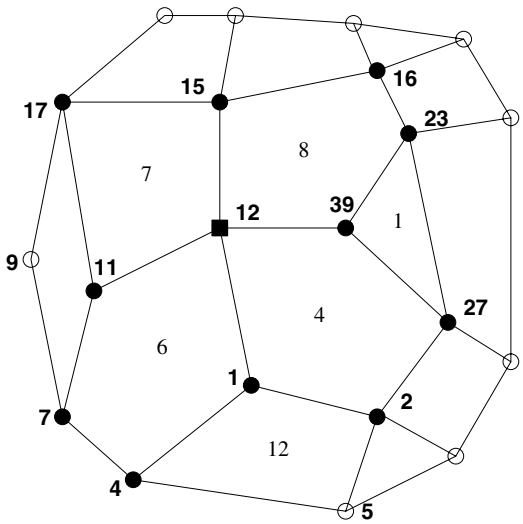

Fig. 5. The stencil for velocity repair. The stencil for node **12** (marked by solid square) consists of the union of the vertices of cells 6, 4, 8, 7, that is, **1, 2, 4, 7, 11, 12, 15, 16, 17, 23, 27, 39**, which are marked by solid circles.

where $\mathscr{E}(\tilde{c})$ is still unknown. If we define

$$E(\tilde{c}) = \sum_{\tilde{n}\in N(\tilde{c})} (\mathscr{E}(\widetilde{cn}) + K(\widetilde{cn})), \tag{5.15}$$

then the conservation of total energy is guaranteed because $\mathscr{E}(\widetilde{cn})$ and $K(\widetilde{cn})$ are obtained as a result of a conservative subcell remapping.

From Eqs. (5.15) and (5.14), we conclude that to conserve total energy, the new internal energy in the cell must be defined as follows:

$$\mathscr{E}(\tilde{c}) = \sum_{\tilde{n}\in N(\tilde{c})} \mathscr{E}(\widetilde{cn}) + \left[\left(\sum_{\tilde{n}\in N(\tilde{c})} K(\widetilde{cn})\right) - K(\tilde{c})\right]. \tag{5.16}$$

The term in the square brackets can be interpreted as the distribution of the change in kinetic energy due to the processes of defining and repairing the nodal velocities. The new specific internal energy is defined by analogy with (1.13) as

$$\varepsilon(\tilde{c}) = \mathscr{E}(\tilde{c})/m(\tilde{c}), \tag{5.17}$$

and in the final step, the specific internal energy is conservatively repaired.

6. Properties of the algorithm

6.1. Conservation

As we have proved in previous sections, mass, momentum, and total energy are all conserved at each stage: gathering, subcell remapping, and scattering. Therefore,

$$\begin{aligned}
&\widetilde{M} = \widetilde{M}^s = M^s = M,\\
&\tilde{\mu}_u = \tilde{\mu}_u^s = \mu_u^s = \mu_u,\\
&\widetilde{\mu}_v = \widetilde{\mu}_v^{\,s} = \mu_v^s = \mu_v,\\
&\tilde{E} = \tilde{E}^s = E^s = E.
\end{aligned}$$

That is, mass, momenta and total energy are conserved by the overall process.

6.2. Reversibility

Reversibility of the remapping means that if the new and old meshes are identical, then the primary variables will not be changed. Reversibility is a very important property that is related to the continuous dependence of the change of primary variables between the old and the new meshes. As mentioned in Section 4, there is no change in the subcell quantities during the subcell remapping stage if the new and old meshes are identical, i.e.,

$$m(\widetilde{cn}) = m(cn), \tag{6.1}$$

$$\mu(\widetilde{cn}) = \mu(cn), \quad v(\widetilde{cn}) = v(cn), \tag{6.2}$$

$$\mathscr{E}(\widetilde{cn}) = \mathscr{E}(cn), \quad K(\tilde{cn}) = K(cn). \tag{6.3}$$

From (6.1) we can immediately conclude that the subcell density has not been changed

$$\rho(\widetilde{cn}) = \frac{m(\widetilde{cn})}{V(\widetilde{cn})} = \frac{m(cn)}{V(cn)} = \rho(cn).$$

Combining (6.1) and (6.2) with the definition of $u(\widetilde{cn})$, we see that $u(\widetilde{cn}) = u(cn)$, and therefore

$$\mathbf{U}^s(\tilde{c}) = \mathbf{U}^s(c). \tag{6.4}$$

Also, (6.1), implies the matrix equality

$$\bar{\mathbf{I}}_{\tilde{c}} = \bar{\mathbf{I}}_c. \tag{6.5}$$

We recall that by definition

$$\mathbf{U}^s(c) = \bar{\mathbf{I}}_c \mathbf{U}(c). \tag{6.6}$$

Thus, combining (5.6), (6.5), (6.4) and (6.6) we derive

$$\mathbf{U}(\tilde{c}) = (\bar{\mathbf{I}}_{\tilde{c}})^{-1}\mathbf{U}^s(\tilde{c}) = (\bar{\mathbf{I}}_c)^{-1}\mathbf{U}^s(c) = (\bar{\mathbf{I}}_c)^{-1} \cdot (\bar{\mathbf{I}}_c)\mathbf{U}(c) = \mathbf{U}(c). \tag{6.7}$$

Eq. (6.7) means that

$$u^{\tilde{c}}(\tilde{n}) = u(n), \tag{6.8}$$

demonstrating that the old and new nodal velocities in node $\tilde{n}$ (from the point of view of all cells sharing this node) are the same. Now combining (5.7), (6.1) and (6.8), with the definition of $m(n)$ in (1.2), we derive

$$u(\tilde{n}) = \frac{1}{m(\tilde{n})} \sum_{\tilde{c} \in C(\tilde{n})} m(\widetilde{cn}) u^{\tilde{c}}(\tilde{n}) = \frac{1}{m(n)} \sum_{c \in C(n)} m(cn) u(n) = u(n) \left(\frac{1}{m(n)} \sum_{c \in C(n)} m(cn) \right) = u(n), \tag{6.9}$$

demonstrating that the nodal velocity stays the same

$$u(\tilde{n}) = u(n).$$

We next prove that $\varepsilon(\tilde{c}) = \varepsilon(c)$. Because of (5.17), it is sufficient to prove that

$$\mathscr{E}(\tilde{c}) = \mathscr{E}(c).$$

Using (5.16) and the fact that after the subcell remapping stage, the subcell internal and kinetic energies are not changed, we derive

$$\mathscr{E}(\tilde{c}) = \sum_{n \in N(c)} \mathscr{E}(cn) + \left[\left(\sum_{n \in N(c)} K(cn) \right) - K(\tilde{c}) \right]. \tag{6.10}$$

By construction

$$\sum_{n \in N(c)} \mathscr{E}(cn) = \mathscr{E}(c), \quad \sum_{n \in N(c)} K(cn) = K(c).$$

Therefore, from (6.10) we can conclude that

$$\mathscr{E}(\tilde{c}) = \mathscr{E}(c) + [K(c) - K(\tilde{c})]. \tag{6.11}$$

Finally, the expression in square brackets in (6.11) is zero, because of the definition of $K(\tilde{c})$ in (5.13), and because $u(\tilde{n}) = u(n)$. Thus we have proved that $\mathscr{E}(\tilde{c}) = \mathscr{E}(c)$, and so

$$\varepsilon(\tilde{c}) = \varepsilon(c).$$

To summarize, we have demonstrated that all of the primary quantities before repair do not change if the new mesh and the old mesh are the same. Further, the repair process does not change anything because all variables are in bounds by definition, if the meshes are identical.

6.3. DeBar consistency condition

As mentioned in Section 3.1, when the nodal velocity on the old mesh is constant $\{u(n) = \mathscr{C}\}$ the subcell velocity on the old mesh is also constant, $\{u(cn) = \mathscr{C}\}$. For a constant subcell velocity $\{u(cn) = \mathscr{C}\}$, it is true by definition that the subcell momentum on the old mesh is $\{\mu(cn) = \mathscr{C} \cdot m(cn)\}$ and the remapped subcell momentum on the new mesh is $\{\mu(\widetilde{cn}) = \mathscr{C} \cdot m(\widetilde{cn})\}$. Now using (5.5) we get $\{u(\widetilde{cn}) = \mathscr{C}\}$. From the property of the matrix $\bar{\mathbf{I}}_{\tilde{c}}$ in (3.16) (which holds for both the old and new meshes) and Eq. (5.6) we conclude that $\{u^{\tilde{c}}(\tilde{n}) = \mathscr{C}\}$. Finally, from Eq. (5.7), and because $\{u^{\tilde{c}}(\tilde{n}) = \mathscr{C}\}$, we derive $u(\tilde{n}) = \mathscr{C}$. Thus we have proved that the DeBar consistency condition is satisfied: if a body has an uniform velocity and a spatially varying density, then the remap procedure exactly reproduces this uniform velocity.

7. Numerical results

In this section we will investigate numerically the performance of our new method. All problem are solved in Cartesian coordinates (x, y).

Our remapping method is unique in the sense that it is intended for a staggered mesh of general polygons using a subcell discretization of the density. As previously mentioned, we are not aware of any other method that can treat such a remapping problem. However, we are still interested in comparing our new remapping method with other known methods for remapping on a staggered mesh. To make such comparisons, we need to identify specific situations where both our new method and other existing methods can be used. One such situation is the case of a 1D staggered discretization, where there are no polygons and there is no hourglass phenomenon. Therefore, in Section 7.1.1, we consider several well-known 1D problems (i.e., where the solution depends only on x): Sod's problem, [30,31]; the blast wave problem of Woodward and Colella, [34,25]; and the LeBlanc shock tube problem, [4,25]. On this set of problems we will compare our new method with three other methods: the Half-Interval-Shift (HIS) method, [3,4]; the Nodal Momentum Remap (NMR) method, [25]; and the Method of Moments (MM), [19,4]. All three methods, HIS,NMR, and MM employ only a cell-centered discretization for density (no subcells) and differ individually in how velocity is remapped.

The 1D HIS method employs a remap of two cell-centered momenta, that are "momenta shifted" from the vertices of the corresponding cell, remapped, and then combined to recover unique velocities at the vertices. The 1D MM employs a cell-centered remap of cell-centered momentum, (1.9), and discrete derivative, $\delta u/\delta x \sim \partial u/\partial x$, and then uses these to recover a unique velocity at the node. The NMR method uses a dual mesh with vertices in the centers of the original cells to directly remap nodal momentum. We will not discuss advantages and disadvantages of these methods, but refer the interested reader to [4]. In our implementations of all these methods, after a unique velocity at each node is recovered, a cell-centered specific internal energy is defined as described in Section 5.3. In all implementations, repair is performed in the same way as described in Section 5. Also, in our 1D implementation of these three methods, the Lagrangian phase is the same for all methods and is consistent with the methodology described in [12] when all subcell densities are equal. In 1D there can be no hourglass deformation, and so no hourglass treatment is necessary. To make the implementation of HIS, MM, and NMR, methods comparable to our method, we also have arranged the order of computations similarly to our method. In all three methods the flux limiter remapper is used with the specific choice of the Barth–Jespersen limiter [2]. We apply all methods in an Eulerian framework, which is described as "Eulerian as Lagrange Plus Remap", [25].

In Section 7.1.2, we will use the same set of 1D test problems to investigate numerically the convergence properties of our new algorithm, always in the Eulerian framework. In Section 7.2, we will use

the well-known 2D Sedov blast wave problem to demonstrate the performance of our new method on both logically rectangular and polygonal meshes. This problem can be run in pure Lagrangian regime as well, and we will use Lagrangian results as a reference. For this problem we will present results for both the Eulerian framework and also for an ALE method using the RJM rezone strategy, [17,28].

7.1. One-dimensional tests

For all 1D test problems we use our new method implemented in a 2D code, but run on an initially square mesh with only two cells in y direction. The length of the computational domain in the y direction, $y_{\max}$, depends on number of cells in x direction (in the x direction initial mesh is always uniform). In our description of the test problems, we will specify only the length of computational domain and the number of cells in the x direction. It is interesting that our numerical experiments produce almost identical results (at the resolution presented) for the MM, the HIS and the NMR methods. For this reason we present only results obtained by NMR method.

7.1.1. Comparison with other methods

7.1.1.1. Sod problem. The Sod problem is a Riemann shock tube with a relatively small discontinuity, and so is very mild test. Its solution consists of a left moving rarefaction, a contact discontinuity and a right moving shock; the exact solution is illustrated in Fig. 6 by the solid line.

In our numerical experiments, the computational domain is $1 \geqslant x \geqslant 0$. The discontinuity is initially at 0.5. The domain is filled with an ideal gas with $\gamma = 1.4$. The density/pressure values on the left side of the discontinuity are 1.0/1.0, while those on the right side are 0.125/0.1. In Fig. 6, we present numerical results for the density at the final time $t = 0.25$ for a run with $N_x = 200$ computational cells. The results obtained by our new method and by the NMR method are very close, but the resolution of the contact discontinuity is slightly better for our method.

7.1.1.2. Woodward–Colella blast wave problem. The computational domain for this problem has length one, with reflecting walls at the both ends. The gas is an ideal gas with $\gamma = 1.4$. At $t = 0.$, the gas is at rest with an uniform density equal to 1.0. The initial pressure is 1000.0 in the leftmost tenth of the domain, 100.0 in the rightmost tenth, and 0.01 everywhere else. The final problem time is $t = 0.038$. Initially, two shocks and two contacts develop at the initial discontinuities and propagate toward one another, while two rarefactions develop, propagate toward the walls, and reflect off them. As time progresses, these six initial waves interact and create additional contact discontinuities. There is no analytical solution for this problem and typically a solution obtained by purely Lagrangian method with very high resolution ($N_x = 3600$ cells in our case) is considered as the reference "truth" (the solid line in Fig. 7). As has been mentioned in [25], the Lagrangian solution has a flaw, a spurious overshoot at $x \approx 0.765$. In Fig. 7 we present numerical results obtained by NMR and our new method for $N_x = 1200$.

A discussion of the results obtained by the NMR method can be found in [25]. It appears that, at least for the density, our method gives better results for this problem. We note however that the difference in the results is accentuated by the use of the Barth–Jespersen limiter. When the minmod limiter is used in both methods, the results are much closer. In presenting these results, we note that in our 2D code, we use the Barth–Jespersen limiter most typically.

7.1.1.3. LeBlanc shock tube problem. In this extreme shock tube problem, the initial discontinuity separates a region of very high energy and density from one of low energy and density. This is a much more severe test than the Sod Problem. The computational domain is $9 \geqslant x \geqslant 0$ and is filled with an ideal gas with $\gamma = 5/3$. The gas is initially at rest. The initial discontinuity is at $x = 0.3$: $(\rho,\varepsilon) = (1, 0.1)$ for $x < 3$ and

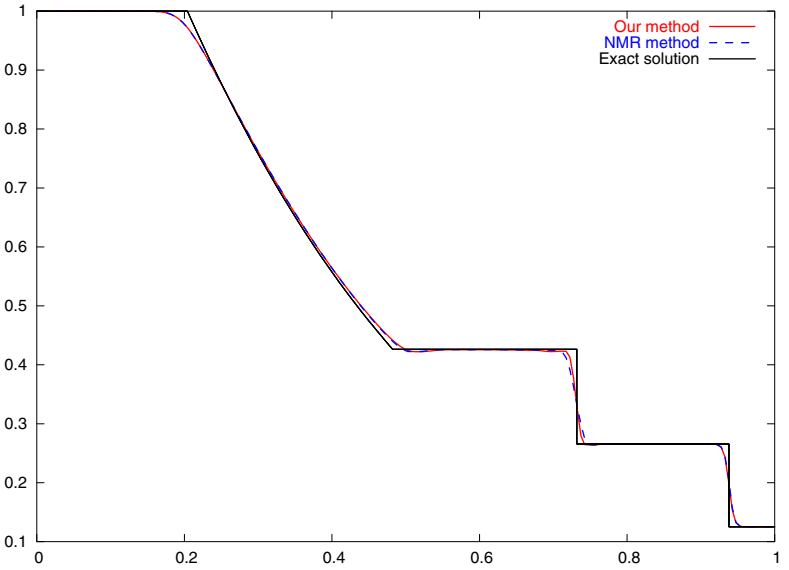

Fig. 6. Sod test problem. Comparison of our new method and the NRM method: density at $t = 0.25$, $N_x = 200$.

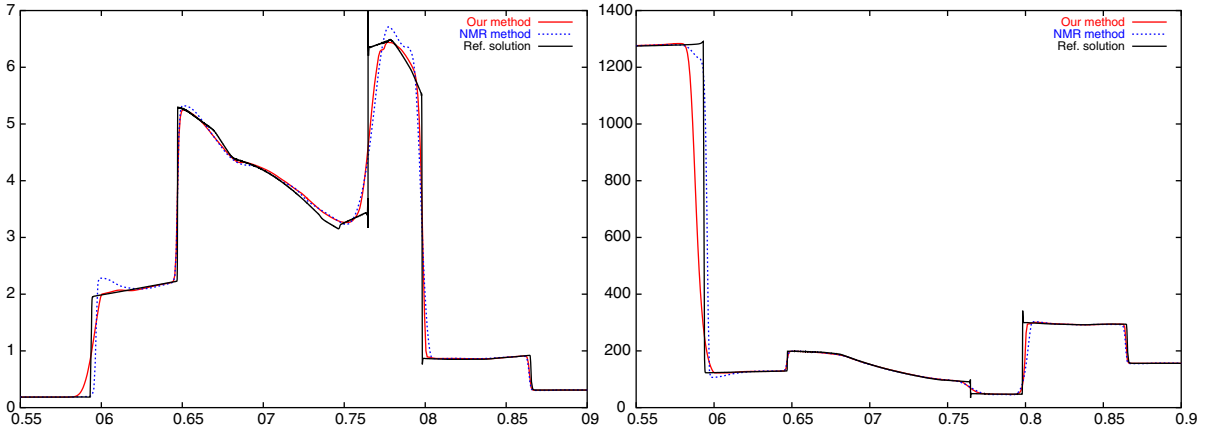

Fig. 7. Woodward–Colella blast wave problem. Comparison of the NMR method and the new method: density (zoom)—left, and specific internal energy (zoom)—right at $t = 0.038$, $N_x = 1200$.

(0.001×10^{-7}) for $x > 3$. The solution consists of a rarefaction moving to the left, and a contact discontinuity and a strong shock moving to the right—solid line in Fig. 8. At the final time of $t = 6.0$, the shock wave is located at $x = 7.975$. In Fig. 8 we present numerical results obtained by NMR and our new method for $N_x = 1400$. In comparison with the NMR method, our new method gives a more accurate position of the contact discontinuity, but shows a larger, relatively narrow, overshoot at the contact. The position of the shock is slightly more accurate for the NMR method.

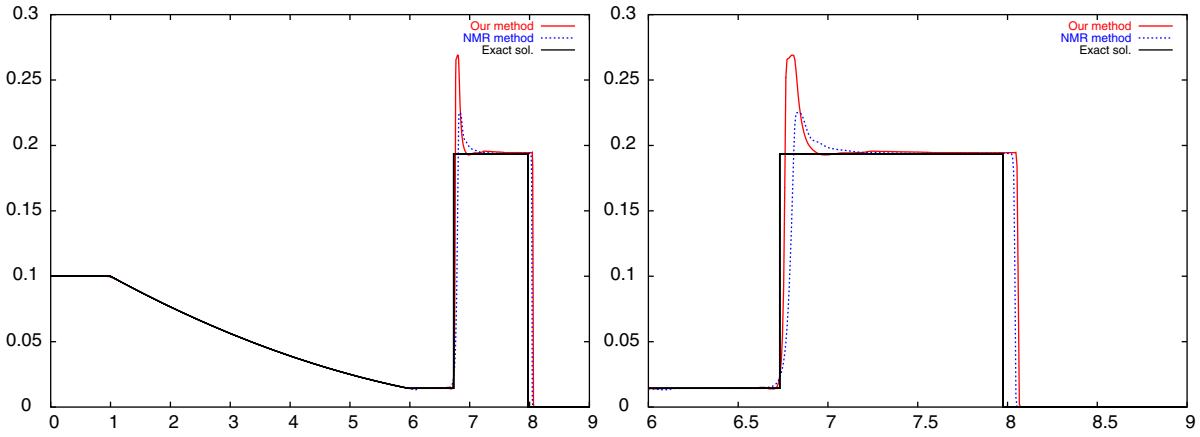

Fig. 8. LeBlanc shock tube problem. Comparison of the NMR method and the new method—specific internal energy at $t = 6.0$, $N_x = 1440$: entire computational domain—left, zoom—right.

The numerical results presented in this subsection demonstrate that, on these 1D problems, our new method shows comparable performance to other known remapping methods on a staggered mesh, i.e., the nodal momentum remap method, the half-interval-shift method, and the method of moments.

7.1.2. Convergence tests

In this subsection we investigate numerically the convergence of our new method for the 1D test problems described in the previous section. Recall that all these problems are run in the Eulerian framework.

7.1.2.1. Sod problem. In Fig. 9 we present the exact solution and numerical results for the density for resolutions $N_x = 50, 100, 200$. In Table 1 we present the L_1 errors for density and corresponding estimates for the convergence rate, which is close to 2.

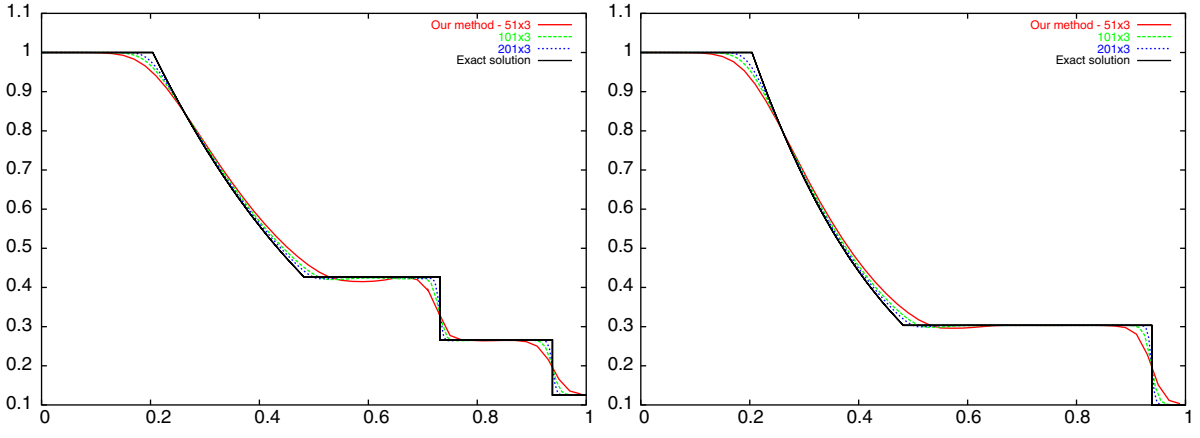

Fig. 9. Convergence for Sod problem: density and pressure at $t = 0.25$ for $N_x = 50, 100, 200$.

Table 1
Sod problem

N_x	L_1 Error	Convergence rate
50	5.63E−4	–
100	1.52E−4	1.88
200	4.22E−5	1.87
400	1.17E−5	1.85
800	3.23E−6	1.86
1600	8.77E−7	1.88

Errors and convergence rate.

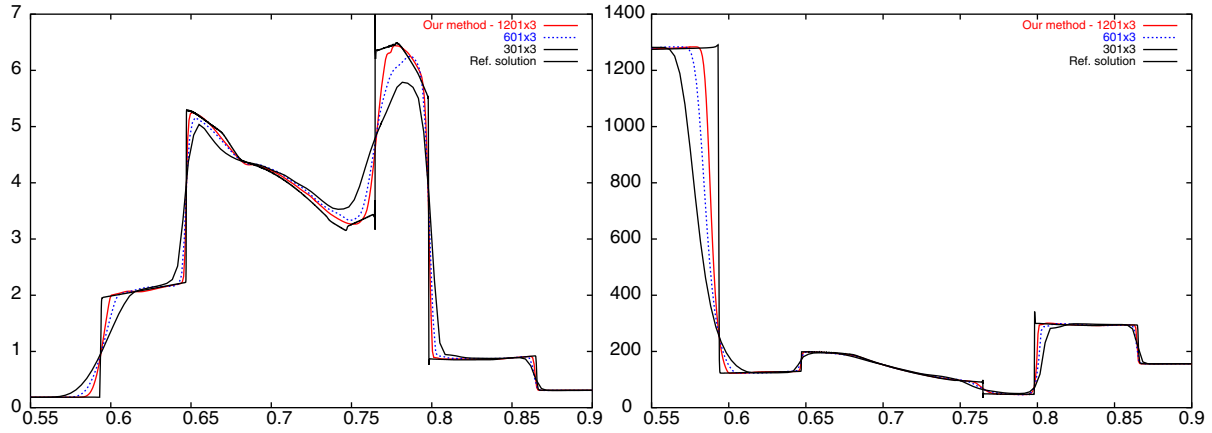

Fig. 10. Convergence for Woodward–Colella blast wave problem. Density (zoom)—left, and specific internal energy (zoom)—right at $t = 0.038$, $N_x = 300, 600, 1200$.

7.1.2.2. Woodward–Colella blast wave problem. In Fig. 10 we graphically demonstrate the convergence rate for the Woodward–Colella blast wave problem on the set of meshes with resolutions $N_x = 300, 600, 1200$. We present numerical results both for density and for specific internal energy. Because there is no analytical solution for this problem we do not present a table with convergence rates.

7.1.2.3. LeBlanc shock tube problem. In Figs. 11 and 12 we present numerical results and the exact solution for the specific internal energy and pressure for the LeBlanc shock tube problem. The convergence rate is analyzed in Table 2. Table 2 demonstrates approximately first-order convergence for the specific internal energy. We note here that the initial spatially uniform mesh for LeBlanc problem creates 10^3 jump in the masses of the cells adjacent to initial discontinuity, which implies a loss of accuracy in the Lagrangian stage at the beginning of calculation. This explains the observed low order of convergence in comparison with the Sod problem.

In summary, the numerical results presented for the Sod problem, the Woodward–Colella blast wave problem, and the LeBlanc shock tube problem, indicate a convergence rate between first and second order.

7.2. Two-dimensional tests

In this subsection we present numerical results for the Sedov blast wave problem, [27], which describes the evolution of a blast wave in a point symmetric explosion; it is an example of a diverging

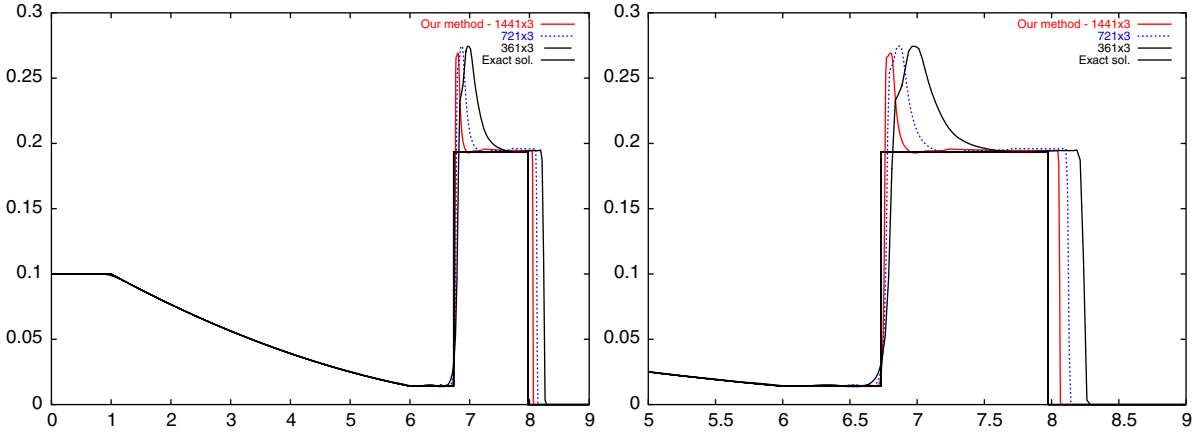

Fig. 11. Specific internal energy for the LeBlanc shock tube problem at $t = 6.0$, $N_x = 360, 720, 1440$: entire computational domain—left, zoom—right.

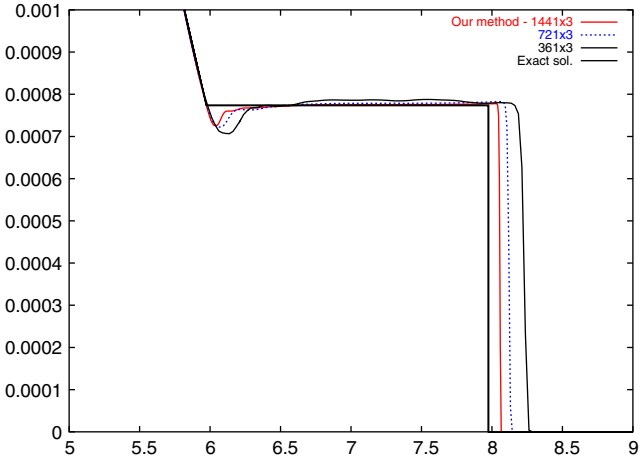

Fig. 12. Zoom for the pressure for the LeBlanc shock tube problem at $t = 6.0$, $N_x = 360, 720, 1440$. The pressure profile for the entire domain is not presented because details of shock resolution (and shock itself) are not visible due to the strong rarefaction wave.

Table 2
LeBlanc problem

N_x	L_1 Error	Convergence rate
180	7.39E−2	–
360	3.38E−2	1.13
720	1.60E−2	1.08
1440	7.79E−3	1.04
2880	3.84E−3	1.02

Errors and convergence rate.

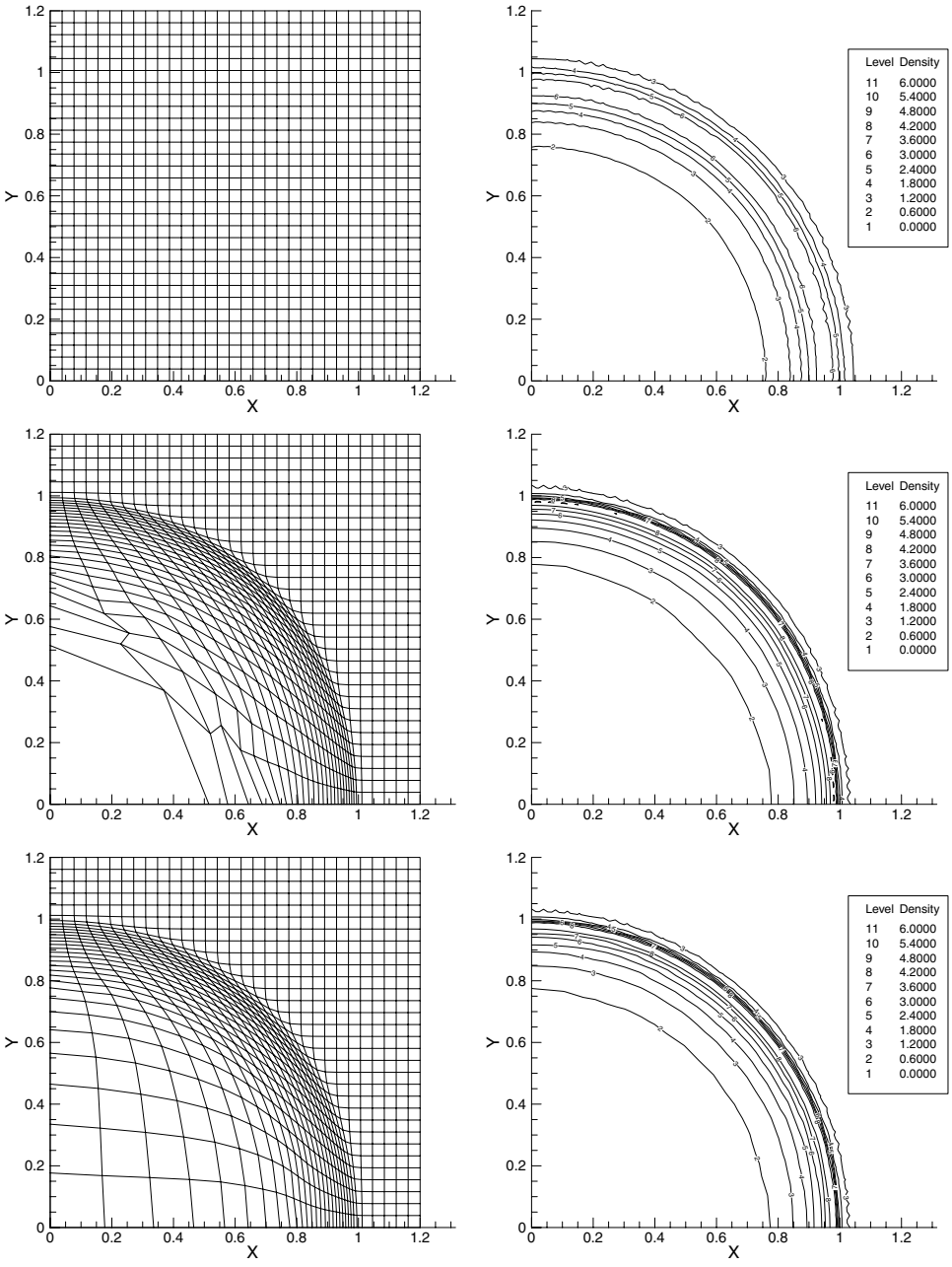

Fig. 13. Sedov problem—quadrilateral mesh. Mesh (left), and density isolines (right) at $t = 1.0$—Eulerian regime (top), Lagrangian regime (middle), ALE regime (bottom).

shock wave. We consider the cylindrically symmetric Sedov problem, in Cartesian coordinates (x, y). The total energy of the explosion is concentrated at the origin and has magnitude $E_{\text{total}} = 0.244816$ (similar to [9]). The material is an ideal gas with $\gamma = 1.4$ and initially is at rest with an initial density

equal to 1. At time $t = 1.0$ the exact solution is a cylindrically symmetric diverging shock whose front is at radius, $r = \sqrt{x^2 + y^2} = 1$ and has a peak density of 6.0 (the solid line in Fig. 14). In our numerical experiments E_{total} is concentrated in one cell located at the origin (that is, containing the vertex $(x, y) = (0, 0)$). The specific internal energy of this cell, c is defined as $\varepsilon(c) = E_{\text{total}}/V(c)$. Therefore the initial pressure is $p = (\gamma - 1)\rho\varepsilon = 0.4E_{\text{total}}/V(c)$. For this problem we compare results obtained by our 2D code in three different frameworks: a purely Eulerian, a purely Lagrangian, and an ALE framework. Simulations in all three frameworks have been carried out on both quadrilateral and polygonal meshes. In the ALE calculation, the rezoning/remapping is performed once every 10 Lagrangian steps. The CFL number is chosen to be equal to 0.25 for all simulations. For each simulation we show both the initial and the final mesh, with 11 density isolines equally distributed in magnitude between 0.0 and 6.0. (Figs. 13 and 15). Each isoline has a label that refers to a density value in the legend scale. Also we show a 1D plot of density as a function of the radius, r, and a corresponding plot of the exact solution (Figs. 14 and 16). The 1D plots demonstrate how well the numerical solution preserves cylindrical symmetry.

7.2.1. Quadrilateral meshes

For this set of simulations, the computational domain is a square $(x, y) \in [0 : 1.2] \times [0 : 1.2]$ whose initial mesh consists of 31×31 square cells (top-left mesh in Fig. 13). The two top panels in Fig. 13 and the left panel in Fig. 14 shows the results of purely Eulerian computations. The symmetry of the solution is preserved quite well but the density peak is diminished ($\rho_{\max} = 3.55$ instead of 6) and the shock wave is spread over several cells. The two panels in the middle of Fig. 13 and the central panel in Fig. 14 shows the results of purely Lagrangian computations. The peak density magnitude, 4.9, is much closer to the correct value than is the Eulerian computational value. Also, the symmetry is better preserved in the Lagrangian calculation, especially near the peak. However, the Lagrangian mesh has a very low geometrical quality near the axis. The two bottom panels in Fig. 13 and right panel in Fig. 14 shows results of the ALE computations. The symmetry of the solution is even better than was found in the Lagrangian calculations and the peak density is 4.75 which is little bit smaller than in the Lagrangian calculations. The geometrical quality of the mesh is significantly improved in comparison with the Lagrangian case. In the top part of Table 3 we present the peak density values and also the number of time steps needed to reach the final time of $t = 1.0$ for the Eulerian, Lagrangian and ALE computations. It is interesting to note that the ALE computation takes the least number of time steps.

The ratio between the CPU time spent for the Eulerian regime versus the Lagrangian regime is $\sim$10, between the ALE regime and the Lagrangian one is $\sim$2. We remark that these timing comparisons are strongly dependent on the details of implementation and are presented to the reader as very ''rough'' estimates.

7.2.2. Polygonal meshes

The computational domain is one quarter of a circular disk with radius of $r_{\max} = 1.2$. A polygonal mesh is constructed in the computational domain using a Voronoi diagrams (see for example, [24]) for the set of point defined as follows:

$$x_{i,j} = r_j \sin(\theta_{i,j}) y_{i,j} = r_j \sin(\theta_{i,j}); \qquad j = 1, \ldots, J; \quad i = 1, \ldots I(j).$$

where

$$r_j = r_{\max} \cdot \frac{j-1}{J}, \quad I(j) = \mathbf{round}((j-1)\frac{\pi}{2}), \quad \theta_{i,j} = \frac{i-1}{I(j)} \cdot \frac{\pi}{2}, \quad J = 31.$$

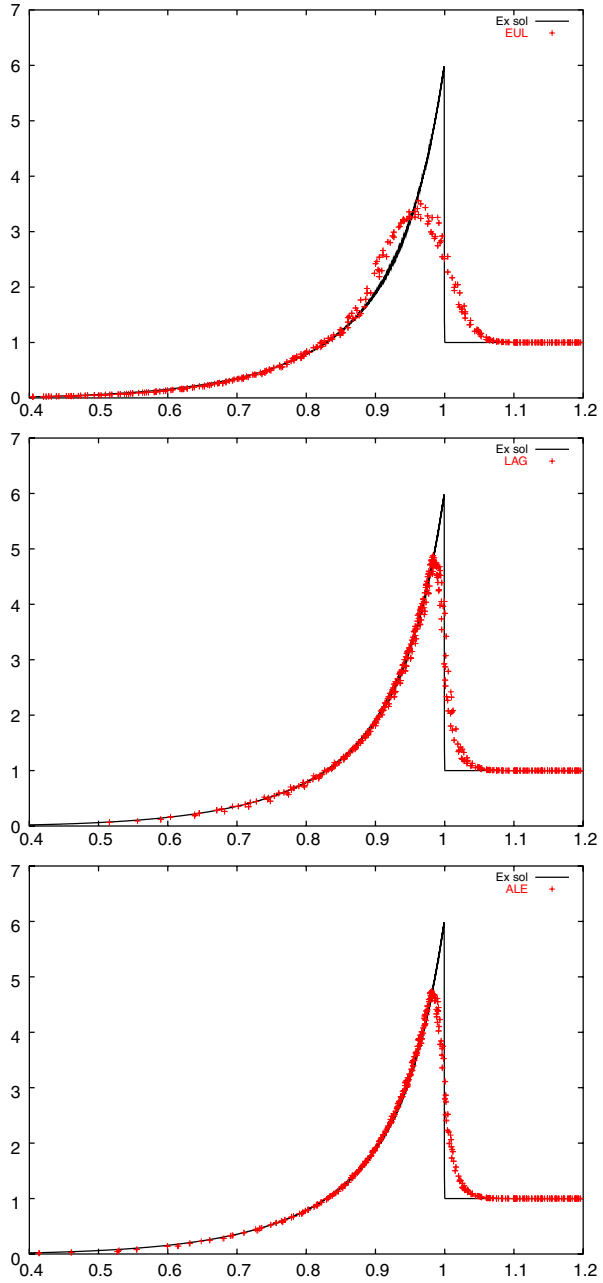

Fig. 14. Sedov problem—quadrilateral mesh. Density at $t = 1.0$ as a function of the radius (solid line exact solution)—Eulerian regime (top), Lagrangian regime (middle), ALE regime (bottom).

and function **round**(x) returns the closest integer to x. According to these formulas, on each circle of radius r_j points are distributed so that the distance between adjacent points along the circle is approximately equal to $\Delta r = r_{\max}/(J-1)$. The total number of points is 775. There is exactly one Voronoi cell corresponding to

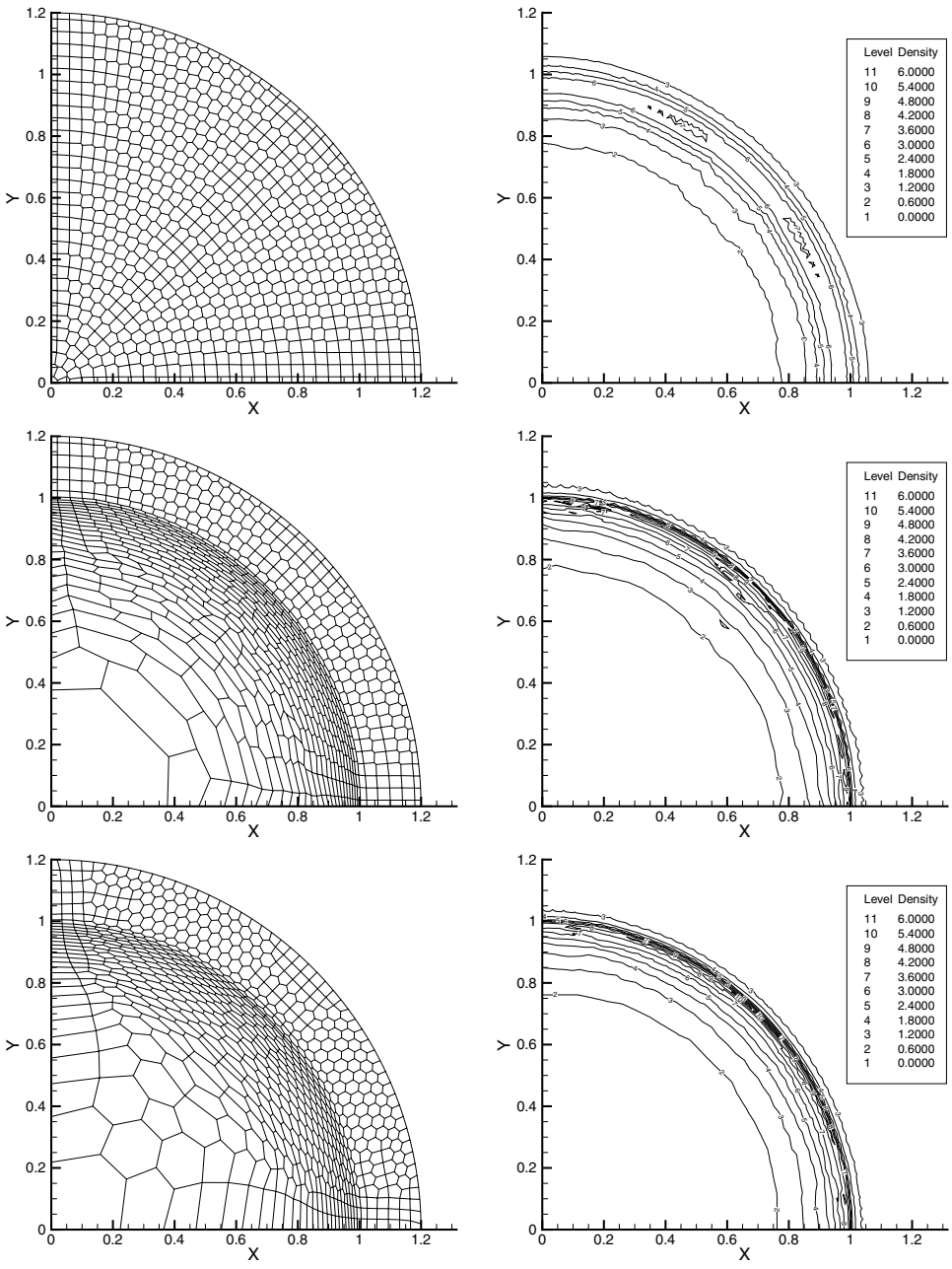

Fig. 15. Sedov problem—polygonal mesh. Mesh (left), and density isolines (right) at $t = 1.0$—Eulerian regime (top), Lagrangian regime (middle), ALE regime (bottom).

each point. The mesh consists of a mixture of convex polygons: quadrilaterals, pentagons and hexagons, and the total number of vertices is 1325; the mesh is shown in Fig. 15 (top-left panel). The resulting polygonal mesh has approximately the same resolution as the quadrilateral mesh presented in Fig. 13. Numerical

Table 3
Sedov problem

	# of time steps	Peak density	Mesh type
Eulerian	477	3.55	Quad
Lagrangian	375	4.90	Quad
ALE-10	338	4.75	Quad
Eulerian	1567	3.69	Poly
Lagrangian	603	6.20	Poly
ALE-10	408	5.70	Poly

Number of time steps needed to reach final time $t = 1.0$ and peak density values.

results for the initially polygonal mesh are arranged in a similar way as was done for our study of the quadrilateral meshes and are presented in Figs. 15 and 16, and bottom of Table 3. Qualitatively, the relative performance of purely Eulerian, purely Lagrangian, and ALE methods on polygonal meshes is the same as for quadrilateral meshes. The results of the purely Eulerian and purely Lagrangian calculations on the polygonal mesh exhibit less symmetry than the corresponding calculations on the quadrilateral meshes. However, the polygonal mesh behaves better near the axes even for purely Lagrangian calculations. In this case the ratio between the CPU time spent for the Eulerian versus the Lagrangian regimes is $\sim$20 and between the ALE and Lagrangian regime $\sim$2.

8. Conclusion

In this paper we have constructed a full ALE method for use on a staggered polygonal mesh. The method combines and generalizes previous work on the Lagrangian and rezoning phases, and includes a new remapping algorithm.

In the Lagrangian phase of the ALE method we use compatible methods to derive the discretizations [8,9]. We assume a staggered grid where velocity is defined at the nodes, and where density and internal energy are defined at cell centers. In addition to nodal and cell-centered quantities, our discretization employs subcell masses that serve to introduce special forces that prevent artificial grid distortion and hourglass-type motions, [10]. This adds an additional requirement to the remap phase—that the subcell densities (corresponding to subcell masses) have to be conservatively interpolated in addition to nodal velocities and cell-centered densities and internal energy.

In the remap phase, we assume that the rezone algorithm produces mesh that is "close" to Lagrangian mesh so that a local remapping algorithm (i.e., where mass and other conserved quantities are only exchanged between neighboring cells) can be used.

Our new remapping algorithm consists of three stages.

- A *gathering stage*, where we define momentum, internal energy, and kinetic energy in the subcells in a conservative way such that the corresponding total quantities in the cell are the same as at the end of the Lagrangian phase.
- A *subcell remapping stage*, where we conservatively remap mass, momentum, internal, and kinetic energy from the subcells of the Lagrangian mesh to the subcells of the new rezoned mesh.
- A *scattering stage*, where we conservatively recover the primary variables: subcell density, nodal velocity, and cell-centered specific internal energy on the new rezoned mesh.

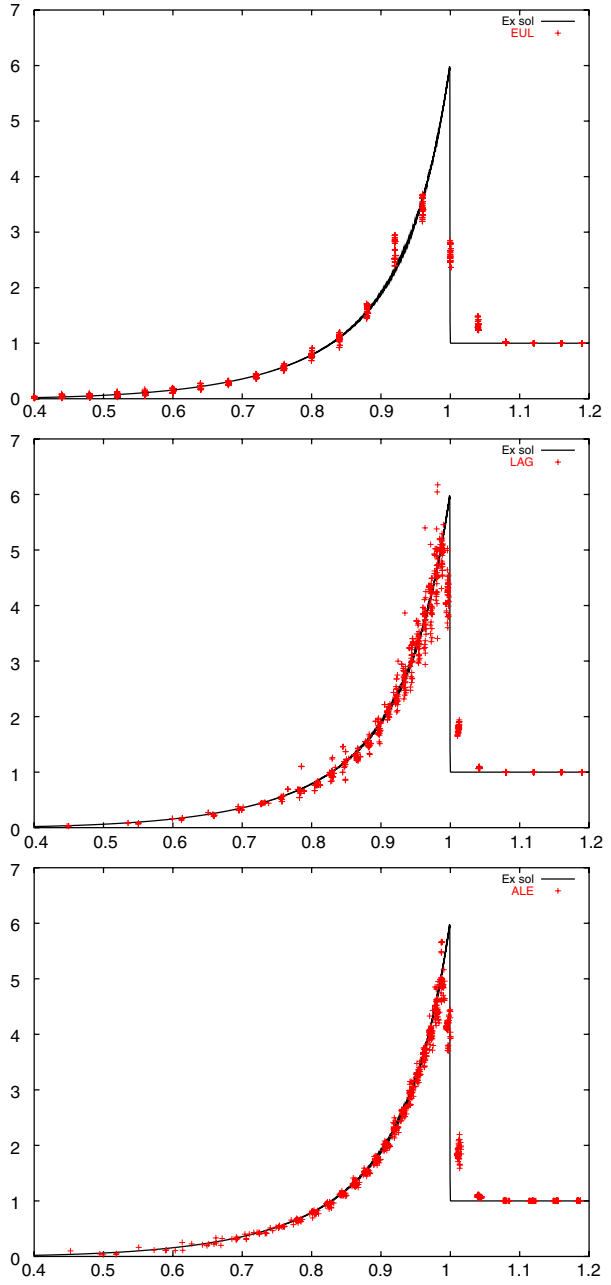

Fig. 16. Sedov problem—polygonal mesh. Density at $t = 1.0$ as a function of the radius (solid line exact solution)—Eulerian regime (top), Lagrangian regime (middle), ALE regime (bottom).

We have proved that our new remapping algorithm is conservative, reversible, and satisfies the DeBar consistency condition.

We have also demonstrated computationally that our new remapping method is robust and accurate for a series of test problems in one and two dimensions.

Acknowledgements

The authors thank L. Margolin, B. Rider, S. Li, B. Wendroff, R. Anderson, R. Pember, D. Benson, and T. Dey for fruitful discussions. Special thanks go to K. Lipnikov for developing the procedure for the analytical inversion of matrices described in Appendix A.

This work was performed under the auspices of the US Department of Energy at Los Alamos National Laboratory, under contract W-7405-ENG-36. The authors acknowledge the partial support of the DOE/ASCR Program in the Applied Mathematical Sciences and the Laboratory Directed Research and Development program (LDRD). The authors also acknowledge the partial support of DOE's Advanced Simulation and Computing (ASC) program.

Appendix A. Details of the velocity gathering

Invertibility: Here we present a constructive proof of the invertibility of $\bar{\mathbf{I}}_c$ (see Eq. (3.15)). The analytical inversion can be performed by taking into account the specific form of the matrix $\bar{\mathbf{I}}_c$, and then rewriting $\bar{\mathbf{I}}_c$ into the form

$$\bar{\mathbf{I}}_c = \bar{\mathbf{S}} + \mathbf{w}\mathbf{r}^t, \tag{A.1}$$

where

$$\bar{\mathbf{S}} = \frac{1}{4}\begin{pmatrix} 2 & \frac{1}{2} & 0 & 0 & \cdots & 0 & \frac{1}{2} \\ \frac{1}{2} & 2 & \frac{1}{2} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{2} & 2 & \frac{1}{2} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{2} & 0 & 0 & 0 & \cdots & \frac{1}{2} & 2 \end{pmatrix}, \tag{A.2}$$

and

$$\mathbf{w} = (1, 1, \ldots, 1)^t, \quad \mathbf{r} = (Q_{N,1,2}, Q_{1,2,3}, \ldots, Q_{N-1,N,1})^t. \tag{A.3}$$

Lemma. $\bar{\mathbf{I}}_c$ is invertible and $(\bar{\mathbf{I}}_c)^{-1}$ is given by the following formula:

$$(\bar{\mathbf{I}}_c)^{-1} = \bar{\mathbf{S}}^{-1} - \mathbf{w}\mathbf{r}^t\bar{\mathbf{S}}^{-1}. \tag{A.4}$$

Proof. Let us first remark that

$$\bar{\mathbf{S}}\mathbf{w} = \frac{3}{4}\mathbf{w}, \tag{A.5}$$

and

$$\mathbf{r}^t\mathbf{w} = (Q_{N,1,2}, Q_{1,2,3}, \ldots, Q_{N-1,N,1}).(1, 1, \ldots, 1)^t = \sum_{n=1}^{N} Q_{n^-,n,n^+} \tag{A.6}$$

$$= \frac{1}{8m(c)}\sum_{n=1}^{N}(-m(cn^-) + 4m(cn) - m(cn^+)) \tag{A.7}$$

$$= \frac{1}{8m(c)}\sum_{n=1}^{N} 2m(cn) = \frac{2m(c)}{8m(c)} = \frac{1}{4}. \tag{A.8}$$

Now we verify that $\bar{\mathbf{I}}_c(\bar{\mathbf{I}}_c)^{-1} = \bar{\mathbf{E}}$, where $\bar{\mathbf{E}}$ is the identity matrix

$$\bar{\mathbf{I}}_c(\bar{\mathbf{I}}_c)^{-1} = \left(\bar{\mathbf{S}} + \mathbf{w}\mathbf{r}^t\right)\left(\bar{\mathbf{S}}^{-1} - \mathbf{w}\mathbf{r}^t\bar{\mathbf{S}}^{-1}\right) = \bar{\mathbf{E}} - \underbrace{\left(\bar{\mathbf{S}}\mathbf{w}\right)}_{=3/4\mathbf{w}}\mathbf{r}^t\bar{\mathbf{S}}^{-1} + \mathbf{w}\mathbf{r}^t\bar{\mathbf{S}}^{-1} - \mathbf{w}\underbrace{\left(\mathbf{r}^t\mathbf{w}\right)}_{=1/4}\mathbf{r}^t\bar{\mathbf{S}}^{-1}$$

$$= \bar{\mathbf{E}} + \underbrace{\left(-\frac{3}{4} + 1 - \frac{1}{4}\right)}_{=0}\mathbf{w}\mathbf{r}^t\bar{\mathbf{S}}^{-1} = \bar{\mathbf{E}}.$$

Therefore $\bar{\mathbf{I}}_c$ is invertible and $(\bar{\mathbf{I}}_c)^{-1}$ is given by Eq. (A.4). □

The matrix $\bar{\mathbf{S}}$ can be exactly inverted for every size (recall that the dimension of $\bar{\mathbf{S}}$ is 3 if cell c is a triangle, 4 for a quadrilateral cell, etc). So once inverse matrices have been stored for every reasonable integer, then $(\bar{\mathbf{I}}_c)^{-1}$ can be easily computed using Eq. (A.4).

1D Analog: The meaning of formula (3.12) becomes more clear in 1D, where it is more natural to return to standard notations, Fig. 17. For nodes, we will use integer indexes, i,$i+1$ and so on. For cells, we will use half indexes, such that cell with end nodes i,$i+1$ has index $i+1/2$. Subcells , *cn*, now will be denoted like $i+1/2, i$. Also in 1D we will use of subscripts. For instance, nodal velocities are denoted by u_i, u_{i+1}, cell-centered velocities are $u_{i+1/2}$, and subcell velocities are $u_{i,i+1/2}$ and similar for other quantities.

In the 1D boundary of cell $c = i + 1/2$ consists of two nodes $n^- = i$, $n^+ = i + 1$. The subcell velocities are defined as follows (analog of formula (3.7))

$$u_{i+\frac{1}{2},i} = \frac{u_i + u_{i+\frac{1}{2}}}{2}, \quad u_{i+\frac{1}{2},i+1} = \frac{u_{i+\frac{1}{2}} + u_{i+1}}{2}. \tag{A.9}$$

This is a natural definition because subcell velocity is associated with the center of the subcell and it is natural to define it as an average of the velocities of the subcell end points.

The 1D analog of formula (3.5) is

$$m_{i+\frac{1}{2},i}u_{i+\frac{1}{2},i} + m_{i+\frac{1}{2},i+1}u_{i+\frac{1}{2},i+1} = m_{i+\frac{1}{2},i}u_i + m_{i+\frac{1}{2},i+1}u_{i+1}. \tag{A.10}$$

The formulas (A.9) and (A.10) give the following definition of cell-centered velocity (analog of (3.8))

$$u_{i+\frac{1}{2}} = \frac{m_{i+\frac{1}{2},i}u_i + m_{i+\frac{1}{2},i+1}u_{i+1}}{m_{i+\frac{1}{2},i} + m_{i+\frac{1}{2},i+1}}, \tag{A.11}$$

which is just the mass average of nodal velocities. One also can consider this formula as result of linear interpolation of momentum between nodes. Finally, using definition (A.9) and formula (A.11) we obtain the following expressions for subcell velocities (analog of formula (3.12)):

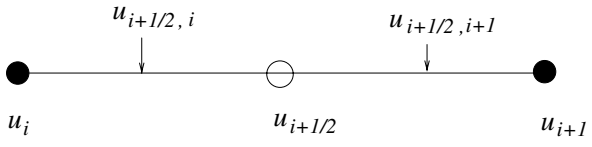

Fig. 17. Illustration to 1D velocity gathering.

$$u_{i+\frac{1}{2},i} = \frac{1}{2}u_i + \left[\frac{1}{2}\frac{1}{m_{i+\frac{1}{2}}}\left(m_{i+\frac{1}{2},i}u_i + m_{i+\frac{1}{2},i+1}u_{i+1}\right)\right],$$

$$u_{i+\frac{1}{2},i+1} = \frac{1}{2}u_{i+1} + \left[\frac{1}{2}\frac{1}{m_{i+\frac{1}{2}}}\left(m_{i+\frac{1}{2},i}u_i + m_{i+\frac{1}{2},i+1}u_{i+1}\right)\right].$$

In 1D the matrix $\bar{\mathbf{I}}_c$, (3.15), is

$$\bar{\mathbf{I}}_c = \frac{1}{2}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} + \frac{1}{2m_{i+\frac{1}{2}}}\begin{pmatrix}m_{i+\frac{1}{2},i} & m_{i+\frac{1}{2},i+1}\\ m_{i+\frac{1}{2},i} & m_{i+\frac{1}{2},i+1}\end{pmatrix} = \frac{1}{2m_{i+\frac{1}{2}}}\begin{pmatrix}2m_{i+\frac{1}{2},i} + m_{i+\frac{1}{2},i+1} & m_{i+\frac{1}{2},i+1}\\ m_{i+\frac{1}{2},i} & m_{i+\frac{1}{2},i} + 2m_{i+\frac{1}{2},i+1}\end{pmatrix}.$$

The determinant of this matrix is equal to 1/2 and therefore it is invertible.

References

[1] A.A. Amsden, H.M. Ruppel, C.W. Hirt, SALE: A Simplified ALE computer program for fluid flow at all speeds, Los Alamos National Laboratory Report LA-8095, 1980.

[2] T.J. Barth, Numerical methods for gasdynamics systems on unstructured grids, in: D. Kroner, M. Ohlberger, C. Rohde (Eds.), An Introduction to Recent Developments in Theory and Numerics for Conservation Laws, Proceedings of the International School on Theory and Numerics for Conservation Laws, Freiburg/Littenweiler, October, 20–24, 1997, Lecture Notes in Computational Science and Engineering, Springer, Berlin, 1997, pp. 195–285.

[3] D.J. Benson, An efficient, accurate, simple ALE method for nonlinear finite element programs, Comput. Meth. Appl. Mech. Eng. 72 (1989) 305–350.

[4] D.J. Benson, Computational methods in Lagrangian and Eulerian hydrocodes, Comput. Meth. Appl. Mech. Eng. 99 (1992) 235–394.

[5] D.J. Benson, Momentum advection on a staggered grid, J. Comput. Phys. 100 (1992) 143–162.

[6] D.E. Burton, Multidimensional discretization of conservation laws for unstructured polyhedral grids, Report UCRL-JC-118306, Lawrence Livermore National Laboratory, 1994.

[7] D.E. Burton, Consistent finite-volume discretization of hydrodynamics conservation laws for unstructured grids, Report UCRL-JC-118788, Lawrence Livermore National Laboratory.

[8] J. Campbell, M. Shashkov, A compatible Lagrangian hydrodynamics algorithm for unstructured grids, Selcuk J. Appl. Math. 4 (2003) 53–70, report version can be found at www.cnls.lanl.gov/~shashkov.

[9] J. Campbell, M. Shashkov, A tensor artificial viscosity using a mimetic finite difference algorithm, J. Comput. Phys. 172 (2001) 739–765.

[10] E.J. Caramana, M.J. Shashkov, Elimination of artificial grid distortion and hourglass-type motions by means of Lagrangian subzonal masses and pressures, J. Comput. Phys. 142 (1998) 521–561.

[12] E.J. Caramana, D.E. Burton, M.J. Shashkov, P.P. Whalen, The construction of compatible hydrodynamics algorithms utilizing conservation of total energy, J. Comput. Phys. 146 (1998) 227–262.

[13] R.B. Demuth, L.G. Margolin, B.D. Nichols, T.F. Adams, B.W. Smith, SHALE : a computer program for solid dynamics, Report Los Alamos National Laboratory Report LA-10236, 1985.

[14] C.W. Hirt, A.A. Amsden, J. Cook, An arbitrary Lagrangian–Eulerian computing method for all flow speeds, J. Comput. Phys. 14 (1974) 227–253, and reprinted in 135 (1997) 203–216.

[15] D.S. Kershaw, M.K. Prasad, M.J. Shaw, J.L. Milovich, 3D unstructured mesh ale hydrodynamics with the upwind discontinuous finite element method, Comput. Meth. Appl. Mech. Eng. 158 (1998) 81–116.

[16] P. Kjellgren, J. Hyvarien, An arbitrary Lagrangian–Eulerian finite element method, Comput. Mech. 21 (1998) 81–90.

[17] P. Knupp, L.G. Margolin, M. Shashkov, Reference Jacobian optimization-based rezone strategies for arbitrary Lagrangian–Eulerian methods, J. Comput. Phys. 176 (2002) 93–112.

[18] M. Kucharik, M. Shashkov, B. Wendroff, An efficient linearity- and bound-preserving remapping method, J. Comput. Phys. 188 (2003) 462–471.

[19] L.G. Margolin and C.W. Beason, Remapping on the staggered mesh, Report UCRL-99682 of Lawrence Livermore National Laboratory, 1988. Available from: <www.llnl.gov/tid/lof/documents/pdf/208550.pdf>.

[20] L.G. Margolin, Introduction to an arbitrary Lagrangian–Eulerian computing method for all flow speeds, J. Comput. Phys. 135 (1997) 198–202.

[21] L.G Margolin, M. Shashkov, Remapping, recovery and repair on staggered grid, Comput. Meth. Appl. Mech. Eng. 193 (2004) 4139–4155.
[22] L.G. Margolin, M. Shashkov, Second-order sign preserving remapping on general grids, LA-UR-02-525. Available from: <www.cnls.lanl.gov/~shashkov/>.
[23] J.M. McGlaun, S.L. Thompson, M.G. Elrick, CTH: A three dimensional shock wave physics code, Int. J. Impact Eng. 10 (1990) 351–360.
[24] A. Okabe, B. Boots et al., Spatial Tessellations. Concepts and Applications of Voronoi Diagrams, 2nd ed., John Wiley, Chichester, England, 2000.
[25] R.B. Pember, R.W. Anderson, A comparison of staggered-mesh Lagrange plus remap and cell-centered direct Eulerian Godunov schemes for Eulerian shock hydrodynamics, Preprint UCRL-JC-139820, Lawrence Livermore National Laboratory, 2000. Available from: <www.llnl.gov/tid/lof/documents/pdf/238843.pdf>.
[26] J.S. Peery, D.E. Carroll, Multi-material ALE methods in unstructured grids, Comput. Meth. Appl. Mech. Eng. 187 (2000) 591–619.
[27] L.I. Sedov, Similarity and Dimensional Methods in Mechanics, Academic Press, New York, 1959.
[28] M. Shashkov, P. Knupp, Optimization-based reference-matrix rezone strategies for arbitrary Lagrangian–Eulerian methods on unstructured grids, Selcuk J. Appl. Math. 3 (2002) 81–99.
[29] M. Shashkov, B. Wendroff, The repair paradigm and application to conservation laws, J. Comput. Phys. 198 (2004) 265–277.
[30] G.A. Sod, A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws, J. Comput. Phys. 27 (1978) 1–31.
[31] E.F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics. A Practical Introduction, Springer, Berlin, 1999.
[32] P. Vachal, R. Garimella, M.J. Shashkov, R. Loubere, Untangling of meshes in ALE simulations, in: Proceedings of the 7th US National Congress on Computational Mechanics, Albuquerque, July 2003, Available from: <www.cnls.lanl.gov/~shashkov>.
[33] P. Vachal, R. Garimella, M. Shashkov, Untangling of 2D meshes in ALE simulations, J. Comput. Phys. 196 (2004) 627–644.
[34] P.R. Woodward, P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, J. Comput. Phys. 54 (1984) 115–173.