An Efficient Linearity and Bound Preserving Conservative Interpolation (Remapping) on Polyhedral Meshes

Rao Garimella, Milan Kucharik and Mikhail Shashkov

Los Alamos National Laboratory, T-7, MS-B284, Los Alamos, NM 87545, USA

Abstract

An accurate conservative interpolation (remapping) algorithm is an essential component of most Arbitrary Lagrangian-Eulerian (ALE) methods. In this paper, we describe an efficient linearity and bound preserving method for polyhedral meshes. The algorithm is based on reconstruction, approximate integration and conservative redistribution. We validate our method with a suite of numerical examples, analyzing the results from the viewpoint of accuracy and order of convergence.

Key words: ALE, Conservative Interpolation, Polyhedral Meshes
PACS: 02.70.-c, 47.11.+j

1 Introduction

In numerical simulations of fluid flow, the choice of the computational grid is crucial. Traditionally, there have been two viewpoints, utilizing the Lagrangian or the Eulerian framework, each with its own advantages and disadvantages. In a pioneering paper [16], Hirt et al. developed the formalism for a grid whose motion could be determined as an independent degree of freedom, and showed that this general framework could be used to combine the best properties of Lagrangian and Eulerian methods. This class of methods has been termed Arbitrary Lagrangian-Eulerian or ALE. Many authors have described ALE strategies to optimize accuracy, robustness, or computational efficiency – see for example [6], [4], [25], [18], [19], [31] and [24].

It is possible to use the ALE method in a mainly Lagrangian mode, with an occasional rezone/remap whenever the grid becomes too distorted. However, it is generally more effective to rezone and remap on each cycle, a strategy termed continuous rezoning. It is possible to formulate the ALE scheme as a single algorithm [9] based on solving the equations in a moving coordinate frame. However it is more usual to separate it into three individual phases. These are: 1) a Lagrangian phase, in which the solution and grid are updated, 2) a rezoning phase, in which the nodes of the computational grid are moved to a more optimal position, and 3) a remapping phase, in which the Lagrangian solution is interpolated onto the rezoned grid. One advantage of continuous rezone is that the individual grid movements can be constrained to be small, allowing the use of a local remapper where mass (and other conserved quantities) is only exchanged between neighboring cells. Local remappers are logically simpler and computationally more efficient than global remappers, where an explicit or implicit overlay of two arbitrary meshes is required [10], [11], [12], [15].

We are interested in developing an ALE methodology for 3D unstructured mesh consisting of *generalized* polyhedra. In this paper, a generalized polyhedron is a 3D solid with arbitrary topology and, possibly, non-planar polygonal faces. This general consideration is necessary because even simple polyhedra such as hexahedra can start out with planar faces at the start of the simulation but end up with non-planar faces due to the movement of its nodes as induced by the flow. Clearly, the geometry of a non-planar polygon is not uniquely defined. We will consider the issues raised by the presence of such faces in Section 2. The use of an unstructured mesh consisting of generalized polyhedra simplifies the setup process for computational domains with complex geometrical shapes and helps to avoid artificial mesh imprinting due to the restrictions of a conventional mesh consisting only of tetrahedra and generalized bricks, [7,8].

This paper focuses primarily on a new remapping algorithm for discrete scalar function on 3D generalized unstructured polyhedral meshes for use in continuous rezone ALE simulations. The algorithm is based on ideas presented in [21] and [26] for remapping on the general unstructured 2D polygonal meshes.

Our new method is linearity and bound preserving. Linearity preservation means that the method is exact if the discrete cell-centered values are constructed from a globally linear function. Bound preservation mean that remapping process does not create new extrema. The algorithm is computationally efficient because it does not require an explicit or implicit intersection of the two arbitrary meshes.

The algorithm is based on piece-wise linear reconstruction, approximate integration and conservative redistribution. We validate our method with a suite of numerical examples, analyzing the results from the viewpoint of accuracy and order of convergence.

The outline of the rest of this paper is as follows. In Section 2, we introduce necessary definitions. In Section 3, we give a precise statement of the remapping problem and list the desired properties of the remapping algorithm. In Section 4, we describe all the stages of the remapping algorithm. Numerical results that demonstrate the accuracy and convergence of the remapping algorithm are presented in Section 5. Finally, we conclude the paper in Section 6.

2 Grids

We consider a three-dimensional domain Ω and a mesh $\{c\}$ of non-overlapping cells c fully covering it. To formally describe the mesh, we need to introduce a set of definitions and corresponding notations. Our description of the mesh is based on boundary representations commonly used in geometric modeling where each object is described in terms of its bounding, lower-dimensional entities [17,30].

An n-dimensional mesh can be completely specified by describing the topology and geometry of its 0D to nD component entities. In this context, the topology of an entity refers to a shape-independent description of the object in terms of lower-dimensional entities and the geometry of an entity is a description of its shape in space [3,35]. The dimension of a mesh entity is the dimension of the parametric space necessary to describe the entity.

The fundamental components of a mesh are its *vertices* or *points* $\{p\}$ which are its zero-dimensional topological components. Each point p has a unique index, which for simplicity we will also denote by p. We will call these *primary* points to distinguish from *derived* points introduced later in this section.

The one-dimensional topological entities of a mesh are its *edges* $\{\mathbf{e}\}$. The topology of an edge is uniquely defined by the ordered set of points, $P(\mathbf{e}) = (p_0^e, p_1^e)$, which bound it. Each edge has a unique index, which we will denote by e. We will assume that geometry of an edge is specified by the straight line segment between p_0^e and p_1^e. We can then associate a vector going from point p_0^e to point p_1^e with the edge $\mathbf{e}$. Then it makes sense to define the following operation:

$$+1 \cdot \mathbf{e} \stackrel{def}{=} \mathbf{e} = (p_0^e, p_1^e)\,, \quad -1 \cdot \mathbf{e} \stackrel{def}{=} -\mathbf{e} \stackrel{def}{=} (p_1^e, p_0^e)\,. \tag{1}$$

We can also apply the standard notation for vector operations to edges; for example, if $\mathbf{e}_1 = (p_0^{e_1}, p_1^{e_1})$ and $\mathbf{e}_2 = (p_0^{e_2}, p_1^{e_2})$ then $\mathbf{e}_1 + \mathbf{e}_2 = (p_0^{e_1} + p_0^{e_2}, p_1^{e_1} + p_1^{e_2})$.

Faces are the two-dimensional topological components of a mesh. The topology of a face f is defined by an ordered set of edges together with the directions in which they are used, such that:

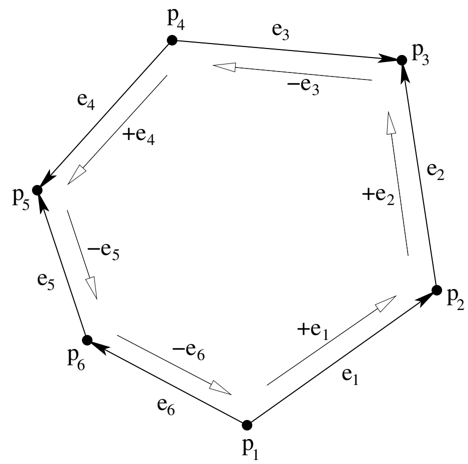


Fig. 1. Definition of the face. Bold arrows represent edges with independent orientation, for example, $\mathbf{e}_3 = \{p_4,\, p_3\}$. The inside arrows represent components of the face, for example, the component which corresponds to edge $\mathbf{e}_3$ is $-\mathbf{e}_3 = \{p_3,\, p_4\}$.

$$f = (d_1^f \cdot \mathbf{e}_1^f,\, d_2^f \cdot \mathbf{e}_2^f, \ldots) \text{ and } \sum_k d_k^f \cdot \mathbf{e}_k^f = \mathbf{0}\,, \tag{2}$$

where $d_k^f = \pm 1$ defines the orientation in which the edge $\mathbf{e}_k^f$ is used in the face and the equation $\sum_k d_k^f \cdot \mathbf{e}_k^f = \mathbf{0}$ means that the loop of directed edge vectors is closed (Fig. 1).

We then say that the kth component of the face is an edge along with an orientation. We will call such an edge an *external* edge of the face to distinguish it from *internal* edges of the face, which will be introduced later in this section.

All edges of the face are denoted by $E(f) = \cup_k \mathbf{e}_k^f$ and all points which define edges of the face are denoted by $P(f) = \cup_k P(\mathbf{e}_k^f)$.

The loop of directed edge vectors then leads to the notion of a *face normal* using the right hand rule of vectors. Note that this is a uniquely defined spatial direction only for planar faces; in all other cases, it is only serves to distinguish the two sides of a face.

The topology of a *face* is uniquely defined by the loop of its edges. The geometry of a face whose edges are not all in one plane, however, is not unique. Therefore, we adopt a faceted representation to obtain a consistent definition of its geometry as follows. First we define the "center" of the face

$$p_c^f = \left(\sum_{p_k \in P(f)} p_k \right) \Big/ \, |P(l)|\,, \tag{3}$$

where $|P(f)|$ is number of points in set $P(f)$, and the sum of points is understood in a coordinate sense (Fig. 2).

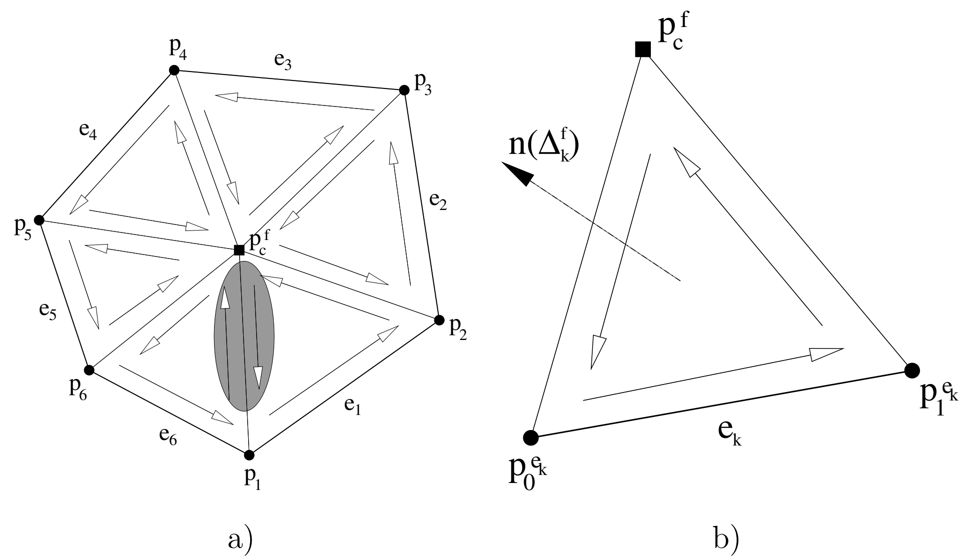


Fig. 2. Triangular elements of a face – a) Center of the face p_c^f, triangles, and internal edges; b) Normal to triangle Δ_k^f.

Next, for the face component $d_k^f \cdot e_k^f = (p_0^k,\, p_1^k)$, we define a triangle (which will be a planar facet of the face) as the ordered set of points $\Delta_k^f = (p_0^k,\, p_1^k,\, p_c^f)$.

We will call edges, $\{p_1^k,\, p_c^f\}$, $\{p_c^f,\, p_0^k\}$ of the triangle $\Delta_k^f = (p_0^k,\, p_1^k,\, p_c^f)$ *internal* edges (Fig. 2), since they are not part of the mesh definition. It is important to note that orientations of the internal edge in the two triangle which share it are opposite to each other (Fig. 2).

Orientation of the boundary of the triangle uniquely defines the unit normal (using the right-hand rule) to the triangle. We will denote this normal as $\mathbf{n}(\Delta_k^f)$, or generically by $\mathbf{n}^f$ if we are not talking about a specific triangle of the face. Note that the normals of the triangular facets will be consistent with the face normal, and if face f is planar then by construction all normals $\mathbf{n}(\Delta_k^f)$ are the same.

Now, we can define the geometry of the face as the union of all triangles (with corresponding normals) which correspond to the edges of the face:

$$f = (\Delta_0^f,\, \Delta_1^f,\, \ldots)\,. \tag{4}$$

We will denote the set of all triangles for a face f by $T(f) = \cup_k \Delta_k^f$.

A cell, c, is a solid or a three-dimensional topological component of a mesh which is bounded by faces, or effectively, by triangles that form the faces. We will call cells defined in this section *generalized* polyhedra or polyhedra for simplicity where it does not lead to ambiguity. Faces of the cell c are denoted by $F(c)$. All cells which share face of vertex with cell c are denoted by $C(c)$.

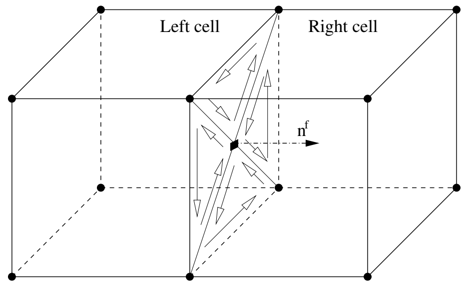


Fig. 3. Left – $c_L(f)$ and right – $c_R(f)$ cells for the face with normal $\mathbf{n}^f$.

Because each face is shared by only two cells, we will refer to the cell into which the face normal is pointing as the "right" cell and the other cell as the "left" cell (Fig. 3). Cells adjacent to face f are denoted by $C(f)$.

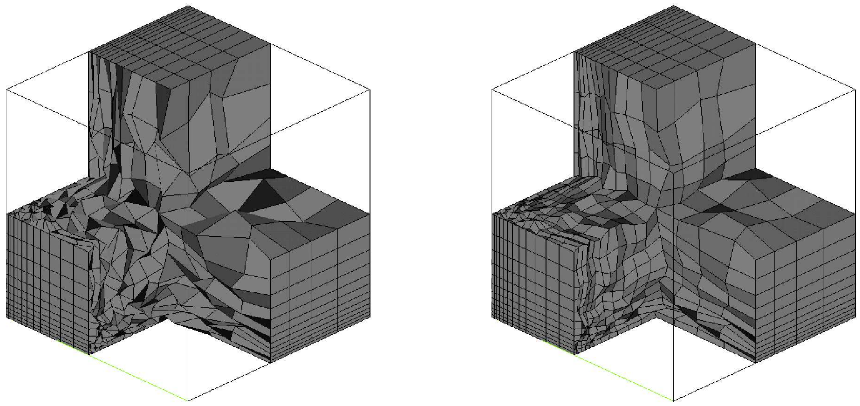
Clearly each cell, c, has an outward and inward normal on its boundary for which we will use generic notations $\mathbf{n}_c^+$ and $\mathbf{n}_c^-$, correspondingly. The outward and inward normals are different for each triangle of the corresponding boundary faces for the cell. We will use notations $\mathbf{n}_c^+(\Delta_k^f), \mathbf{n}_c^+(\Delta_k^f)$ if we are referring to particular face and triangle.

3 The Remapping Problem

3.1 Lagrangian and Rezoned Meshes

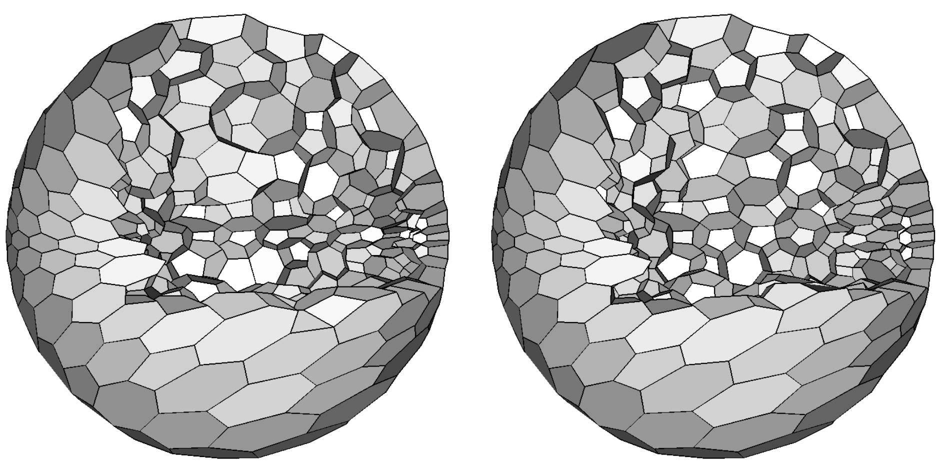
In the context of ALE methods, we will denote components and variables of the Lagrangian (old) mesh by symbols without tilde; for example, the cells of the Lagrangian mesh will be denoted by $\{c\}$. Components of the rezoned (new) mesh will be denoted by symbols with tilde; for example, cells of the new mesh will be denoted by $\{\tilde{c}\}$. The new mesh has the same connectivity as the original mesh and differs only in the positions of mesh vertices. Let us note that during Lagrangian and rezone stages of the ALE method only primary points of the faces are moved independently; positions of the face centers used to define the face geometry are enslaved to the positions of the external points as described in previous section.

In Figures 4 and 5, two pairs of Lagrangian and rezoned meshes are presented. In Fig. 4, we show a hexahedral mesh, and in Fig. 5, we show a mesh consisting of general polyhedra. The rezoned grids were generated using the Reference Jacobian Matrix based smoothing algorithm (RJM method) described in [20,13]. The goal of the RJM method is to improve the quality of the mesh while keeping the mesh as close as possible to the Lagrangian mesh. As we can see in the figures, the geometrical quality of the rezoned meshes is better



a) b)

Fig. 4. a) Lagrangian hexahedral mesh, and b) rezoned mesh obtained by using Reference Jacobian Matrix based rezoning.



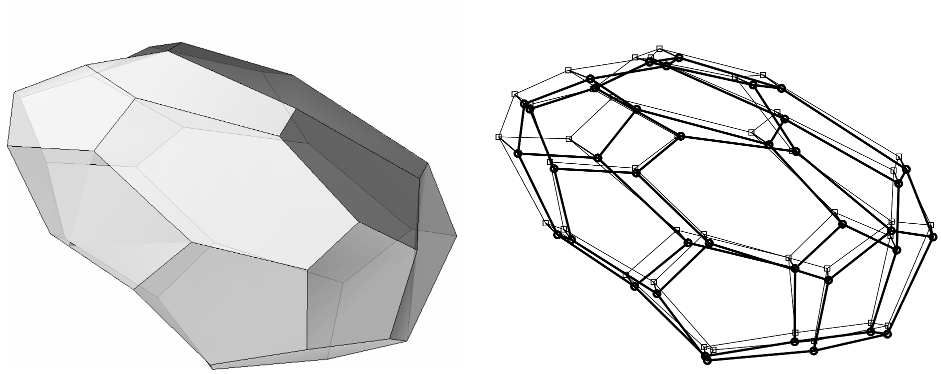
a) b)

Fig. 5. a) Lagrangian polyhedral mesh, and b) rezoned mesh obtained by using Reference Jacobian Matrix based rezoning.

than the original ones, but the nodes of the rezoned meshes stay very close to their original positions. This is very important, since the numerical error of the remapping part of the ALE algorithm depends on the relative movement of the mesh nodes.

In Fig 6 a) we present one polyhedron from polyhedral mesh presented in Fig. 5. This polyhedron has twenty one faces, which are clearly not all flat.

In Fig. 6 b) we show an overlap of the cell presented in Fig. 6 a) with its new configuration.



a) b)

Fig. 6. a) Generalized polyhedron with twenty one faces – Lagrangian mesh; b) Overlap of Lagrangian (thick line) and rezoned (thin line) cells

3.2 Statement of the Remapping Problem

We assume that there is an underlying function $g(\mathbf{r})$, $\mathbf{r} = (x, y, z)$, (in the context of ALE methods it can be density of mass, density of momentum, or density of the total energy) that is defined throughout the problem domain. For definiteness we will refer to the function $g(\mathbf{r})$ as density. The only information that we are given about this function is its mean value in each of the cells of the old grid:

$$\overline{g}(c) = \frac{\int_c g(\mathbf{r})\, dV}{V(c)}\,, \tag{5}$$

where $V(c)$ is the volume of the cell c. We will refer to the numerator of (5) as the cell mass:

$$m(c) \stackrel{def}{=} \int_c g(\mathbf{r})\, dV\,, \tag{6}$$

and therefore

$$\overline{g}(c) = \frac{m(c)}{V(c)}\,. \tag{7}$$

The total problem mass is

$$M \stackrel{def}{=} \int_\Omega g(\mathbf{r})\, dV = \sum_c \int_c g(\mathbf{r})\, dV \ = \sum_c m(c) = \sum_c \overline{g}(c)\, V(c)\,. \tag{8}$$

The problem statement is to find accurate approximations for the masses of the new cells $\hat{m}(\tilde{c})$:

$$\hat{m}(\tilde{c}) \approx m(\tilde{c}) = \int_{\tilde{c}} g(\mathbf{r})\, dV\,, \tag{9}$$

where $m(\tilde{c})$ is the unknown exact mass in the new cell. The approximate mean values of density in the new cells are defined by

$$\overline{g}(\tilde{c}) = \frac{\hat{m}(\tilde{c})}{V\,(\tilde{c})}\,. \tag{10}$$

We require that, if the underlying function $g(\mathbf{r})$ is global linear function – that is,

$$g(\mathbf{r}) = a + b\,x + c\,y + d\,z \tag{11}$$

– then our remapping process is exact, that is,

$$\hat{m}(\tilde{c}) = \int_{\tilde{c}} g(\mathbf{r})\,dV\,. \tag{12}$$

This property of the remapping process is referred to here as linearity-preservation.

We will require that remapping be conservative:

$$\tilde{M} = \sum_{c} \hat{m}(\tilde{c}) = M\,, \tag{13}$$

that is, the total mass remains the same after remapping.

Another important requirement is local bound preservation

$$\overline{g}_{\tilde{c}}^{\max} \geq \overline{g}(\tilde{c}) \geq \overline{g}_{\tilde{c}}^{\min}\,, \quad \overline{g}_{\tilde{c}}^{\max} = \max_{c' \in C(c)} \overline{g}(c')\,, \;\; \overline{g}_{\tilde{c}}^{\min} = \min_{c' \in C(c)} \overline{g}(c')\,, \tag{14}$$

which means that remapping process does not create new extrema.

4 Remapping Algorithm

Our algorithm consists of three stages:

- Piecewise-linear reconstruction of the underlying function on the original mesh.
- Approximate integration of the reconstructed function on the new grid to obtain mean values in the cells.
- Repair to ensure local bound preservation.

In the first stage, gradient of the function in all cells are computed to obtain the reconstructed piecewise linear function. This can be done using different methods, with or without limiters. During the second stage we integrate this reconstructed function to obtain new mean values on the new grid. The most natural approach is to use exact integration [26], [21], but it requires computation of all the intersections of the Lagrangian grid with the rezoned one – a time consuming process in 2D and almost infeasible in 3D. Therefore, we use a numerical quadrature method called swept area integration, which is much faster as it does not require finding any intersections. The problem, however, is that it is an approximate method, and as a result local bounds of the function may be violated or new extrema may be created. Therefore, we need a repair procedure which ensures satisfaction of the local bound preservation condition.

4.1 Stage 1 – Piecewise Linear Reconstruction

We want to reconstruct the underlying function in the form

$$g_c(\mathbf{r}) = g_c(x,y,z) = \bar{g}(c) + \sum_{\alpha=x,y,z} \left(\frac{\partial g}{\partial \alpha}\right)_c (\alpha - \alpha_c), \tag{15}$$

where

$$\alpha_c = \frac{\int_c \alpha \, dV}{V(c)} \tag{16}$$

are the coordinates of the cell centroid and

$$V(c) = \int_c 1 \, dV \tag{17}$$

is the volume of the cell. In other words, we want to compute the slopes $(\partial g/\partial \alpha)_c$ in the directions of the coordinate axes $\{\alpha\}$ in each grid cell.

4.1.1 Unlimited Slopes

In 1D, we can use just central difference as an unlimited slope. To compute an unlimited slope in 2D, we construct a contour surrounding the cell and use the Green's Theorem. In 3D, this would be too slow, because it would require the computation of the intersections of this neighborhood with the original grid. So, we use an alternative method based on least squares reconstruction of gradients [28] as described below.

Let us construct the functional

$$F\left(\left(\frac{\partial g}{\partial x}\right)_c^{\mathrm{UL}}, \left(\frac{\partial g}{\partial y}\right)_c^{\mathrm{UL}}, \left(\frac{\partial g}{\partial z}\right)_c^{\mathrm{UL}}\right) = \sum_{c' \in C(c)} \left(\bar{g}(c') - \frac{\int_{c'} g_c(\mathbf{r}) \, dV}{V(c')}\right)^2 \tag{18}$$

for each cell, which measures the sum of differences between the mean values in the neighboring cells and average values of the reconstructed function from the original cell in the same neighboring cell. The superscript "UL" in the equation indicates that these are unlimited slopes. The neighborhood $C(c)$ of cell c is defined in Section 2. We want to minimize this functional, so we want the reconstructed function to be as close to the mean values in the neighboring cells as possible.

To find the minimum we easily differentiate this functional with respect to all three variables $(\partial g/\partial \alpha)_c^{\mathrm{UL}}$ and let these derivatives be equal to zero. This gives us a linear system

$$\mathbb{A}\begin{pmatrix}\left(\frac{\partial g}{\partial x}\right)_c^{\mathrm{UL}}\\ \left(\frac{\partial g}{\partial y}\right)_c^{\mathrm{UL}}\\ \left(\frac{\partial g}{\partial z}\right)_c^{\mathrm{UL}}\end{pmatrix}=\begin{pmatrix}a_{xx} & a_{xy} & a_{xz}\\ a_{xy} & a_{yy} & a_{yz}\\ a_{xz} & a_{yz} & a_{zz}\end{pmatrix}\cdot\begin{pmatrix}\left(\frac{\partial g}{\partial x}\right)_c^{\mathrm{UL}}\\ \left(\frac{\partial g}{\partial y}\right)_c^{\mathrm{UL}}\\ \left(\frac{\partial g}{\partial z}\right)_c^{\mathrm{UL}}\end{pmatrix}=\begin{pmatrix}b_x\\ b_y\\ b_z\end{pmatrix} \tag{19}$$

for unknown partial derivatives, where

$$a_{\alpha\beta}=2\sum_{c'\in C(c)}\frac{I_\alpha^{c'}\,I_\beta^{c'}}{V(c')^2} \tag{20a}$$

$$b_\alpha=2\sum_{c'\in C(c)}\frac{I_\alpha^{c'}}{V(c')}\,\left(\bar{g}(c')-\bar{g}(c)\right) \tag{20b}$$

for $\alpha,\beta\in\{x,y,z\}$, and where the integrals $I_\alpha^{c'}$ are defined as

$$I_\alpha^{c'}=\int_{c'}(\alpha-\alpha_c)\,dV. \tag{21}$$

For solving it, we use a direct formula obtained from the inverse matrix computation. If the matrix $\mathbb{A}$ is singular then the stencil of cells can be extended to use additional levels of cells in the neighborhood. Issues related to least-squares procedures for gradient reconstruction on unstructured meshes are considered, for example, in [32], [1], [28].

We will denote linear reconstruction in the cell using unlimited slopes by

$$g_c^{\mathrm{UL}}(\mathbf{r})=\bar{g}(c)+\sum_{\alpha=x,y,z}\left(\frac{\partial g}{\partial\alpha}\right)_c^{\mathrm{UL}}\,(\alpha-\alpha_c). \tag{22}$$

4.1.2 Limited Slopes

For computation of the final slopes we limit the slopes $(\partial g/\partial\alpha)_c^{\mathrm{UL}}$ described in 4.1.1 as follows

$$\left(\frac{\partial g}{\partial\alpha}\right)_c=\Phi_c\left(\frac{\partial g}{\partial\alpha}\right)_c^{\mathrm{UL}}, \tag{23}$$

where Φ_c is Barth-Jespersen (BJ) limiter, [2].

The BJ limiter is constructed in such a way that values of the reconstructed function at the cell vertices are within the bounds defined by the maximum and the minimum of the mean values of the set $C(c)$, consisting of the cell c and its nearest neighbors. Here, we need to mention that there are some pathological situations when one needs to extend the stencil for reconstruction to be exact for a global linear function, [34].

At each cell vertex p we evaluate the limiter

$$\Phi_p = \begin{cases} \min\left(1, \frac{\bar{g}_{\tilde{c}}^{\max} - \bar{g}(c)}{g_c^{\mathrm{UL}}(p) - \bar{g}(c)}\right) & \text{for } g_c^{\mathrm{UL}}(p) - \bar{g}(c) > 0\,, \\ \min\left(1, \frac{\bar{g}_{\tilde{c}}^{\min} - \bar{g}(c)}{g_c^{\mathrm{UL}}(p) - \bar{g}(c)}\right) & \text{for } g_c^{\mathrm{UL}}(p) - \bar{g}(c) < 0\,, \\ 1 & \text{for } g_c^{\mathrm{UL}}(p) - \bar{g}(c) = 0\,, \end{cases} \tag{24}$$

and then we take their minimum as a cell limiter

$$\Phi_c = \min_{p \in P(c)} \Phi_p\,. \tag{25}$$

4.2 Stage 2 – Swept Region Integration

4.2.1 Volume Integrals of Polynomial Functions

Here we show that, for a density function $g(\mathbf{r})$, $\mathbf{r} = \{x, y, z\}$ that has bounded second derivatives, the mass in a cell (up to fifth-order in mesh size h) can be written as the sum of the surface integrals of polynomial functions over the boundaries of the cell.

Consider the Taylor series expansion for $g(\mathbf{r})$ with origin at the point $\mathbf{r}_0$

$$g(\mathbf{r}) = g(\mathbf{r}_0) + \sum_{\alpha \in \{x,y,z\}} \left(\frac{\partial g}{\partial \alpha}\right)\Bigg|_{\mathbf{r}_0} (\alpha - \alpha_0) + O(h^2)\,. \tag{26}$$

Using (26) with $\mathbf{r}_0 \in \tilde{c}$ and the definition of mass, we derive

$$\begin{aligned} m(\tilde{c}) &= \int_{\tilde{c}} g(\mathbf{r})\, dV \qquad\qquad (27) \\ &= g(\mathbf{r}_0) \int_{\tilde{c}} 1\, dV + \sum_{\alpha \in \{x,y,z\}} \left\{ \left(\frac{\partial g}{\partial \alpha}\right)\Bigg|_{\mathbf{r}_0} \cdot \int_{\tilde{c}} (\alpha - \alpha_0)\, dV \right\} + O(h^5)\,. \end{aligned}$$

Now, using the divergence theorem, we can express the volume integrals of linear functions participating in (27) as surface integrals over the boundary of $\tilde{c}$. For example,

$$\int_{\tilde{c}} 1\, dV = \int_{\tilde{c}} \mathrm{div}\ \mathbf{\Phi}_1\, dV = \oint_{\partial\tilde{c}} \mathbf{\Phi}_1 \cdot \mathbf{n}_{\tilde{c}}^{+}\, dS\,, \tag{28}$$

where $\mathbf{n}_{\tilde{c}}^{+}$ is outward normal to $\partial\tilde{c}$ and $\mathbf{\Phi}_1 = \frac{1}{3} \cdot (x, y, z)^T$. Similarly for $\int_{\tilde{c}} x\, dV$ we use

$$\int_{\tilde{c}} x\, dV = \oint_{\partial\tilde{c}} \mathbf{\Phi}_x \cdot \mathbf{n}_{\tilde{c}}^{+}\, dS\,, \quad \mathbf{\Phi}_x = \frac{1}{4} \cdot (x^2, x\,y, x\,z)^T\,, \tag{29}$$

and so on. For more details about integration of polynomial function see the Appendix.

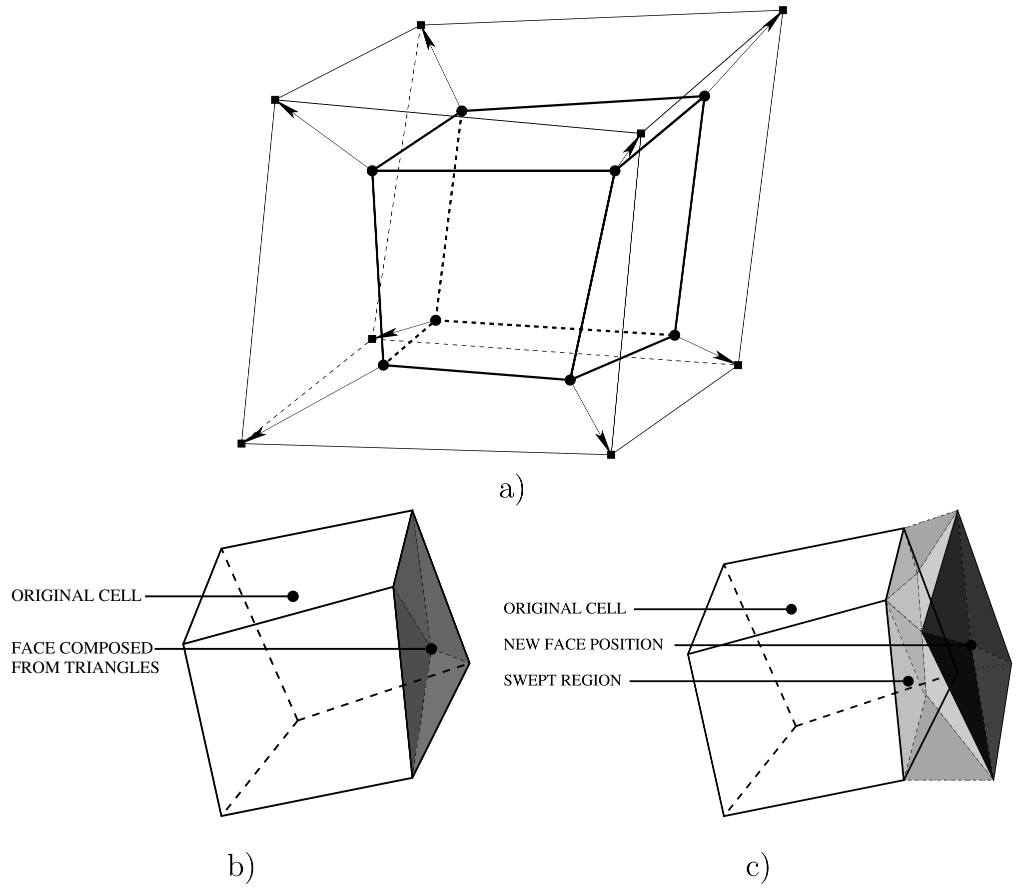


Fig. 7. Swept regions: a) Original cell is presented in thick lines, and new cell is presented in thin lines, arrows show movement of the vertices; b) one face of generalized cube composed of the four triangles; c) swept region.

4.2.2 Swept Regions

Let us introduce notion of swept region for the face f of cell c. The swept region is a generalized prism, which is formed by face f itself (bottom base), and corresponding face $\tilde{f}$ of the new cell $\tilde{c}$ (top base). The side faces of the prism are generalized quadrilaterals formed by a pair of corresponding edges of f and $\tilde{f}$ and straight line segments connecting old and new positions of the vertices corresponding to these edges. In Fig. 7 we demonstrate the notion of swept region for a generalized cube. In the case presented in Fig. 7c, the right face moves outward from the original cell and the region in-between the original and new positions of the face is the swept region. In fact, all faces can move in different ways and swept regions can be tangled.

We will denote the swept region corresponding to face f by R_f.

Now let us recall that each face f has an independently defined normal $\mathbf{n}^f$ as described in Section 2. We will define the *signed* volume of the swept region

R_f to face f as follows

$$V(R_f) = \oint_{\partial R_f} \mathbf{\Phi}_1 \cdot \mathbf{n}^f_{R_f} \, dS \,. \tag{30}$$

Here $\mathbf{n}^f_{R_f}$ is the normal to boundary of R_f (outward or inward) which is consistent with $\mathbf{n}^f$, that is, on face f it coincides with $\mathbf{n}^f$

$$\mathbf{n}^f_{R_f}(f) = \mathbf{n}^f \,. \tag{31}$$

Depending on the sign of signed volume of the swept region we can say that the most part of the swept region lies in the left cell or the right cell with respect to face f.

4.2.3 Flux Form of Surface Integral

Using the notion of a swept region, the surface integral of a vector, $\mathbf{\Phi}$, over the boundary of cell $\tilde{c}$ can be written as follows

$$\oint_{\partial \tilde{c}} \mathbf{\Phi} \cdot \mathbf{n}^+_{\tilde{c}} \, dS = \oint_{\partial c} \mathbf{\Phi} \cdot \mathbf{n}^+_c \, dS + \sum_{f \in F(c)} \oint_{\partial R_f} \mathbf{\Phi} \cdot \mathbf{n}^+_{R_f} \, dS \,. \tag{32}$$

We will define flux corresponding to face f as follows

$$\mathcal{F}_f = \oint_{\partial R_f} \mathbf{\Phi} \cdot \mathbf{n}^f_{R_f} \, dS \,. \tag{33}$$

Using definition (33) we can write

$$\oint_{\partial R_f} \mathbf{\Phi} \cdot \mathbf{n}^+_{R_f} \, dS = -\,\mathrm{sign}\{\mathbf{n}^f \cdot \mathbf{n}^+_c(f)\} \cdot \mathcal{F}_f \,, \tag{34}$$

where $\mathbf{n}^+_c(f)$ is normal to the face f corresponding to outward normal to boundary of cell c.

Using formula (34) we can rewrite expression (32) as follows

$$\oint_{\partial \tilde{c}} \mathbf{\Phi} \cdot \mathbf{n}^+_{\tilde{c}} \, dS = \oint_{\partial c} \mathbf{\Phi} \cdot \mathbf{n}^+_c \, dS - \sum_{f \in F(c)} \mathrm{sign}\{\mathbf{n}^f \cdot \mathbf{n}^+_c(f)\} \cdot \mathcal{F}_f \,. \tag{35}$$

4.2.4 Conservative Swept Region Quadrature

Considerations in previous Sections 4.2.1, 4.2.2, and 4.2.3 suggest the following conservative swept region quadrature for computation of mass of new cell

$$\hat{m}(\tilde{c}) = m(c) - \sum_{f \in F(c)} \mathrm{sign}\{\mathbf{n}^f \cdot \mathbf{n}_c^+(f)\} \, \hat{\mathcal{F}}_f \,, \tag{36}$$

where $\hat{m}(\tilde{c})$ is approximation of the mass of the new cell, and $\hat{\mathcal{F}}_f$ are approximate mass fluxes, whose computation is explained later in this section. Clearly this formula guarantees conservation of mass because it is in flux form (note that signs of $\mathbf{n}^f \cdot \mathbf{n}_c^+(f)$ are opposite to each other for left and right cells for given face f).

Approximate mass flux is computed using linear reconstruction on the old mesh as follows

$$\hat{\mathcal{F}}_f = \int_{R_f} g_f(\mathbf{r}) \, dV \,, \tag{37}$$

where

$$g_f(\mathbf{r}) = \begin{cases} g_{c_L}(\mathbf{r}) \,, \; V(R_f) \geq 0 \,, \\ g_{c_R}(\mathbf{r}) \,, \; V(R_f) < 0 \,. \end{cases} \tag{38}$$

The volume integral of linear function in formula (37) is understood in the sense of corresponding surface integral as explained in the Section 4.2.1.

The meaning of formulas (37), (38) is that the flux corresponding to face f is computed from reconstruction in the left or right cell depending on the sign of the signed volume of the swept region R_f. In other words, the flux is computed from the cell containing most of the swept region.

In the appendix, we explain how to analytically compute surface integrals of the form $\oint_{\partial c} \mathbf{\Phi} \cdot \mathbf{n} \, dS$ over polyhedra when the components of vector $\mathbf{\Phi}$ are polynomial functions.

The new mean value of function g is computed as follows:

$$\bar{g}(\tilde{c}) = \frac{\hat{m}(\tilde{c})}{V(\tilde{c})} \,. \tag{39}$$

Analysis of the accuracy of swept region integration can be performed similar to the 2D case as presented in [26]. For a smooth function it can be shown that $\bar{g}(\tilde{c})$ is second-order accurate and is exact if $g(\mathbf{r})$ is a global linear function.

As it was noticed before, the local bounds for $\bar{g}(\tilde{c})$ can be violated due to approximate character of the swept region integration. So the third stage is necessary to enforce local bound preservation.

4.3 Stage 3 – Repair

The goal of the repair stage is to conservatively redistribute mass in such a way that mean values on the new mesh satisfy local bounds as stated in (14).

Here we demonstrate how repair works when the lower bound is violated

$$\bar{g}(\tilde{c}) < \bar{g}_{\tilde{c}}^{\min}\,; \tag{40}$$

the upper bound is handled similarly. At first we compute the mass, which is needed in the cell $\tilde{c}$ to bring the mean value back to the lower bound

$$\delta m_{\tilde{c}}^{\text{needed}} = (g_c^{\min} - \bar{g}(\tilde{c}))\,V(\tilde{c})\,. \tag{41}$$

Because our method has to be conservative, this needed mass has to be taken from neighboring cells. We start the search for the mass in the immediate neighborhood. First we need to check how much total mass can be taken from all cells in the neighborhood without violating their lower bound. Therefore for each cell in the neighborhood we compute available mass

$$\delta m_{\tilde{c}'}^{\text{avail}} = \max\Big((\bar{g}(\tilde{c}') - \bar{g}_{\tilde{c}'}^{\min})\,V(\tilde{c}), 0\Big)\,, \tag{42}$$

which can safely be taken from the cell without violating the local bound also. The total available mass in the neighborhood is

$$\delta m_{C(c)}^{\text{avail}} = \sum_{c' \in C(c)} \delta m_{\tilde{c}'}^{\text{avail}}\,. \tag{43}$$

If there is not enough available mass ($\delta m_{C(c)}^{\text{avail}} < \delta m_{\tilde{c}}^{\text{needed}}$), we extend the stencil and look for the available mass in cells which are neighbors of the neighbors. If there is enough mass available, we perform the repair. We bring the value in the cell $\tilde{c}$ to its lower bound

$$m'(\tilde{c}) = \bar{g}_{\tilde{c}}^{\min}\,V(\tilde{c}) \tag{44}$$

and we take the mass from the neighborhood cells proportionally to the mass available in these cells

$$m'(\tilde{c}') = m(\tilde{c}') - \frac{\delta m_{\tilde{c}'}^{\text{avail}}}{\delta m_{C(\tilde{c})}^{\text{avail}}}\,\delta m_{\tilde{c}}^{\text{needed}}\,. \tag{45}$$

In [21] we proved that this algorithm succeeds in a finite number of steps and the repair stage corrects all local bound violations.

There are other variants of repair algorithm that can be found in [33], [22].

5 Numerical Tests

Our remapping algorithm is intended to be coupled with a rezoner, and used in the context of ALE simulations. However it is instructive to test it in a simpler environment, where there are no partial differential equations, nor any Lagrangian algorithm. We will test the remapper in the context of interpolation. That is, we will choose an underlying function, prescribe a grid motion, and compare the exact integrals of this function on the new grids to the numerical simulations.

For completeness, we also present comparison of the results of our remapping method with the results obtained by variants of the method where we use reconstruction with unlimited slopes and do not perform repair.

5.1 Cyclic remapping

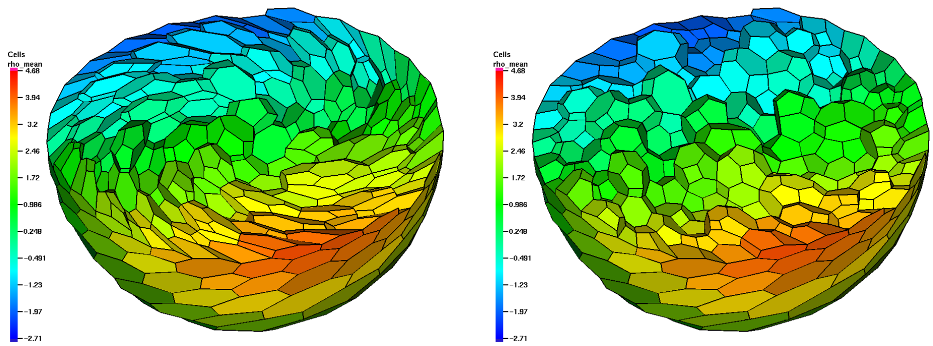
Here we assume that we have a sequence of grids $\{c\}_k^n$, $k = 1, \ldots k_{\max}$, $n = 0, \ldots n_{\max}$, where the subscript k is index which identifies a cell on the grid and the superscript identifies a particular grid. It is convenient to think of the index n as representing a fictitious time t^n, which is a parameter to define the grid motion. We begin with a test function $g(\mathbf{r})$ and compute its mean values on grid $\{c\}_k^0$, remap the mean values of the function from grid $\{c\}_k^0$ to grid $\{c\}_k^1$, and then remap the resulting values from grid $\{c\}_k^1$ to grid $\{c\}_k^2$, etc. This allows us to look at the cumulative effects of many remappings.

Let us note that in our real calculations, instead of computing integral averages over cells we just compute its value at the center of mass, which is equal to the integral average with second-order accuracy for smooth function $g(\mathbf{r})$.

5.2 Reference-Jacobian Movement of Polyhedral Mesh

In first series of experiments we use meshes obtained by using Reference Jacobian Matrix (RJM) method, [13], [20]. Recall that the goal of the RJM method is to produce mesh which as close as possible to Lagrangian mesh but smoother.

In our experiments we start with a twisted polyhedral mesh consisting of about 700 various polyhedra in unit sphere with its center at the origin, Fig. 8 a). The next mesh is obtained by 10 consecutive application of RJM method. The mesh resulting from 180 consecutive applications of RJM algorithm is presented in Fig. 8 b). As result we have 19 meshes (original twisted mesh and



a) b)

Fig. 8. RJM mesh smoothing and color map for linear function (46): a) top view of the meshes in unit sphere whose centers are below plane $z = 0$ and color map of linear function, b) the same view of the meshes after 180 applications of RJM method – 18-th mesh in sequence and color map of linear function after 18 remappings.

18 smoothed ones). Our cycling remapping consists of consecutive remapping (in 18 steps) from the mesh presented in Fig. 8 a) to the mesh presented in Fig. 8 b) and then another 18 steps in reverse, back to the original mesh..

5.2.1 Linear Function

The first function is the linear function

$$g(x, y, z) = 1 + 3\,x + y + 2\,z. \tag{46}$$

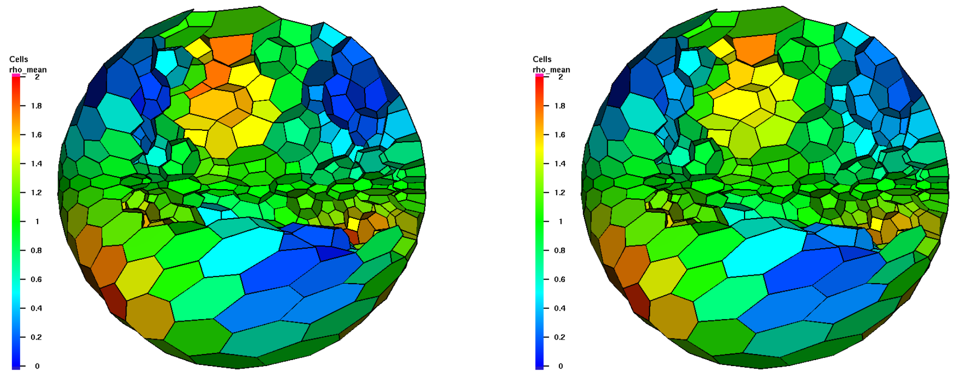
Our numerical experiment confirms that our algorithm is exact for linear function. In Fig. 8 we present a color map for linear function on the original mesh and the 18-th mesh (maximal smoothing) in the sequence of meshes.

5.2.2 Smooth Sine Function

The second function, which we use for the verification of algorithm properties, is the smooth sine function defined as follows

$$g(x, y, z) = 1 + \sin(\pi\,x)\,\sin(\pi\,y)\,\sin(\pi\,z). \tag{47}$$

In Fig. 9 a), we present the color map for sine function $g(x, y, z)$ on the original mesh, where values are computed from using Eq. 47 (initial data). In the Fig. 9 b), we present the color map for results of cycling remapping (36 remapping) on a final mesh which coincides with initial mesh. The result of the remapping process presented in Fig. 9 b) (as well as all remapping results presented in all the other Figures), is obtained by method with limited slopes using Barth-Jespersen limiter and repair.



a) b)

Fig. 9. Remapping of the smooth sine function. Color map of initial data – a); and result of cycling remapping on the original mesh. For visualization purposes we have removed cells, which have their centroids inside the box $\langle -0.32, 1\rangle^3$.

	L_1	$\mathrm{L}_{\max}$	$g_{\min}$	$g_{\max}$
unlimited, no repair	4.68E-2	2.02E-1	2.74E-2	1.99E0
unlimited, repair	4.70E-2	2.02E-1	7.30E-2	1.96E0
BJ limiter, no repair	4.83E-2	2.24E-1	7.30E-2	1.93E0
BJ limiter, repair	4.83E-2	2.24E-1	7.30E-2	1.93E0

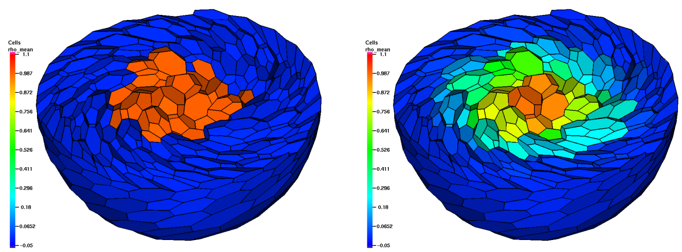
Table 1
The L_1 and $\mathrm{L}_{\max}$ errors and global final extrema of the smooth sine function.

In Table 1, we present the errors for different methods with respect to slopes reconstruction methods as well as using or not using repair. As we can see, there are no big differences in the errors, but the limited version with repair stage has the advantage of local-bound preservation, which is not guaranteed by other methods. In two last column we also present minimal and maximal values of the discrete functions after remapping. In this particular example of smooth sine function, repair works only when we are using unlimited slopes.

5.2.3 Discontinuous Color Function

To show the ability of the algorithm to deal with the discontinuous function, we present here the example of color function, which has the value 1 inside a spherical region of radius 1/4, and the value of zero outside of this region. In Fig. 10, we present the color map for the initial and remapped discrete functions.

In Table 2, we present the errors for different methods. We can see again, that the values of the errors are comparable. The discontinuous behavior of the function causes many repairs in the case of unlimited reconstruction,



a) b)

Fig. 10. Color map for discontinuous color function: a) Initial discrete function; b) Discrete function after 36 remapping.

	L_1	L_{max}	g_{min}	g_{max}
unlimited, no repair	7.99E-2	5.69E-1	-4.37E-2	1.10E0
unlimited, repair	7.01E-2	5.76E-1	0.00E0	1.00E0
BJ limiter, no repair	8.13E-2	6.29E-1	0.00E0	9.87E-1
BJ limiter, repair	8.13E-2	6.29E-1	0.00E0	9.87E-1

Table 2
L_1 and L_{max} errors and global final extrema of the discontinuous color function.

which almost doubles the computational time when compared with the Barth-Jespersen limited reconstruction. When using the unlimited variants, the local bound preservation condition is not satisfied. This example clearly shows the importance of the limiting process and the repair stage.

5.3 Sine Movement of Cubic Mesh

To demonstrate convergence properties of our algorithm we consider movement of originally cubic meshes in $\langle 0,1\rangle^3$. In this case we easily can refine the mesh.

For this type of a mesh, we introduce a "sine movement" where the position of the kth node is computed as:

$$x_k^n = x_k^0 + \alpha(t^n)\,\sin(2\,\pi\,x_k^0)\,\sin(2\,\pi\,y_k^0)\,\sin(2\,\pi\,z_k^0)\,, \tag{48}$$
$$y_k^n = y_k^0 + \alpha(t^n)\,\sin(2\,\pi\,x_k^0)\,\sin(2\,\pi\,y_k^0)\,\sin(2\,\pi\,z_k^0)\,, \tag{49}$$
$$z_k^n = z_k^0 + \alpha(t^n)\,\sin(2\,\pi\,x_k^0)\,\sin(2\,\pi\,y_k^0)\,\sin(2\,\pi\,z_k^0)\,, \tag{50}$$

the coefficient $\alpha(t)$ is defined as follows

$$\alpha(t) = \begin{cases} t/5 & \text{for} \qquad t \le 1/2\,, \\ (1-t)/5 & \text{for} \qquad t > 1/2\,, \end{cases} \tag{51}$$

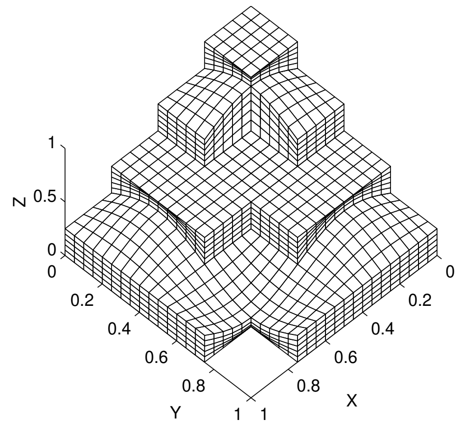


Fig. 11. Sine movement of logically rectangular mesh. The fragment of the mesh, obtained by removing some cells at $t^n = 1/2$, is shown.

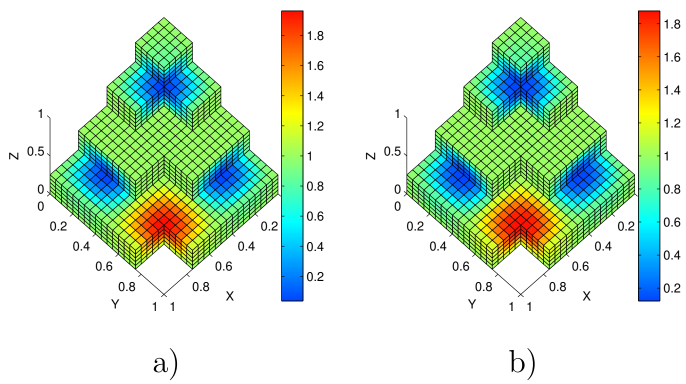


Fig. 12. Remapping of the smooth sine function: a) original discrete function; b) remapped discrete function.

and t^n has the meaning of the fictitious time in the nth remapping step, $t^n = n/n_{\max}$. Note that at $t^{n_{\max}} = 1$ mesh returns to its initial position.

The mesh at time $t^n = 1/2$ is presented the Fig. 11. We want to note here that cells obtained by such movement are generalized hexahedra with non-flat faces.

5.3.1 Smooth Sine Function

Because the computational domain is the unit cube, we use the following sine function

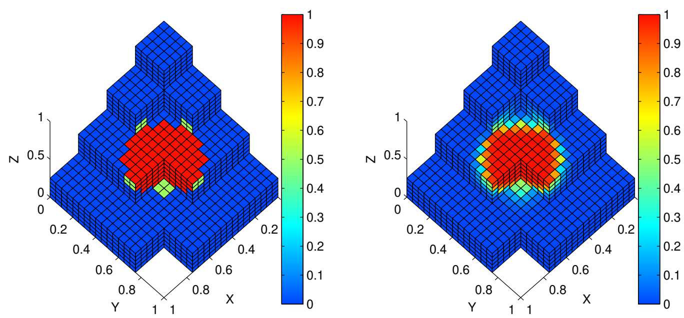
$$g(x, y, z) = 1 + \sin(2\,\pi\,x)\,\sin(2\,\pi\,y)\,\sin(2\,\pi\,z).$$

The original discrete function and result of remapping on initial mesh is shown in Fig. 12.

The order of convergence is demonstrated in Table 3, where we present L_1 and $L_{\max}$ norms of the errors. Note that number of time steps increases when

Size of grid	$n_{\max}$	L$_1$ error	L$_{\max}$ error
10^3	10	4.06E-2	2.92E-1
20^3	20	7.70E-3	8.98E-2
40^3	40	1.15E-3	3.78E-2
80^3	80	1.65E-4	1.49E-2
160^3	160	2.25E-5	5.55E-3

Table 3
Numerical errors of the smooth sine function for sine movement of the mesh.



a) b)

Fig. 13. Remapping of the discontinuous color function: a) original discrete function; b) remapped discrete function.

Size of grid	$n_{\max}$	L$_1$ error
10^3	10	2.27E-2
20^3	20	1.96E-2
40^3	40	1.20E-2
80^3	80	7.46E-3
160^3	160	4.58E-3

Table 4
Numerical errors of the color function over for sine movement of the mesh.

we increase the resolution. It is needed to guarantee that two consecutive meshes are close to each other. This table suggests that algorithm is at least second-order accurate for this example.

5.3.2 Discontinuous Color Function

Here we present results for discontinuous color function. Color maps are presented in Fig. 13 and convergence study is presented in Table 4. We present only L_1 in this case. The algorithm preserves local extremes and Table 4 suggests that convergence rate is first-order for this discontinuous function.

Size of grid	$n_{\max}$	L_1 error	$L_{\max}$ error
10^3	10	3.93E-2	1.41E-1
20^3	20	9.07E-3	6.45E-2
40^3	40	1.64E-3	2.56E-2
80^3	80	2.79E-4	9.86E-3
160^3	160	4.97E-5	3.73E-3

Table 5
Numerical errors of the smooth sine function over a series of randomly moving grids.

Size of grid	$n_{\max}$	L_1 error
10^3	10	3.20E-2
20^3	20	2.59E-2
40^3	40	1.57E-2
80^3	80	9.60E-3
160^3	160	5.76E-3

Table 6
Numerical errors of the color function over a series of randomly moving grids.

5.4 Random Mesh Movement

The last series of calculations is performs for so-called random mesh movement. In every time step, the new mesh is obtained from the original orthogonal cubical mesh by small random movement of all nodes. By small, we mean, that the node cannot move more than a half of the length of the edge of the cubical mesh.

We present convergence analysis for the smooth sine function (Table 5) and the discontinuous color function (Table 6). The numerical errors in this case have the same qualitative behavior as for the other types of mesh movement.

6 Conclusion

It this paper, we have constructed an algorithm for efficient linearity and bound preserving conservative interpolation (remapping) on generalized polyhedral meshes. We have presented a series of numerical examples to verify the theoretical properties of our algorithm and demonstrate its performance.

7 Acknowledgments

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). R. Garimella and M. Shashkov acknowledge the partial support of DOE's Advanced Simulations and Computing (ASC). M. Kuchařík also acknowledges the partial support of the Czech Grant Agency grant GACR 202/03/H162 and Czech Ministry of Education grants FRVS 1987G1/2005, MSM 6840770022, MSM 6840770010, and LC528. The authors thank L. Margolin, B. Wendroff, B. Swartz, R. Liska, M. Berndt and V. Dyadechko for fruitful discussions and constructive comments.

A Appendix: Integration of Polynomial Function over Arbitrary Polyhedron

In this Appendix, we briefly discuss the problem of integration of a linear function over an arbitrary polyhedron, which is needed for the computation of cell volumes $V(c')$ (17), coordinates of centroids of cells $\alpha(c)$ (16), and integrals I_α^c in the functional (18). This algorithm is also used for the computation of the volumes of the swept regions, (30), and mass fluxes, (37). Our presentation is based on the results from [29].

Let us first demonstrate the integration procedure on the example of the computation of the cell volume. The cell volume can be written in the form

$$V(c) = \int_c 1\, dV = \int_c \operatorname{div} \mathbf{\Phi}_1 \, dV \ = \oint_{\partial c} \mathbf{\Phi}_1 \cdot \mathbf{n}_c^+ \, dS \,, \quad \mathbf{\Phi}_1 = \frac{1}{3} \cdot (x, y, z)^T, \tag{A.1}$$

The surface integral can be represented as sum of the face integrals, which in turn are sums of integrals over corresponding triangles. Therefore, the previous expression for volume can be rewritten as follows

$$V(c) = \frac{1}{3} \sum_{f \in F(c)} \sum_{\Delta \in T(f)} \sum_{\kappa = x,y,z} n_c^{+,\kappa}(\Delta) \int_\Delta \kappa \, dS, \tag{A.2}$$

where $n_c^{+,x}$, $n_c^{+,y}$, $n_c^{+,z}$, are components of the normal $\mathbf{n}_c^+(\Delta)$ to triangle Δ, whose orientation is consistent with outward normal to boundary of cell c.

To evaluate the integrals $\int_\Delta \kappa \, dS$, $\kappa = x,\, y,\, z$, we project each triangle to one of the coordinate plane as described in [29]. Suppose, that triangle Δ belongs to the plane $n_c^{+,x}(\Delta)\, x + n_c^{+,y}(\Delta)\, y + n_c^{+,z}(\Delta)\, z + \omega = 0$, where $\omega = -\mathbf{n}_c^+ \cdot \mathbf{p}$

and $\mathbf{p}$ is arbitrary point in the plane (for example, one of the vertices of the triangle). To reduce the error of computations, for the given triangle Δ we choose $\alpha - \beta - \gamma$ as right-handed permutation of $x - y - z$ coordinates, such that $|n_c^{+,\gamma}|$ is maximized. If we denote projection of triangle Δ to $(\alpha,\, \beta)$ plane by Π_Δ, then integrals over Δ can be reduced to integrals over Π_Δ as follows

$$\begin{aligned} &\int_\Delta \alpha\, dS = |n_c^{+,\gamma}|^{-1}\, J_\alpha^{\Pi_\Delta}, \quad \int_\Delta \beta\, dS = |n_c^{+,\gamma}|^{-1}\, J_\beta^{\Pi_\Delta}, \\ &\int_\Delta \gamma\, dS = -|n_c^{+,\gamma}|^{-1}\, (n_c^{+,\gamma})^{-1}\, (n_c^{+,\alpha}\, J_\alpha^{\Pi_\Delta} + n_c^{+\,\beta}\, J_\beta^{\Pi_\Delta} + \omega\, J_1^{\Pi_\Delta}), \end{aligned} \tag{A.3}$$

where the integral $J_g^{\Pi_\Delta}$ is defined as $J_g^{\Pi_\Delta} = \int_{\Pi_\Delta} g\, d\alpha\, d\beta$.

Finally, Green's Theorem in $(\alpha,\, \beta)$ plane is used to reduce integrals $\int_{\Pi_\Delta} g\, d\alpha\, d\beta$ of polynomial function g to 1D integrals over the edges.

Computation of integrals of arbitrary linear functions follows the same path. The only difference is that at the final stage one needs to compute 1D integrals of higher degree polynomial function, which still can be done explicitly.

References

[1] M. Aftosmis, D. Gaitonde, T. S. Tavares, *Behavior of Linear Reconstruction Techniques on Unstructured Meshes*, AIAA Journal, **33** (1995), pp. 2038–2049.

[2] T. J. Barth, *Numerical Methods for Gasdynamics Systems on Unstructured Grids*, in Lecture Notes in Computational Science and engineering, "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, D. Kroner, M. Ohlberger, C. Rohde (Eds.) Springer, Berlin (1997), pp. 195–285.

[3] M. W. Beall and M. S. Shephard, *A general topology-based mesh data structure*, International Journal for Numerical Methods in Engineering, **40(9)**:1573–1596, May 1997.

[4] D. J. Benson, *Computational Methods in Lagrangian and Eulerian Hydrocodes*, Computer Methods in Applied Mechanics and Engineering, **99** (1992), pp. 235–394.

[5] D. J. Benson, *Momentum Advection on a Staggered Grid*, Journal of Computational Physics, **100** (1992), pp. 143–162.

[6] D. J. Benson, *An Efficient, Accurate, Simple ALE Method for Nonlinear Finite Element Programs*, Computer Methods in Applied Mechanics and Engineering, **72** (1989), pp. 305–350.

[7] D. E. Burton, *Multidimensional Discretization of Conservation Laws for Unstructured Polyhedral Grids*, Report UCRL-JC-118306, Lawrence Livermore National Laboratory, 1994.

[8] D. E. Burton, *Consistent Finite-Volume Discretization of Hydrodynamics Conservation Laws for Unstructured Grids*, Report UCRL-JC-118788, Lawrence Livermore National Laboratory, 1994.

[9] J. Donea, S. Giuliani, and J. P. Halleux, *An Arbitrary Lagrangian-Eulerian Finite Element Method for Transient Dynamic Fluid-Structure Interactions*, Computer Methods in Applied Mechanics and Engineering, **33** (1982), pp. 689–723.

[10] J. K. Dukowicz and J. W. Kodis, *Accurate Conservative Remapping (Rezoning) for Arbitrary Lagrangian-Eulerian Computations*, SIAM Journal of Scientific and Statistical Computing, **8** (1987), pp. 305–321.

[11] J. K. Dukowicz, M. C. Cline and F. L. Addessio, *A General Topology Godunov Method*, Journal of Computational Physics, **82** (1989), pp. 29–63.

[12] J. Dukowicz and N. Padial, *REMAP3D: A Conservative Three-Dimensional Remapping Code*, LA-12136-MS, Report of Los Alamos National Laboratory, Los Alamos, NM, USA, 1991.

[13] V. Dyadechko, R. Garimella, and M. Shashkov, *Reference Jacobian Rezoning Strategy for Arbitrary Lagrangian-Eulerian Methods on Polyhedral Meshes*, Proceedings, 13th International Meshing Roundtable, Williamsburg, VA, Sandia National Laboratories, SAND #2004-3765C, pp. 459–470, 2004.

[14] R. Garimella, *Mesh Data Structure Selection for Mesh Generation and FEA Applications*, International Journal for Numerical Methods in Engineering, **55** (2002), pp. 451–478.

[15] J. Grandy, *Conservative Remapping and Regions Overlays by Intersecting Arbitrary Polyhedra*, Journal of Computational Physics, **148** (1999), pp. 433–466.

[16] C. Hirt, A. Amsden and J. Cook, *An Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds*, Journal of Computational Physics, **14** (1974), pp. 227–253, reprinted in **135** (1997), pp. 203–216.

[17] C. M. Hoffmann, *Geometric and Solid Modeling: An Introduction*. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann Publications, 1989.

[18] D. S. Kershaw, M. K. Prasad, M. J. Shaw, and J. L. Milovich, *3D Unstructured Mesh ALE Hydrodynamics with the Upwind Discontinuous Finite Element Method*, Computer Methods in Applied Mechanics and Engineering, **158** (1998), pp. 81–116.

[19] P. Kjellgren and J. Hyvarien. *An Arbitrary Lagrangian-Eulerian Finite Element Method*, Computational Mechanics, **21** (1998), pp. 81–90.

[20] P. Knupp, L. Margolin, and M. Shashkov, *Reference Jacobian Optimization-Based Rezone Strategies for Arbitrary Lagrangian-Eulerian Methods*, Journal of Computational Physics, **176** (2002), pp.93–128.

[21] M. Kucharik, M. Shashkov, and B. Wendroff, *An Efficient Linearity-and-Bound-Preserving Remapping Methods*, Journal of Computational Physics, **188** (2003), pp. 462–471.

[22] R. Loubere, M. Staley, and B. Wendroff, *The Repair Paradigm: New Algorithms and Applications to Compressible Flow*, LA-UR-04-7095, Report of Los Alamos National Lab., Journal of Computational Physics, to appear.

[23] R. Loubere and M. Shashkov,*A Subcell Remapping Method on Staggered Polygonal Grids for Arbitrary Lagrangian Eulerian Methods,* Journal of Computational Physics, **209** (2005), pp. 105–138.

[24] H. U. Mair, *Review: Hydrocodes for Structural Response to Underwater Explosions*, Shock and Vibration, **6** (1999), pp. 81–96.

[25] L. G. Margolin, *Introduction to "An Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds"*, Journal of Computational Physics, **135** (1997), pp. 198–202.

[26] L. G. Margolin and M. Shashkov, *Second-order Sign-preserving Conservative Interpolation (Remapping) on General Grids*, Journal of Computational Physics, **184** (2003), pp. 266–298.

[27] L. G. Margolin and M. Shashkov, *Remapping, Recovery and Repair on Staggered Grid*, Computer Methods in Applied Mechanics and Engineering, **193** (2004), pp. 4139–4155.

[28] D. J. Mavriplis, *Revisiting the Least-squares Procedure for Gradient Reconstruction on Unstructured Meshes*, National Institute of Aerospace, Report No. 2003-06, NASA/CR-2203-212683.

[29] B. Mirtich, *Fast and Accurate Computation of Polyhedral Mass Properties*, Journal of Graphics Tools, **1** (1996), pp. 31–50.

[30] M. Mortenson, *Geometric Modeling*, J. Wiley and Sons, New York, 1985.

[31] J. S. Peery and D. E. Carroll, *Multi-Material ALE Methods in Unstructured Grids*, Computer Methods in Applied Mechanics and Engineering, **187**, (2000), pp. 591–619.

[32] N. B. Petrovskaya, *The Impact of Grid Cell Geometry on the Least-Squares Gradient Reconstruction*, Preprint of Institute of Applied Mathematics, #21, 2003, Russian Academy of Sciences, Moscow, Russia.

[33] M. Shashkov and B. Wendroff, *The Repair Paradigm and Application to Conservation Laws*, Journal of Computational Physics, **198** (2004), pp. 265–277.

[34] B. Swartz, *Good Neighborhoods for Multidimensional Van Leer Limiting*, Journal of Computational Physics, **154** (1999), pp. 237–241.

[35] T. J. Tautges, R. E. Meyers, K. Merkley, Clint Stimpson, and Corey Ernst, *SMOAB: A Mesh-Oriented Data Base T*, Sandia National Laboratories Report SAND2004-1592, Sandia National Laboratories, Albuquerque, New Mexico, 2004.