INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids (2007)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1577

Extension of efficient, swept-integration-based conservative remapping method for meshes with changing connectivity‡

M. Kucharik*,† and M. Shashkov

T-7 Group, MS B284, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545, U.S.A.

SUMMARY

Remapping is one of the essential parts of most arbitrary Lagrangian–Eulerian methods. Here, we extend the idea of swept integration introduced in (*J. Comput. Phys.* 2003; **184**(1):266–298) to meshes with connectivity changing in Voronoi-like manner. To demonstrate properties of the developed method, we present several numerical examples. Published in 2007 by John Wiley & Sons, Ltd.

Received 10 April 2007; Revised 19 June 2007; Accepted 22 June 2007

KEY WORDS: ALE; conservative interpolations; Voronoi meshes

1. INTRODUCTION

In numerical simulations of fluid flow, the choice of the computational mesh 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 [1], Hirt *et al*. developed the formalism for a mesh 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.

It is most usual to separate the ALE algorithm into three individual phases. These are the following: (1) a Lagrangian phase, in which the solution and mesh are updated, (2) a rezoning phase, in which the nodes of the computational mesh are moved to a more optimal position, and (3) a remapping phase, in which the Lagrangian solution is interpolated onto the rezoned mesh. We are interested in the development of staggered ALE methods for meshes whose connectivity may

*Correspondence to: M. Kucharik, T-7 Group, MS B284, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545, U.S.A.
†E-mail: kucharik@lanl.gov

Contract/grant sponsor: U.S. Department of Energy; contract/grant number: DE-AC52-06NA25396

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. Changing connectivity adds another degree of freedom to the method—in the case of shear flows, initially close cells may not be close to each other in later stages, and methods without reconnections fail due to mesh tangling. We focus here on the Voronoi meshes [2] constructed from arbitrary set of points (Voronoi generators) in the computational domain. Each cell of the Voronoi mesh corresponds to one of the Voronoi generators, and is defined as a set of points, which are closer to the particular generator than to all the other ones. Generally, each node of the Voronoi mesh connects three different edges. However, quadrilateral meshes (with four edges in each mesh node) can be constructed by degeneration of two nodes to one physical location. By movement of the Voronoi generators, the mesh nodes and edges move, and reconnection in the particular node can appear. We allow the topology changes caused by the prescribed movement of the Voronoi generators.

This paper focuses primarily on the last phase of the ALE algorithm—remapping. We are looking for the remapping algorithm, which satisfies several important conditions: (1) conservation, (2) local-bound preservation, (3) linearity preservation, and (4) efficiency. The complete remapping algorithm based on approximate swept integration was presented in [3] for the case of 2D logically orthogonal computational meshes with the same connectivity. This algorithm does not require finding the cell intersections, and is face-based and thus more efficient than the natural exact integration method. In this paper, we describe the process of dealing with Voronoi-like intersections, and extending the swept-integration-based remapping algorithm to similar meshes with changing connectivity.

2. SWEPT-INTEGRATION-BASED METHOD FOR MESHES WITH IDENTICAL TOPOLOGY

Our remapping algorithm [3] includes three steps: (1) piecewise-linear reconstruction of the unknown quantity function (density of mass, total energy, total momenta in both directions) in each cell from the mean values of the particular quantity in the neighboring cells; (2) approximate integration of the reconstructed function interpolating new mean values on the new mesh; and (3) repair, mass redistribution procedure enforcing local-bound preservation. The reconstruction and repair stages are performed exactly the same way, as described in [3] for 2D and in [4] for 3D.

Let us focus on the second step, the approximate swept integration. Suppose, we know the mean values $\bar{g}_c$ (and masses $m_c = \bar{g}_c\, V_c$) in cells of the original mesh $\{c\}$, and during the reconstruction stage, we have computed slopes in each cell c. Here, V_c stands for volume of cell c, and is computed analytically as integral of 1 over the particular cell. Our goal is to compute new mean values $\bar{g}_{\tilde{c}}$ (and masses $m_{\tilde{c}}$) in the cells of the new mesh $\{\tilde{c}\}$. The same cell c in original mesh and new mesh $\tilde{c}$ are shown in Figure 1(a) and (b). The swept integration method is based on the idea that the mass in the new cell $m_{\tilde{c}}$ can be written in the flux form as the original mass m_c plus masses of swept regions δm_e (mass fluxes) corresponding to each edge e of the cell

$$\tilde{m}_{\tilde{c}} = m_c + \sum_{e \in \partial c} \Omega_c(e)\, \delta m_e \tag{1}$$

where $\Omega_c(e)$ is equal to 1 if edge e is in counter-clockwise orientation according to cell c and -1 otherwise. Swept masses $\delta m_e = \int_{\delta V_e} g_{c^*}(\mathbf{r})\, \mathrm{d}S$ are computed as integrals over swept region δV_e

Published in 2007 by John Wiley & Sons, Ltd.

DOI: 10.1002/fld

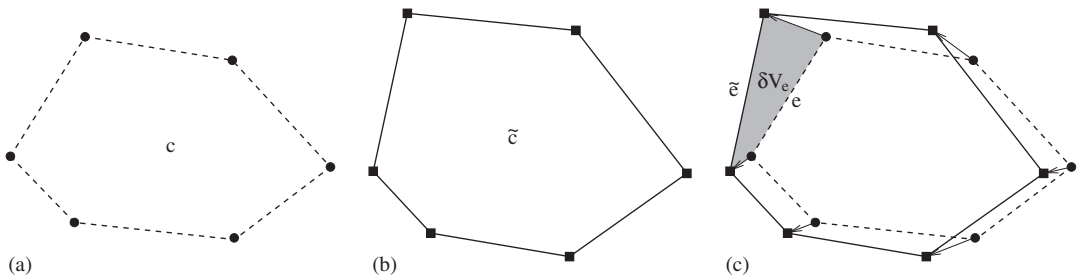

Figure 1. One cell (a) c in original dashed mesh; (b) $\tilde{c}$ in new solid mesh; and (c) swept region δV_e corresponding to old edge e and new edge $\tilde{e}$.

attached to edge e (see Figure 1(c)). The reconstruction is taken from cell c^*, which is selected according to the sign of the volume of swept region δV_e. If the edge e moves inwards the cell c, the swept volume is negative and we take the reconstruction from cell $c^* = c$. In the opposite case, the reconstruction is taken from the cell neighboring with c over edge e. The described swept integration algorithm is approximate, and does not guarantee preservation of local bounds. Therefore, repair stage [5] enforcing this property must be added. In [3] we have shown that these algorithms followed by the repair stage satisfy all our conditions stated in Section 1.

3. SWEPT-INTEGRATION-BASED METHOD FOR MESHES WITH CHANGING TOPOLOGY

After detecting, that there is no connectivity change around the particular cell (its neighborhood remains the same), we perform the algorithm as described in Section 2. Now, let us discuss the possibility that the connectivity changes (the neighborhood of the particular cell is different in old and new meshes). The typical situation is shown in Figure 2, showing four cells in original mesh (a), new mesh (b), and both meshes plotted over each other (c). As we can see, the original bottom-left (BL) cell was neighbor of the upper-right (UR) cell. In the new mesh, their common edge disappeared, and new edge was added, causing the upper-left (UL) and bottom-right (BR) cells to be neighbors in the new mesh. We call this type of reconnection (one removed and one added edge) Voronoi-like reconnection.

When such a reconnection is detected, we find the center of reconnection C by averaging the coordinates of vertices of removed and added edges, as shown in Figure 3(a). Then, we follow the swept integration algorithm, as described before, in two steps. At first, we shrink the removed edge to this central point and perform swept integrations corresponding to all involved edges—the removed edge and four edges connected to it (see Figure 3(b)). In the second step, similar five swept integrations are performed by extending the central point to the created edge (see Figure 3(c)).

For correct functioning of the described algorithm, we must ensure that each mesh edge is involved in at most one reconnection. This is not generally fulfilled, but it can be achieved in the case of Voronoi meshes obtained by generators movement (as in our case). When more (two) reconnections at one edge are detected, the generators movement can be reduced and performed in two or more steps. Time-step reduction causes higher total number of remappings, but it guarantees satisfaction of the single reconnection condition.

Published in 2007 by John Wiley & Sons, Ltd. *Int. J. Numer. Meth. Fluids* (2007)
DOI: 10.1002/fld

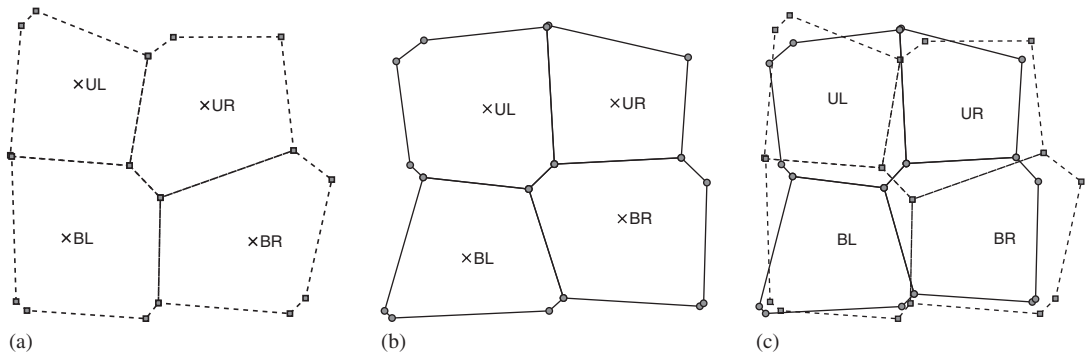

Figure 2. Reconnection in Voronoi mesh: (a) piece of original; (b) new mesh; and (c) both meshes over each other.

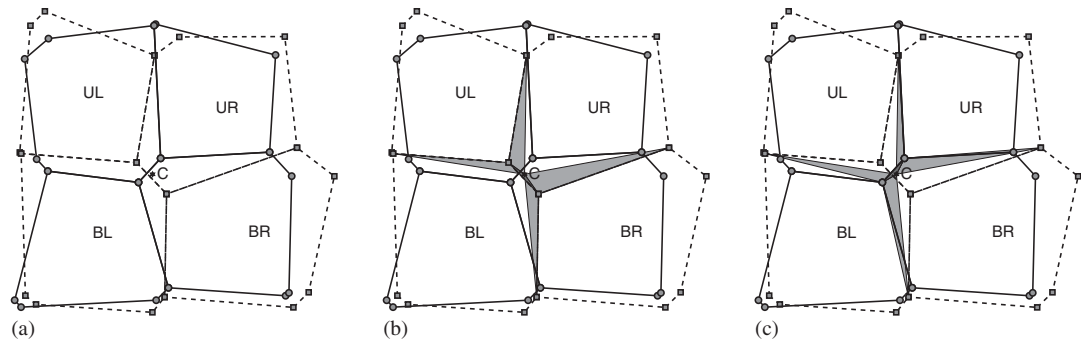

Figure 3. Three steps for handling of reconnection: (a) location of center of reconnection C; (b) shrinking of removed edge to center C; and (c) expanding of center C to new edge.

4. NUMERICAL EXAMPLES

In this section, we present several numerical tests to demonstrate properties of our remapping algorithm. The complete method was implemented in C programing language using MSTK [6] environment for mesh representation.

As the testing mesh, we present here the Voronoi mesh generated by the uniformly spread 32^2 generators in the $\langle 0, 1\rangle^2$ computational domain. The generated Voronoi mesh is the regular logically orthogonal mesh, in fact each vertex is degenerated from two Voronoi vertices. To look at the cumulative effects of many remappings, we remap the function over a sequence of meshes (usually called cyclic remapping [7]), generated as Voronoi meshes from the moved generators. The velocity of the Voronoi generator of cell c in time t^{n+1} is described by the stream-like formula:

$$u^{n+1} = -\alpha(t^{n+1})(\sin(\pi x_c^n))^2 \sin(2\pi y_c^n), \quad v^{n+1} = +\alpha(t^{n+1})(\sin(\pi y_c^n))^2 \sin(2\pi x_c^n) \tag{2}$$

where the parameter $\alpha(t) = \cos(\pi t)$, and x_c^n, y_c^n are coordinates of the cell generator in time level t^n. Computation ends in final time $t = 3$. The generated sequence of meshes includes many connectivity changes, testing all capabilities of our algorithm.

Published in 2007 by John Wiley & Sons, Ltd. *Int. J. Numer. Meth. Fluids* (2007)
DOI: 10.1002/fld

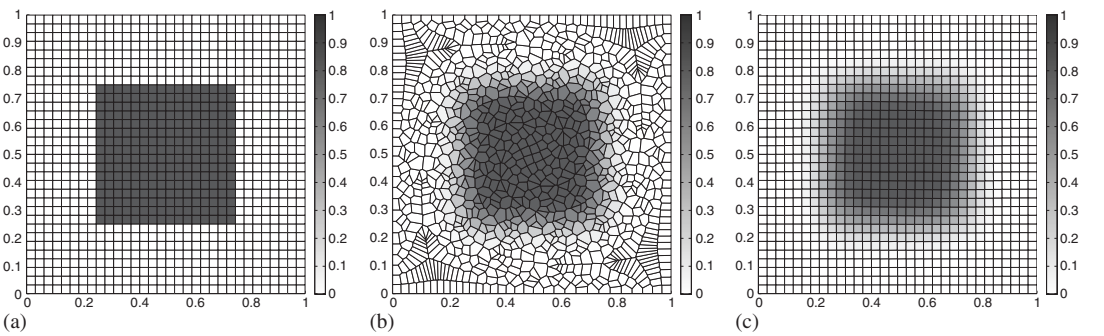

Figure 4. Square color function in 1024 cells of (a) initial; (b) middle; and (c) final computational mesh.

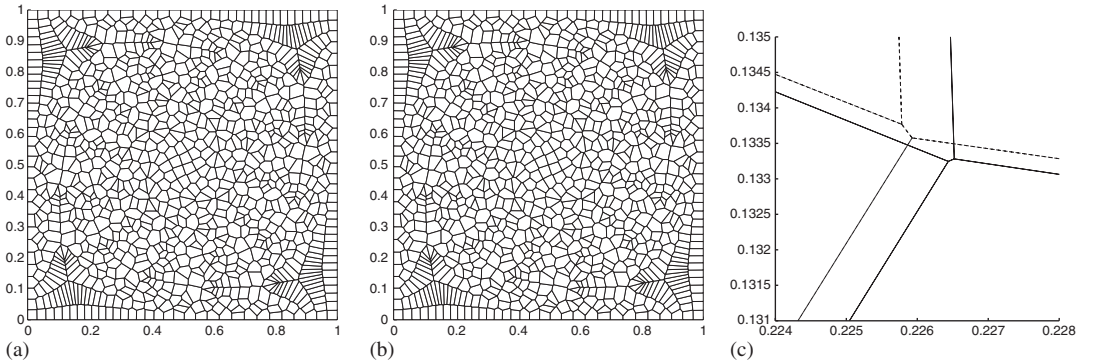

Figure 5. Two consecutive meshes in the middle of the stream-like mesh movement: (a) previous mesh in time $t = 1.4352$; (b) following mesh in time $t = 1.5000$; and (c) zoom to overlap of both meshes in lower-left region of the computational domain including one typical reconnection.

As the first test function, we use a linear function $g(x, y) = 1 + x + 2y$. The numerical error is zero (up to the round off error) in all simulations (with different initial meshes and different generators movements), which confirms linearity preservation of our algorithm.

The mean values of the second test function in the cells of the initial mesh are shown in Figure 4(a). This color square function is equal to 1 inside the square of edge length $\frac{1}{2}$ located in the center of the computational domain, and 0 otherwise. In Figure 4(b), we can see the situation in the middle of the remapping process in time $t = \frac{3}{2}$. As we can see, the mesh topology is completely different from the initial one. In Figure 5, we can see two consecutive meshes in the middle of the simulation, in times $t = 1.4352$ and 1.5000. As we can see, both meshes are very similar to each other. To demonstrate that the meshes have different topologies, we zoomed one typical reconnection in the lower-left part of the computational domain in Figure 5(c).

The remapped function mean values in the final time are shown in Figure 4(c). We can observe the dissipation around the square edge, accumulated from many remappings. In Table I, we present a comparison of numerical relative L_1 errors and times of computation for our swept-integration-based method and the natural method based on exact integration. As we can see, the new method is more than three times faster than the natural method, which is caused by the fact that exact

Table I. Comparison of relative L_1 error and time of computation T in seconds for standard method based on exact integration and new method using approximate swept integrations.

$\|\{c\}\|$	L_1^{exact}	L_1^{swept}	T^{exact} (s)	T^{swept} (s)
16	3.6×10^{-1}	3.7×10^{-1}	1.1×10^{0}	1.0×10^{0}
64	2.1×10^{-1}	2.1×10^{-1}	9.0×10^{0}	5.4×10^{0}
256	1.3×10^{-1}	1.3×10^{-1}	1.2×10^{2}	4.7×10^{1}
1024	7.8×10^{-2}	7.8×10^{-2}	1.9×10^{3}	6.0×10^{2}
4096	4.8×10^{-2}	4.8×10^{-2}	3.1×10^{4}	9.7×10^{3}
16 384	3.0×10^{-2}	3.0×10^{-2}	4.9×10^{5}	1.5×10^{5}

Note: Comparison is shown for several initially uniform meshes with $|\{c\}|$ cells. Simulations performed on standard PC with 2.0 GHz AMD Opteron processor.

integration method requires finding all intersections of both meshes. The numerical errors of both methods are almost the same, the order of convergence is close to the first order for the non-smooth function. The local extrema were not overshot in any cell of the computational mesh.

We have also performed several tests with smooth 2D sine function. As in the presented color square function example, numerical errors of swept and exact integration methods are almost identical. For smooth functions, we numerically achieved second order of convergence of our algorithm in L_1 error. This is the consequence of the linearity preservation property of our algorithm.

5. CONCLUSION

In this paper, we have described the extension of the swept-integration-based remapping algorithm proposed in [3] to the general polygonal meshes with connectivity changing in Voronoi-like manner. We have also presented several numerical examples including numerical errors, which verifies linearity and local-bound preservation, conservation, and applicability to general 2D polygonal meshes. Comparison of our algorithm with the classical exact integration algorithm was performed, showing comparable numerical errors and higher efficiency of new method.

ACKNOWLEDGEMENTS

This work was carried out under the auspices of the National Nuclear Security Administration of the U.S. Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396. The authors thank R. Garimella, L. Margolin, B. Wendroff, B. Swartz, R. Liska, M. Berndt and V. Dyadechko for fruitful discussions and constructive comments.

REFERENCES

1. Hirt CW, Amsden AA, Cook JL. An arbitrary Lagrangian–Eulerian computing method for all flow speeds. *Journal of Computational Physics* 1974; **14**(3):227–253.
2. Aurenhammer F, Klein R. Voronoi diagrams. In *Handbook of Computational Geometry*, Chapter 5. Elsevier: Amsterdam, The Netherlands, 2000; 201–290.
3. Kucharik M, Shashkov M, Wendroff B. An efficient linearity-and-bound-preserving remapping method. *Journal of Computational Physics* 2003; **188**(2):462–471.

4. Garimella R, Kucharik M, Shashkov M. An efficient linearity and bound preserving conservative interpolation (remapping) on polyhedral meshes. *Computers & Fluids* 2007; **36**(2):224–237.
5. Shashkov M, Wendroff B. The repair paradigm and application to conservation laws. *Journal of Computational Physics* 2004; **198**(1):265–277.
6. Garimella R. Mesh data structure selection for mesh generation and FEA applications. *International Journal for Numerical Methods in Engineering* 2002; **55**(4):451–478.
7. Margolin LG, Shashkov M. Second-order sign-preserving conservative interpolation (remapping) on general grids. *Journal of Computational Physics* 2003; **184**(1):266–298.

Published in 2007 by John Wiley & Sons, Ltd.

DOI: 10.1002/fld