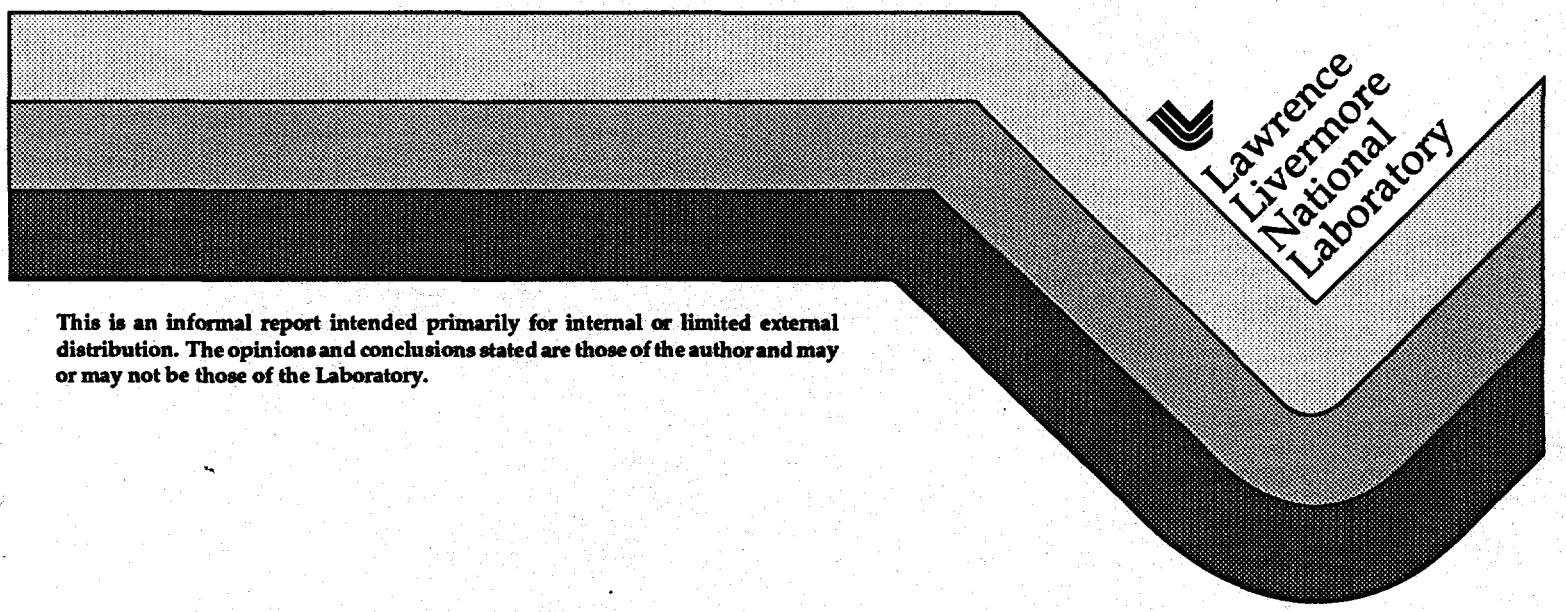
UCRL-ID-123530

Efficient Second Order Remapping on Arbitrary Two Dimensional Meshes

Douglas S. Miller
Donald E. Burton
Joseph S. Oliviera

March 18, 1996



Efficient Second Order Remapping on Arbitrary Two Dimensional Meshes

Douglas S. Miller, Donald E. Burton, Joseph S. Oliviera
Lawrence Livermore National Laboratory

March 18, 1996

Abstract

We have developed an efficient method of remapping physical variables from one unstructured grid composed of arbitrary polygons to another, based on the work of Ramshaw and Dukowicz. Eulerian cycles are used to convert the mesh into a single chain of connected edges, which eliminates grid searching. The error is second order in the zone size. The algorithm handles degenerate meshes well. Computational effort to perform a remap scales linearly with the number of zones in the two grids, which is an improvement over typical $N \log N$ methods.

1 Introduction

Except for the one dimensional case, any physics code in which the mesh moves as the problem evolves in time faces the difficulty of the mesh tangling, zones inverting, or the mesh in other ways becoming unsuitable to continue solving the problem at hand. Lagrangian hydrodynamics codes frequently encounter this situation. To continue running the problem, the user or the code must generate a new mesh and then map the physics variables from the old mesh to the new one. This paper deals with the mapping process, often called "remapping" or "rezoning".

Previous authors, notably Dukowicz [3] and Ramshaw [6], have dealt with the question of how to do this remapping for a grid composed of triangular or quadrilateral zones, but the methods they proposed were not applicable to

arbitrary, unstructured grids. We extend their work by providing a method that works on totally unstructured meshes composed of arbitrary polygons. We also improve the efficiency of the remapping process by converting each mesh into a single long chain of connected edges over which to integrate. We detail our method for dealing with degeneracies in the mesh as well.

2 Review of Ramshaw and Dukowicz

We begin with an explanation of our version of the method of Ramshaw and Dukowicz. It differs only slightly from their original work [2][5]. Let us suppose that we want to map an intensive physical quantity $q(\mathbf{x})$ from one mesh, the "old mesh", to a different one, which we will call the "new mesh". Let q'_k denote the value of q on the new mesh in the k^{th} zone. Then we have

$$q'_k = \frac{1}{A_k} \int_{A_k} q(\mathbf{x})\, dA \tag{1}$$

Dukowicz's first essential contribution was to point out that this problem is easier to solve if we make a substitution of variable and use the divergence theorem to reduce the dimensionality of the above integral. We have chosen to specialize to two dimensions in this paper and so we will replace q with the curl of a vector $\mathbf{F}$ where Dukowicz used the divergence of $\mathbf{F}$. This leads us to use Green's theorem in the next step instead of the divergence theorem but the basic idea is the same.

$$\nabla \times \mathbf{F} = q \tag{2}$$

Then we use Green's theorem to convert the two-dimensional integral in equation (1) to a line integral around the zone k.

$$\int_{A_k} \nabla \times \mathbf{F}\, dA = \int_{A_k} q(\mathbf{x})\, dA \tag{3}$$

$$\oint_{E_k} \mathbf{F} \cdot d\mathbf{l} = \int_{A_k} q(\mathbf{x})\, dA \tag{4}$$

2.1 Choosing an F

$\mathbf{F}$ can be thought of as a vector potential. It has no important physical meaning, and any zero curl vector field can be added to it without changing our results.

An obvious choice for $\mathbf{F}$ is

$$\mathbf{F} = \frac{1}{2} q_0 \hat{\mathbf{k}} \times \mathbf{x} + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) \hat{\mathbf{k}} \times \nabla q \tag{5}$$

for each zone, where $\mathbf{x}_0$ is the zone center and ∇q is a constant within each zone. This $\mathbf{F}$ is discontinuous across zone boundaries, which changes the results of Green's theorem as used above; one must add delta function contributions generated at the discontinuity boundaries. The strength of the delta function is precisely the difference in $\mathbf{F}$ across the boundary, so the final result is that one integrates $\mathbf{F}$ over the new mesh edges and the quantity $(\mathbf{F}_{\text{left}} - \mathbf{F}_{\text{right}})$ over the old mesh edges, where "left" and "right" are defined by the direction of integration.

$$q_k' = \frac{1}{A_k} \Bigg(\oint_{\substack{\text{faces} \\ \text{of new} \\ \text{zone } k}} \mathbf{F} \cdot d\mathbf{l} + \int_{\substack{\text{old faces} \\ \text{contained} \\ \text{in zone } k}} (\mathbf{F}_{\text{left}} - \mathbf{F}_{\text{right}}) \cdot d\mathbf{l} \Bigg) \tag{6}$$

2.2 Integrating F

$\mathbf{F}$ is integrated over closed curves, but these curves are in practice a series of straight lines. Hence the integral is just the sum

$$\oint_{zone} \mathbf{F} \cdot d\mathbf{l} = \sum_i \int_{a_i}^{b_i} \mathbf{F} \cdot d\mathbf{l} \tag{7}$$

where a_i and b_i are the beginning and end points of zone edge i. The form of $\mathbf{F}$ makes this straightforward to evaluate.

$$\int_{a_i}^{b_i} \mathbf{F} \cdot d\mathbf{l} = \frac{1}{2} q_0 \hat{\mathbf{k}} \cdot (\mathbf{a}_i \times \mathbf{b}_i) + \frac{1}{2} \hat{\mathbf{k}} \cdot \nabla q \times (\mathbf{b}_i - \mathbf{a}_i) \left[\left(\mathbf{x}_0 - \frac{(\mathbf{a}_i + \mathbf{b}_i)}{2} \right)^2 + \frac{1}{12} (\mathbf{a}_i - \mathbf{b}_i)^2 \right] \tag{8}$$

The first term on the right hand side corresponds to doing a pure donor cell remapping. The second term adds a second order correction. The gradient factor, ∇q, can be defined in a variety of ways, and is usually limited in some fashion to avoid non-monotonic behavior. A Van Leer-type limiter [4], generalized to arbitrary polygonal zones by Dukowicz and Kodis [3] has been used in our second order test problems. The value of q in a zone is set to $q = q_z + \alpha \nabla q$, where q_z is the constant average q in the zone and α is set such that the value of q in a zone is never outside the range of the q_z of its neighbors.

3 Remapping Method

At this point we have a method in hand for doing remapping. First we integrate the edges of the new mesh through the old mesh. In integrating these edges, each edge is broken up into segments; each edge segment of the new mesh passes through exactly one zone of the old mesh. We accumulate the contribution from each edge segment to the right and left zones of the new mesh.

$$\Delta_{\text{new}}^{\text{left}} = \int_{a_i}^{b_i} \mathbf{F} \cdot d\mathbf{l} \tag{9}$$

$$\Delta_{\text{new}}^{\text{right}} = -\int_{a_i}^{b_i} \mathbf{F} \cdot d\mathbf{l} \tag{10}$$

Then we integrate the edges of the old mesh through the new mesh. Each old edge segment passes through only one zone of the new mesh. We accumulate the contribution of each old edge segment to the new zone containing it.

$$\Delta_{\text{old}} = \int_{a_i}^{b_i} (\mathbf{F}_{\text{left}} - \mathbf{F}_{\text{right}}) \cdot d\mathbf{l} \tag{11}$$

The combination of old and new mesh contributions to a new mesh zone, divided by the area of the new mesh zone, results in the value of the remapped quantity in the new mesh zone.

4 Eulerian Cycles

Having reduced the remapping problem to performing a collection of line integrals, the question arises of how to perform those integrals in an efficient way. Each edge of both meshes must be integrated, or "traced". The difficulty with naive approaches to this problem is that the zones containing the edge must be found in the other mesh for each integration. Point location in an arbitrary mesh is an expensive operation. Ideally we would like to locate a starting point once, integrate that edge to its endpoint, then use that endpoint as the starting point for another edge, and so on. A perfect mesh trace would visit every edge exactly once and cover the entire mesh.

This is precisely the definition of an "Eulerian Cycle" or "Eulerian Walk". The algorithm and prerequisites for its application are well known. An Eulerian Walk exists on a mesh if and only if 1) the degree of each vertex (i.e., the

number of edges coming into it)is even, or 2) there are exactly two vertices of odd degree in the entire mesh. A mesh with these properties is called "Eulerian".

Clearly, most computational meshes are not Eulerian. This need not be a hindrance, however; any mesh can be made Eulerian through the simple artifice of doubling each edge. The extra edges are not a significant burden to computation if each edge is marked as it is integrated. The second time an edge is encountered, the integration step is skipped, the algorithm jumps to the edge endpoint and continues.

Through this technique, the expensive point location operation is incurred only once at the beginning of each mesh trace, and relatively inexpensive line intersection calculations suffice to cover the entire mesh. Further, no unnecessary edges are considered when doing the intersection calculations.

5 Handling Degeneracies

Degeneracies are defined as instances where the intersection between an edge of the old mesh and an edge of the new mesh is not unique. Examples include an edge that exits a zone through a vertex, or an edge that lies exactly on top of an edge in the other mesh. Two methods are often employed to deal with degeneracies; the first is to write special code to handle each type of degeneracy on a case by case basis, the second is to perturb the vertex positions by an amount large enough to break the degeneracies but small compared to some scale length of the problem. Handling degeneracies as special cases leads to awkward and error prone code. We use the perturbation approach for identifying intersecting edges, but insist on computing areas of intersection based on the unperturbed points.

Our method for handling degeneracies is to perturb each point in the new mesh by a nonzero random amount that is δ times smaller than the shortest edge in the mesh. δ is an adjustable parameter that we have set to 10^{-6} in our test problems. Once all degeneracies have been broken, the mesh traces are done. The edge intersection points are stored not as (x, y) pairs but as $(e_{\text{new}}, e_{\text{old}})$ pairs, where e_{new} is the index of an edge in the new mesh and e_{old} the index of an old mesh edge. An essential ingredient in handling degeneracies is that the same intersection point be calculated for $(e_{\text{new}}, e_{\text{old}})$ as $(e_{\text{old}}, e_{\text{new}})$. We achieve this by ordering the edge pair according to mesh

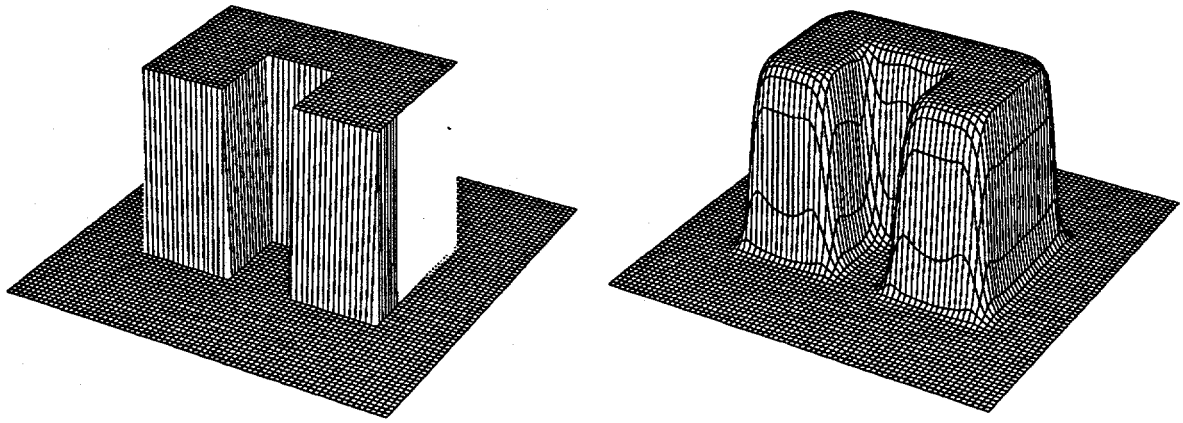


Figure 1: Keyhold test problem: 160 second order rezones with modified Van Leer limiting of the gradient. The blank region in the initial data is a plotting error.

number in the case of degenerate edges (for an unambiguous mesh number we use the memory location of the mesh structure, which does not change throughout the remapping process). We define a unique (x, y) intersection point in space for two edges even if they are parallel and overlay one another in the unperturbed representation (in that instance the intersection point will be an extreme point of one of the edges). In this fashion we can perturb the new mesh to eliminate degeneracies but obtain the unperturbed areas for the q' calculation.

6 Examples

In figure 1 we have run a version of the Bailey keyhole test problem [1] with regular quadrilateral zoning. Our "key" is a 44x27 zone block with a 14x14 hole taken out of it. In this problem each point on the mesh moves through a circle five zone in radius. We broke this motion into 160 steps. The mesh has been rezoned once each cycle, or 160 times. The remapping was done with the second order limited scheme described above.

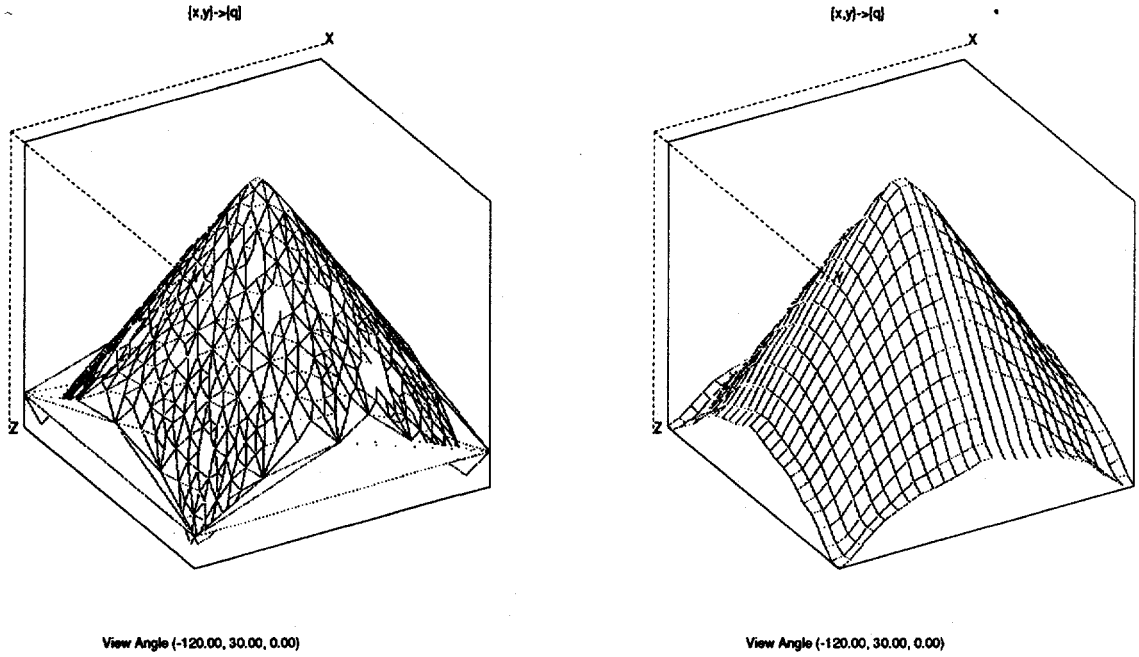


Figure 2: Remapping a ramp function from the arbitrary polygonal mesh on the left to the regular quadrilateral grid on the right. The errors near the edges are due to the poor resolution (very few zones) at the edges of the arbitrary mesh.

Figure 2 shows the remapping algorithm working on meshes composed of completely arbitrary polygons. The left grid was made starting with a triangular grid of random points which was then relaxed, and several nodes were removed to create a new grid composed of arbitrary polygons. A linear function $f = \sqrt{x^2 + y^2}$ has been defined on the left grid. The left mesh was then remapped to the grid on the right, which is composed of regular quadrilaterals. The poor appearance at the edges of the remapped grid results from the large zones in the donor mesh near its edges.

7 Performance is Linear in Number of Zones

Current global mesh remapping methods in typical codes run in time order N^2 or $N \log N$, where N is the number of zones in both meshes. Our method improves upon this. The algorithm performance is linear in the number of zones in both the old and new meshes.

The algorithm can be analyzed in three pieces; constructing the Eulerian

cycle, tracing the cycle to find the edge intersections points, and using the intersections list to compute areas. Constructing the Eulerian cycle is well known to be require order N_e time, where N_e is the number of edges in the mesh. Now we consider the other two components of the remapping algorithm.

It is helpful to define e_{1z} and e_{2z} as the average number of edges per zone in meshes 1 and 2, respectively. Also define z_{1e2} as the average number of zones in mesh 1 intersected by an edge of mesh 2. Then the average number of edges of mesh 1 checked against an edge of mesh 2 for intersection is $z_{1e2}e_{1z}$. We will define E_{21} to be the total number of edges checked in tracing mesh 2 through mesh 1;

$$E_{21} = N_{e2}e_{1z}z_{1e2} \tag{12}$$

where N_{e2} is the number of edges in mesh 2. Since N_{e2} is related to the number of zones in mesh 2 by the factor e_{2z}, the work required to trace mesh 2 through mesh 1 is just

$$E_{21} = Z_2e_{2z}e_{1z}z_{1e2} \tag{13}$$

where Z_2 is the number of zones in mesh 2. The work is linearly dependent on the number of zones in mesh 2. The same argument applies to tracing mesh 1 through mesh 2, so the total computational effort is of order $Z_1 + Z_2$.

Computing the areas of intersection is done by processing one edge segment at a time, with constant work for each segment. The number of segments is the number of edges times the number of zones crossed per edge. Hence the total work for both mesh traces is $Z_1e_{1z}z_{2e1} + Z_2e_{2z}z_{1e2}$, again of order $Z_1 + Z_2$. Since every step of the algorithm is linear in the total number of zones, the entire algorithm executes in linear time (see Fig. 3).

8 Future Work

We are currently extending the algorithm described here to unstructured three dimensional grids composed of arbitrary polyhedra. The bookkeeping is considerably more complicated but the same basic method of reducing the dimensionality and integrating the potential function still applies.

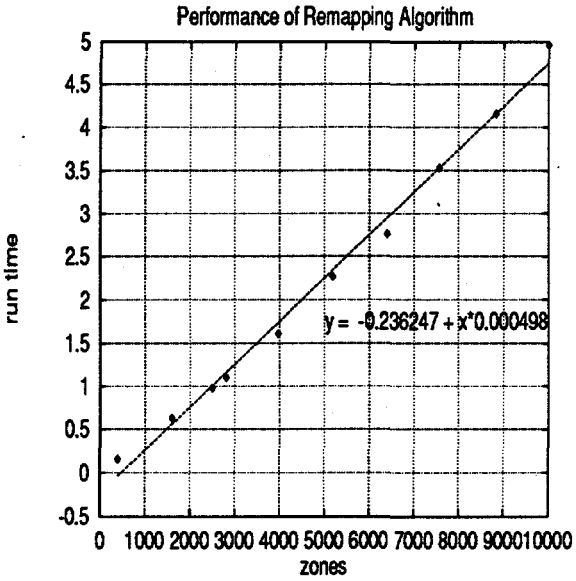


Figure 3: Run time in seconds versus the number of zones being remapped (on a 100 MHz MIPS R4000 processor). The remapping algorithm is remapping a mesh back on to itself, which is the worst degenerate case.

9 Acknowledgments

We would like to gratefully acknowledge the advice and support of Todd Palmer, at Oregon State University. This work was performed under the auspices of the U.S. Department of Energy, by Lawrence Livermore National Laboratory under Contract #W-7405-Eng-48.

References

[1] D.S. Bailey and B.bA. Wellnitz. La-10112-c: Rezoning workshop—1983. pages 1,54–60. Los Alamos National Laboratory, 1983.

[2] John K. Dukowicz. Conservative rezoning (remapping) for general quadrilateral meshes. *Journal of Computational Physics*, 54:411–424, 1984.

[3] John K. Dukowicz and J.W. Kodis. Accurate conservative remapping for arbitrary lagrangian-eulerian computations. *LA-UR 2646*, 1985.

[4] Bram Van Leer. Toward the ultimate conservative difference scheme ii. monotonicity and conservation combined in a second-order scheme. *Journal of Computational Physics*, 14:361–370, 1974.

[5] John D. Ramshaw. Conservative rezoning algorithm for generalized two-dimensional meshes. *Journal of Computational Physics*, 59:193–199, 1985.

[6] John D. Ramshaw. Simplified second-order rezoning algorithm for generalized two-dimensional meshes. *Journal of Computational Physics*, 67:214–222, 1986.