LA-UR 98-1697

CONF- 980769--

TITLE: MAINTAINING TETRAHEDRAL MESH QUALITY IN RESPONSE TO TIME-DEPENDENT TOPOLOGICAL AND GEOMETRICAL DEFORMATION

AUTHOR(S): Andrew Kuprat and Denise George, T-1, LANL

RECEIVED
OCT 05 1998
OSTI

SUBMITTED TO: Proceedings - at 6th International Conference on Numerical Grid Generation in Computational Field Simulation

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Maintaining Tetrahedral Mesh Quality in Response to Time-dependent Topological and Geometrical Deformation *

Andrew Kuprat
Denise George
Los Alamos National Laboratory, Theoretical Division, T-1
Los Alamos, NM 87545
email: kuprat@lanl.gov, dgeorge@lanl.gov

Abstract

When modeling deformation of geometrically complex regions, unstructured tetrahedral meshes provide the flexibility necessary to track interfaces as they change geometrically and topologically. In the class of time-dependent simulations considered in this paper, multimaterial interfaces are represented by sets of triangular facets, and motion of the interfaces is controlled by physical considerations. The motion of "interior" points in the conforming tetrahedral mesh (i.e., points not on interfaces) is arbitrary and may be chosen to produce good element shapes.

In the context of specified boundary motion driven by physical considerations, we have found that a rather large "glossary" of mesh changes is required to allow our simulation to survive all the transitions of interface geometry and topology that occur during time evolution. This paper will describe mesh changes required to maintain good element quality as the geometry evolves, as well as mesh changes required to capture changes in topology that occur when material regions collapse or "pinch off".

In the first category of mesh changes required to maintain good element quality under a physically induced flow, we have found that we require a combination of node merging, edge refinement, and face swapping to keep the simulation from prematurely terminating. Interestingly, we have also found that it is impossible to completely decouple the physical motion of the interface triangles from the conforming tetrahedral mesh that contains them, because of the inevitable presence of "all-surface" tetrahedra (i.e. tetrahedra all of whose vertices lie on the same multimaterial interface). These tetrahedra are very flat and have near zero volume and hence are prone to invert as the interface deforms. To prevent element inversion, we have found it is necessary to add artificial forces to the physically justified forces on the interfaces. These artificial forces keep the simulation "alive" until periodically the worst offending "all-surface" tetrahedra are removed by merging of their vertices. In addition to maintaining the "all-surface" tetrahedra, refinement and face swapping tools are employed to ensure good mesh quality.

The second category of mesh changes are those required to capture changes in interface topology. We have found that it is necessary to keep

* Work supported by the U.S. Department of Energy.

track of the various connected topological components and determine whether they are on the verge of disappearing or changing as the simulation evolves. For example, for all pairs of material regions that touch each other, we monitor the topological component consisting of connected interface triangles between the two materials. If the total surface area of one such component is about to go to zero, this marks an imminent physical topological change, which must be simulated by performing topological changes to the computational tetrahedral mesh. Similar strategies are required to detect other forms of topological change.

This paper will present a detailed description of mesh changes necessary for capturing the aforementioned geometrical and topological changes, as implemented in our code GRAIN3D, and will provide examples from a metallic grain growth simulation in which the normal velocity of the grain boundary is proportional to mean curvature.

"Gradient Weighted Moving Finite Elements" Applied to Metallic Grain Growth

Our application which defines a physically driven interface motion is metallic grain growth approximated by Gradient Weighted Moving Finite Elements. This application and approximation is explained in this section; in the following sections, we present developments needed for efficient movement of tetrahedra attached to the physically driven interfaces.

We use Gradient Weighted Moving Finite Elements [1,2,3] to move a multiply-connected network of triangles for the modeling of deformation of 3-D grains. In one model of metallic grain growth [4,5], interface surfaces obey the simple equation

$$v_n = K, \tag{1}$$

where v_n is the normal velocity of the interface, and K is the local mean curvature. We represent interfaces as parametrized surfaces:

$$\mathbf{u}(s_1, s_2) = \sum_{\text{nodes } j} \alpha_j(s_1, s_2)\mathbf{u}_j. \tag{2}$$

Here, (s_1, s_2) is the surface parametrization, the sum is over interface nodes j, $\alpha_j(s_1, s_2)$ is the piecewise linear basis function ("hat function") which is unity at node j and zero at all other interface nodes, and $\mathbf{u}_j$ is the vector position of node j.

We have that

$$\dot{\mathbf{u}}(s_1, s_2) = \sum_j \alpha_j(s_1, s_2)\dot{\mathbf{u}}_j, \tag{3}$$

and

$$v_n = \dot{\mathbf{u}}(s_1, s_2) \cdot \hat{\mathbf{n}} \quad (\hat{\mathbf{n}} \text{ is local surface normal}). \tag{4}$$

So

$$v_n = \sum_j (\hat{\mathbf{n}}\alpha_j) \cdot \dot{\mathbf{u}}_j. \tag{5}$$

In effect, we have that the basis functions j for v_n are $n_i\alpha_j$, where $\hat{\mathbf{n}} = (n_1, n_2, n_3)$. These basis functions are discontinuous piecewise linear, since the n_i are piecewise constant.

Gradient Weighted Moving Finite Elements minimizes

$$\int (v_n - K)^2 \, dS \tag{6}$$

over all possible values for the derivatives $\dot{\mathbf{u}}_j$. (The integral is over the surface area of the interfaces.) We thus obtain

$$\begin{aligned} 0 &= \frac{1}{2}\frac{\partial}{\partial \dot{u}_j^i} \int (v_n - K)^2 \, dS, \quad i \in \{1, 2, 3\} \\ &= \int (v_n - K) n_i \alpha_j \, dS. \end{aligned} \tag{7}$$

This leads to a system of $3N$ ODE's:

$$(n_i\alpha_j, n_k\alpha_l)\dot{u}_l^k = (K, n_i\alpha_j), \tag{8}$$

or

$$\mathbf{C}(\mathbf{u})\dot{\mathbf{u}} = \mathbf{g}(\mathbf{u}), \tag{9}$$

where $\mathbf{u} = (u_1^1, u_1^2, u_1^3, u_2^1, \ldots, u_N^3)^T$ is the $3N$-vector containing the x, y, and z coordinates of all N interface nodes, $\mathbf{C}(\mathbf{u})$ is the matrix of inner products of basis functions, and $\mathbf{g}(\mathbf{u})$ is the right-hand side of inner products involving surface curvature.

Although $\mathbf{g}(\mathbf{u}) = (K, n_i\alpha_j)$ appears ill-defined for piecewise linear manifolds, being the inner product of a distribution (K) with discontinuous functions $(n_i\alpha_j)$, we can replace it by a well-defined sum of surface tensions over the triangular facets of the interfaces using an integral identity for manifolds [1].

Artificial Grid Movement Equations

System (9) is integrated using a second-order implicit variable time step ODE solver [2]. However, it gives velocities of the interface nodes only $\left(\dot{\mathbf{u}} = (\dot{u}_l^k)_{,1\le l\le N}^{,1\le k\le 3}\right)$, and so the system must be enlarged to include velocities for M interior nodes that are not part of the interface. That is, we extend $\mathbf{u}$ to

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_{interface} \\ \mathbf{u}_{interior} \end{pmatrix}, \tag{10}$$

where $\mathbf{u}_{interface} = (u_l^k)_{,1\le l\le N}^{,1\le k\le 3}$, and $\mathbf{u}_{interior} = (u_l^k)_{,N+1\le l\le N+M}^{,1\le k\le 3}$. With this extension, we enlarge system (9) to be order $N+M$.

Since interface physics only tells us how to evolve the N interface nodes, we must "artificially" construct the extra elements in the enlarged $\mathbf{C}(\mathbf{u})$, $\mathbf{g}(\mathbf{u})$ to allow for orderly (tetrahedra orientation preserving) evolution of the mesh. That is, given a physically meaningful method of evolving the triangular interfaces, we are free to develop auxiliary equations for moving the attached tetrahedra with the only requirement being that these equations lead to efficient solution of the system (9), and that they maintain positive orientation of tetrahedra.

We have found that viable extra equations are formed by adding to the left-hand side of (9) the term

$$\epsilon_1 \tilde{\mathbf{C}}(\mathbf{u})\dot{\mathbf{u}} \equiv \epsilon_1 \{(\nabla\tilde{\alpha}_j, \nabla\tilde{\alpha}_l)\delta^{i,k}\}\dot{u}_l^k, \tag{11}$$

and to the right-hand side of (9) adding the term

$$\epsilon_2 \tilde{\mathbf{g}}(\mathbf{u}) \equiv -\epsilon_2 \nabla_{\mathbf{u}_j}\Big(\sum_{tets\ p} Q_p\Big). \tag{12}$$

Here ϵ_1 and ϵ_2 are set sufficiently small so that the artificial node movement equations have small magnitude in comparison to the magnitude of the physical interface node movement terms. With regard to the left-hand side term, $\tilde{\alpha}_j = \tilde{\alpha}_j(\mathbf{x})$ is a piecewise linear "hat function" defined for $\mathbf{x} \in \mathbb{R}^3$. $\tilde{\alpha}_j$ is 1 on the j'th node in the tetrahedral mesh, and zero on all other nodes. $\nabla\tilde{\alpha}_j$ is the gradient of $\tilde{\alpha}_j$ over the tetrahedra (i.e., $\nabla = \nabla_{\mathbf{x}}$, where

$\mathbf{x} \in I\!R^3$.) $\delta^{i,k}$ is the Kronecker delta, and $(\nabla\tilde{\alpha}_j, \nabla\tilde{\alpha}_l) \equiv \int \nabla\tilde{\alpha}_j \cdot \nabla\tilde{\alpha}_l \, d\mathbf{x}$, where integration is over the tetrahedra in the mesh.

Term (11) has the effect of viscously damping node movement in the mesh. In fact, it is easily seen to be equal to

$$\epsilon_1 \frac{\partial}{\partial \dot{u}_j^i} \int ||\nabla \mathbf{v}||^2 \, d\mathbf{x}, \tag{13}$$

where $\mathbf{v}(\mathbf{x}) = \sum_l \dot{\mathbf{u}}_l \tilde{\alpha}_l$ is interpolated mesh velocity. Inclusion of this term thus tends to minimize $\int ||\nabla \mathbf{v}||^2 \, d\mathbf{x}$ which is a measure of nonuniformity in the velocity of the mesh.

With regard to the right-hand side term (12), Q_p is a measure of "quality" of tetrahedron p, and $-\nabla_{\mathbf{u}_j}(\sum_{tets\ p} Q_p)$ is a "quality force" on node j in a direction which will cause the greatest increase in the qualities of tetrahedra that contain node j. For Q_p, we currently use

$$Q_p = \frac{\sum_{n=1}^{6} (L_n^{(p)})^2 \sum_{n=1}^{4} (A_n^{(p)})^2}{(V^{(p)})^2}, \tag{14}$$

where the $L_n^{(p)}$ are the edge lengths, the $A_n^{(p)}$ are the face areas, and $V^{(p)}$ is the volume of tetrahedron p. Q_p is a dimensionless quality measure which is minimal if p is a regular tetrahedron, but approaches infinity if p has a worsening aspect ratio. (Note: for Q_p, the "quality force" acts to increase quality, which is to say it acts to *decrease* Q_p.)

Thus our artificial grid forces move the grid by (1) acting to minimize nonuniformities in grid velocity, and (2) acting to continually improve grid element aspect ratios.

The artificial grid forces have the effect of necessarily overriding physically justified node movement when it is necessary to prevent inversion of tetrahedra. For instance, if a tetrahedron p has all 4 nodes on an interface, the motion given by (8) might cause the tetrahedron to invert, especially if the interface changes its sense of curvature. By adding the "quality force" (12), the tetrahedron will "lock up" at some minimum volume (dependent on ϵ_2) and will be prevented from inverting. The effect of this on the simulation is acceptable with regard to accuracy, since the "lock up" of a small number of tetrahedra simply removes a small fraction of numerical degrees of freedom from the simulation. The effect is further reduced if the locked up tetrahedra are effectively removed by merge and swap operations mentioned in the next section.

Grid Maintenance Operations

To model 3D processes which exhibit physically based boundary motion, mesh maintenance and mesh optimization tools are used to assure good element quality. Primitive grid operations provided by LaGriT, the Los Alamos Grid Toolbox [6], provide a basis for mesh maintenance and optimization. The "merge" primitive accepts as input lists of pairs of neighboring nodes: merge candidate nodes and survivor nodes. If the merge is completed, only one of the pair survives and the mesh connectivity is repaired to reflect that one node has been deleted. Before the merge takes place, LaGriT verifies that the merge will not cause tetrahedra to become inverted and that the node types and surface constraints of the survivor and merged nodes will lead to a legal merge. The "refine" primitive used in these simulations adds nodes at the midpoints of selected edges. LaGriT sets the node type and surface constraints of the added nodes by determining if the added node is in the interior, on a material interface or on an exterior boundary. The grid connectivity is repaired to include the new elements created by connecting the new node to the other vertices of the elements which contained the refined edge. Depending on what is desired by the user, the "recon" primitive interchanges connections to either improve a measure of the geometric quality of the elements [closely related to Q_p in (14)], or to maximize the number of elements satisfying the familiar "Delaunay" criterion.

The three primitives, "refine", "merge", and "recon" are combined into the LaGriT grid optimization operation called "massage". ("Massage" is similar to the algorithm presented in [7].) "Massage" accepts three arguments: a creation edge length, an annihilation edge length and a damage tolerance. Refinement is carried out such that no edge in the grid has length greater than the creation edge length. The "refine" primitive is invoked using a version of an algorithm due to Rivara [8]. In the algorithm, an edge marked for refinement is placed on a stack. The algorithm then checks that the elements containing the marked edge have no other edges longer than the marked edge. If longer edges are encountered, they are placed on the stack ahead of the marked edge. The process continues recursively. The refinement candidates are then popped off the stack and refined, resulting in a refinement pattern proceeding from the longest edges to the shortest; this pattern is desirable since it usually does not degrade element quality. "Massage" may attempt to "merge" pairs of points only if the resulting grid has no edge length greater than the annihilation length. The damage tolerance specifies the amount of change to the shape of a material interface that is permissible. If a node that sits on a material interface is merged out, the interface will become flatter at that point. If the distance from the merged node's original position to its projected position on the flattened interface is greater than the damage tolerance, the merge will not be allowed. Since "merge" and (less commonly) "refine" can produce poorly shaped tetrahedra, "recon" is used to restore well shaped elements.

Mesh Response to Topology Changes

As grain boundaries move, topology changes must be detected and the mesh must be modified to reflect these topological changes. The topology changes are detected by assembling and monitoring the rate of change of sets of topological components. To detect grain collapse, we assemble sets of connected elements of the same material; for interface surface collapse we assemble sets of connected interface triangles between two materials; for boundary surface detachment, we assemble sets of connected boundary triangles that lie on a given boundary surface; for triple line collapse where a line is surrounded by three or more materials, we assemble sets of connected edges. We monitor the rate of collapse of these sets, and when a collapse or detachment is imminent, the mesh is adjusted. We identify a neighborhood that completely surrounds the collapsing feature and assign a new material to the elements in this neighborhood. The encroaching material that is accumulating most rapidly is chosen to be the new material. Soon after the material reassignment, the curvature driven interface motion will effectively straighten the interfaces. Figure 1 is a schematic of three types of topological change. The first frame in each sequence shows the event as it is detected by GRAIN3D; the dotted line demarcates the neighborhood to receive a new material assignment. The second frame shows the mesh just after the material reassignment, and the third frame shows the mesh after the interfaces have straightened.

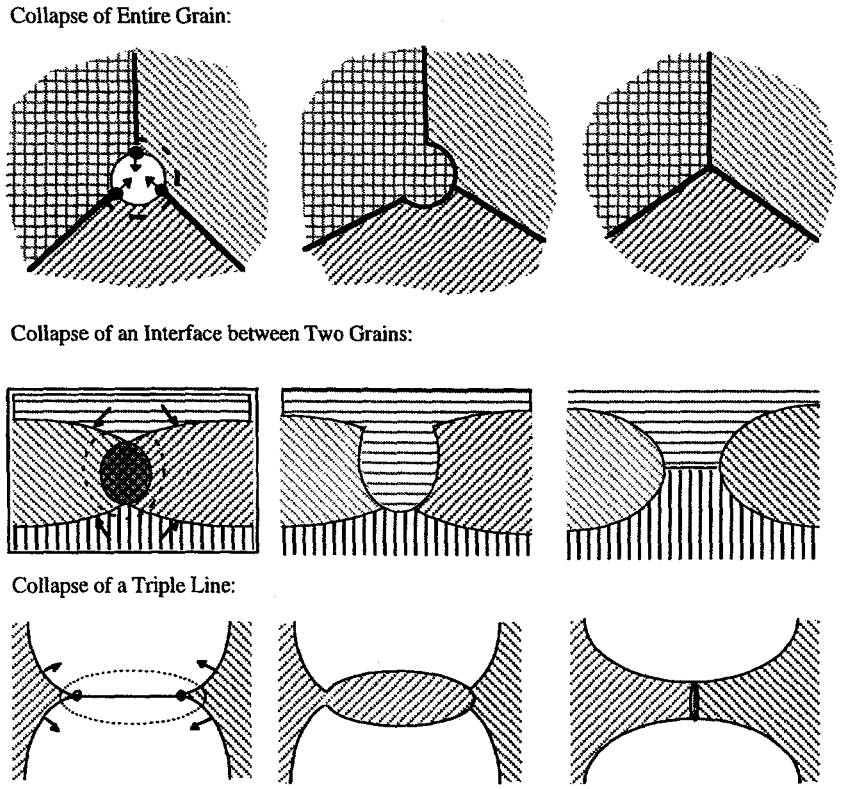

Figure 1. Topological collapse detection and resolution by GRAIN3D.

Numerical Example

As an application of GRAIN3D, in Figure 2 we model a 5 grain microstructure in which the surfaces of the grains move under mean curvature. The figure shows a time sequence in which one of the grains shrinks and disappears. When run to completion the simulation results in a single grain structure. Note that the grid is maintained throughout the simulation to keep the number and spacing of the nodes approximately constant. Although only the surface grid is displayed, the quality of the mesh is maintained in the volume as well.

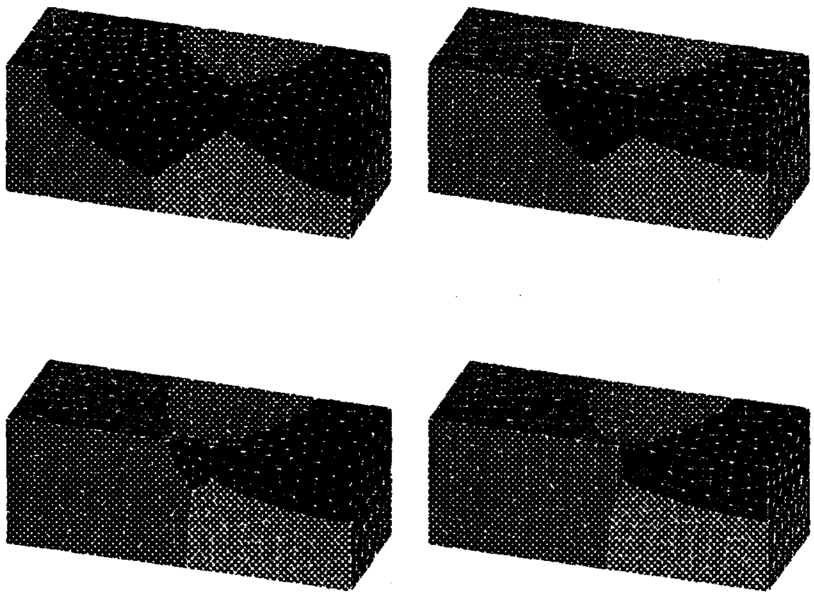

Figure 2. Grain evolution time sequence computed by GRAIN3D.

Conclusions

We have investigated the modeling of 3D processes involving physical boundary motion. We have found that for a front-tracking simulation to be effective, it must exhibit at least three capabilities. There must be a method of moving boundary nodes to track boundary motion, while adjusting positions of "interior" nodes to prevent inversion of elements. There must be a grid maintenance capability to assure well shaped elements and even node spacing. There must be a method of detecting topology changes and of modifying the grid accordingly. The Los Alamos code GRAIN3D incorporates these three capabilities.

References

[1] K. Miller, *A Geometrical-Mechanical Interpretation of Gradient-Weighted Moving Finite Elements,* SIAM J. Num. Anal., Vol. 34, pp. 67-90, 1997.

[2] N. Carlson and K. Miller, *Design and Application of a Gradient-Weighted Moving Finite Element Code, Part I, in 1D,* SIAM J. Sci. Comput., to appear.

[3] N. Carlson and K. Miller, *Design and Application of a Gradient-*

Weighted Moving Finite Element Code, Part II, in 2D, SIAM J. Sci. Comput., to appear.

[4] D.A. Porter and K.E. Easterling, *Phase Transformations in Metals and Alloys,* Great Britain: Van Nostrand Reinhold, 1988, pp. 130-136.

[5] A. Kuprat and J.T. Gammel, *Modeling Metallic Microstructure Using Moving Finite Elements,* http://xxx.lanl.gov/abs/physics/9705041.

[6] D.C. George, *LaGriT User's Manual,* http://www.t12.lanl.gov/~lagrit.

[7] A. Kuprat, *Creation and Annihilation of nodes for the moving finite element method,* Ph.D. thesis, Department of Mathematics, University of California, Berkeley, May 1992.

[8] M. Rivara, *New Longest-Edge Algorithms For the Refinement and/or Improvement of Unstructured Triangulations,* Int. J. Num. Meth. Eng., Vol. 40, pp. 3313-3324, 1997.