SoftwareX 12 (2020) 100595

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

# MicroStructPy: A statistical microstructure mesh generator in Python

Kenneth A. Hart[1], Julian J. Rimoli[*,2]

Georgia Institute of Technology, Atlanta, GA 30332, USA

ARTICLE INFO

*Article history:*
Received 8 July 2020
Received in revised form 25 August 2020
Accepted 10 September 2020

*Keywords:*
Microstructure
Mesh generation
Polycrystal
Laguerre tessellation

ABSTRACT

MicroStructPy is a statistical microstructure mesh generator written in Python. This software package includes classes and methods to generate a mesh by (i) creating a list of grain seed geometries, (ii) packing them into a domain, (iii) performing a Laguerre tessellation of the seed geometries, and (iv) performing quality unstructured meshing. Results can be visualized and compared with the specified microstructural properties. MicroStructPy accurately reproduces 2D and 3D polycrystalline microstructures with arbitrary number of phases, volume fractions and distributions, each including the possibility of elongated grains. It can also generate meshes for amorphous phases and porous materials. Meshes are suitable for direct numerical simulation, a prevalent technique in computational mechanics of materials and geomechanics. The package contains extensive documentation, including guides and demonstrations to facilitate adoption for new users.

## Code metadata

| | |
|---|---|
| Current code version | v1.3.2 |
| Permanent link to code/repository used for this code version | https://github.com/ElsevierSoftwareX/SOFTX-D-20-00001 |
| Code ocean compute capsule | |
| Legal code license | MIT License |
| Code versioning system used | Git |
| Software code languages, tools, and services used | Python |
| Compilation requirements, operating environments & dependencies | Python 3.6+; Windows, MacOS, or Linux; aabbtree, lsq-ellipse, matplotlib, meshpy, numpy, pybind11, pyquaternion, pyvoro-mmalahe, scipy, xmltodict |
| If available Link to developer documentation/manual | https://docs.microstructpy.org |
| Support email for questions | support@microstructpy.org |

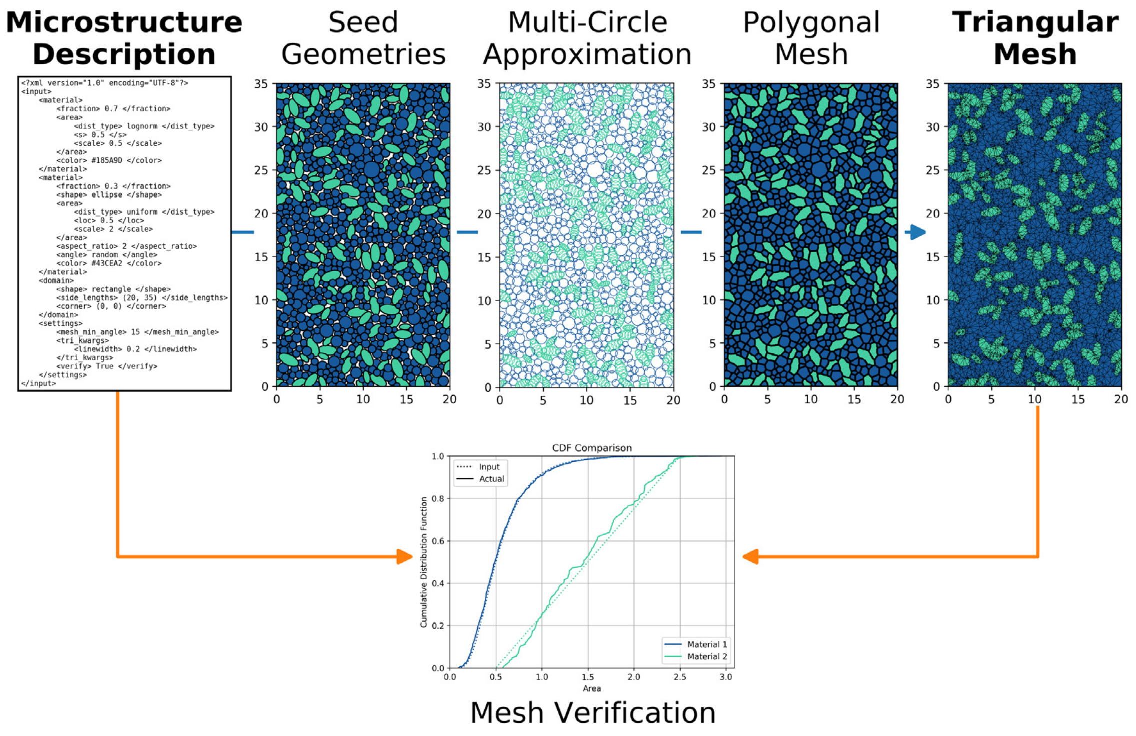
## 1. Motivation and significance

The response of a material is significantly influenced by the properties and composition of its constituents, hereafter referred to as the microstructure of the material. Observations of steel indicate that specimen with smaller grains have higher ultimate strength [1,2]. Grain size and aspect ratio also influence the thermal conductivity of materials [3,4]. Finally, additive manufacturing can create *anisotropic* bulk material properties from an *isotropic* build material [5]. Precise and reliable prediction of macroscale material properties from the microscale constituents is currently an active area of research. Modern prediction techniques often rely on direct numerical simulation (DNS), whereby relevant properties and boundary conditions are applied to a representative volume element (RVE) mesh.

There are several techniques to characterize the microstructure of a material. Each of these characterizations is associated with several more techniques to virtually reconstruct the microstructure. Modern scanning techniques include 3-D X-ray diffraction (3D-XRD) and X-ray tomography, which generate a

* Corresponding author.
*E-mail address:* rimoli@gatech.edu (J.J. Rimoli).
[1] Ph.D. Candidate, Daniel Guggenheim School of Aerospace Engineering.
[2] Associate Professor, Daniel Guggenheim School of Aerospace Engineering.

https://doi.org/10.1016/j.softx.2020.100595



```
<?xml version="1.0" encoding="UTF-8"?>
<input>
    <material>
        <fraction> 0.7 </fraction>
        <area>
            <dist_type> lognorm </dist_type>
            <s> 0.5 </s>
            <scale> 0.5 </scale>
        </area>
        <color> #185A9D </color>
    </material>
    <material>
        <fraction> 0.3 </fraction>
        <shape> ellipse </shape>
        <area>
            <dist_type> uniform </dist_type>
            <loc> 0.5 </loc>
            <scale> 2 </scale>
        </area>
        <aspect_ratio> 2 </aspect_ratio>
        <angle> random </angle>
        <color> #43CEA2 </color>
    </material>
    <domain>
        <shape> rectangle </shape>
        <side_lengths> (20, 35) </side_lengths>
        <corner> (0, 0) </corner>
    </domain>
    <settings>
        <mesh_min_angle> 15 </mesh_min_angle>
        <tri_kwargs>
            <linewidth> 0.2 </linewidth>
        </tri_kwargs>
        <verify> True </verify>
    </settings>
</input>
```

**Fig. 1.** The microstructure mesh generation process. A textual description of the microstructure is converted, through steps described below, into an unstructured triangular or tetrahedral mesh. Statistics of the mesh can be compared against those prescribed, to ensure microstructure fidelity.

voxel raster of the microstructure [6,7]. These rasters can be directly converted into a microstructural mesh for high-fidelity DNS [8]. The raster can also be converted into $n$-point correlation functions, which compress the microstructure data into several functions [9]. Correlation functions can be inverted to generate multiple realizations of the same microstructure using global optimization and phase recovery [10]. The raster can also be converted into statistical distributions, such as volume fractions and grain volume distributions [11]. These distributions can be used to reconstruct the microstructure by packing seed geometries together, then creating a dual such as the Laguerre–Voronoi diagram [12,13] or the barycentric dual [14]. The work presented follows this paradigm due to the prevalence and availability of volume fractions and grain size distributions for multiple materials.

In this paper we introduce MicroStructPy, a statistical microstructure mesh generator written in Python [15]. MicroStructPy is part of a growing ecosystem of open source microstructure mesh generators which cover different needs through varying functionalities. Neper, for instance, produces statistically representative microstructures by optimizing the location and weight of tessellation points [13,16]. The Dream.3D software produces microstructural meshes using a general system of pipelines [17]. Finally, PuMA is a microstructure software that is specialized for porous materials, such as fiber-reinforced composites and foams [18]. MicroStructPy is unique in that it reproduces statistical microstructure distributions without the need for optimization and it can create elongated grains, using a multi-sphere approach [19].

The reminder of this paper is organized as follows. Section 2 describes the software architecture and functionality, including a snippet of the most important methods. Section 3 contains an illustrative example. Finally, Section 4 discusses the impact that MicroStructPy has on the research community and Section 5 concludes the work.

## 2. Software description

### 2.1. Software architecture

MicroStructPy follows four principal steps to generate the mesh of a microstructure, as shown in Fig. 1: (i) packing seed geometries into the domain, (ii) converting seeds into multi-circle/multi-sphere approximations, (iii) tessellating the domain to create a polygonal/polyhedral mesh, and (iv) quality unstructured meshing to create a triangular/tetrahedral mesh. For verification, the statistics of the output mesh are compared against the desired statistics for the microstructure, which are specified in the input file. The methods associated with each of these steps are described in the following subsections.

#### 2.1.1. Seed geometries

Generating seed geometries and packing them into a domain are the first steps in creating a microstructure. In MicroStructPy, the information about the microstructure is included in an input file, which contains specifications for each constituent phase, as well as a definition of the geometry of the domain. This information is used to create an initial list of un-positioned seeds. The volume fraction, $V_f$, and average grain volume, $\mathbb{E}[V]$, are used to determine the population density, $n$, of each phase according to Eq. (1), where the subscript $i$ indicates the population density of the $i$th phase. Since $n_i$ is the number of grains of phase $i$ per unit volume, the probability that a given grain belongs to this phase is given by Eq. (2).

$$n_i = \frac{V_{f,i}}{\mathbb{E}[V_i]} \tag{1}$$

$$p_i = \frac{n_i}{\sum_j n_j} \tag{2}$$

The list of un-positioned grains is therefore created by sampling a categorical distribution with weights $p_i$ to determine the phase of a seed, sampling the distributions associated with that phase, and repeating the process until the volume of seeds in the list matches the volume of the domain. For example, the second material in Fig. 1 contains elliptical seeds with uniformly distributed area, on the distribution $U[0.5, 2.5]$, a constant aspect ratio of 2, and uniformly random rotation angle. When material 2 is chosen, an ellipse is created by sampling the area distribution and the rotation angle distribution, solving for the length and height of the ellipse based on its area and aspect ratio, then rotating it through the sampled rotation angle.

Once the list has been generated, the seeds are positioned within the domain in decreasing order of grain volume. The first seed is positioned by sampling the position distribution associated with its phase, which defaults to uniform random throughout the domain. Subsequent seeds are positioned by sampling a position, checking for overlap with previously positioned seeds, then resampling if the overlap exceeds a relative tolerance. Allowing the seeds to overlap results in a better correlation between the seed and grain volumes, compared to not allowing overlap. Voids between seeds will add their volume to the grains in the tessellation, increasing the overall grain size distribution. The overlapping volume of nearby seeds is subtracted from the grain volume, balancing the added volume from nearby voids. Overlap is determined using the multi-circle/multi-sphere approximations for the seeds, since circle–circle overlap detection is straightforward. Let the subscript $i$ denote a circle in the approximation for the grain being positioned and the subscript $j$ denote a circle in the approximation for a previously positioned grain, the two grains are overlapping if

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\| + \alpha \min(r_i, r_j) < r_i + r_j \tag{3}$$

where $\boldsymbol{x}$ is the center of a circle, $r$ is the radius, and $\alpha$ is the relative overlap tolerance. A calibration curve is included in MicroStructPy to set an appropriate value of $\alpha$ so that the errors between prescribed and obtained microstructural parameters are minimized [19]. This overlap check is performed with a subset of all of the previously positioned seeds, to reduce computational cost. The subset is determined using an axis-aligned bounding box (AABB) tree, a spatial hierarchy designed to group seeds with those in their vicinity. Each seed is bounded by an AABB, so if two seeds overlap then their AABBs also overlap. These AABBs are organized into a binary tree, where two AABBs are surrounded by a larger AABB, then that AABB is grouped with another, and this continues until there is one AABB that surrounds all of the seed AABBs. In this binary tree, overlap testing requires fewer computations because descending the tree effectively reduces the number of potential overlaps by half [19]. This hierarchy accelerates the seeding process, resulting in a microstructure domain densely packed with seed geometries.

### 2.1.2. Multi-circle approximation

Decomposing seed geometries into sets of circles/spheres streamlines the overlap checking process in the previous step and provides a suitable list of weighted points for the next step. This is an extension of the work by Ilin and Bernacki [20], who generated elliptical grains using multiple Laguerre–Voronoi cells. As shown in Fig. 2, from [19], the Ilin and Bernacki method is first applied to two cross sections of the ellipsoid, labeled (a) and (b). The circle centers from (a) and (b) are then mapped into the third cross-section, which is labeled (c). Next, centers are added at locations that are offset from the outer curve of the third cross section, labeled (d). Radii are assigned to these centers based on the minimum distance between the points and the surface of the ellipsoid. Lastly, the results from this section of the ellipsoid are mirrored into the other quadrants to create a multi-sphere approximation of the ellipsoid. Rectangular grains are also approximated by a series of circles, with additional circles resolving the corners. Through multi-circle approximation, the dense pack of seed geometries is transformed into a collection of circles/spheres.

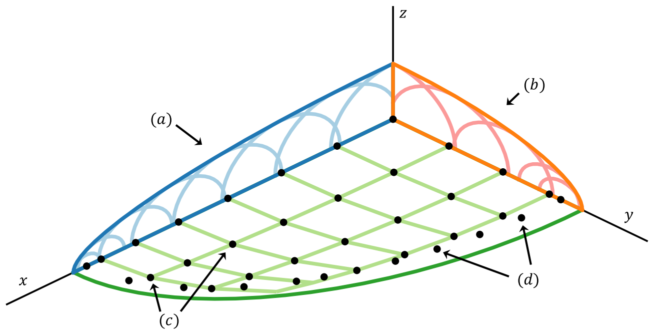


**Fig. 2.** Multi-sphere centers for an ellipsoid. (a) Multi-circle approximation of the $xz$ plane, (b) Multi-circle approximation of the $yz$ plane, (c) Grid of non-terminal circle centers, (d) Centers of spheres tangent at the $xy$ plane.

### 2.1.3. Polygonal mesh

The collection of circles/spheres generated in the previous step is used to create a Voronoi diagram in Laguerre geometry, also known as a power diagram [21]. The Laguerre–Voronoi diagram is a tessellation of $d$-dimensional Euclidean space, $E^d$, based on the distance to the nearest seed point. Let the sphere $s$ have center $\boldsymbol{c}$ and radius $r$, the power distance from any point $\boldsymbol{p}$ to $s$ is given by Eq. (4), where $\|\boldsymbol{p} - \boldsymbol{c}\|$ is the Euclidean distance between $\boldsymbol{p}$ and $\boldsymbol{c}$. If $S$ is the collection of spheres $s$, the region of $E^d$ associated with sphere $s$ is defined by Eq. (5).

$$\mathrm{pow}(\boldsymbol{p}, s) = \|\boldsymbol{p} - \boldsymbol{c}\|^2 - r^2 \tag{4}$$

$$R(s) = \left\{\boldsymbol{p} \in E^d \mid \mathrm{pow}(\boldsymbol{p}, s) < \mathrm{pow}(\boldsymbol{p}, t), \forall\, t \in S - \{s\}\right\} \tag{5}$$

Since the seeds geometries are densely packed, the Laguerre–Voronoi cells are very close polygonal approximation of the seed geometries. The seed geometries are each samples of statistical distributions and therefore the polygonal cells will have empirical distributions that are very close to the input distributions, so long as there is a statistically significant number of grains.

MicroStructPy creates polygonal/polyhedral meshes using the Voro++ package with a Python wrapper [22]. This mesh may be sufficient for some users, however DNS typically does not support elements of arbitrary geometry. The following step creates unstructured triangular/tetrahedral meshes of the microstructure, which can readily be used in DNS.

### 2.1.4. Triangular mesh

The unstructured triangular/tetrahedral mesh fills the microstructure grains with multiple triangular/tetrahedral elements. The quality of these meshes is controlled by three parameters: (i) the maximum element volume, (ii) the minimum interior angle of an element, and (iii) the maximum edge length at grain boundaries. The last quality control parameter is included to determine the resolution of stress fields or cohesive zones between grains, necessary for common techniques in computational mechanics. Before calling the unstructured mesh generators, Laguerre–Voronoi cells of the same grain are merged together. If a material is flagged as amorphous, cells of the same material phase are also merged together. Finally, materials flagged as voids are removed from the domain prior to unstructured meshing. Unstructured meshing in 2D is performed by Triangle [23] and in 3D by TetGen [24], both of which are accessed through a Python wrapper.

### 2.1.5. Mesh verification

Once the microstructure mesh has been generated, statistics can be gathered on the grains and compared against the input statistics. The volume fraction of each phase is computed by summing the volumes of each element of a given phase and dividing by the volume of the domain. For void phases, the volumes in the polygonal mesh are used instead. Next, the shape and orientation parameters of the grains are computed by fitting a geometry to the mesh vertices at the grain boundaries. For circular and spherical grains, the center and radius of best fit can be determined through linear least squares regression, using a method known as Bullock's algorithm. The center of the circle of best fit is the solution to Eq. (6) and the radius is given by Eq. (7). For elliptical grains, a numerically stable algorithm is applied which guarantees the output is elliptical, rather than the other quadratics [25,26]. Ellipsoidal grains are fit by minimizing the sum squared error in geometric distance between the vertices and the ellipsoid [27].

$$\begin{bmatrix} \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_i y_i \\ \sum_{i=1}^{N} x_i y_i & \sum_{i=1}^{N} y_i^2 \end{bmatrix} \cdot \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \sum_{i=1}^{N} x_i (x_i^2 + y_i^2) \\ \sum_{i=1}^{N} y_i (x_i^2 + y_i^2) \end{bmatrix} \tag{6}$$

$$r = \sqrt{x_c^2 + y_c^2 + \frac{1}{N} \sum_{i=1}^{N} (x_i^2 + y_i^2)} \tag{7}$$

The coefficient of determination, $R^2$, value of the fit geometries, relative to the seed geometries, quantifies how well each seed is represented in the output microstructure. Additionally, non-parametric distributions are developed for the output microstructure to evaluate statistical significance metrics such as Kolmogorov–Smirnov and energy distance [28–30]. If an input distribution is parametric, maximum likelihood estimates (MLEs) for its parameters are reported based on the output microstructure. Verification of grain orientations is limited to 2D microstructures only.

### 2.2. Software functionalities

MicroStructPy generates statistically representative 2D and 3D microstructures. The software supports arbitrarily many material phases, which can be either crystalline, amorphous, or voids. The size, shape, orientation, and position of these grains can be a combination of deterministic and randomly distributed quantities, with any of the random variable distributions included in SciPy including empirical distributions. MicroStructPy includes mesh visualization and verification methods, as well as output to common mesh file types. Finally, the mesh generation process in MicroStructPy is segmented into multiple classes and methods for users to create custom workflows such as an alternative seeding scheme or including a new primitive geometry.

### 2.3. Sample code snippet

Microstructures can be generated either via XML input files or Python scripting. The script in Listing 1 creates an equivalent microstructure to the one shown in Fig. 1. The inputs on Lines 4–23 correspond to the XML in the first panel of Fig. 1. Line 26 creates an un-positioned list of seeds that has the same area as the domain. Line 27 positions those seeds within the domain, similar to the second panel of Fig. 1. The multi-circle approximations of the seeds, shown in the third panel, are stored in the `breakdown` property of each seed. Line 29 creates the polygonal mesh (Laguerre tessellation) as shown in the fourth panel of Fig. 1, where the tessellation edges between cells of the same grain are not plotted. Line 30 creates the triangular mesh, shown in the fifth panel, with a minimum interior angle quality control on the triangular elements. Finally, Lines 32–36 create the output plots and perform one of the verification tests. Microstructures in 3D use the same classes and methods as in 2D.

**Table 1**
Input mesh parameters and output maximum likelihood estimates for an example microstructure.

| Property | Input | Output |
|---|---|---|
| Volume fraction | | |
| Large grains | 0.1250 | 0.1216 |
| Fine grains | 0.6875 | 0.6847 |
| Intrusions | 0.1875 | 0.1937 |
| Grain volume | | |
| Large grains | Triang(0.5, 3.0, 0.5) | Triang(0.214, 2.853, 0.740) |
| Fine grains | Lognormal(1.105, 0.64) | Lognormal(1.097, 0.679) |
| Intrusions | Triang(0, 1.5, 0) | Triang(0, 1.653, 0.030) |

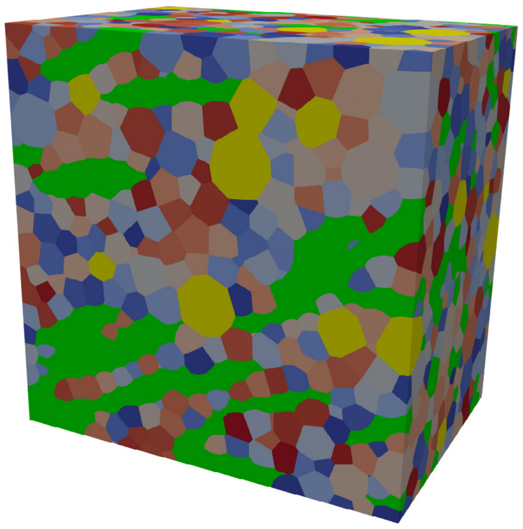


**Fig. 3.** An example microstructure generated by MicroStructPy. The large grains are highlighted in yellow, the intrusion veins in green, and the fine grains shown on a scale from red to blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

## 3. Illustrative example

A polycrystalline material with a venous intrusion is included to demonstrate the capabilities of MicroStructPy. The two crystalline phases contain large and small grains, which are approximately spherical. The veins are modeled as amorphous ellipsoids, meaning that neighboring ellipsoids are merged together. The parameters defining this microstructure are presented in the first column of Table 1, while the output microstructure is shown in Fig. 3.

The volume fractions of the microstructure mesh were computed by summing the volumes of each element of a material phase and dividing by the total. The volume fractions in the output mesh, reported in the second column of Table 1, match the input values to within 0.006. The grain volumes are similarly calculated, however in amorphous phases the volumes are calculated from the tessellation because mesh elements could span multiple grains. The maximum likelihood estimators (MLEs) are computed used the SciPy statistics module and reported in Table 1. These output distributions agree with the input distributions, with the $p$-values for the Kolmogorov–Smirnov test all above 0.05.

## 4. Impact

DNS is perhaps the most prevalent approach when modeling the response of complex materials in order to understand their performance and underlying physics. A microstructure mesh generator such as MicroStructPy, converts a description of the microstructure into a mesh suitable for DNS. The code presented can produce microstructure meshes with any number of material phases. These phases can be crystalline, amorphous, or voids in the material. The size, shape, orientation, and position of these constituents can be directly controlled using any distribution in the SciPy statistical module. . Other microstructure codes individually possess a subset of these capabilities. This positions MicroStructPy very well in the scientific community. We believe its capabilities will enable the research of phenomena not studied before, e.g., the influence of combined microstructural statistics such as grain size, elongation, and orientation simultaneously, on the effective response of advanced materials.

MicroStructPy is unique not only because it provides all of these capabilities but also because it is written in the Python language. The Python implementation of the code enables rapid installation and customization. Extensive documentation, examples, demonstrations, and guides are available online and continuously updated to facilitate adoption of the code. MicroStructPy has been used in unpublished research on the thermal spallation of rocks and diffusion in batteries at the Georgia Institute of Technology, and hydraulic fracture modeling at Louisiana State University, to cite a few examples. The software is included in the NASA Software Catalog and there are dozens of users worldwide. We expect the user base to expand significantly in time and with this publication. MicroStructPy is a capable microstructure mesh generator that is easy to adopt, use, and customize by materials researchers.

## 5. Conclusions

We presented an original microstructure mesh generator written in Python. MicroStructPy can create meshes for a wide variety of material microstructures, supporting (i) multiple material phases, (ii) polycrystalline, amorphous, and void phases, (iii) direct control of grain size, shape, orientation, and position using any of the SciPy statistical distributions, (iv) mesh verification and visualization, and (v) customizable workflows. The software is open source and extensively documented to facilitate code adoption and future improvements. MicroStructPy produces meshes that enable direct numerical simulation of material microstructures.

## Declaration of competing interest

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 National Aeronautics and Space Administration, USA, for providing financial support under cooperative agreement 80NSSC17M0058, and the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, for providing research cyber infrastructure resources and services.

## Appendix. Sample Code Snippet

```python
import microstructpy as msp
import scipy.stats

# Inputs - Material Phases
phase_0 = {
    'color': '#185A9D',
    'shape': 'circle',
    'area': scipy.stats.lognorm(s=0.5, scale=0.5),
    'fraction': 0.7,
}
phase_1 = {
    'color': '#43CEA2',
    'shape': 'ellipse',
    'area': scipy.stats.uniform(loc=0.5, scale=2),
    'aspect_ratio': 2,
    'angle': 'random',
    'fraction': 0.3,
}
p = [phase_0, phase_1]
# Inputs - Domain
d = msp.geometry.Rectangle(limits=[(0, 20), (0,
    35)])
# Inputs - Settings
min_ang = 15  # degrees

# Create Seed Geometries
s = msp.seeding.SeedList.from_info(p, d.area)
s.position(d)
# Create Polygonal & Triangular Meshes
pmesh = msp.meshing.PolyMesh.from_seeds(s, d)
tmesh = msp.meshing.TriMesh.from_polymesh(pmesh, p
    , min_ang)
# Plot & Verify Results
msp.cli.plot_seeds(s, p, d, ['seeds.pdf'])
msp.cli.plot_poly(pmesh, p, ['poly.pdf'])
msp.cli.plot_tri(tmesh, p, s, pmesh, ['tri.pdf'],
    lw=0.2)
fit_s = msp.verification.seeds_of_best_fit(s, p,
    pmesh, tmesh)
msp.verification.plot_distributions(fit_s, p, ext=
    'pdf')
```

**Listing 1:** Workflow snippet to reproduce the plots in Figure 1.

## References

[1] Hall EO. The deformation and ageing of mild steel: III Discussion of results. Proc Phys Soc Sec B 1951;64(9):747–53. http://dx.doi.org/10.1088/0370-1301/64/9/303.

[2] Petch NJ. The cleavage strength of polycrystals. J Iron Steel Inst 1953;174:25–8. http://dx.doi.org/10.1007/BF01972547.

[3] Maldovan M. Micro to nano scale thermal energy conduction in semiconductor thin films. J Appl Phys 2011;110(3):034308–1–6. http://dx.doi.org/10.1063/1.3607295.

[4] Bouquet J-BP, Rimoli JJ. A length-dependent model for the thermomechanical response of ceramics. J Mech Phys Solids 2015;82:82–96. http://dx.doi.org/10.1016/j.jmps.2015.05.018.

[5] Wang Z, Palmer TA, Beese AM. Effect of processing parameters on microstructure and tensile properties of austenitic stainless steel 304L made by directed energy deposition additive manufacturing. Acta Mater 2016;110:226–35. http://dx.doi.org/10.1016/j.actamat.2016.03.019.

[6] Groeber M, Ghosh S, Uchic MD, Dimiduk DM. A framework for automated analysis and simulation of 3D polycrystalline microstructures. Part 1: Statistical characterization. Acta Mater 2008;56(6):1257–73. http://dx.doi.org/10.1016/j.actamat.2007.11.041.

[7] Ludwig W, Reischig P, King A, Herbig M, Lauridsen EM, Johnson G, Marrow TJ, Buffire JY. Three-dimensional grain mapping by X-ray diffraction contrast tomography and the use of friedel pairs in diffraction data analysis. Rev Sci Instrum 2009;80(033905):1–9. http://dx.doi.org/10.1063/1.3100200.

[8] Langer SA, Fuller ER, Carter W. Oof: An image-based finite-element analysis of material microstructures. Comput Sci Eng 2001;3(3):15–23. http://dx.doi.org/10.1109/5992.919261.

[9] Torquato S. Random heterogeneous materials: Microstructure and macroscopic properties. Interdisciplinary applied mathematics. New York: Springer Science and Business Media; 2013, http://dx.doi.org/10.1115/1.1483342, arXiv:9809069v1.

[10] Fullwood DT, Niezgoda SR, Kalidindi SR. Microstructure reconstructions from 2-point statistics using phase-recovery algorithms. Acta Mater 2008;56(5):942–8. http://dx.doi.org/10.1016/j.actamat.2007.10.044.

[11] Groeber M, Ghosh S, Uchic MD, Dimiduk DM. A framework for automated analysis and simulation of 3D polycrystalline microstructures. Part 2: Synthetic structure generation. Acta Mater 2008;56(6):1274–87. http://dx.doi.org/10.1016/j.actamat.2007.11.040.

[12] Fan Z, Wu Y, Zhao X, Lu Y. Simulation of polycrystalline structure with Voronoi diagram in Laguerre geometry based on random closed packing of spheres. Comput Mater Sci 2004;29(3):301–8. http://dx.doi.org/10.1016/j.commatsci.2003.10.006.

[13] Quey R, Renversade L. Optimal polyhedral description of 3D polycrystals: Method and application to statistical and synchrotron X-ray diffraction data. Comput Methods Appl Mech Engrg 2018;330:308–33. http://dx.doi.org/10.1016/j.cma.2017.10.029.

[14] Rimoli JJ, Ortiz M. A duality-based method for generating geometric representations of polycrystals. Internat J Numer Methods Engrg 2011;86(9):1069–81. http://dx.doi.org/10.1002/nme.3090.

[15] Hart KA, Rimoli JJ. Microstructpy (version 1.3.2). 2020, http://dx.doi.org/10.5281/zenodo.3940607.

[16] Quey R, Dawson PR, Barbe F. Large-scale 3D random polycrystals for the finite element method: Generation, meshing and remeshing. Comput Methods Appl Mech Engrg 2011;200(17–20):1729–45. http://dx.doi.org/10.1016/j.cma.2011.01.002.

[17] Groeber MA, Jackson MA. DREAM.3D: A digital representation environment for the analysis of microstructure in 3D. Integrating Mater Manuf Innov 2014;3(1):5. http://dx.doi.org/10.1186/2193-9772-3-5.

[18] Ferguson JC, Panerai F, Borner A, Mansour NN. Puma: the porous microstructure analysis software. SoftwareX 2018;7:81–7. http://dx.doi.org/10.1016/j.softx.2018.03.001.

[19] Hart KA, Rimoli JJ. Generation of statistically representative microstructures with direct grain geomety control. Comput Methods Appl Mech Engrg 2020;370:113242. http://dx.doi.org/10.1016/j.cma.2020.113242.

[20] Ilin DN, Bernacki M. Advancing layer algorithm of dense ellipse packing for generating statistically equivalent polygonal structures. Granul Matter 2016;18(3):43. http://dx.doi.org/10.1007/s10035-016-0646-9.

[21] Aurenhammer F. Power diagrams: Properties, algorithms and applications. SIAM J Comput 1987;16(1):78–96. http://dx.doi.org/10.1137/0216006.

[22] Rycroft CH. VORO++: A three-dimensional Voronoi cell library in c++. Chaos 2009;19(4):041111. http://dx.doi.org/10.1063/1.3215722, arXiv:0402209.

[23] Shewchuk JR. Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In: Lin MC, Manocha D, editors. Applied computational geometry towards geometric engineering. Berlin: Springer; 1996, p. 203–22. http://dx.doi.org/10.1007/BFb0014497.

[24] Si H. Tetgen, a delaunay-based quality tetrahedral mesh generator. ACM Trans Math Software 2015;41(2):11. http://dx.doi.org/10.1145/2629697.

[25] Halır R, Flusser J. Numerically stable direct least squares fitting of ellipses. In: Proc. 6th international conference in central europe on computer graphics and visualization. Plzen, Czech Republic. 1998. p. 125–32.

[26] Hammel B, Sullivan-Molina N. Bdhammel/least-squares-ellipse-fitting: v2.0.0. 2020, http://dx.doi.org/10.5281/zenodo.3723294.

[27] Turner D, Anderson I, Mason J, Cox M. An algorithm for fitting an ellipsoid to data. Tech. rep., Teddington, United Kingdom: National Physical Laboratory; 1999.

[28] Kolmogorov A. Sulla determinazione empirica di una legge di distribuzione. Istituto Italiano degli Attuari, Giornale 1933;4:83–91.

[29] Smirnov NV. Estimate of deviation between empirical distribution functions in two independent samples. Bull Moscow Univ 1939;2(2):3–16.

[30] Székely GJ. E-statistics: The energy of statistical samples. Tech. rep., Bowling Green, Ohio: Bowling Green State University, Department of Mathematics and Statistics; 2003.