Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 1 (2010) 1879–1886

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

Using the feasible set method for rezoning in ALE

Markus Berndt

Los Alamos National Laboratory
Computational Physics and Methods Group, Mail Stop D413
Los Alamos, NM 87545, U.S.A.

Milan Kucharik

Department of Physical Electronics
Faculty of Nuclear Sciences and Physical Engineering
Czech Technical University in Prague
Brehova 7, Praha 1, 115 19, Czech Republic

Mikhail J. Shashkov

Los Alamos National Laboratory
Applied Mathematics and Plasma Physics Group, Mail Stop B284
Los Alamos, NM 87545, U.S.A.

Abstract

One of the steps in the Arbitrary Lagrangian Eulerian (ALE) algorithm is the improvement of the quality of the computational mesh. This step, commonly referred to as rezoning, is essential for maintaining a mesh that does not become invalid during a simulation. In this paper, we present a new robust and computationally efficient 2D mesh relaxation method. This feasible set method is a geometric method for finding the convex polygon that represents the region of coordinates that a vertex in a mesh can occupy while the mesh around it remains valid. After the feasible set has been computed for a vertex in a mesh, a new vertex location can be chosen that lies inside this feasible set. As a result, the mesh after relaxation is guaranteed to be valid. We present an example ALE simulation, that highlights the robustness of the feasible set method when used as a rezoning method in ALE.

Keywords: mesh smoothing, arbitrary Lagrangian Eulerian method, ALE, rezoning, feasible set

1. Introduction

Arbitrary Eulerian Lagrangian (ALE) methods are popular for the simulation of continuum mechanics problems with large shear deformation. These ALE methods [1, 2] consist of a Lagrangian step in which the mesh nodes move along with the fluid motion and a rezone step in which the mesh is modified to improve its quality, and a remap step

Email addresses: berndt@lanl.gov (Markus Berndt), kucharik@newton.fjfi.cvut.cz (Milan Kucharik), shashkov@lanl.gov (Mikhail J. Shashkov) in which the solution is transferred from the old mesh to the new improved mesh. Many mesh improvement methods generate rather small vertex displacements when the mesh is nearly invalid. This can eventually lead to a situation, where the rezone step cannot keep up with the mesh motion that is the result of the Lagrangian steps, and the mesh tangles. Mesh untangling is an approach that can be taken to repair such a situation. However, this does not address the question of how to remap physical quantities from a tangled mesh to a repaired mesh.

Several researchers have investigated the problem of untangling unstructured meshes by node repositioning [3, 4]. Freitag and Plassmann [5, 6] untangle meshes by optimization of a local function based on maximizing the minimum element area at each mesh vertex. Knupp [7] performs a global optimization of the difference between the absolute and signed values of element volumes in order to untangle the mesh. Kovalev et al. [4] visit each vertex connected to at least one invalid element and reposition the vertex directly to a point in its feasible set (or kernel) to make all connected elements valid. They define the feasible set of a vertex to be the set of all locations of the vertex for which all elements connected to the vertex will be valid, however, in the process of finding a new location for a tangled vertex, this feasible set is not explicitly calculated. The approach presented in [8], is based on calculating the feasible set explicitly, and not just an approximation. This provides more flexibility for choosing a vertex location, since the feasible region is larger than its subset that is computed in [4].

We observe that, in the case of a valid mesh, the feasible set is non-empty for all vertices, and propose to use this feasible set method as a mesh improvement method, rather than a mesh repair method for tangled meshes.

Even theoretically robust optimization based mesh relaxation methods such as the reference Jacobian or the condition number mesh smoothing method (for both, see [9]) may in some cases be inadequate as rezoning methods in an ALE algorithm. Problems in ALE typically occur, when the mesh displacements that are generated by the rezoner are not large enough to keep up with the mesh displacements that the Lagrangian step yields. In such a case, the mesh can eventually tangle, requiring user intervention such as frequent restarts to fine tune the ALE strategy. In contrast, the feasible set method when used as a rezoner yields by design the largest possible mesh displacements, since these displacements are computed based on the feasible set. It can thus be used as a rezoning method of last resort, to more rapidly recover mesh quality around mesh nodes that are close to tangling.

The rest of this paper is organized as follows. In Section 2, we review the feasible set method and explain how it can be used as a mesh smoothing method. In Section 3, we give an example of a simple ALE simulation that utilizes the feasible set relaxer.

2. The feasible set method

While the feasible set method was introduced in [4, 8], it is essential to this presentation and, thus, we begin by giving a brief overview of it here.

A mesh M in 2D consists of vertices $V_i, i = 1, \ldots, N_V$, edges $E_i, i = 1, \ldots, N_E$ each of which connects two distinct vertices, $E_i = (V_{i_1}, V_{i_2})$, and cells $C_i, i = 1, \ldots, N_C$ each of which is a polygon with a boundary that can be written as the union of a sequence of distinct edges $\partial C_i = \bigcup_{j=1}^{N_{C_i}} E_{i_j}$ such that $E_{i_j} \cap E_{i_{j+1}} \neq \emptyset$ and the sequence of edges traces the boundary of C_i in counter clockwise direction. For each vertex V_i we define its patch of cells as $P_i = \{C_j | C_j \cap V_i \neq \emptyset\}$.

If every cell in a mesh is convex, then we call that mesh valid. An equivalent definition of mesh validity uses corners in the mesh. A corner is an ordered pair of adjacent edges. The corners in a mesh are generated by the corners of the cells in the mesh. A cell C that has n_c edges $E_{i_j}, j = 1, \ldots, n_c$ that are ordered in a counter clock-wise direction, gives rise to n_c ordered pairs of edges $(E_{i_j}, E_{i_{j+1}}), j = 1, \ldots, n_c - 1$, and $(E_{i_{n_c}}, E_{i_1})$. Clearly, if the angle between the two edges in each ordered pair that was generated by a cell C is positive but smaller than π then C is convex. We call such an angle valid and use this to decide whether a vertex is tangled.

We note that each vertex V that is part of the boundary of a cell C has three adjacent corners that are also part of the cell. We denote the corner that has this vertex at its apex as the *apex connected* corner and the other two as the *far connected* corners to vertex V in the cell C.

Definition 1. *If any of the angles that are attached to a vertex V is not valid, then we call V tangled.*

In order to untangle a vertex, its coordinates must be changed such that all of its attached angles are valid. To that end, we define the Cartesian coordinates of a vertex V as $(x(V), y(V))$.

For each vertex V in the mesh we can define the set

$$\mathcal{F}_M(V) = \{(\xi,\eta) \in R^2 \mid V \text{ with } (x(V), y(V)) = (\xi,\eta) \text{ is not tangled in } M\} \tag{1}$$

We refer to $\mathcal{F}_M(V)$ as the feasible set of the vertex V in the mesh M. This definition is computationally impractical since it cannot be used to construct the feasible set. An alternative definition that can be used to construct the feasible set is based on the intersection of half planes.

We consider a corner K that is comprised of two distinct edges E_1 and E_2 and three vertices V_1, V_2, and V_3, where $E_1 = (V_1, V_2)$ and $E_2 = (V_2, V_3)$ (see Figure 1). The first case of a apex connected corner is depicted on the left in

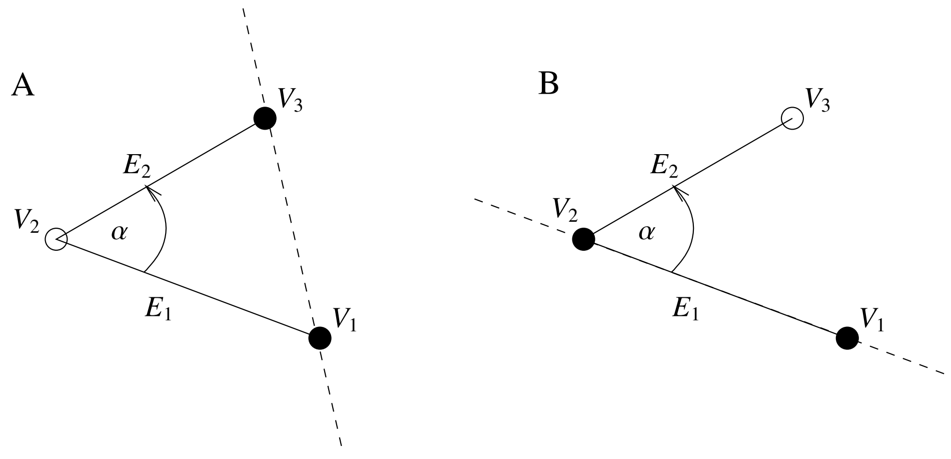

Figure 1: A: For a corner that is apex connected to a vertex, that vertex (in this case V_2) can be moved to any position on the positive side of the line defined by the other two vertices (here, V_1 and V_3), while the corner remains valid. B: For a corner that is far connected to a vertex, that vertex (in this case V_3) can be moved to any position on the positive side of the line that is defined by the other two vertices (in this case V_1 and V_2).

Figure 1. The angle α between edges E_1 and E_2 is between zero and π if the coordinates of vertex V_2 are on the positive side of the line through vertices V_1 and V_3. The second case of a far connected corner is depicted on the right in Figure 1. The angle α between edges E_1 and E_2 is between zero and π if the coordinates of vertex V_3 are on the positive side of the line through vertices V_2 and V_1. The case for varying the position of vertex V_2 is analogous.

To say it more abstractly, in each of these three cases the free vertex must lie inside a half plane for the corner to be valid, i.e.

$$\mathcal{H}_1(K) = \{(\xi,\eta) \mid 0 < \cos^{-1}\left(\frac{\overrightarrow{(V_2,(\xi,\eta))} \times \overrightarrow{(V_2,V_3)}}{||\overrightarrow{(V_2,(\xi,\eta))}||||\overrightarrow{(V_2,V_3)}||}\right) < \pi\} \tag{2}$$

$$\mathcal{H}_2(K) = \{(\xi,\eta) \mid 0 < \cos^{-1}\left(\frac{\overrightarrow{((\xi,\eta),V_1)} \times \overrightarrow{((\xi,\eta),V_3)}}{||\overrightarrow{((\xi,\eta),V_1)}||||\overrightarrow{((\xi,\eta),V_3)}||}\right) < \pi\} \tag{3}$$

$$\mathcal{H}_3(K) = \{(\xi,\eta) \mid 0 < \cos^{-1}\left(\frac{\overrightarrow{(V_2,V_1)} \times \overrightarrow{(V_2,(\xi,\eta))}}{||\overrightarrow{(V_2,V_1)}||||\overrightarrow{(V_2,(\xi,\eta))}||}\right) < \pi\} \tag{4}$$

For a particular vertex V in the interior of mesh M, we can construct $\mathcal{F}_M(V)$ by intersecting a number of half planes. As noted above, each cell that is adjacent to V contains three corners that are adjacent to V. Enumerate the V adjacent cells as $C_{j_1}, \ldots, C_{j_{n_c}}$ and the corners in the adjacent cell with index j_i as $K^{j_i}_{k_1}, K^{j_i}_{k_2}, K^{j_i}_{k_3}$ in counter clock-wise direction relative to cell C_{j_i}, then the feasible set for vertex V can be written as

$$\mathcal{F}_M(V) = \bigcap_{i=1}^{n_c} \left(\mathcal{H}_1(K^{j_i}_{k_1}) \cap \mathcal{H}_2(K^{j_i}_{k_2}) \cap \mathcal{H}_3(K^{j_i}_{k_3})\right) \tag{5}$$

In Figure 2, we give an example of a patch that consists of four cells. In part A, we label the cells and the corners in each cell that are adjacent to the vertex in the center according to the labeling scheme described above. In the parts

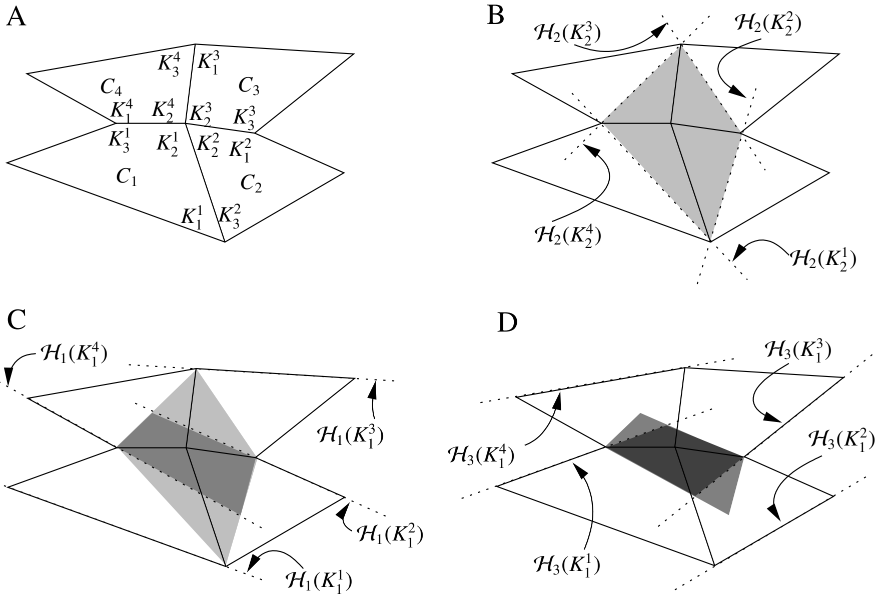

Figure 2: An example of the construction of the feasible set by the successive intersection of half planes. In part A, the labeling of cells and corners is defined, and in parts B through D the half planes that are generated by the corners of the three types K_2^j, K_1^j, and K_3^j are intersected. The half planes $\mathcal{H}_i(K_k^j)$ are each delineated by a dotted line. The arrow from the label points to the side of the line inside the half plane. The feasible set for the center vertex is the most darkly shaded polygon in part D.

B through C, we generate the feasible set by successive intersections of half planes. In part B, we begin by intersecting the half planes that are generated by the apex connected corner in each cell. Recall that these apex connected corners are labeled $K_2^{i_j}$ in cell C_{i_j} and that the associated half plane is denoted by $\mathcal{H}_2(K_2^{i_j})$. The polygon that results as the intersection of these four half planes is lightly shaded. In part C of Figure 2, we illustrate the further successive intersection of the half planes that are generated by the far connected corners of the type (3). The resulting polygon is shaded darkly, while we also for reference display the polygon that resulted in part B. In part D of Figure 2, we illustrate the remaining intersections of with half planes that are generated by the far connected corners of type (4). Again, for reference, we display the polygon that resulted in part C. Most darkly shaded is the polygon that is the feasible set.

Note that none of the half planes in (5) depend on the location of V. Therefore, the feasible set is really a function of the patch of cells that is adjacent to vertex V. In particular, the specific location of the edges that are adjacent to V plays no role in the construction of the feasible set.

For a vertex that is interior to the mesh, the feasible set is not empty if the vertex only has valid adjacent corners. However, if some of its adjacent corners are not valid, its feasible set may be empty. Also, by construction, the feasible set is a convex polygon, if it is not empty.

For vertices that lie on the boundary of the mesh, the same construction of the feasible set can be used as for interior vertices. The difference is that the feasible set of a boundary vertex, if it is not empty, may not be bounded. If it is bounded, and not empty, the feasible set is, by construction a convex polygon. Analogously to the case of an interior vertex, the feasible set of a boundary vertex is not empty, if all its adjacent corners are valid.

2.1. The feasible set method as a mesh improvement method

The feasible set method can be employed as a robust local mesh relaxation technique. For a valid mesh, the feasible set method will by design always yield a non empty feasible set. This requires that vertices are updated one at a time (Gauss Seidel type sweep) and not all at once (Jacobi type sweep). The Gauss Seidel sweep is difficult to parallelize, and will in addition lead to mesh imprinting that is driven by the order in which the positions of mesh vertices are updated. An alternative approach that is easily parallelizeable and does not produce any vertex order induced mesh imprinting is the damped Jacobi sweep. First, the feasible set and the associated new vertex position and its associated vertex displacement is computed for all eligible vertices. This new vertex position is based on old mesh data, as the mesh is not updated during this step. Then a global damping parameter $0 < \delta < 1$ is multiplied with all vertex displacements. This damped update constitutes the damped Jacobi mesh displacement. In our experience, a damping factor of $\delta = 0.5$ will work well in most cases. If the damping factor is chosen too close to one, 'ringing' can result, where the neighboring mesh vertices start oscillating back and forth from rezone step to rezone step. In our experience, the feasible set method can generate rather large vertex displacements. This behavior can be counteracted by stronger damping of the displacements, however, as we will see in Section 3, these large displacements can be a desirable feature.

An alternative approach that has proved to be successful is to mark mesh cells that are close to being invalid and only apply the feasible set relaxer to vertices that are part of these marked cells. The rather large vertex displacements tend to bring such nearly invalid mesh cells back from the brink, and place the marked vertices far away from an invalid/tangled position.

2.2. Vertex placement inside the feasible set

For a particular valid vertex and its patch in a mesh, the feasible set can be computed and then an improved placement for the vertex determined. By construction, this feasible set is a convex polygon. A simple choice that will always yield a valid vertex position inside its convex feasible set is the arithmetic average of all corners of the feasible set. Denote by $(x_i, y_i), i = 1, \ldots, N$ the corners of a convex polygon, then the arithmetic average vertex position (x_{avg}, y_{avg}) is

$$x_{avg} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{6}$$

$$y_{avg} = \frac{1}{N} \sum_{i=1}^{N} y_i. \tag{7}$$

While this arithmetic average is very easy to compute, it has one obvious drawback that we illustrate in Figure 3. In the case where corners of the feasible set are clustered in one area, the arithmetic average of its corners will be biased toward that cluster, since it depends on the coordinates of the corners of the feasible set. A better choice for a relaxed

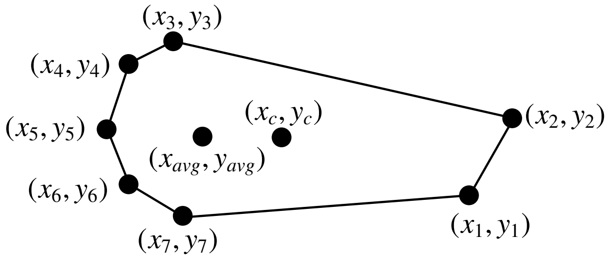

Figure 3: A convex polygon with corners that are clustered. The arithmetic average of its corners (x_{avg}, y_{avg}) is biased toward that cluster, while its centroid (x_c, y_c) is not (the positions of the arithmetic average and the centroid in this schematic are approximate).

vertex position inside the feasible set is its centroid. The centroid of a convex polygon is always inside that polygon, and its location depends only on the shape of the polygon and is not biased by clusters of corners of the polygon. The centroid of a closed polygon (x_c, y_c) can easily be calculated by

$$x_c = \frac{1}{6A}\sum_{i=1}^{N}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{8}$$

$$y_c = \frac{1}{6A}\sum_{i=1}^{N}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i), \tag{9}$$

where A is the area of the polygon

$$A = \frac{1}{2}\sum_{i=1}^{N}(x_i y_{i+1} - x_{i+1} y_i), \tag{10}$$

and for convenience of notation $(x_{N+1}, y_{N+1}) \equiv (x_1, y_1)$. We use the centroid of the feasible set as the target vertex position.

3. A hydro example

We now give an example of a simple ALE calculation that employs our feasible set relaxer. We implemented the relaxer in an ALE code that is based on the staggered grid compatible hydrodynamics scheme presented in [10].

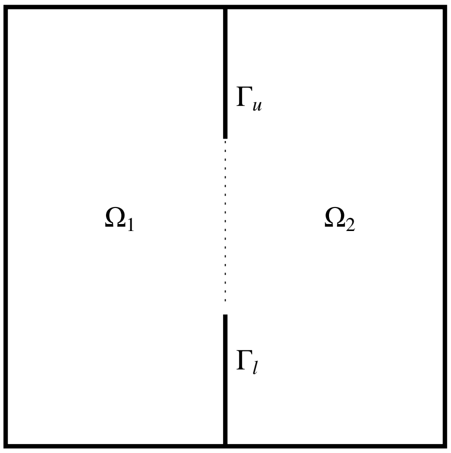

Figure 4: The computational domain for the example is the unit square. Vertices are allowed to slide along the outside boundary, however they are fixed along the interior boundaries Γ_u and Γ_l. The gas inside the computational domain is initialized to have higher energy in Ω_1 than in Ω_2. As a result, the gas will rapidly expand from Ω_1 into Ω_2 and almost immediately cause the computational mesh to become nearly tangled near the opening (the upper end of Γ_l and the lower end of Γ_u.

Figure 4 depicts the computational domain $\Omega(0, 1) \times (0, 1)$. The computational domain is divided into two subdomains $\Omega_1 = (0.5, 1) \times (0, 1)$ and $\Omega_2 = (0.5, 1) \times (0, 1)$. The initial mesh on each of these two subdomains is logically rectangular; on Ω_1, the mesh consists of 25×50 cells, and on Ω_2, the mesh consists of 5×50 mesh cells. As a result, the mesh resolution is much lower in the x-direction on Ω_2 than on Ω_1. Vertices in the mesh are allowed to slide along the outside boundary of the domain $\partial\Omega$, but are fixed along the interior boundaries $\Gamma_l = \{0.5\} \times [0, 0.3]$ and $\Gamma_u = \{0.5\} \times [0.7, 1]$.

The problem domain is occupied by a gamma law gas with $\gamma = 5/3$, a minimum sound speed of 0.001, and a density of 1.0. It is initialized with an energy of 0.1 in Ω_1 and 0.0 in Ω_2.

When run in pure Lagrangian mode, the computational mesh tangles after only a few time steps, at the tips of the interior boundaries Γ_l and Γ_u. We use the feasible set relaxer with triggers: A vertex is marked as to be relaxed, when

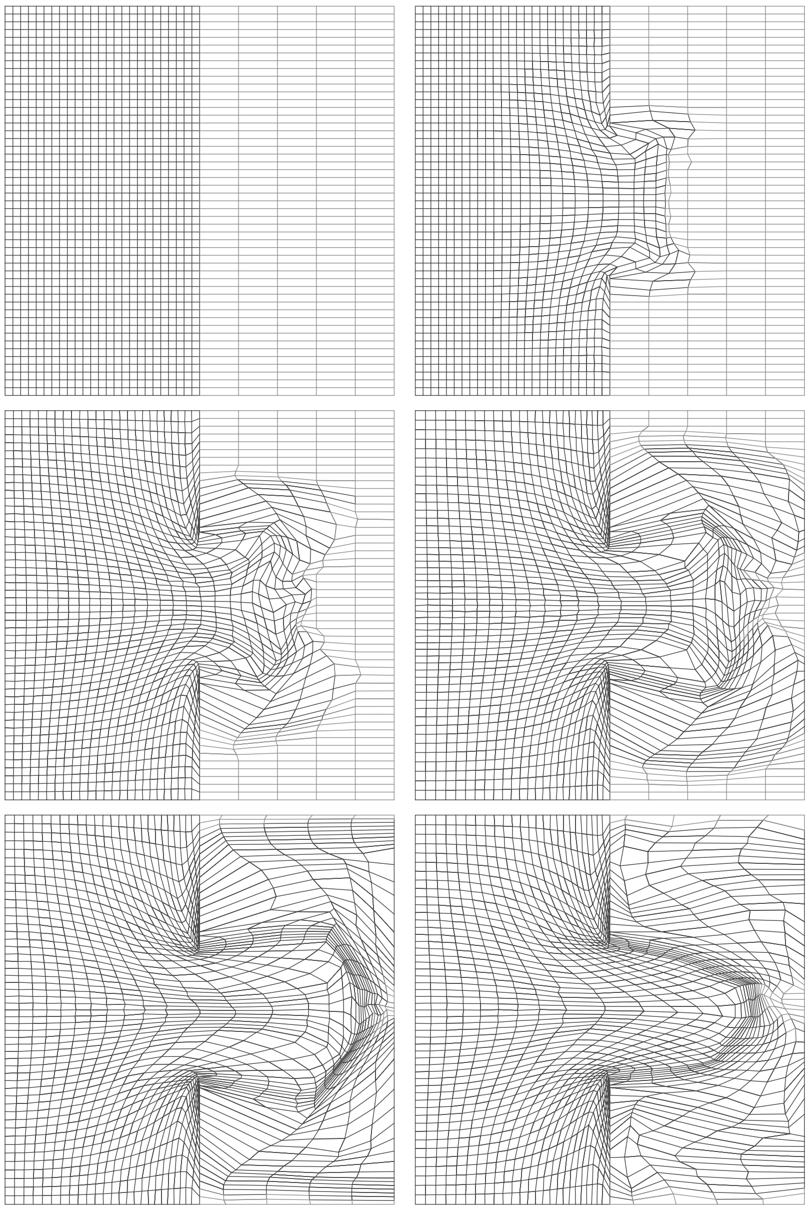

Figure 5: Snapshots of the initial computational mesh (left top) and of the mesh after 93 (right top), 1198 (left middle), 5273 (right middle), 11350 (left bottom) and 15882 cycles (right bottom) of the example problem outlined in Section 3. The meshes show quite a severe distortion around the corners of the opening between Γ_l and Γ_u. With the help of the feasible set relaxer, it is possible to run the simulation to impressively late time.

one of its attached angles is smaller than 30° or larger than 150°, or when the aspect ratio is larger than ten. We use one damped Jacobi sweep per time step over these marked vertices with a damping factor of 0.7.

Figure 5 depicts six snapshots of the computational mesh of the initial mesh and the mesh after cycles 93, 1198, 5273, 11350 and 15882, which corresponds to problem times 0.0, 1.0090388, 2.0006902, 3.0002201, 4.0002856 and

5.0003796 seconds. This example highlights that with this rezoning strategy alone, the simulation can run to an impressively late time. Note that, in this example, the emphasis is not creating a high quality mesh, but simply on preserving a valid mesh. Consequently, the feasible set method should be combined with another mesh improvement method that can generate meshes of better quality.

4. Conclusions

We have presented a new 2D rezone strategy for ALE methods, which is based on the feasible set method. If used in a Gauss-Seidel type sweep, this rezone method is robust; it will never create a tangled mesh from a mesh that is not tangled. In practice, the perhaps more desirable damped Jacobi sweep proves to be quite robust as well. This feasible set rezone strategy if employed by itself can generate meshes that are valid but of low quality. We recommend to combine it with another rezone strategy such as Winslow, to improve the quality of the resulting meshes.

5. Acknowledgments

This work was performed under the auspices of the US Department of Energy by Los Alamos National Laboratory under contract DE-AC52-06NA25396, LA-UR 09-07380.

References

[1] C. W. Hirt, A. A. Amsden, J. L. Cook, An Arbitrary Lagrangian-Eulerian computing method for all flow speeds, Journal of Computational Physics 14 (1974) 227–253.

[2] L. G. Margolin, Introduction to "An arbitrary Lagrangian-Eulerian computing method for all flow speeds", Journal of Computational Physics 135 (1997) 198–202.

[3] T. S. Li, Y. C. Wong, S. M. Hon, R. M. M. C. G. Armstrong, Smoothing by optimisation for a quadrilateral mesh with invalid elements, Finite Elements in Analysis and Design 34 (2000) 37–60.

[4] K. Kovalev, M. Delanaye, C. Hirsch, Untangling and optimization of unstructured hexahedral meshes, in: S. A. Ivanenko, V. A. Garanzha (Eds.), Proceedings of Workshop on Grid Generation: Theory and Applications, Russian Academy of Sciences, Moscow, Russia, 2002, http://www.ccas.ru/gridgen/ggta02/papers/Kovalev.pdf.

[5] L. Freitag, P. Plassmann, Local optimization-based simplicial mesh untangling and improvement, International Journal of Numerical Methods for Engineering 49 (2000) 109–125.

[6] L. Freitag, P. Plassmann, Local optimization-based untangling algorithms for quadrilateral meshes, in: Proceedings of the Tenth Anniversary International Meshing Roundtable, Sandia Report, SAND 2001-2976C, Newport Beach, CA, 2001, pp. 397–406.

[7] P. M. Knupp, Hexahedral and tetrahedral mesh untangling, Engineering with Computers 17 (2001) 261–268.

[8] P. Vachal, R. V. Garimella, M. J. Shashkov, Untangling of 2D meshes in ALE simulations, Journal of Computational Physics 196 (2004) 627–644.

[9] P. M. Knupp, L. G. Margolin, M. J. Shashkov, Reference Jacobian optimization-based rezone strategies for arbitrary Lagrangian Eulerian methods, Journal of Computational Physics 176 (2002) 93–128.

[10] E. J. Caramana, D. E. Burton, M. J. Shashkov, P. P. Wahlen, The construction of compatible hydrodynamics algorithm utilizing conservation of total energy, Journal of Computational Physics 146 (1998) 227–262.