Method Article

Volumetric analysis and mesh generation of real and artificial microstructural geometries

David J. Walters*, Darby J. Luscher, John D. Yeager

Los Alamos National Laboratory, Los Alamos, NM 87545, USA

A B S T R A C T

Producing a viable finite element mesh of realistic microstructural structural geometry is a critical step in analyzing the thermo-mechanical behavior of complex multi-material composites. Advancements in imaging technology such as micro computed tomography have allowed modelers to access high resolution mesoscale geometries for direct numerical simulation. However, converting from voxel based 3D images to usable finite element meshes has been challenging. A robust method including algorithms and software scripts for generating finite element meshes from 3D imaged microstructures is presented. It includes a routine for inserting cohesive elements around material interfaces to enable modeling of interface properties including delamination and damage. The algorithms and procedures presented in this method leverage currently available software packages for processing surface based geometry into volume based meshes. In addition to converting real geometry from physical imaging systems, algorithms for producing numerically generated and statistically equivalent microstructural geometry are also included. These artificial microstructures can be a valuable resource for modelers when physical specimens do not exist or are limited in quantity.

- Method establishes a workflow from voxel data to viable finite element mesh including interface information
- Includes a method for synthetic geometry generation based on metrics from real microstructures

© 2020 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license. (http://creativecommons.org/licenses/by-nc-nd/4.0/)

A R T I C L E   I N F O

Method name: Mesh conversion from voxel to finite element mesh
Keywords: Mesoscale, Finite Element Mesh, Cohesive Elements, Polycrystalline Composites
Article history: Received 17 December 2019; Accepted 5 March 2020; Available online 23 March 2020

* Corresponding author.
E-mail address: djwalters@lanl.gov (D.J. Walters).

https://doi.org/10.1016/j.mex.2020.100856
2215-0161/© 2020 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license. (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Specifications Table

| Subject Area:  | Engineering |
|---------------|-------------|
| More specific subject area | Modeling and simulation mesh creation |
| Method name:  | Mesh conversion from voxel to finite element mesh |
| Name and reference of original method: | [1] (Software). Sandia National Laboratories |
| Resource availability: | Cubit: https://cubit.sandia.gov Meshlab v1.3.2_64bit Cohesive Element Mesh Tools: https://github.com/lanl/MicroMesh |

Method Details

Advances in computational infrastructure have enabled the ability to model increasingly complex geometric structures through direct numerical simulation (e.g. finite element, finite volume). In conjunction with increased computing power, experimental imaging techniques have also enabled an unprecedented view into the microstructures of complex heterogeneous materials [e.g. 2,3,4,5]. In many studies [2,6–11], directly meshing and analyzing the grain scale structures of heterogeneous composite materials is a way to elucidate important mesoscale phenomena (e.g. grain/matrix delamination, inter- and intra-granular fracture, meso-scale response to dynamic loads, etc.). Although accurately predicting the macro-scale thermomechanical response of these materials is important, discretely capturing meso-scale phenomena is also desired since many relevant dynamic processes occur at this scale (e.g. void nucleation/growth/collapse, hot-spot formation, fracture initiation).

The geometry for many of these models originates from a three dimensional imaging technique such as micro Computed Tomography (µCT), Magnetic Resonance Imaging (MRI), or 3D laser scanning. These types of datasets can be directly converted to a computational surface mesh or grid for to define the surfaces of multiple constituents in a multi-body microstructure. Finite element analyses conducted in three dimensions require a volumetrically discretized mesh — not just a definition of the exterior surfaces. A robust method for converting microstructures defined by a surface mesh to a usable discretized volume is described in detail here. This method is applicable to multiphase composites (e.g. one or more types of granular materials embedded in a matrix), multigrain single materials (e.g. grain scale structures of metals and other crystalline materials), and complex single phase microstructures (e.g. foams, snow, bone). Included in this methodology is a procedure for implanting special elements between different materials to accurately capture interface behavior. Illustrations and examples provided throughout this article are conducted on a multiphase granular composite. Specifically, the examples here are of plastic bonded explosives (PBXs) which are highly loaded (80 wt% or greater) with single crystals of explosives (5–520 µm) surrounded by a solid polymer matrix.

The methods and procedures described below utilized the following software: Meshlab [12] and Cubit [1]. The selection of software was based on availability, compatibility with import and export file formats in use by this research team, and capability (smooth hex dominant meshes and zero thickness cohesive elements). No new capabilities were added to these existing software packages. The methods presented here primarily consist of scripts and processes that flow through existing software packages to convert from surface descriptions of solid materials (especially polycrystalline composites) to volumetric finite element meshes with the capability to model interface behavior with the inclusion of cohesive elements. The methods shown here may also inspire other workflows or scripts for converting microstructural data into volumetric meshes using alternative software (e.g. Gmsh [13], Neper [14], etc.)

In addition to directly converting real microstructures, a technique is included below with which to create numerically generated synthetic structures when real geometries are unavailable or in limited supply. These synthetic structures are generated in a way to vary individual constituents’ geometric complexity to enable the study of artificially generated microstructures, and whether they can capture meso- and macro-scale phenomena exhibited by real microstructures. Software [e.g. 14,15] with the capability to generate synthetic microstructures and meshes already exist for the cases involving closed polycrystalline materials (i.e. individual grain boundaries are adjacent and in contact with other grains in the microstructure; no filler material or matrix). These existing packages are not currently
capable of synthetically generating microstructures containing matrix material in the interstitial space between individual grains. A method for generating and meshing these types of synthetic composite structures is also described here. Fig. 1 illustrates the general steps of the procedures, whether using real or synthetic microstructures. Notes on general use cases (e.g. microstructure with or without matrix material) are provided in applicable areas of the text.

Initial surface topology

Microstructural images are used either to construct a mesh of the actual microstructure, or analyzed for characteristics to use for synthetic geometry construction. The PBX examples in this article were imaged using cone beam μCT and reconstructed into a three dimensional image stack using the [16] reconstruction algorithm. The digitized image stack provides a 3D representation of the original material in which the intensity (gray scale value) of each voxel is a function of the density of the material in the spatial position of said voxel. If the overall volume fraction of each constituent is known, thresholds can be applied to the images to segment individual materials [e.g. 17]. If sufficient contrast exists between different materials, distinct material definitions are automatically separated out as shown in the 2-D cross-section in Fig. 2.

In granular structures with highly loaded aggregate particles such as PBXs, the resulting gray scale thresholding can produce erroneously connected individual particles when separated by widths at or below the scanned voxel resolution. Directly modeling this type of structure containing artificially clustered grains can be problematic since individual grains could be falsely connected during the volume meshing process. Watershed segmentation [18,19] provides relief from the problems introduced during segmentation and appropriately separates most erroneously connected grains.

If the real geometry is to be modeled directly, the 3-D image stack is processed using a surface reconstruction technique (e.g. marching cubes [20]) to produce a watertight surface representation of the desired material. Watertight means that each independent constituent (e.g. grain, crystal) is fully enclosed within a continuous surface without holes. When the surface is non-manifold (holes, intersecting faces), definitions for the boundary of a volume become ambiguous. In the case shown here, since void spaces in the binder material were negligible, only the explosive grains' surfaces were reconstructed into a common file format for 3-D surfaces called stereo-lithography (STL). The STL file format is commonly used in the additive manufacturing community assuming the reconstructed geometry is convex and watertight. Fig. 3 shows the raw surface reconstruction of just the explosive grains with a spatial domain with a cubic side length of 0.52 cm. From here, the surface reconstruction in the form of an STL file is either processed for production of a volumetric mesh (valid for finite element analysis), or its morphology is analyzed to generate a set of statistical parameters with which an artificially generated microstructure can be produced with similar features to the real microstructure.

Morphometric analysis of microstructure

Being able to utilize numerically generated artificial geometry could be advantageous in certain situations. If imaging of real experimental samples is limited or unavailable, numerically generated geometry is one pathway to modeling similar representative microstructures. Additionally, modeling of artificial geometries may be able to mimic behaviors exhibited by real geometries while simplifying the complex features of individual constituents reducing the computational cost of a similar sized representative volume element.

Metrics such as the distribution of particle sizes, shapes, and number density can be utilized to generate probability distribution functions from which information for numerically generated geometry can be sampled. If a sufficient set of these factors is reasonably reproduced from the real sample, other factors such as volume fraction and mean particle separation should also be reproduced.

Number density of particles simply represents the total number of independent particles/grains/ crystals of a certain type within a volumetric domain. As shown in Fig. 4 for the PBX sample used for demonstration, several subvolumes at different volumetric domain sizes are displayed with their
**Fig. 1.** Summary of the volume mesh generation process for FEA analysis originating from a surface mesh of 3D imaged mesoscale geometries. This procedure also allows for the generation of statistically equivalent synthetic microstructures based on the originally imaged geometry.
Fig. 2. Two dimensional cross-section of a reconstructed and segmented μCT image of a PBX sample. Dark grey represents the explosive grains and red represents the polymer binder. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 3. Raw surface reconstruction of explosive grains with a cubic domain edge length of 0.52 cm.

respective number densities. If generating an artificial microstructure numerically, one could reference this metric to determine the appropriate number of particles to seed a specific sized domain.

The description of the distribution of particle sizes offers more choice in which metric to utilize. Several options are described below. Each of these metrics apply to individual objects (e.g. grains, crystals), and are tabulated as a distribution across the entire domain.

The major diameter is a measure of the distance between the two most distant voxels in an individual object. The major diameter has a tendency to over estimate the true sizes of particles if
the the population of particles are highly eccentric. This would be an appropriate metric for shapes nearly spherical in nature.

If matching the real geometries’ volume fraction is important, the volume-equivalent sphere diameter could be a useful metric. The equivalent sphere diameter is calculated as:

\[ D_v = \left( \frac{6V}{\pi} \right)^{1/3}, \]  

(1)

where \( V \) is the object’s volume obtained from the marching cubes algorithm \cite{20}. In situations with a high packing density and eccentric particles (e.g. plate- or needle-like), using this particular metric without adjusting the shape of the particles may produce a geometry containing particles with frequent overlaps with other particles.

The surface-equivalent sphere diameter is similar, however the surface area of the object is utilized as the matching criteria. The surface-equivalent sphere diameter is described by:

\[ D_A^s = \sqrt[3]{\frac{A}{\pi}}, \]  

(2)

where \( A \) is the surface are of the individual object, also obtained from the marching cubes algorithm \cite{20}.

A common metric utilized in fluid dynamics calculations is the Sauter diameter. This metric represents a sphere with the same volume to surface area ratio as the analyzed object.

\[ D_s^2 = \left( \frac{D_v}{D_A} \right)^3 = \frac{V}{A} \]  

(3)

Probability distribution functions can be generated from the tabulated size metrics. In the case demonstrated here, a generalized gamma function fit the size distributions best as seen in Fig. 5. A log normal distribution could also be a candidate, however since particle diameters are voxel based, the minimum value would be 1, not 0. The gamma distribution works better with this physical lower bound. Generally, different probability distribution functions should be chosen to best match the specific data. The generalized gamma density function takes the form:

\[ f(x,a,c) = \frac{c^{a-1} x^{a-1} \exp(-x^c)}{\gamma(a)}, \]  

(4)

where \( a \) and \( c \) are function shape parameters, and \( x \) represents the equivalent diameter. The generalized gamma function parameters for the density functions shown in Fig. 5 are listed in Table 1.
Fig. 5. Probability density distributions of different metrics to describe grain size distributions for the example PBX material demonstrated here. The curves represent best fit generalized gamma functions for sampling later during the generation of artificial microstructures. Metrics represented from left to right are: major diameter, volume-equivalent sphere diameter, surface-equivalent sphere diameter, and Sauter diameter.
Table 1
Generalized gamma distribution parameters for density functions shown in Fig. 5.

|                  | a        | c        | l     | β          |
|------------------|----------|----------|-------|------------|
| Major Diam.      | 9.6277   | 0.2689   | 0.0   | 0.001301   |
| Vol. Equiv. Diam.| 9.9077   | 0.2716   | 0.0   | 0.0007602  |
| Surf. Equiv. Diam.| 11.0452 | 0.2489   | 0.0   | 0.0002579  |
| Sauter Diam.     | 11.0759  | 0.2788   | 0.0   | 0.0004803  |

Fig. 6. Probability density distribution of sphericity to describe grain shape for the example PBX material demonstrated here. The curve represents the best fit generalized gamma function for sampling later during the generation of artificial microstructures. The parameters are: $a = 0.2375$, $c = 35.5843$, $l = 0.0$, $\beta = 0.8655$.

The location and scaling of the gamma function can be accomplished by substituting $x$ with:

$$y = \frac{x - l}{\beta},$$

where $l$ is a shift in the $x$ location, and $\beta$ is a vertical scaling factor. Individual particle sizes for the numerically generated geometry can then be randomly sampled from the probability density function with the requisite number of particles for a given domain size, as given by the number density of grains.

The overall shape distribution of the microstructure can be analyzed using a metric called sphericity. Sphericity is a measure of an object’s resemblance of a sphere. It is defined as [21]:

$$\Psi = \pi^{1/3} \left( \frac{6V_p}{A_p} \right)^{2/3},$$

where $V_p$ is the particle’s volume and $A_p$ is the particle’s surface area. The same distribution function described earlier for particle sizes can be formulated to generate a distribution of particle sphericities. Using the example volume in this article, the distribution of grain sphericities is shown in Fig. 6.

Artificial geometry generation

The first step in creating numerically generated artificial microstructural geometry is to generate an array of radii which define a bounding sphere that will contain each grain of arbitrary shape. The number of desired grains can be specified arbitrarily or using an analyzed real microstructural volume of the same size (c.f. Fig. 4). Grain sizes can be uniform, or sampled from a probability density function. In the ongoing example in this article, grain sizes for the artificial microstructure...
were sampled from a generalized gamma distribution from a \( (400 \text{ voxel})^3 \) subvolume of a real microstructure described in the previous section.

Since this generation routine is designed for heterogeneous, random microstructures containing a mixture of suspended aggregates, placement of each grain within the domain is a random process. Each spherical sub-domain which contains an individual grain is assigned a random position within the RVE/SVE domain and checked for overlap with other existing grain domains. If overlap is detected with existing granular sub-domains, a new random position is drawn and the checks are completed again. The routine has a settable parameter for the maximum number of tries to place an individual particle before moving on to the next grain. Due to the random nature of grain placement, the grain placement is ordered from largest to smallest to increase the likelihood of successfully placing all particles without overlap. It would be improbable to fit the largest grain sizes into the domain without overlap if several smaller grains were pre-placed. The routine for mapping grain sizes and grain centers is described in Algorithm 1. All grain centers are stored along with the size of each grain in an array labeled \( S \). The procedure in Algorithm 1 requires three inputs: \( R \), a list or array containing a sorted list of equivalent sphere grain radii; \( X \), a set of tuples defining the overall domain boundaries; and \( \omega \), the maximum number of attempts to try to place an individual particle. The function RandomDraw randomly chooses a grain center coordinate within the specified domain boundaries. The remaining operations perform distance checking to verify no overlap between particle boundary spheres that have already been placed. These methods differ from iterate process described by discrete element methods or other iterative approaches e.g. [22].

Following mapping the size and location of each grain in the domain, each grain center is assigned a shape. Each of the grains is a polyhedron with a desired number of faces — more faces results in a closer approximation to an ellipsoid. Polyhedrals are generated using a procedure developed by [23], which generates a shape of desired complexity that is fully circumscribed by a sphere with unit radius \( (r=1) \). All of the shapes produced are convex hulls with triangular faces. Since the generated shape is fully contained in a sphere with unit radius, some loss of volume fraction can be expected.

Algorithm 1  Grain placement algorithm.

1: \textbf{procedure} GRAINCENTERS\( (R, (X), \omega) \) \> R, grain size array; \( (X) \), domain boundary; \( \omega \), placement attempt per particle
2: \textbf{for all} \( r \leftarrow R \) \> \text{Index through sorted size array}
3: \hspace{1em} attempt \leftarrow 0
4: \hspace{1em} \textbf{while} attempt < \( \omega \) \> \text{Try placing grain until max. attempts exceeded.}
5: \hspace{2em} \( x, y, z, r \leftarrow \text{RANDOMDRAW}(\langle X \rangle^- \langle X \rangle^+) \) \> Gen. coords. inside bound box
6: \hspace{2em} \textbf{if} \( S \) \textbf{not exist then} \> \text{If no grains created, place first one}
7: \hspace{3em} \( S \leftarrow (x, y, z) \)
8: \hspace{2em} \textbf{continue}
9: \hspace{2em} \hspace{1em} \textbf{end if}
10: \hspace{2em} distance = \( (S[:,0]-x)^2 + (S[:,1]-y)^2 + (S[:,2]-z)^2 \)^{0.5}
11: \hspace{2em} dist\_check = distance = \( (S[:,0]-1) + r \)
12: \hspace{2em} \textbf{if} any \( \text{dist\_check < 0 then} \) \> Overlap detected, try again
13: \hspace{3em} attempt = attempt + 1
14: \hspace{2em} \textbf{continue}
15: \hspace{2em} \hspace{1em} \textbf{else} \> No overlap, append coords. to list, get new particle
16: \hspace{3em} attempt = 0
17: \hspace{3em} APPEND\( (S \) with \( x, y, z, r) \)
18: \hspace{2em} \textbf{break}
19: \hspace{2em} \textbf{end if}
20: \hspace{2em} \textbf{end while}
21: \hspace{2em} \textbf{end for}
22: \textbf{end procedure}
as shown in Fig. 7. Fig. 7 shows the normalized volume of a convex polyhedron containing \( n \) vertices. Volume is normalized to that of a perfect sphere. As the number of vertices increases, the normalized volume increases in an inverted exponential fashion. The fit function and parameters are shown in Fig. 7. The function fits the normalized volume data with \( r^2 = 0.9994 \). The coordinates of the polyhedral vertices can be scaled accordingly using the provided inverted exponential function for determining the normalized volume of a polyhedron with \( n \) vertices to adjust for the volume loss attributed to the number of vertices utilized. This helps maintain the previously analyzed volume fraction assuming volume equivalent sphere diameters were used as a metric to generate the random placement of each grain.

Distortion of the synthetic grain can also be introduced by making use of the sphericity parameter. Sphericity can be sampled from the gamma distribution of grain sphericities as seen in Fig. 6. A few assumptions must be made in order to convert this scalar sphericity parameter back into distortion parameters on the synthetic grain. The general definition of an ellipsoid is under-constrained when only the sphericity and volume are known since an ellipsoid has three independent axes \((j, k, \text{ and } l)\). However, if assuming a spheroid shape, only two axial values are independent with two of the three axes being identical \((j = k)\). A prolate spheroid is produced when \((j > l)\), and an oblate spheroid is produced when \((j < l)\). This distinction is important since both produce different surface areas, i.e.:

\[
A_{\text{prolate}} = 2\pi j^2 \left( 1 + \frac{l}{j^2} \sin^{-1} e \right) \\
A_{\text{oblate}} = 2\pi j^2 + \pi \frac{l^2}{j^2} \ln \left( \frac{1 + e}{1 - e} \right),
\]

where \(e\) is defined as the eccentricity \((e = \sqrt{1 - l^2/j^2})\).

The axes \(j\) and \(l\) are scaled from the spherical radius by scaling factors \(\alpha\) and \(\xi\) respectively. In order to maintain the same volume, these scaling factors are dependent, i.e.:

\[
\xi = \frac{1}{\alpha^2}.
\]

Applying \(\xi\) and \(\alpha\) to distort the axes of the original sphere produces spheroids with identical volumes but varying surface areas. After choosing either a prolate or oblate shape, \(\xi\) can be numerically solved for which satisfies the sampled sphericity by substituting in the original spherical radius, scaling factors from Eq. 8, surface area from Eq. 7, and ellipsoid volume into Eq. 6.
The vertices that make up each polyhedral are scaled (volume), deformed (spheroid), and randomly rotated (rotations sampled from a uniform distribution if assumed random) to each position defined in the size and location array generated in Algorithm 1.

Since the procedure above defines each shape as a set of triangular faces, little effort is required to translate the data into a standard STL formatted file. At this point, the numerically generated artificial geometry is in the same state of processing as the surface reconstruction of the real microstructural geometry. The artificial geometry can be analyzed using the same techniques described in the previous subsection to verify statistical equivalence to the real microstructure in terms of morphometry. Two artificial microstructures with identical grain sizes and locations are shown in Fig. 8 alongside the real microstructure analyzed to generate the artificial microstructures.

Conversion to volume mesh

Both real and artificially generated microstructural volumes are treated the same from this point forward. Often, the surface reconstruction of the real microstructure leaves the surface jagged and stair-stepped as a result of inadequate smoothing of the voxel edges during the initial surface reconstruction phase. In addition to the required smoothing, some STL formatted files suffer from duplicated vertices (e.g. vertices defining adjacent surface facets) depending on the software that wrote the STL file. MeshLab [12] is an open-source surface mesh viewer and editor which provides capabilities to repair vertex duplication and has several options to smooth and reconstruct meshes depending on the initial quality of the surface mesh of the reconstructed microstructure. Both real and numerically generated microstructures are subjected to the repair procedure described below to ensure compliant geometries are utilized in the remaining processes.

Upon opening the surface reconstruction STL file in Meshlab, duplicate vertices are unified. Following this, the surface is smoothed using a two step smoothing filter based on the algorithm in [24]. The two step smoothing algorithm is a feature preserving filter where similar face normals are averaged together with a step in which vertices are fit to the new face normals. The filter contains four options: feature angle threshold, number of face normal smoothing steps, number of vertex fitting steps, and number of global smoothing steps (normal smoothing and vertex fitting). For the demonstration presented here, the feature angle was set at 45°, twenty iterations of the normal smoothing and vertex fitting was completed in only one global iteration. A before and after comparison of the smoothed mesh is illustrated in Fig. 9. Following smoothing, isolated particles three voxels in diameter or less are filtered out since features of this size cannot be reasonably accommodated later during volumetric meshing. The filtering of isolated pieces this small did not appreciably affect the overall volume fraction of the grains in several iterations of the example followed here.
Fig. 9. (Left) Initial surface reconstruction of a real microstructure before smoothing and (right) after smoothing using the TwoStep filter implemented in MeshLab [12,24].

All of the above MeshLab operations can be performed in the Graphical User Interface (GUI). Limited functionality exists to automate the process through batch scripting, although only a specific version of MeshLab (v1.3.2_64bit running on Ubuntu v14.04 LTS and 16.04 LTS) was able to perform these functions in batch mode without errors. The batch script used to operate MeshLab in batch mode is shown in Algorithm 2.

Algorithm 2 XML formatted Meshlab procedure script.

```
<!DOCTYPE FilterScript>
<FilterScript>
  <filter name="Merge Close Vertices">
    <Param type="RichAbsPerc" value="0" min="0"
      name="Threshold" max="1.72052"/>
  </filter>
  <filter name="TwoStep Smooth">
    <Param type="RichInt" value="1" name="stepSmoothNum"/>
    <Param type="RichFloat" value="45" name="normalThr"/>
    <Param type="RichInt" value="20" name="stepNormalNum"/>
    <Param type="RichInt" value="20" name="stepFitNum"/>
    <Param type="RichBool" value="false" name="Selected"/>
  </filter>
  <filter name="Remove Isolated pieces (wrt Diameter)">
    <Param type="RichAbsPerc" value="3.0" min="0"
      name="MinComponentDiag" max="100.0"/>
  </filter>
</FilterScript>
```

Following these procedures, the surface representation of the microstructural geometry will be watertight and convex — making it valid for volumetric meshing. Initial volumetric meshing is accomplished by a parallel-computing capable program called Sculpt [25,26] which comes packaged with Sandia National Laboratories' Cubit program [1]. Cubit is available to U.S. government research facilities and a commercially licensed version is available through Csimsoft as Trelis. Sculpt [25,26] is built to convert surface geometry represented by one or more STL files to fully filled volumetric meshes containing predominantly hex elements.

The procedure for producing high quality hex meshes from complex geometries in Sculpt is detailed in [25] and [26] and illustrated in Fig. 10. Sculpt begins by imposing the geometric surfaces of the input STL files over a regular Cartesian grid (Fig. 10a). The nodes on the base grid nearest to the boundaries of the projected geometry are distorted to roughly follow the underlying geometry
(Fig. 10b). A new layer of hex elements are inserted in an action called pillowing, whereby geometric boundary nodes are duplicated and displaced away from the boundary (Fig. 10c,d). Smoothing procedures could be applied to the newly formed mesh to improve the quality of mesh near geometric boundaries (Fig. 10e).

Several command line options control base grid size, grid adaptation for refining the mesh in complex areas (described in [26]), smoothing procedures, and an option to mesh the void. Meshing the void enables the user to provide only the aggregate particles of a composite and the void space in the specified domain volume is filled with a separate material (e.g. the matrix material). If trying to reproduce actual voids found in the sample, separate surface representations of the voids can be provided, meshed as if they were solid materials, then removed from the final mesh resulting in discrete void spaces. Depending on the complexity of the meshed geometry and levels of grid adaptivity selected by the user, a fully conforming and usable mesh can be produced for post-processing into a FEA/CFD preprocessor.

In the example provided here, the base grid specified to Sculpt contained a grid with individual elements with a dimension of 4 voxels (pertaining to the original μCT image resolution) and 2 levels of adaptive refinement allowed. The resulting mesh contained approximately 1.5 million hex elements. The random selection of the RVE boundary location in conjunction with grain boundaries located near the edges of the RVE produced thin slivers of material (either solid crystals or matrix material) around the corners of the RVE. Limiting the adaptive refinement to 2 levels, thin elements with poor aspect ratios were placed in these troublesome regions. Special localized refinement exceeding the global specifications of grid adaptivity is not permissible in the Sculpt meshing application and increasing the global refinement results in an over-refined mesh. To overcome these difficulties, Sculpt also provides the option to detect and automatically remove malformed elements (e.g. negative scaled jacobian). In our example, this option manifests in single element voids on parts of the corners of the boundary of a meshed microstructure as seen in an example in Fig. 11. The removal process deleted about 20 elements out the 1.5 million elements produced. Element deletion is not necessary in all cases but provides a reasonable alternative to the user when additional mesh refinement is not practical or desired.

The Sculpt software outputs the volumetric mesh in exodusII format (*.e). This format is supported by the parent program, Cubit. Cubit (or its commercial equivalent, Trelis) is used to perform post-processing operations such as node and element set labeling, insertion of cohesive elements (if modeling cohesive behavior between materials), and exporting the mesh in formats for common FEA and CFD packages (e.g. Abaqus, Fluent, Nastran, etc.).

Volume mesh manipulation and processing

Although a volumetric mesh is established using the Sculpt parallel mesher, several steps remain to post-process the resulting mesh to make it useful to an FEA/CFD preprocessor. The exodusII mesh produced by sculpt only contains $n$ number of volume definitions where $n$ is the number of STL files
Fig. 11. Final volumetric hex mesh of a real 3D microstructure of a PBX material with a 1/8 section removed to show internal mesh structure. Refinements to the mesh are controlled by global refinement parameters to adapt the base grid in a stepped pattern within a specified limit. Adaptations exceeding a pre-specified limit cannot be applied locally resulting in poor element formulations on the corners of this RVE. This often results from a thin sliver of material running near the edge of the domain that would require higher levels of refinement than specified to achieve properly formed elements. This mesh contains a base grid with dimensions of 4 voxels (from original μCT image) per grid element and 2 levels of adaptive refinement. This results in approximately 20 poorly formed elements removed from a mesh containing approximately 1.5 million elements.

provided as input to Sculpt $(n + 1$ volumes if the void is meshed). This is the case even if a single STL file contains multiple independent volumes.

When importing the volumetric mesh into Cubit, the user has an option to enable creation of Mesh Based Geometry (MBG). The creation of MBG enables identification of separate volumes: a critical step when assigning single crystal properties to individual grains in a heterogeneous microstructure. Following this step, each grain in the microstructure is identified with a unique volume label. In a binary material (e.g. explosive crystals in a polymer binder), sorting the bodies based on volume provides a robust way to easily identify the volume associated with the matrix separate from the individual grains since, in this scenario, the matrix material will always be the single largest continuous volume in the domain. The volume labels from the MBG are transferred to the associated element blocks so the names are carried in the exported mesh.
In our application, bond behavior between the aggregate grains and the polymer matrix is of great importance. Cohesive elements provide a solution to allow the interface between individual grains and the binder to degrade based on traction-separation criteria specific to each material pair interface [27]. Since the volumes have been sorted, the python interface within Cubit allows the user to efficiently loop through all of the volumes associated with individual grains and insert a skin of zero thickness elements using the pillow command in Cubit. The pillow command performs an operation similar to that in Sculpt and illustrated in Fig. 10, except that the thickness can be controlled. When specifying a zero thickness pillow of elements, the nodes on the affected interface are duplicated creating a new 8 noded hex element in which the top and bottom surfaces have zero separation creating a pseudo-quadrilateral element that can also act as a cohesive element between regular element boundaries. The new skin of elements are free elements (not associated with an identified block of elements).

Cubit provides command line based selection tools to select these newly created free elements in a single block. This new skin of elements receives a unique element set name which identifies them as cohesive elements when subsequently read by the preprocessor of the FEA package Abaqus.

The python interface within Cubit also provides an efficient way to automatically label special features of the mesh for easier application of boundary conditions, material properties, and constraint equations. Boundary conditions and constraint equations benefit from having labels for nodes on the boundary identified as either being on a face, edge, or corner of the rectangular domain. After processing in Cubit and export as an Abaqus mesh (outputs may differ for other package formats), the following labeled sets exist:

- Numbered element set and label for all elements and nodes associated with each unique grain/particle in the microstructure,
- One element set and label identifying the nodes and elements within the matrix material.
- One element set labeling cohesive elements inserted between grains and the binder.
- Node sets identifying each node on the corner of the domain.
- Node sets identifying nodes on each edge of the domain exclusive of those identified in the corner node sets, and
- Node sets identifying nodes on each face of the domain exclusive of both nodes in the previously defined corner and edge node sets.

Cubit provides robust batch capabilities with its own python package and interface. The work flow described in this section is provided as an open-source code by this author at https://github.com/lanl/MicroMesh.

The result of this process is a valid volumetric mesh for finite element analysis with cohesive behavior between particles in a heterogeneous bound composite. Fig. 11 shows a sectioned view of the final volumetric mesh illustrating the individual explosive crystals, cohesive elements, and binder material ready for finite element analysis. Although important for our analysis purposes, the procedure for producing cohesive elements could be omitted and still produce a valid volumetric mesh. This article outlines the steps and procedures taken by this research group to collect three dimensionally imaged PBX microstructures and convert them to usable finite element meshes. Additional tools were developed to produce numerically generated microstructures designed to perform similarly to the real microstructures both at the micro- and macro- scales (described in Algorithms 1 and 2).

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors acknowledge the support of the U.S. Department of Energy under U.S. Government contract 89233218CNA000001 for Los Alamos National Laboratory (LANL) and specifically the
Laboratory Research and Development program at LANL under project number 20160619ECR for supporting this work and associated software releases.

References

[1] Cubit. (Software), Sandia National Laboratories (2017).
[2] V.W. Manner, J.D. Yeager, B.M. Patterson, D.J. Walters, J.A. Stull, N.L. Cordes, D.J. Luscher, K.C. Henderson, A.M. Schmalzer, B.C. Tappan, In situ imaging during compression of plastic bonded explosives for damage modeling, Materials 10 (6) (2017) 638, doi:10.3390/ma10060638.
[3] D.J. Walters, E.E. Adams, Micromechanical applications to radiation recrystallized snow, in: International Snow Science Workshop, 2014, pp. 608–615. Banff, AB.
[4] E.E. Adams, D.J. Walters, Fine structure layering in radiation recrystallized snow, in: International Snow Science Workshop, 2014. Banff, AB.
[5] E.H. Saenger, M. Lebedev, D. Uribe, M. Osorno, S. Vialle, M. Duda, S. Iglauser, H. Steeb, Analysis of high-resolution X-ray computed tomography images of Bentheim sandstone under elevated confining pressures, Geophys. Prospect. 64 (4) (2016) 848–859, doi:10.1111/1365-2478.12400.
[6] A. Barua, Y. Horie, M. Zhou, Energy localization in HMX-Estane polymer-bonded explosives during impact loading, J. Appl. Phys. 111 (5) (2012) 054902(11), doi:10.1063/1.3688350.
[7] E. Tarleton, M. Charalambides, C. Leppard, Image-based modelling of binary composites, Comput. Mater. Sci. 64 (2012) 182–186, doi:10.1016/j.commatsci.2012.02.048.
[8] M.M. Roozbahani, L. Graham-Brady, J.D. Frost, Mechanical trapping of fine particles in a medium of mono-sized randomly packed spheres, Int. J. Numer. Anal. Methods Geomech. 38 (17) (2014) 1776–1791, doi:10.1002/nag.2276.
[9] H. Arora, E. Tarleton, J. Li-Mayer, M. Charalambides, D. Lewis, Modelling the damage and deformation process in a plastic bonded explosive microstructure under tension using the finite element method, Comput. Mater. Sci. 110 (2015) 91–101, doi:10.1016/j.commatsci.2015.08.004.
[10] D.J. Walters, D.J. Luscher, J.D. Yeager, B.M. Patterson, Cohesive finite element modeling of the delamination of HTPB binder and HMX crystals under tensile loading, Int. J. Mech. Sci. 140 (2018a) 151–162, doi:10.1016/j.ijmecsci.2018.02.048.
[11] D.J. Walters, D.J. Luscher, J.D. Yeager, B.M. Patterson, Investigating deformation and mesoscale void creation in HMX-based composites using tomography-based grain scale FEM, in: AIP Conference Proceedings Shock Compression of Condensed Matter 2017, 070031, AIP Publishing, St. Louis, MO, USA, 2018b, p. 6, doi:10.1063/1.5044840.
[12] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, MeshLab: an open-source mesh processing tool, in: V. Scarnano, R.D. Chiara, U. Erra (Eds.), Sixth Eurographics Italian Chapter Conference, The Eurographics Association, 2008, pp. 129–136, doi:10.2312/LocalChapterEvents/ItalianChap/ItalianChapConf2008/129-136.
[13] C. Geuzaine, J.-F. Remacle, Gmsh: a 3-d finite element mesh generator with built-in pre- and post-processing facilities, Int. J. Numer. Methods Eng. 79 (11) (2009) 1309–1331, doi:10.1002/nme.2579.
[14] R. Quey, P. Dawson, F. Barhe, Large-scale 3d random polycrystals for the finite element method: generation, meshing and remeshing, Comput. Methods Appl. Mech. Eng. 200 (17) (2011) 1729–1745, doi:10.1016/j.cma.2011.01.002.
[15] M.A. Groeber, M.A. Jackson, DREAM3d: a digital representation environment for the analysis of microstructure in 3d, Integr. Mater. Manuf. Innov. 3 (1) (2014) 56–72, doi:10.1186/2193-9772-3-5.
[16] L.A. Feldkamp, L.C. Davis, J.W. Kress, Practical cone-beam algorithm. J. Opt. Soc. Am. A 1 (6) (1984) 612, doi:10.1364/JOSAA.1.100062.
[17] B. Patterson, K. Henderson, P. Gibbs, S. Imhoff, A. Clarke, Laboratory micro- and nanoscale X-ray tomographic investigation of deformation structures, Mater. Charact. 95 (2014) 18–26, doi:10.1016/j.matchar.2014.06.004.
[18] S. Beucher, F. Meyer, The morphological approach to segmentation: the watershed transformation, in: E. Dougherty (Ed.), Mathematical Morphology in Image Processing, CRC Press, New York, NY, 1993, pp. 433–481.
[19] F. Meyer, Topographic distance and watershed lines, Signal Process. 38 (1) (1994) 113–125, doi:10.1016/0165-1684(94)90060-4.
[20] W.E. Lorenzen, H.E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, in: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH ’87, 21, ACM Press, New York, New York, USA, 1987, pp. 163–169, doi:10.1145/37401.37422.
[21] H. Wadell, Volume, shape, and roundness of quartz particles, J. Geol. 43 (3) (1935) 250–280, doi:10.1086/624298.
[22] M. Roozbahani, R. Borela, J. Frost, Pore size distribution in granular material microstructure, Materials 10 (12) (2017) 1237, doi:10.3390/ma10111237.
[23] S. Tatham, Constructing Polyhedra from Repelling Points on a Sphere, 2007.
[24] A. Belyaev, Y. Ohtake, A comparison of mesh smoothing methods, in: Israel-Korea Bi-national Conference on Geometric Modeling and Computer Graphics, 2003, pp. 83–87. 10
[25] S.J. Owen, T.R. Shilton, Validation of grid-based hex meshes with computational solid mechanics, in: 22nd International Meshing Roundtable, Orlando, FL, USA, 2014, pp. 39–56, doi:10.1007/978-3-642-33573-0.
[26] S.J. Owen, R.M. Shih, C.D. Ernst, A template-based approach for parallel hexahedral two-refinement, in: 24th International Meshing Roundtable, Austin, TX, USA, 2015, p. 13, doi:10.1007/10.1016/j.cad.2016.09.005.
[27] A. Needleman, A continuum model for void nucleation by inclusion debonding, J. Appl. Mech. 54 (3) (1987) 525, doi:10.1115/1.3173064.