

Provided for non-commercial research and education use.  
Not for reproduction, distribution or commercial use.



**This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.**

**Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.**

**In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:**

**<http://www.elsevier.com/copyright>**



## Image-based finite element mesh construction for material microstructures

Andrew C.E. Reid<sup>a,b</sup>, Stephen A. Langer<sup>a,\*</sup>, Rhonald C. Lua<sup>c</sup>, Valerie R. Coffman<sup>c</sup>,  
Seung-Ill Haan<sup>d</sup>, R. Edwin García<sup>e</sup>

<sup>a</sup> Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899, United States

<sup>b</sup> Computer Integration and Programming Solutions, 4520 East West Highway, Suite 620, Bethesda, MD 20814, United States

<sup>c</sup> Materials Science and Engineering Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899, United States

<sup>d</sup> Samsung Corning Precision Glass, 544 MyungArm-Rhi, TaangJung-Myun, Asan-Si, ChungNam-Do, South Korea

<sup>e</sup> School of Materials Engineering, Purdue University, West Lafayette, IN 47907-2044, United States

Received 8 January 2008; received in revised form 8 February 2008; accepted 16 February 2008

Available online 18 April 2008

---

### Abstract

One way of computing the macroscopic behavior of a material sample with complex microstructure is to construct a finite element model based on a micrograph of a representative slice of the material. The quality of the results produced with such a model obviously depends on the quality of the constructed mesh. In this article, we describe a set of routines that modify and improve the quality of a 2D mesh. Most of the routines are guided by an effective element “energy” functional, which takes into account the shape quality of the elements and the homogeneity of the elements as determined from an underlying segmented image. The interfaces and boundaries in the image arise naturally from the segmentation process. From these routines, we construct a close-to-automatic mesh generator that requires only a few inputs, such as the linear sizes of the largest and smallest features in the micrograph.

© 2008 Elsevier B.V. All rights reserved.

PACS: 02.70.Dh; 02.70.–c

Keywords: Meshing; Mesh refinement; Microstructures; Finite element modeling; Shape quality; Homogeneity

---

### 1. Introduction

Finite element modeling is a technique which is at its best where analytical models are inapplicable because of the complex spatial geometry of the modeling domain. Even so, most finite element packages require as input a numerical representation of the model geometry in terms of simple building blocks. The boundaries of the domain are described by points, straight lines, planes, and simple curves. In materials science, the starting point for the modeling effort is often a micrograph or other “analog” representation of the structure. An example is shown in Fig. 1. For these types of images, converting the boundaries to a

simple numerical representation by hand is a tedious task. In this paper, we will describe how the software package OOF2 (named for “Object Oriented Finite-Elements”, version 2) [1–4] circumvents this difficulty by creating meshes directly from images, simultaneously identifying boundaries in the image and bringing the finite element mesh into correspondence with these boundaries. With a unique, image-based, adaptive meshing technique, OOF2 is capable of parsing experimental data relating to polyphase, polydomain materials with complex geometries into a representation that is appropriate for use in a finite element simulation. Using this method, an explicit mathematical description of the image boundaries is not required, and is not explicitly constructed.

The OOF tool (in its original version, OOF1, and the current OOF2) has been used successfully in a wide variety

---

\* Corresponding author.

E-mail address: [stephen.langer@nist.gov](mailto:stephen.langer@nist.gov) (S.A. Langer).

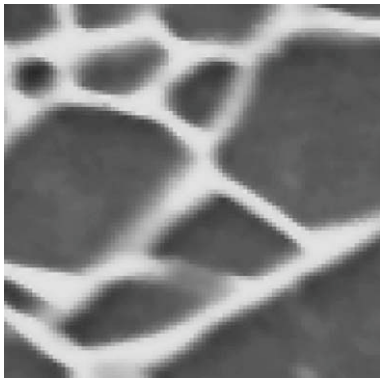


Fig. 1. *Microstructure image example*: A micrometer scale SEM image of plasma-etched  $\text{Si}_3\text{N}_4$ , provided by Chun-Hway Hsueh at Oak Ridge National Laboratory [5].

of applications, including studying the relationship between thermal properties and the structure of plasma-sprayed zirconia coatings [6], the effects of microstructure on the macroscopic mechanical properties of glass-matrix composites [7], the role of texture in the macroscopic response of polycrystalline piezoelectric materials [8], and the modeling and design of electrode microstructures in rechargeable lithium-ion batteries [9].

One option for forming a finite element mesh from an image is to do so directly, defining a square element for each pixel [10,11]. This has two major pitfalls. First, it usually creates too many elements – homogeneous regions can adequately be described by larger elements. Second, it introduces jagged edges where the boundary in the real material is smooth, which can lead to artificial features such as stress concentrations at pixel corners. It is useful to distinguish between the process of making a mesh of the image itself, which this direct approach does, and the process of making a mesh which approximates the underlying microstructure, using the image as necessarily approximate source data. The latter process gives higher quality meshes and is not encumbered by the discretization process that created the image [1].

The OOF2 meshing scheme begins with a coarse, regular, well-formed, space-filling mesh on the image, and then brings that mesh into correspondence with the image by a series of mesh-modifying steps which preserve the space-filling and well-formed character of the mesh. Mesh-modifying steps may refine elements, replacing them with smaller elements, or may move nodes and boundaries around to align them with image features. The elements are generally at least several pixels in size, which promotes efficient use of computational resources, and avoids accidental modeling of image artifacts. User judgement may be employed in selecting mesh modification steps and their parameters to help ensure that it is the underlying physical structure which drives this process. Because the well-formed, space-filling character of the mesh is preserved, unmeshable voids cannot arise, and illegal (inverted or concave) elements can be avoided.

## 2. Image-based meshing

The starting point for the OOF2 meshing scheme is an image which has been segmented, that is, broken up into distinct sets of pixels, each of which corresponds to a homogeneous part of the image. These image parts presumably are microstructural features such as grains or inclusions for which the bulk material properties are known (or can be assigned.) Image segmentation is a rich topic beyond the scope of this paper. For our purposes here, we will assume that a segmentation of the image exists, and also treat “pixel group”, “pixel category”, and “material” as synonyms, although they have slightly differing technical meanings within the OOF2 program.<sup>1</sup>

The goal of the meshing process is then to create a mesh whose element boundaries lie approximately along the pixel group boundaries, and for which the elements themselves are approximately regular in shape and homogeneous, enclosing pixels of exactly one pixel group. The technique used is to create an initial mesh that is a regular, space-filling grid of rectangles or triangles, with user-specified dimensions, and then optimize this mesh and align it with the boundaries of the microstructure using a variety of tools described below. Several methods move the nodes in order to improve homogeneity or shape quality. Other methods change the topology of the mesh by refining inhomogeneous elements, merging homogeneous elements, or correcting for badly shaped elements.

### 2.1. Quantifying the quality of the mesh

For an automated meshing procedure to work, a figure of merit, or a quantitative measure of the quality of the elements in the mesh, must be introduced that has as few components as possible. This quantitative measure must reflect the degree to which the mesh represents the microstructure, and must also reflect the degree to which the mesh will give rise to good convergence behavior in the finite element solution step.

To this end, we define two element functionals, which we call “energies”. The shape energy quantifies the quality of the shapes of elements, and the homogeneity energy measures how well the mesh matches the pixel regions. We use the word “energy” because some of our mesh modification routines move the mesh nodes as if they were physical particles with potential energies given by the shape and homogeneity functionals. For example, the anneal routine (see Section 2.2.1) moves nodes randomly and accepts moves that lower the energy. A mesh is a good finite ele-

<sup>1</sup> A “pixel group” is a set of pixels from the image. Each pixel may belong to many pixel groups simultaneously. A “material” defines the set of physical and crystallographic properties belonging to the microstructure at the location of a pixel. Each pixel can belong to at most one material. A “pixel category” is a label indicating the set of groups and materials to which a pixel belongs. All pixels with the same category have the same material and belong to the same groups.

ment representation of the microstructure if a weighted average of the two energies, summed over elements, is low.

For the shape energy, it is useful to have an energy function which has high values for large aspect ratios, since high-aspect-ratio elements can lead to slow convergence of the finite element solver[12]. With such a function, the energy minimization scheme will favor low-aspect-ratio elements. Equilateral triangles and squares, which are ideal, have a shape energy of zero while thin and flat elements have higher energies. Degenerate elements with colinear corners have the maximum shape energy, which is normalized to 1.0. For triangular elements, the shape energy is taken to be

$$E_{\text{shape}} = 1 - 4\sqrt{3} \frac{A}{L_0^2 + L_1^2 + L_2^2}, \quad (1)$$

where  $A$  is the area of the triangle and  $L_i$  are the lengths of the sides. For quadrilateral (quad) elements, OOF calculates a quality measure for each corner [13]

$$q = 2 \frac{A_{\parallel}}{L_0^2 + L_1^2}, \quad (2)$$

where  $A_{\parallel}$  is the area of the parallelogram formed by the two edges adjacent to the corner in question.  $L_0$  and  $L_1$  are the lengths of the adjacent sides. OOF2 calculates the shape energy for a quad by constructing a weighted average from the  $q$ 's at the corner with the minimum  $q$  ( $q_{\text{min}}$ ) and at the corner opposite it ( $q_{\text{opp}}$ ).

$$E_{\text{shape}} = 1 - ((1 - w_{\text{opp}})q_{\text{min}} + w_{\text{opp}}q_{\text{opp}}), \quad (3)$$

where  $w_{\text{opp}} = 10^{-5}$  is an arbitrary small parameter. The weighted average is taken to ensure that every node movement affects the energy. If instead  $E_{\text{shape}}$  were taken to be  $1 - q_{\text{min}}$ , without involving  $q_{\text{opp}}$ , it would be possible to construct a mesh in which the shape energy is completely independent of the position of some nodes. Some of the mesh modification routines behave badly in the presence of such “free” nodes.<sup>2</sup>

The homogeneity (see Fig. 2) is a measure of how close an element is to enclosing a region that contains a single material:

$$H = \frac{\max_i \{a_i\}}{A}, \quad (4)$$

where  $a_i$  is the area within the element that belongs to material  $i$  and  $A$  is the total area of the element. (When computing  $a_i$ , it is important to count the fractional area of pixels that intersect the element boundaries, so that the functional is a continuous function of the node positions.) If an element is filled with only one material,  $H = 1$ . The corresponding homogeneity energy is defined by

$$E_{\text{hom}} = 1 - H. \quad (5)$$

<sup>2</sup> The precise value of  $w_{\text{opp}}$  is unimportant. It needs to be large enough so that it is not lost in the noise, and small enough that  $E_{\text{shape}}$  is dominated by  $q_{\text{min}}$ . We have found that a value of  $10^{-5}$  works well.

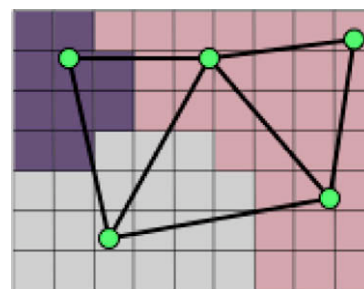


Fig. 2. If the three different pixel colors represent different materials, then the triangle in the upper right is completely homogeneous ( $H = 1.0$ ), the central triangle is inhomogeneous ( $H \approx 0.5$ ) and the leftmost triangle is even more inhomogeneous ( $H \approx 0.3$ ).

These two measures can be combined into an effective element energy

$$E = \alpha E_{\text{hom}} + (1 - \alpha) E_{\text{shape}}, \quad (6)$$

where  $\alpha$  is a tuneable parameter. In several of the mesh adaptation routines, the user may adjust  $\alpha$  according to whether the priority is improving shape quality or homogeneity. We have often found it useful to first use a large  $\alpha$  (say 0.8), which quickly moves mesh edges onto material boundaries, and then use a small value (say 0.2) to fix the shapes of the elements.

## 2.2. Adaptive methods

The mesh adaptation routines used in OOF2 can be classified in a number of ways, summarized in Table 1. The primary distinction is between routines that (1) move nodes but preserve the connectivity or topology of the starting

Table 1  
Summary of the OOF mesh adaptation routines

Routine	Preserving	Homog.	Shape	MC	Snap	Fix
Anneal	★	★	★	POT		
Fix illegal elements	★		★	O		★
Merge triangles			★	O		
Rationalize			★	O		★
Refine		★	★			
Relax	★	★	★			
Smooth	★		★	OT		
Snap Anneal	★	★		OT	★	
Snap Nodes	★	★		O	★	
Snap Refine		★	★		★	
Split Quads		★		O		
Swap Edges		★	★	O		
Manual node motion	★	★	★		★	★

*Preserving* indicates that the routine preserves mesh topology. *Homog.* indicates that the routine primarily improves element homogeneity. *Shape* indicates that it primarily improves element shape. *MC* indicates how the routine uses Monte Carlo methods: *P* means that it chooses new node positions randomly, and *O* means that it addresses the nodes in a random order, and *T* means that it accepts or rejects moves randomly from a Boltzmann distribution. *Snap* indicates that new node positions must lie on material boundaries. *Fix* indicates that the routine fixes problem elements.

mesh (e.g. Anneal, Snap Nodes), and (2) routines that change the topology by adding, removing, or reconnecting nodes (e.g. Refine, Snap Refine). Routines in both groups are used to improve overall element energy.

Topology-preserving routines can improve the homogeneity or shape energy of the elements in the mesh without increasing the number of elements. Unless the initial grid is sufficiently fine, they usually *cannot* be used to resolve the small features in a microstructure. Non-preserving routines are used to achieve this task by subdividing existing elements and segments and spawning new elements and nodes. Once the desired element resolution is reached, topology-preserving routines can again be used to further improve the mesh.

The columns marked *Homog.* and *Shape* in Table 1 indicate whether the routines are best at improving the homogeneity or shape of the mesh elements. Routines with a star in both columns can be used for either purpose, or both at once, by adjusting  $\alpha$  in Eq. (6).

Because of the similarity of topology-preserving routines to steps in a molecular dynamics simulation, some of these routines (e.g. Anneal, Smooth, Snap Anneal) incorporate a type of Metropolis Monte Carlo method [14]. The Monte Carlo randomness appears in three ways: (1) the algorithm may address nodes or elements sequentially in a random order; (2) nodes may be moved to randomly chosen positions; and (3) mesh modifications may be accepted or rejected at random, using a Boltzmann probability distribution. The quantity  $\Delta E$  in the exponent of the Boltzmann factor

$$P = \exp(-\Delta E/T) \quad (7)$$

is the change in the energy (Eq. (6)) of the affected elements. A change in the shape of some elements can be accepted even if it leads to an increase in the overall element energy, with a probability given by  $P$ . This analogy with molecular dynamics simulations can be pursued further by explicitly applying fictitious forces on the nodes that may drive the elements to improve their homogeneity or shape (e.g. the relax routine).

The *Snap* column in Table 1 marks routines that place nodes precisely at the interface between two different pixel groups. Snap routines are better at improving element homogeneity than element shape. Routines with a star in the *Fix* column of the table are specialized for correcting or improving an element or group of elements created by the other routines.

Most of these routines can be flexibly applied to an explicitly selected set of elements, segments or nodes. Most of the routines can also be targeted to elements or segments that satisfy certain criteria, such as having a homogeneity below a certain threshold.

In the following sections, we describe each mesh adaptation routine used in OOF2. We also describe a combination of these routines that seems effective in creating a mesh for a categorized microstructure. This sequence of routines is

applied in OOF2 after providing just a little input and once started requires no feedback from the user.

### 2.2.1. Anneal

The *Anneal* routine moves nodes to random positions (see Fig. 3(1A) and (1B)), accepting or rejecting moves according to a given criterion. It is similar to a simulated annealing simulation in statistical mechanics, from which it gets its name. Instead of minimizing the free energy of a system of particles, it minimizes the effective energy of a mesh. Annealing is useful when the mesh elements are small enough to resolve the features of a microstructure, but are not in quite the correct positions. It can usually find a happy medium in situations in which a trade-off must be made between element shape and homogeneity.

The general procedure for a single iteration of anneal is as follows:

- (1) Create a list of target nodes and reorder them randomly to remove any potential artifacts from the original ordering of nodes. This reordering is repeated at every iteration.
- (2) Give each node in the list a single chance to move to a randomly assigned new position. OOF2 computes the new position from

$$x_{\text{new}} = x_{\text{old}} + \delta x$$

$$y_{\text{new}} = y_{\text{old}} + \delta y$$

$\delta x$  and  $\delta y$  are random numbers chosen from a Gaussian distribution of width  $\delta$  and mean 0.0. Fig. 3(1A) shows a target node that is about to move to a new position. Before making the move, OOF2 computes the total effective energy of all the neighboring elements of the node ( $E_1, E_2, E_3, E_4$ ).

- (3) After moving each node, an acceptance criterion is used to decide whether the move is acceptable. Any moves that create illegal elements (see Section 2.2.5) are automatically rejected. OOF2 has a number of different criteria to choose from: accepting all moves, only those that decrease the energy, or only those that decrease energy and meet specified additional constraints on homogeneity and shape.
- (4) If a legal move is unacceptable according to the acceptance criterion, OOF2 may still accept the move if the annealing is being done at a non-zero temperature  $T$ . The parameter  $T$  sets the effective temperature of the annealing process. These moves are accepted with the probability  $P$  given in Eq. (7), where  $\Delta E$  is the difference in the effective energies of the new and old element configurations.

Successful annealing usually requires a number of iterations. On each iteration, OOF2 makes one attempt to move each node. OOF2 can be set to perform a fixed number of iterations of anneal or to stop after some condition is satisfied. As with conventional simulated annealing or

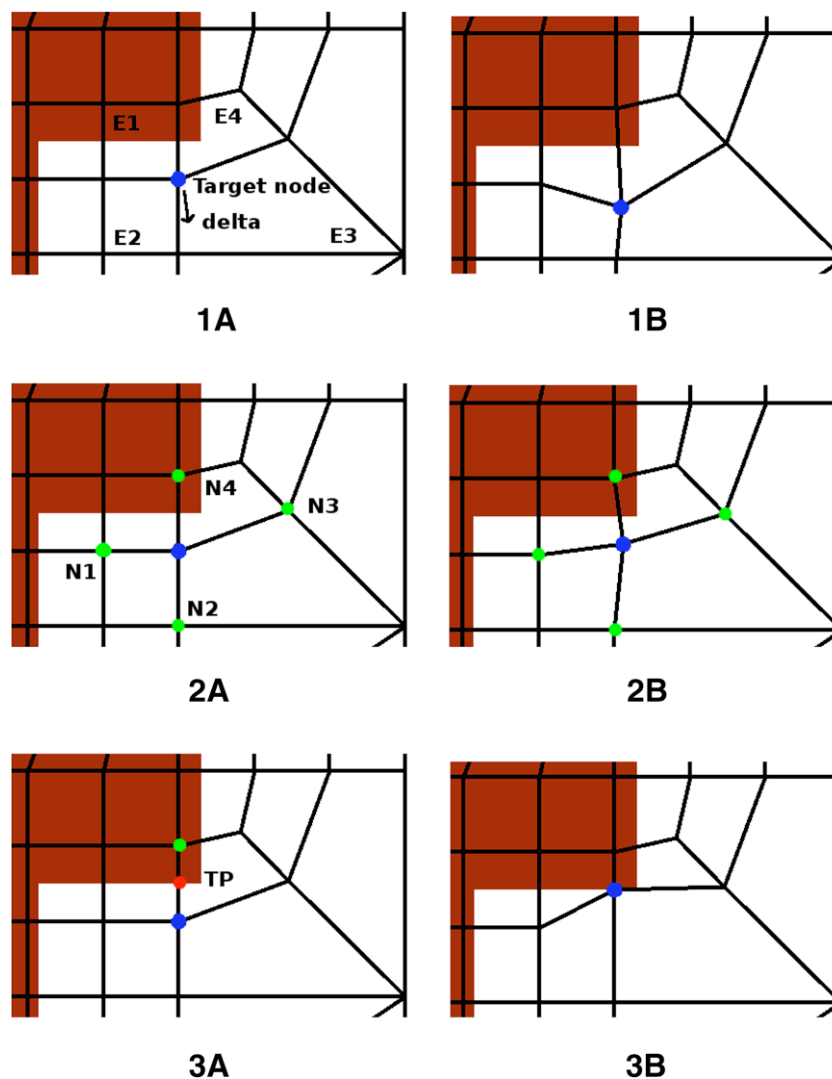


Fig. 3. Illustration of some node moves used in OOF2. (1A) and (1B) show a trial move of the Anneal routine. A target node is displaced by a random vector whose components are taken from a Gaussian distribution. (2A) and (2B) show the trial move for the Smooth routine. The target node is placed at the center of the four neighboring nodes  $N1$ ,  $N2$ ,  $N3$  and  $N4$ . (3A) and (3B) show the trial move for the Snap Anneal routine. The target node is shifted to the transition point ( $TP$ ) along one of the edges incident at the target node.

Monte Carlo simulations, if the change in energy or the acceptance rate gets too small, the annealing process has reached a point where it is no longer improving the mesh effectively.

### 2.2.2. Smooth

The Smooth routine implements a “smart Laplacian” algorithm[15]. It moves nodes to the average position of their neighbors (see Fig. 3(2A) and (2B)), if the move meets a given criterion (such as lowering the local energy) and does not create illegal elements. This tends to smooth out gradients in node density and improve element shapes. Typically, Smooth ceases to make substantial changes after about 3 iterations over the mesh. The general procedure for a single iteration is very similar to that for Anneal, except that in step 2, each target node is given a single chance to move to the average position of its neighbors. For this purpose, neighbors are defined as

nodes that share a segment with the target node ( $N1$ ,  $N2$ ,  $N3$ ,  $N4$ ).

Smooth is most useful in applying the final touches to a mesh. After the mesh has been adjusted so that all of the material boundaries correspond to element edges, applying smooth to all of the nodes that *are not* on material boundaries will improve the quality of the elements in the interior of each region, without affecting homogeneity.

### 2.2.3. Snap Anneal

Snap Anneal is very similar to anneal. In Snap Anneal, the trial move consists of placing the target node at a nearby pixel boundary, called a transition point, located along a segment that contains the target node as an endpoint (see Fig. 3(3A) and (3B)). Snap Anneal would not explore as many configurations as Anneal, but it can be useful when a mesh is nearly correct but needs to be forced into alignment.

### 2.2.4. Snap Nodes

Snap Nodes moves nodes to improve the elements' homogeneities. If an element edge crosses over regions of the microstructure belonging to different pixel categories, then Snap Nodes tries to move one of the element corners to the crossing point. These points are precisely the transition points used in Snap Anneal and illustrated in Fig. 3(3B). Snap Nodes differs from Snap Anneal by having correlated snaps of two nodes. That is, more than one node of an element may be moved at the same time in order to increase the homogeneity.

The general procedure for snapping nodes is as follows:

- (1) Scan the current mesh to find candidate nodes for snapping.
- (2) Loop over the elements containing snappable nodes, and identify the transition points along each segment of each element.
- (3) Find all possible snaps involving one or two nodes of a single element, assigning a priority to each according to the arrangement of its transition points. The priorities are listed in Fig. 4.
- (4) Starting with the highest priority snaps (and choosing randomly from snaps with equal priorities), attempt to actually move the nodes. The move is accepted or rejected according to a given criterion. When more than one snap is possible for one element, all snaps are attempted and the best is chosen. After successfully snapping the nodes of an element, the nodes of any neighboring snappable elements are attempted, regardless of their priority. (This ensures that successive nodes along a material boundary are “zipped” up consistently, without the risk of making incompatible choices at different points on the boundary.)

Snapping nodes tends to produce badly shaped elements when the weighting in the element energy ( $\alpha$ ) favors homogeneity (see Eq. 6). However, Snap Nodes is really a homogeneity-oriented operation, and works best when given a large  $\alpha$ . It is usually best to allow it to create badly shaped elements, and to clean up afterwards by applying the Rationalize routine.

### 2.2.5. Fix Illegal Elements

Illegal elements are elements having corners that subtend an angle greater than  $180^\circ$  or less than  $0^\circ$ . For quadrilateral elements, these include elements that have non-convex shapes. Fix Illegal Elements modifies and corrects these elements by moving one or more nodes in the element to the average position of its neighboring nodes, just like in smooth. The move is accepted or rejected solely by whether or not it reduces the number of illegal elements in the vicinity of the node.

In many cases, it is sufficient to fix an element by applying this smoothing procedure to the nodes at the offending corners (those with angles outside the range  $0-180^\circ$ ). However, the procedure may fail when the smoothing is applied to a node located at a corner with an angle very close to zero. In this case, the nodes at the other corners should be tested and moved as well.

The Fix Illegal Elements procedure is illustrated in Fig. 5. On one pass through the mesh, Fix Illegal Elements attempts to move each node in an illegal element once, choosing nodes in a random order. It may not be successful on the first pass, so it repeats the process until there are no more illegal elements, or when the routine has tried too many times.

After an element has been fixed, it is possible that the mesh is no longer a good representation of the microstruc-

Priority	1	2	3	1	2
Type					
Possible Snaps					

Fig. 4. Snapping nodes.

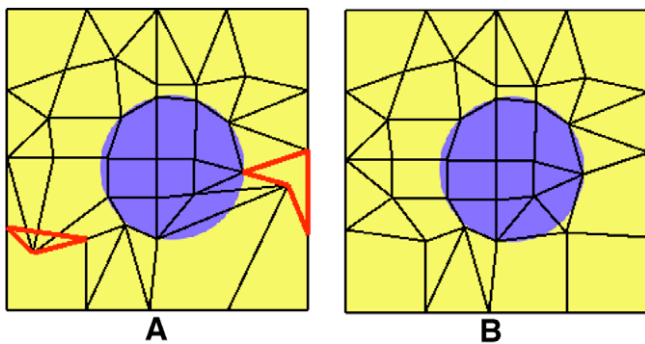


Fig. 5. Before (A) and after (B) application of the fix illegal elements routine on a mesh. In (A), the elements that are fixed include a flipped triangle (left), and a non-convex quadrilateral. Both were intentionally created using OOF2's interactive graphical toolboxes to select and move nodes.

ture. This is because the Fix Illegal Elements routine is a brute force repair that does not pay attention to either element homogeneity or shape.

### 2.2.6. Relax

Relax is like a deterministic and cooperative version of Anneal. One problem with annealing is that it operates on only one node at a time, and it is possible for the algorithm to fail to find low energy states that require moving two nodes together. For example, Fig. 6 shows an initial mesh (A), and an annealed version of it (B) (annealed for 50 iterations using the energy functional, Eq. (6), with  $\alpha = 0.8$ ). If all of the nodes on the bottom row had been moved up or down together, the boundary of the blue region would have been resolved correctly. The annealing did not give the best solution because some nodes were moved up and others were moved down, and the two local solutions could not be combined into a global one.

OOF2's Relax algorithm attempts to solve this problem by moving all nodes at once, by constructing and solving an artificial time-dependent elasticity problem on the mesh. It gives each element an isotropic elastic modulus and an elastic driving force with two components. The first component is an isotropic compression proportional to the element's homogeneity energy. This causes inhomogeneous

elements to shrink. The second component is like an anisotropic stress-free strain, proportional to the traceless part of the element's moment of inertia tensor (assuming equal masses concentrated at the nodes). This drives the elements' shapes toward squares and equilateral triangles. The relative weight of the two terms can be adjusted with the  $\alpha$  parameter. Using these two driving forces and a mobility parameter (which only serves to normalize the time scale), the positions of the nodes are computed for a fixed number of time steps.

Fig. 6(C) shows the result of relaxing the original mesh (A) for 7 time steps. The routine did a poor job on the red and yellow circles, but performed better than anneal on the blue stripe. The resolution of the circles would have been improved if we had started with a finer mesh.

One problem with the Relax algorithm, visible in Fig. 6, is that it can create very small or badly shaped elements if run for too many time steps. It is usually best not to use relax by itself, but to use it to nudge a mesh toward the desired solution, and follow it with Anneal or Snap Nodes.

### 2.2.7. Refine

Refine is a routine that chops its target elements and their neighbors into smaller pieces (see Fig. 7(B)). This adds more degrees of freedom to the mesh, which allows it to adapt better to the microstructure. Refinement by itself is rarely sufficient to create an acceptable mesh – it must be combined with other routines that move nodes or place nodes at pixel boundaries, such as Anneal or Snap Refine.

Meshes are refined in two steps. First, the element edges are marked for subdivision. The marking indicates whether the edge is to be bisected or trisected. Second, each element with marked edges is replaced by a suitable assembly of smaller elements. This two-pass approach allows element replacement to be a purely local process, using only information contained within an element and its edges, and ensures that there are no compatibility problems when subdividing neighboring elements.

The user must specify the following parameters when refining a mesh:

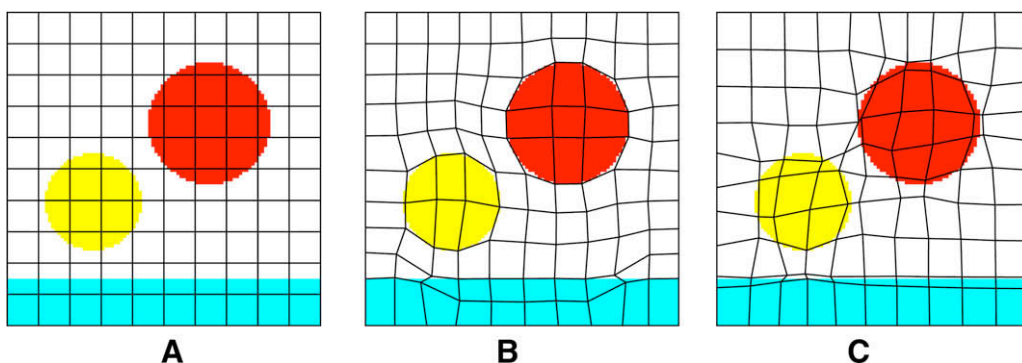


Fig. 6. A crude mesh (A) and the results of Annealing (B) and Relaxing (C).

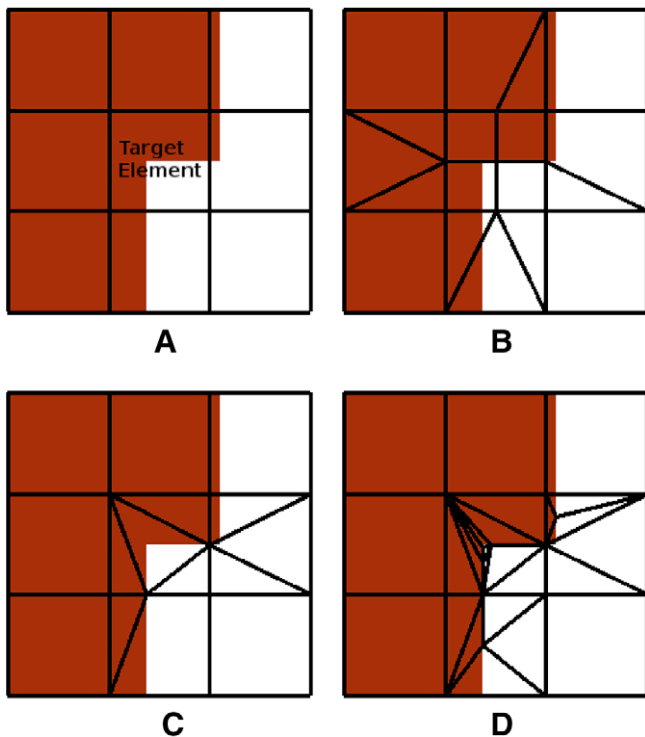


Fig. 7. Illustration of the refinement of a target element using refine and Snap Refine. (A) The target for refinement is the region defined by the central element. (B) One application of the refine routine splits elements and bisects edges. (C) One application of Snap Refine subdivides elements and splits edges at transition points. (D) Four successive applications of Snap Refine improves the homogeneity but produces thin elements.

- (1) A *qualification criterion* specifying which elements are to be considered for refinement. For example, elements smaller than a given area can be excluded from future operations.
- (2) A *target specification*, indicating which element edges to mark. For example, the user can choose to mark all edges of heterogeneous edges, or the long edges of skinny elements.
- (3) A *refinement ruleset*, which specifies how the marked edges are to be divided (bisected or trisected) and how to place new elements within an old one. The ruleset determines whether or not the refinement preserves element shape: it is possible to replace triangles with triangles and quads with quads, or to generate a mixed mesh.

The steps involved in refining a mesh are:

- (1) Find the element edges to be marked, according to the qualification criterion and the target specification.
- (2) Mark the edges for bisection or trisection, according to the refinement ruleset.
- (3) Sometimes, the ruleset requires extra segments to be marked in neighboring elements. In these cases a second pass is made through the mesh, marking additional segments. (For example, in general it is not

possible to bisect edges and maintain a triangle-free mesh without bisecting extra edges that are not part of the originally marked set.)

- (4) Using the ruleset, replace each element with a suitable collection of smaller elements, such that each marked edge of the old element turns into a number of edges of new elements. The rules sometimes allow more than one way of subdividing an element. In those cases, the total effective energy of the new elements is computed, and the configuration with the lowest energy is used.

#### 2.2.8. Snap Refine

Snap Refine attempts to combine the most desirable features of Refine and Snap Nodes into a single method. The routine operates like Refine, except that the element segments are split by points located at the interface of two different pixel categories (see Fig. 7(C)). These points become the corners of new elements that subdivide the original elements.

Snap Refine can yield a mesh with a comparable homogeneity to a mesh produced by Refine, but often with significantly fewer elements. Snap Refine can also be more flexible than Snap Nodes in fitting a mesh to a microstructure, because Snap Refine introduces new nodes and edges that can be made to follow the boundaries of pixel categories and resolve smaller features in the microstructure.

It may seem obvious that one should always use Snap Refine instead of Refine. Unfortunately, Snap Refine can lead to the creation of very thin elements (see Fig. 7(D)). The creation of such badly shaped elements may be prevented in some cases by telling Snap Refine to not place a new node within a certain distance of one of the corners of an element (although this is not a scale-invariant solution). In general, iterating Snap Refine produces very bad results, with a lot of very thin elements. A good strategy is often to use the regular Refine routine first, and to apply Snap Refine just once at the end to Snap Nodes and edges to the interface.

Snap Refine finds at most two transition points along an edge, by scanning the edge starting from each endpoint. The subdivision of an element may also introduce new nodes within the element, which will also be located at the interface of pixel categories if possible.

#### 2.2.9. Rationalize

Rationalize fixes badly shaped elements in a mesh by either removing them or modifying them and their immediate neighbors, as shown in Fig. 8. It should be applied whenever other mesh modifications have resulted in badly shaped elements.

Rationalize has three sub-algorithms (called rationalizers):

- (1) Remove short sides: Eliminate the shortest side of a quadrilateral by merging two nodes and replacing it with a triangle, if the ratio of the lengths of the sec-

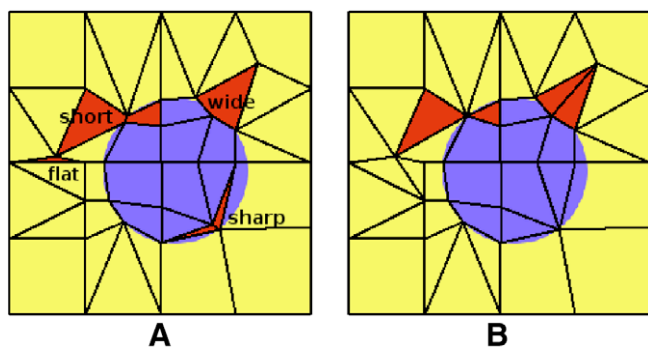


Fig. 8. Before (A) and after (B) application of the rationalize routine on the highlighted elements. In (A), the elements that are fixed by the rationalize routine include (clockwise from left) a flat triangle, two quadrilaterals that share a short side, a wide quadrilateral, and two sharp or thin triangles.

ond-shortest and shortest sides is greater than a user-defined amount. If the other element sharing the two nodes is a quadrilateral, it is replaced with a triangle as well. If it is a triangle, it is eliminated.

- (2) Split wide quads: Split quadrilaterals with large interior angles into two triangles, if the interior angle is greater than a specified size.
- (3) Remove bad triangles: Remove triangles with extreme interior angles, either by merging two nodes or by replacing a triangle and one of its neighbor elements with a set of new triangular elements. This is applied to all triangles with interior angles outside of a user-defined range.

### 2.2.10. Split Quads and Swap Edges

Split Quads and Swap Edges are useful ways of cleaning up a mesh after refining or annealing. They perform small local modifications to mesh elements that can be used to improve element homogeneity along material boundaries.

When a quadrilateral straddles a boundary along its diagonal, as shown in Fig. 9(A), annealing and other node motion routines will not be able to match element edges to the boundary. Split Quads looks for such quadrilaterals, and divides them into two triangles, as in Fig. 9(B).

Swap Edges performs a similar task when the shared edge of a pair of triangles or quadrilaterals intersects a

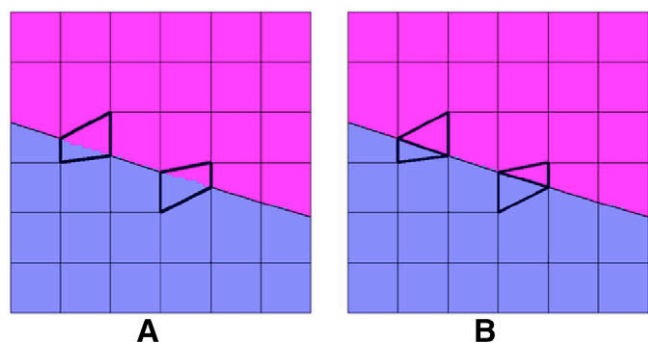


Fig. 9. Before (A) and after (B) application of the Split quads routine.

material boundary. In this case, the homogeneity of the pair can often be improved by merging the triangles into a quadrilateral and redividing it along the other diagonal of the quad, as shown in Fig. 10. The figure illustrates that swapping edges can also improve the shape energy of pairs of quadrilateral elements.

### 2.2.11. Merge Triangles

Merge Triangles merges two triangular elements into one quadrilateral element, as illustrated in Fig. 11. It is a simple way of improving a mesh's shape energy and reducing the number of elements (which may or may not be desired, depending on the specific problem at hand). It should only be applied to homogeneous triangles in order to avoid decreasing homogeneity.

### 2.3. Automatic mesh generation

OOF2 has a close-to-automatic mesh generation feature built from a sequence of the mesh adaptation routines described in the previous sections. It was originally constructed in order to use OOF2 as part of the automated MatCASE [16,17] framework for computational materials design.

OOF2's automatic mesh generation feature requires three parameters:

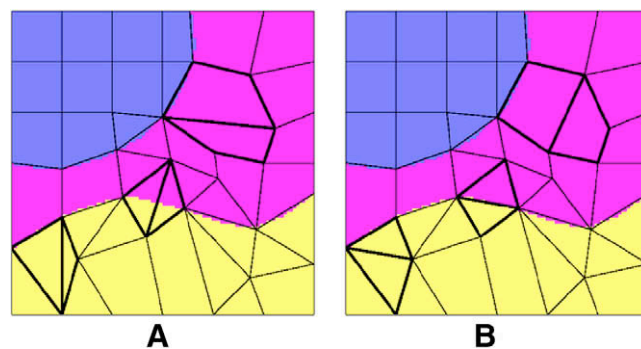


Fig. 10. Before (A) and after (B) application of the Swap edges routine.

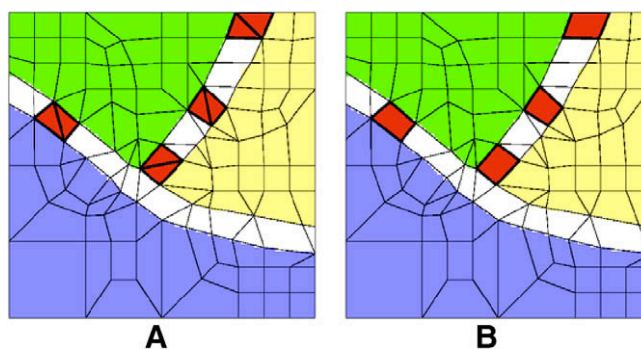


Fig. 11. Before (A) and after (B) application of the Merge triangles routine.

- (1) A rough size of the elements in the initial mesh (*maxscale*). This should correspond to the linear size of the largest features in the microstructure.
- (2) A rough size for the smallest elements in the desired mesh (*minscale*). This is the resolution to which the elements have to be refined by the mesh adaptation routines, and should correspond to the linear size of the smallest features in the microstructure.
- (3) A homogeneity *threshold* that determines which elements will be refined. All elements whose homogeneity is smaller than the threshold will be subdivided until the new elements' homogeneities are above *threshold* or the element sizes fall below *minscale*.

These parameters may be estimated for a class of microstructures based on the intuition of a materials engineer or from the nature of the physical problem.

As an example, we used OOF2 to generate a mesh for a microstructure consisting of the 484 pixel  $\times$  484 pixel image shown in Fig. 12(A). The light and dark regions define different phases or materials. For this microstructure, the parameters used were *maxscale* = 60 pixels, *minscale* = 20 pixels and *threshold* = 0.9. The resulting mesh is overlaid on the microstructure in Fig. 12(B).

The main sequence of the automatic mesh generation algorithm is depicted in Fig. 13. OOF2 first creates a mesh with quadrilateral elements. The size of the elements is set so that the initial mesh resolves features of size *maxscale* in the microstructure. The mesh is then refined iteratively with the Refine routine (edges are bisected with  $\alpha = 0.8$  in Eq. 6), until the linear dimensions of the smallest elements are smaller than *minscale*. All refinement operations are applied only to elements whose homogeneity is less than that specified by *threshold*. To get rid of rough corners and to Snap Nodes and edges onto boundaries, Snap Refine is applied once. After refining, OOF2 cleans up the mesh by applying the Rationalize routine twice, removing short sides of quadrilaterals (those with ratio of the longest

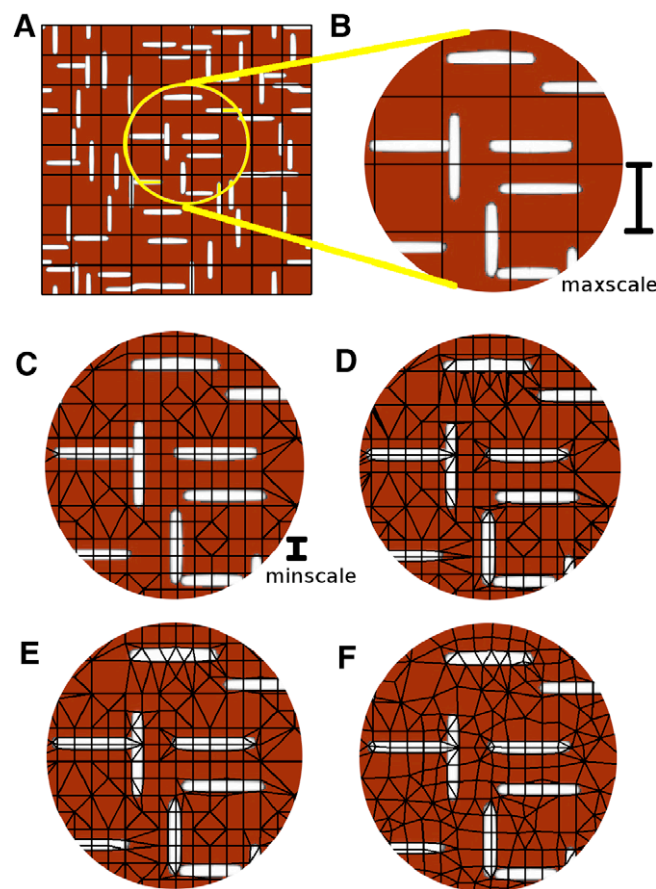


Fig. 13. (A) Microstructure with initial mesh. (B) Close-up of a portion of the microstructure and mesh. *maxscale* is the rough size of each element. The area-weighted average homogeneity of the elements is 0.846. (C) The mesh after a few applications of the Refine routine. Elements with homogeneity less than *threshold* were refined until the resulting elements reached a resolution given approximately by *minscale*. At this stage the average homogeneity is 0.890. (D) The mesh after one application of the Snap Refine routine. Note that nodes and edges have been placed at pixel boundaries. The average homogeneity is 0.980. (E) The mesh after two applications of the Rationalize routine that fixes badly shaped elements. The average homogeneity is 0.981. (F) The mesh after 5 iterations of the Smooth routine, keeping the nodes on material boundaries fixed.

side to the shortest side larger than 5.0), splitting wide quadrilaterals (those with angle greater than  $150^\circ$ ), and removing bad triangles (those with acute angle less than  $15^\circ$  and obtuse angle greater than  $150^\circ$ ). Finally, OOF2 applies the Smooth routine while keeping the positions of the nodes on material boundaries fixed. Smoothing is done for 5 iterations (with  $\alpha = 0.3$ ). We have found this sequence of operations to work well on a wide variety of microstructures, although we make no claim to universality.

### 3. Three dimensions

OOF2 only performs two-dimensional calculations on two-dimensional microstructure images. Three-dimensional micrographs, both from experiments and simulations, are becoming more common, so it is natural to

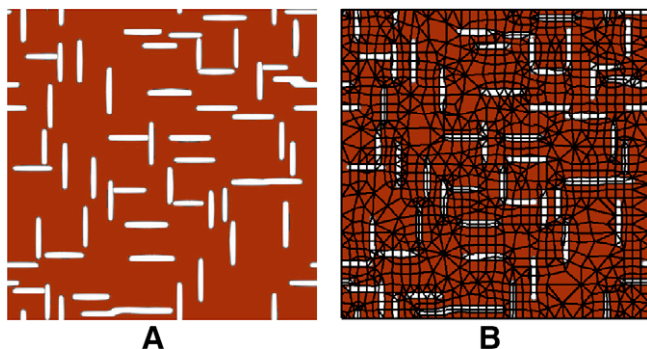


Fig. 12. (A) Microstructure created from a 484 pixel  $\times$  484 pixel image. (B) Mesh with 1851 elements created with OOF2's automatic mesh generation feature. The parameters used were *maxscale* = 60 pixels, *minscale* = 20 pixels and *threshold* = 0.9.

extend OOF to three dimensions. The meshing tools used in OOF2 were designed with extensions to 3D in mind, and the OOF developers are currently working on a 3D version, OOF3D.

One of the biggest challenges in extending our meshing techniques to three dimensions is in defining the energy functional. The definition of homogeneity energy transfers to 3D in a straightforward way, but the shape energy poses more problems. One possibility is to define a quality measure analogous to the expression in Eq. (1):

$$q_{3D} = CV^2/S^3 \quad (8)$$

where  $V$  is the volume of the element,  $S$  is its surface area, and  $C$  is a normalization constant. ( $C = 216\sqrt{3}$  for a tetrahedron). Another possibility is to define  $q_{3D}$  as the ratio of the radii of the inscribing and circumscribing spheres. Defining good quality measures for three-dimensional elements is an area of active research [18,12].

3D meshing is bound to be more computationally demanding, but many of the 2D mesh adaptation routines have straightforward extensions to 3D. For most of the topology-preserving routines, the difference between three and two dimensions is simply the difference between doing Monte Carlo or molecular dynamics simulations for particles in 3D instead of 2D. As for element refinement in 3D, there are clearly many more refinement possibilities so the rule sets will be more complicated, even if we only consider a single type of element (*e.g.* tetrahedrons). We have developed a ruleset for refining a tetrahedron into smaller tetrahedrons for all possible ways of bisecting some of its edges [19].

#### 4. Summary

We have described a suite of 2D mesh adaptation routines available in the software package OOF2. These routines can be combined into a close-to-automatic mesh generator, where the input consists of a segmented image, an upper size scale, a lower size scale, and a homogeneity threshold. Future work may include comparing the quality of elements generated using OOF2 with that of other methods. Work extending the 2D mesh adaptation routines to 3D meshes is also in progress.

The OOF2 program, including source code and users manuals, is available at <http://www.ctcms.nist.gov/oof/oof2/>.

#### Acknowledgement

Rhonald Lua and Edwin García were supported in part by the National Science Foundation through Information Technology Research Grant DMR-0205232. OOF2 was produced by NIST, an agency of the US government, and by statute is not subject to copyright in the United States. Recipients of the software assume all responsibilities associated with its operation, modification and maintenance.

#### References

- [1] S.A. Langer, E.R. Fuller Jr., W.C. Carter, *Computers in Science and Engineering* 3 (3) (2001) 15–23.
- [2] S.A. Langer, A.C.E. Reid, R.E. García, S.-I. Haan, R.C. Lua, W.C. Carter, E.R. Fuller Jr., A. Roosen, <<http://www.ctcms.nist.gov/oof/index.html>>.
- [3] S.A. Langer, A. Reid, S.-I. Haan, R.E. García, <<http://www.ctcms.nist.gov/langer/oof2man/index.html>>.
- [4] R.E. García, A.C.E. Reid, S.A. Langer, W.C. Carter, in: D. Raabe, F. Roters, F. Barlat, L.-Q. Chen (Eds.), *Continuum Scale Simulation of Engineering Materials*, Wiley-VCH, 2004.
- [5] F.B. Paul, E.Y. Sun, K.P. Plucknett, K.B. Alexander, C.-H. Hsueh, H.-T. Lin, S.B. Waters, C.G. Westmoreland, E.-S. Kang, K. Hirao, M.E. Brito, *J. Am. Ceram. Soc.* 81 (11) (1998) 2821–2830.
- [6] A.D. Jadhav, N.P. Padture, E.H. Jordan, M. Gell, P. Miranzo, E.R. Fuller Jr., *Acta Materialia* 54 (12) (2006) 3343–3349.
- [7] V. Cannillo, T. Manfredini, M. Montorosi, A. Boccaccini, *Journal of Non-Crystalline Solids* 344 (2004) 88–93.
- [8] R.E. García, W.C. Carter, S.A. Langer, *Journal of the American Ceramic Society* 88 (3) (2005) 750–757.
- [9] R.E. García, Y.-M. Chiang, W.C. Carter, P. Limthongkul, C.M. Bishop, *Journal of the Electrochemical Society* 152 (1) (2005) A255–A263.
- [10] E. Garboczi, D. Bentz, N. Martys, in: P.-Z. Wong (Ed.), *Methods in the Physics of Porous Media*, Academic Press, San Diego, CA, 1999, pp. 1–41.
- [11] A. Roberts, E. Garboczi, *J. Am. Ceram. Soc.* 83 (2000) 3041–3048.
- [12] J. Shewchuck, Eleventh International Meshing Roundtable, 115–126, Sandia National Laboratories, Ithaca, New York, 2002.
- [13] GiD Manual, <<http://gid.cimne.upc.es/>>.
- [14] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, *Journal of Chemical Physics* 21 (1953) 1087–1092.
- [15] L.A. Freitag, in: *Trends in Unsaturated Mesh Generation*, ASME Applied Mechanics Division, vol. AMD 220, 1997, pp. 37–44.
- [16] <<http://matcase.psu.edu/index.html>>.
- [17] Z. Liu, L.-Q. Chen, P. Raghavan, Q. Du, J. Sofo, S. Langer, C. Wolverton, *Journal of Computer-Aided Materials Design* 11 (2004) 183–199.
- [18] J. Shewchuck, Preprint, 2002.
- [19] Java Applet, <<http://www.ctcms.nist.gov/~rlua/FE/refinery/>>.