An Automated Workflow for Meshing Evolving Microstructures from High-Throughput Grain Growth Simulations

M.C. Golt and E. Hernández-Rivera
Weapons and Materials Research Directorate, U.S. Army Research Laboratory, APG, MD, U.S.A.

Introduction

Microstructure evolution simulations using the Potts Monte Carlo (PMC) and phase-field models are routinely used to simulate sintering and densification of ceramic and metallic materials. Understanding the evolution of microstructural features is important as these are known to influence different materials properties, e.g., strength through the Hall-Petch effect. An approach to understanding how microstructure evolves under different conditions (e.g., processing) is to simulate the microstructural evolution at mesoscales. After obtaining a microstructure, analysis of a finite element mesh of the microstructure is often desired to determine the properties or the response. In addition, certain properties such as the electrical conductivity might be desired at each time-step of the microstructural evolution simulation. Therefore, the process of obtaining the microstructure output at a given simulation time-step, meshing that microstructure, and solving for its properties through finite element analysis software needs to be automated. SPPARKS [1] is an open-source, parallelized Monte Carlo-based framework that enables rapid production of 3D microstructures. SPPARKS's AppSinter "application" simulates the densification of structures through pore removal and can be automated through command line interfacing. COMSOL has powerful solvers that can determine the properties of these microstructures and can be automated through the LiveLink for MATLAB interface. However, adequately meshing these structures is challenging as there can be a multitude of material's features and properties, as well as interfaces that must be accounted for reliably. ISO2MESH is a popular MATLAB open-source meshing toolbox that can create a 3D tetrahedral finite element mesh from gray-scale volumetric images, such as those obtained from computed tomography scans [2]. However, the high-level functions of this toolbox will fail to produce COMSOL compatible meshes where more than two domains are present at a joint interface: a common occurrence in microstructures where three grains come together to form a triple-point.

This paper details how some lower-level functions from the ISO2MESH toolbox can be used to provide robust, COMSOL compatible meshes from voxelized 3D matrices obtained from SPPARKS microstructure simulations. Simulations of the microstructure's conductivity during the densification process are produced using the COMSOL AC/DC module.

Microstructure Generation using SPPARKS

Microstructural evolution during sintering is complex and not easily obtained from experimental results. Therefore, synthetic microstructures were generated using the SPPARKS's AppSinter, which simulates the densification of microstructures during sintering. SPPARKS is a kMC framework that was initially developed to model curvature-driven grain growth as simulated by the Potts Monte Carlo model. It employs a uniform meshless grid where each voxel ("node") represents a microstructural feature (i.e., grain orientation and pore-state). Within SPPARKS, a model based on mechanisms similar to Ashby's classic 6-paths for diffusion known to describe solid-state sintering was developed by Cardona et al. [3]. The solid-state sintering processes observed leading to microstructural evolution are: grain growth, surface/pore migration, and vacancy annihilation at grain boundaries. Each of these processes is defined by a kinetic rate (relative frequency) at which each process is attempted. Furthermore, the model defines a process-specific "temperature" which influences their probability of acceptance as defined by the Boltzmann distribution. As with the Potts Monte Carlo model, the microstructure evolves through interfacial energy minimization by allowing grain growth. This kMC implementation has been validated against sintering of Cu-powder compacts [4]. An example of a SPPARKS generated porous microstructure is shown in Figure 1, where the domain shown and used for the FEM simulations was limited to the internal volume as to avoid external surface roughness.

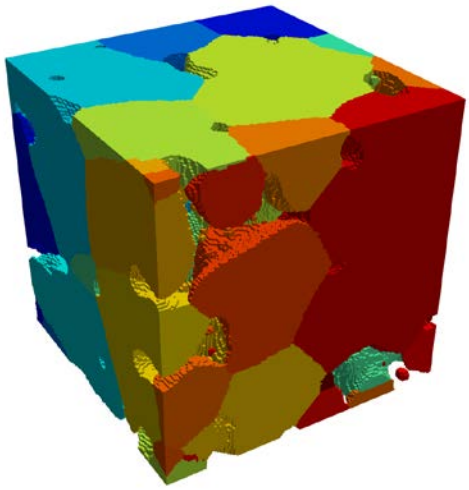

Figure 1. Microstructure obtained using SPPARKS. Colors indicate different grain orientations. Missing voxels indicate pores. Volume has a density of 91.4%.

Mesh Generation and Export to COMSOL

Despite the intent of the ISO2MESH toolbox to automatically generate a mesh of multi-domain geometries, mesh generation of these microstructures will often fail to produce a COMSOL compatible mesh when using the high-level functions of the toolbox. For simple binary or grayscale volumes these functions will provide for easy generation of compatible meshes, but they tend to fail when there are instances with multiple domains converging at a single point. However, many of the lower-level functions of this toolbox are convenient for mesh handling and analysis. As such, several of these functions were used to generate COMSOL compatible meshes through MATLAB on the challenging microstructure domains generated through SPPARKS. The mesh generation process begins with a coarse tetrahedral element mesh that, other than being the same size as the bounding box, is not refined to specific features in the geometry. Then, elements that are located near boundaries within the geometry are identified and progressively refined such that there are finer elements at the grain boundaries and coarse elements in the grain interiors. When the final mesh is obtained, the elements are assigned to a domain (grain ID or pore) depending on where their centroids reside in the microstructure. The following steps detail this process:

(1) Remove extremely small grains (if <0.04% the total volume) by converting them to pores. Grains this small are essentially a free atom transporting mass through space, from one grain to another.
(2) Create a default box with a coarse mesh using *meshabox*, the same size as the microstructure bounds.
(3) For each node of the mesh, determine which domain (grain ID or pore) it would reside in according to its x, y, z position.
(4) Determine which tetrahedral are at a grain boundary interface (where one or more of the tet's nodes are in a different domain).
(5) Refine the mesh at the grain boundary interface nodes using *meshrefine* with an order-of-magnitude reduced volume.
(6) Repeat once steps 3 through 5 with the refined mesh.
(7) Assign each tetrahedral to a domain (grain ID or pore) according to the x, y, z position of its centroid (as found via *meshcentroid*) in the microstructure.

The ISO2MESH toolbox includes the function *savemphtxt* that could be used to write the mesh to a '*.mphtxt' file that can be imported within the COMSOL graphical user interface. However, the aim of this work was to generate a high-throughput workflow to enable analysis of large numbers of simulated microstructures. Because MATLAB is already handling the SPPARKS simulations and the mesh generation, the LiveLink for MATLAB interface is a convenient and efficient tool for automating the analysis of the structures. The mesh was setup within a COMSOL v5.3 model with the AC/DC module using the following MATLAB syntax:

```
%Create model in COMSOL v5.3
1   import com.comsol.model.*
2   import com.comsol.model.util.*
3   model = ModelUtil.create('Model');
4   model.component.create('comp1', true);
5   model.component('comp1').geom.create('geom1', 3);
6   model.component('comp1').mesh.create('mesh1');
7   model.component('comp1').physics.create('ec',
'ConductiveMedia', 'geom1');
8   model.study.create('std1');
9   model.study('std1').create('stat', 'Stationary');
10  model.study('std1').feature('stat').activate('ec', true);
%Upload the mesh
11  model.mesh('mesh1').data.setElem('tet', elem(:, 1:4)'-1);
12  model.mesh('mesh1').data.setVertex(node');
13  model.mesh('mesh1').data.setElemEntity('tet', elem(:,5));
14  model.mesh('mesh1').data.createMesh;
15  disp('COMSOL mesh created.')
```

Here, *node* is a $m \times 3$ matrix of x, y, z coordinates and *elem* is $n \times 5$ matrix, where columns 1 through 4 reference the 4 nodes of the tetrahedral and column 5 is the domain type (either pore or grain ID). Instructions for setting the mesh data can be found in the LiveLink for MATLAB users guide [5]. The node list in COMSOL begins with a node 0, hence the subtraction of 1 (command 11, *data.setElem*) from the element list *elem*. The *setElemEntity* function defines which elements belong to which domains. This

information is essential for there to be multiple domains within the mesh and was determined in step (7) of the mesh generation procedure. The mesh of two microstructures, one with the initial packing (nearly 60%) of spherical grains, and a later densified structure of nearly 91.4% is shown in Figure 2. Despite the large difference in microstructure, the described meshing method was able to successfully import the meshed geometry into COMSOL for all time steps during the densification simulation.

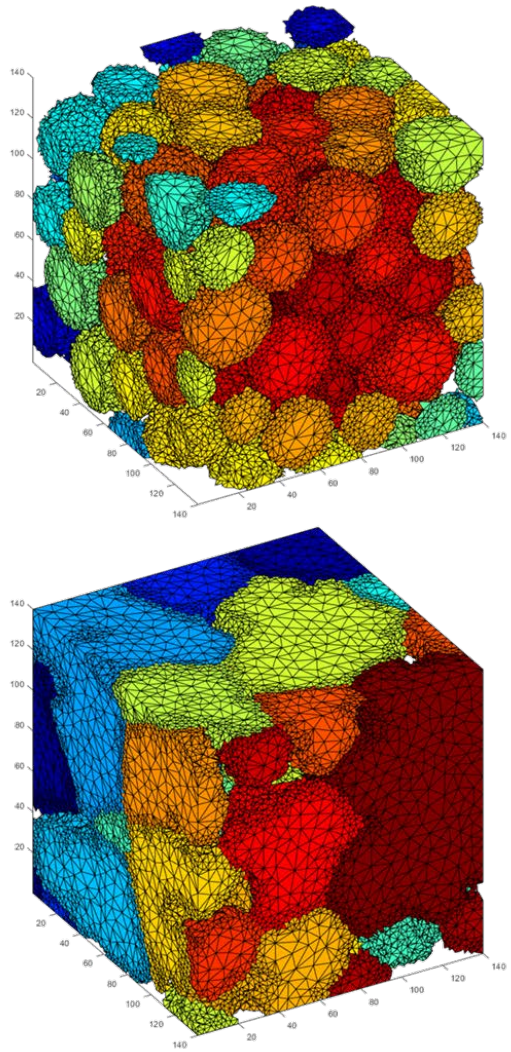

Figure 2. Mesh of the microstructure at initial 59.2% density (top) and a later 91.4% density (bottom). The mesh is refined at the grain boundaries. Colors indicate different grain orientations.

Model Development

With the mesh determined and the mesh data applied to the model, the material properties and boundary conditions can be applied. COMSOL gives each entity (point, edge, surface, domain) a reference number. These numbers can be used to programmatically apply a material property or boundary condition to a particular entity. While the number given to each entity of the geometry by COMSOL is methodical, without knowing the exact procedure, the numbering may appear to the user as a random assignment. Therefore, the approach often taken is to discover COMSOL's entity ID assignments. Within the LiveLink for MATLAB interface the function *mphselectcoords* can be used to determine the number assignments for point, edge, boundary, and domain entities when their approximate coordinates are known. In this example, each tetrahedral element belongs to either a pore or a grain, as determined by step (7) of the mesh building procedure. The domain for each tetrahedral can then be determined via *mphselectcoords* by passing in the coordinates of the 4 nodes defining the tetrahedral. The MATLAB syntax used for finding the corresponding domain ID for the nodes of a tetrahedral (*i*) that was identified as belonging to a pore was:

```
id=mphselectcoords(model,'geom1',node(porenodes(i, :), :)',
'domain', 'include', 'all');
```

In practice, obtaining the domain entity ID that every tetrahedral element belongs to using this command takes considerable time. This processes is redundant as multiple tetrahedral elements will report the same domain ID. Once the COMSOL assigned entity ID is discovered for a domain, it does not need to be found again. For this example there are only two domain types of interest: pore or grain type. If all the domain entity IDs for the pore domains are found, then all the remaining domain entity IDs are assumed to belong to grains. It was found that especially for high porosity microstructures all the domain entity IDs could be found through quickly evaluating at random just 300 tetrahedral coordinates that were identified as belonging to pores (according to column 5 of the *elem* array). After the domain entity IDs are known for the pores vs. the grains, the material properties can be defined and applied using the examples provided in the LiveLink for MATLAB user's guide [5].

Boundary properties can be applied once the COMSOL assigned boundary entity IDs are known. These can be obtained in an approach similar to that used to find the domain entity IDs, but through using the *mphselectbox* function to grab all entities within a bounding box. In this example the top and bottom boundaries of the microstructure are needed to apply an electrostatic potential and ground. For example, the boundary element IDs of the top boundaries were found using the following MATLAB syntax:

```
top_boundaries = mphselectbox(model,'geom1',[0, sx; 0, sy; sz-1, sz], 'boundary', 'include', 'any');
```

Here, $sx, sy,$ and sz are the size of the model in $x, y,$ and $z,$ where the bottom corner of the model is at [0,0,0]. A similar syntax can be used to find the bottom and interior boundary element IDs. Once found, the entity IDs can be used to apply the desired boundary properties.

Simulation Setup and Results

For this example, a measurement of the electrical current through an alumina ceramic microstructure that is densifying during thermal treatment is simulated. SPPARKS was used to provide a series of 90 microstructures at different time steps throughout the densification process. MATLAB was then used to automate the workflow of importing the structures, meshing via the previously described method, and solving for the current across the structure through the LiveLink for MATLAB interface and the AC/DC module. A sub-volume of $140 \times 140 \times 140$ voxels was taken from the inner volume of the densifying structure to analyze as the outermost surface of the volume has large topological variations due to the shrinking volume. A 1 volt terminal was applied to all top boundaries, and a ground condition was applied to all bottom boundaries. Alumina is known to have a grain boundary conductivity that is significantly higher than the grain conductivity at sintering temperatures [6]. Additionally, these grain boundaries (few nanometers) are orders of magnitude thinner than the grain size (microns). A full fidelity model of the grain boundaries would require an unnecessarily large amount of elements to resolve thin grain boundaries. Therefore, in this case, an electric shielding condition can be applied to all interior boundaries to reduce the resources needed to compute the simulation [7]. The following table lists the material properties used in this example.

Feature	σ (S/m)	ε	thickness
Grain	0.105	9.7	as given
Grain boundary	2	9.7	3 (nm)
Pore	1E-15	1	as given

Table 1: Material electrical properties assigned to features of the microstructure.

SPPARKS simulated the sintering and densification of a microstructure that first began with a 3D compact of randomly distributed particles with a normal distribution. As in reference [8], the LAMMPS “fix pour” command was used to "pour" ~700 spherical particles that represent powder particulates into a container. This powder compact was placed within a $300 \times 300 \times 300$ voxels domain where these spheres were discretized into the voxelized grid. As previously mentioned, the microstructure evolves based on how the different frequencies are defined. Here, the event relative frequencies were defined as 2:1:10 for grain growth : pore migration : vacancy annihilation, respectively. The temperatures were defined as 1.5, 1, and 15 [a.u.] for grain growth, pore migration, and vacancy annihilation, respectively. Both the grain growth frequency and temperature were slightly over that of pore migration, as larger grains were desired.

Simulation Results

The densification range of the SPPARKS microstructure simulation is divided into a low density regime, between 60% and 90% density, a high density regime, between 90% and 99.9% density, and a full (100%) density regime. The bulk of the densification occurs during the low-density regime, and slows as it approaches full density in the high-density regime. Large grains continue to grow at the expense of smaller grains (coarsening) during the full density regime.

The conductivity of the microstructures determined by the COMSOL model for the three density regimes is shown in Figure 3. The low density regime conductivity increases as the pore volume is removed and can be viewed as having two regions where there is a near-linear relationship between density and conductivity. This is an expected result and it follows the relationship between density and conductivity of the analytical percolation model:

$$\sigma = \sigma_m \left(\frac{\varphi_m - \varphi_c}{1 - \varphi_c} \right)$$

where σ_m is the conductivity of the material (in this case the effective conductivity of alumina grains and grain boundaries), φ_m is the volume concentration of the alumina, and φ_c is the critical or tap density of the powder. At the lowest, initial stages of densification the slope of the conductivity vs. density result is steeper (curve fit σ_m of 1.05) than that of the later stages (curve fit σ_m of 0.11). This suggests that the early gains in conductivity are mostly due to network formation of the higher conductivity grain boundaries. At later stages, increases in conductivity with density slows as it is dominated by gains in the volume fraction of the bulk, lower-conductivity grains. In the high density regime there is a slowing to the increase in conductivity with density as there are two competing mechanisms of equal contribution. The remaining pores are eliminated, raising the

conductivity, however an increasing amount of the more conductive grain boundaries are eliminated as the grains coarsen. The conductivity decreases during the full density regime as the higher-conductivity grain boundaries are gradually eliminated during grain coarsening.

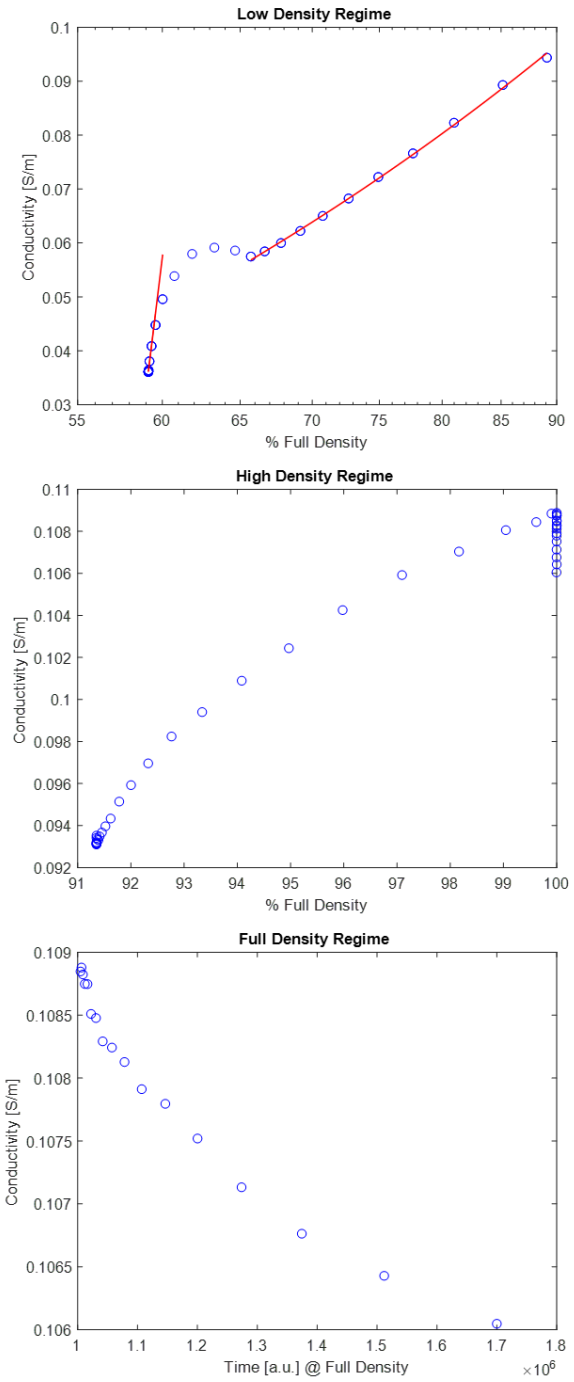

Figure 3: (Top) Conductivity increase during the low density regime. Red lines indicate fit to percolation model. (Mid) Conductivity increase during the high density regime. (Bot) Conductivity decrease during the full density regime.

Summary

Automated and robust meshing of complex geometries for COMSOL has been described that uses functions from the ISO2MESH toolbox and the LiveLink for MATLAB interface. This method has been applied to a sequence of volumes from a microstructure densification simulation, where it was able to successfully import a meshed geometry to COMSOL for the wide-range of microstructures provided. Also described is a method to automatically identify the regions of the geometry to assign the desired material and boundary conditions. An illustrative example of how to use this method for COMSOL model development and solving is given with a simulation of the change in electrical conductivity of the microstructure during the densification process.

References

1. S. Plimpton, C. Battaile, M. Chandross, L. Holm, A. Thompson, V. Tikare, G. Wagner, E. Webb, X. Zhou, C. Garcia Cardona, A. Slepoy, “Crossing the Mesoscale No-Man's Land via Parallel Kinetic Monte Carlo”, *Sandia Report: SAND2009–6226* (Oct 2009).

2. Qianqian Fang and David Boas, "Tetrahedral mesh generation from volumetric binary and gray-scale images," *Proc. of IEEE International Symposium on Biomedical Imaging 2009*, pp. 1142-1145, 2009

3. Cristina Garcia Cardona, Veena Tikare, Steven J. Plimpton, “Parallel simulation of 3D sintering”, *Int. Journal of Computational Materials Science and Surface Engineering*, **Vol. 4**, 37-54 (2011)

4. Tikare, Veena, et al. "Numerical simulation of microstructural evolution during sintering at the mesoscale in a 3D powder compact." *Computational Materials Science* **48.2** (2010): 317-325.

5. LiveLink™ for MATLAB® User’s Guide, ©Comsol (2009-2017)

6. Kitazawa, K., and R. L. Coble. "Electrical Conduction in Single-Crystal and Polycrystalline Al2O3 at High Temperatures." *Journal of the American Ceramic Society* **57.6** (1974): 245-250.

7. AC/DC Module Application Library Manual, ©Comsol (2009-2017)

8. Bjørk, Rasmus, et al. "The effect of particle size distributions on the microstructural evolution during sintering." *Journal of the American Ceramic Society* 96.1 (2013): 103-110.

APPENDEX – Mesh Generation MATLAB code

```
%%Spin is the voxelized volume from SPPARKS
%Pore voxels have value of 1
%Grain orientation voxels are integers > 1
[sx, sy, sz] = size(spin);

%Step 1: Remove small grains
disp('removing small features...')

tspin = int64(spin==1);
tctr = 0;
uids = unique(spin);
for i = 1:length(uids)
    if uids(i) ~= 1
        t = sum(sum(sum(spin==uids(i))));
        %Identify small grains < 0.04% of total volume
        tspin = tspin + uids(i)*int64(bwareaopen(spin==uids(i), 0.0004*(sx*sy*sz), 18));
        tctr = tctr + (t - sum(sum(sum(tspin==uids(i)))));
    end
end
tspin(tspin==0) = 1; %Convert small grains to pores
spin = tspin;
disp([num2str(tctr), ' voxels removed.  ', num2str(tctr/(size(spin,1)*size(spin,2)*size(spin,3))), '% volume converted.'])

%% Create mesh with functions from the ISO2MESH toolbox

%Step 2: Mesh a box with initial max tetrahedral volume of 0.0001, size
%scaled by 100
[node,face,elem]=meshabox([0.0001,0.0001,0.0001], [sx/100, sy/100, sz/100], 0.0001, 0.0001);
node = node*100;

vol=elemvolume(node,elem(:,1:4));
disp(['Mean Tet Volume: ', num2str(mean(vol))])
nodeids = spin(sub2ind(size(spin), ceil(node(:,1)), ceil(node(:,2)), ceil(node(:,3))));

%Step 3: identify the domains (grain IDs or pores) that the nodes of each
%tetrahedral belong to
tetids = zeros(size(elem,1),4);
for i = 1:length(elem)
    tetids(i, :) = nodeids(elem(i,1:4));
end
%Step 4: identify tetrahedrals that are not at grain boundary interfaces
intids = (tetids(:,1)==tetids(:,2) & tetids(:,3)==tetids(:,4) & tetids(:,1)==tetids(:,3));

%Step 5: refine tetrahedrals that are at grain boundary interfaces
[newnode, newelem,newface] = meshrefine(node,elem,(~intids)*20);
node = newnode; elem = newelem;
vol=elemvolume(node,elem(:,1:4));
disp(['Mean Tet Volume: ', num2str(mean(vol))])

%Repeat Step 3: For the new mesh, identify the domains (grain IDs or pores)
%that the nodes of each tetrahedral belong to
nodeids = spin(sub2ind(size(spin), ceil(node(:,1)), ceil(node(:,2)), ceil(node(:,3))));
tetids = zeros(size(elem,1), 4);
for i = 1:length(elem)
    tetids(i, :) = nodeids(elem(i,1:4));
end
%Repeat Step 4: identify tetrahedrals that are not at grain boundary interfaces
intids = (tetids(:,1)==tetids(:,2) & tetids(:,3)==tetids(:,4) & tetids(:,1)==tetids(:,3));

%Repeat Step 5: refine tetrahedrals that are at grain boundary interfaces
vrefine = 3;
[newnode, newelem,newface] = meshrefine(node,elem,(~intids)*vrefine);
node = newnode; elem = newelem; face = newface;
vol=elemvolume(node,elem(:,1:4));
disp(['Mean Tet Volume: ', num2str(mean(vol))])

%Step 7: Assign each tetrahedral to a domain according to the position of
%its centroid
```

```
centroids=meshcentroid(node,face(:,1:3));
face(:,4) = spin(sub2ind(size(spin), ceil(centroids(:, 1)), ceil(centroids(:,2)), ceil(centroids(:, 3))));
centroids=meshcentroid(node,elem(:,1:4));
elem(:,5) = spin(sub2ind(size(spin), ceil(centroids(:, 1)), ceil(centroids(:,2)), ceil(centroids(:, 3))));
voidpts = elem((elem(:,5)==1), 1:4);
```