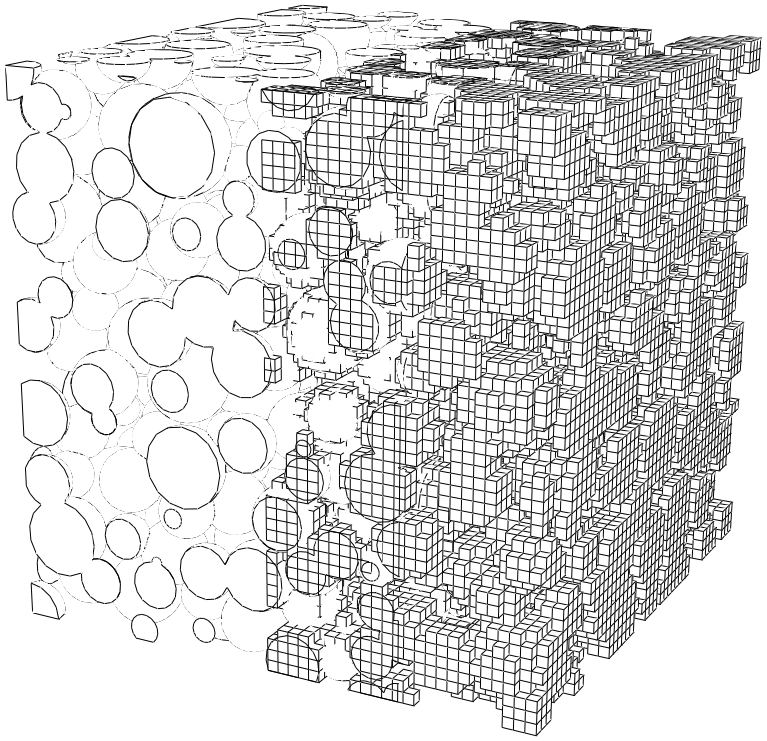
Mote3D

A toolbox for particulate microstructure modelling



User Guide

MOTE3D User Guide Version 1.2

September 10, 2016

Contents

1 Preface

MOTE3D is an open-source software toolbox for the generation of random particulate microstructure models with periodic boundaries. It evolved from a collection of scripts originally written to model the microstructure of partially-sintered, porous alumina. MOTE3D can be used to generate models that represent the microstructure of various inhomogeneous engineering materials such as particle-reinforced composites, modified alloys, ceramics, powders or open-cell foams. These models can be employed, for example, to analyse the relation between microstructure and mechanical, electrical or thermal properties of the aforementioned inhomogeneous materials by numerical simulations.

The MOTE3D toolbox works by randomly positioning spherical particles with user-defined minimum inter-particle distance in a cubical computational domain. The generated microstructure models can be exported in different formats, either as lists of particle centre coordinates and radii or as input scripts for generating solid geometric models or regular hexahedral meshes ('voxel meshes') in the commercial finite-element software Abaqus® [1].

The current version of MOTE3D is version 2.1.

2 Legal Notices

3 Configuring Mote3D

Mote3D is written in GNU Octave [2]. Please download GNU Octave, which is available from http://www.gnu.org/software/octave/download.html and install it onto your system prior to using Mote3D.

After successfully installing GNU Octave, unpack the folder containing the Mote3D files to the current GNU Octave working directory. Execute Mote3D by typing the commands 'run Mote3D_main' or 'Mote3D_main' at the GNU Octave command prompt. Additional information can be obtained using the command 'help Mote3D_main'.

4 Using Mote3D

4.1 Input parameter specification

After launching Mote3D, the user is requested to specify the following input parameters for the generation of a random particulate microstructure model:

- the edge length of the cubical computational domain (default setting is 5.0),
- the total number of spherical particles to be positioned within the cubical computational domain (default setting is 8),
- both the mean and the standard deviation of the particle diameter distribution (default settings are 1.0 and 0.5, respectively),
- the particle overlap factor, which defines the minimum distance between adjacent particles (default setting is 0.9),
- the maximum number of trials Mote3D should undertake to position all spherical particles within the cubical computational domain (default setting is 10000; **note:** large values may require increased computational resources and cause long running times),
- an indication whether the list of particle diameters should be sorted prior to particle positioning to increase the packing efficiency (default setting is 'n'),
- an indication whether the generated microstructure model should be plotted in GNU Octave (default setting is 'n'; **note:** this operation may require increased computational resources for large numbers of particles).

The user can input these parameters directly via a dialog box, as shown in Fig. 1, which proposes the above default settings. An internal routine checks the specified input parameters for completeness and consistency.

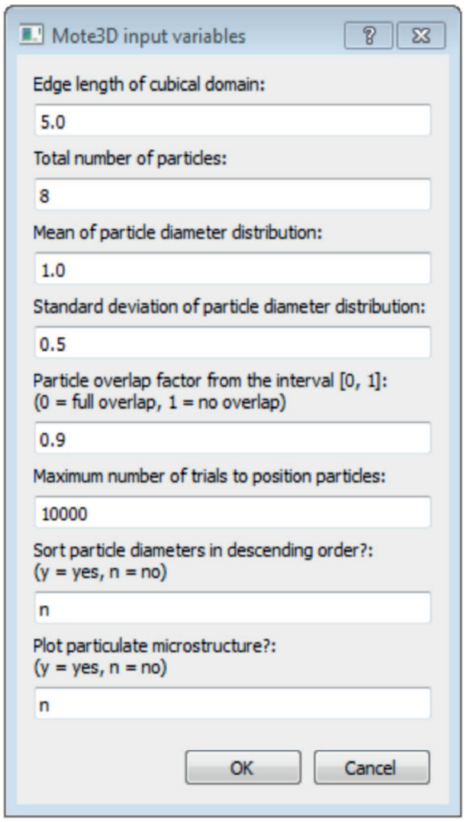


Figure 1: Dialog box for input parameter specification.

After confirming the input parameter selection by pressing the 'OK' button, microstructure model generation starts. Pressing the 'Cancel' button terminates MOTE3D.

4.2 Microstructure model generation

During execution of MOTE3D, progress is shown in a status bar. Three text files are automatically generated, in which the following information is stored:

- the file 'Statistics.txt' contains an overview of the user-specified input parameters, as well as statistical data on the generated particle diameter distribution and inter-particle distances,

- the file 'Positions.txt' contains a list of the centre coordinates of the spherical particles positioned in the cubical computational domain and
- the file 'Radii.txt' contains a list of the corresponding particle radii.

4.3 Microstructure model export

Mote3D provides the functionality to export the generated particulate microstructure model for use with external finite-element software. If this option is selected, the microstructure model geometry is converted and stored in a file format readable by the commercial finite-element software Abaqus® [1] or similar preprocessors. Two output formats are available, which can be selected via a dialog box, as shown in Fig. 2.

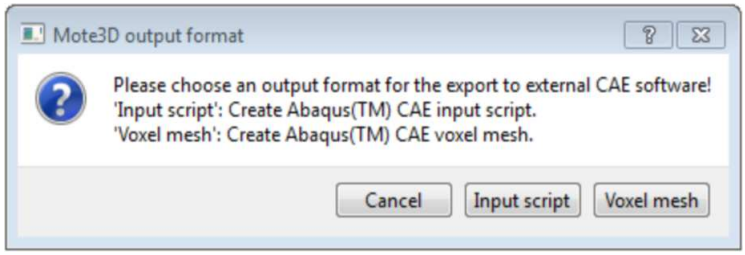


Figure 2: Dialog box for export format specification.

By pressing the 'Input script' button, the particulate microstructure model is converted into a Python-based (http://www.python.org/) [3] input script and saved as 'Abq_input_script.py'. This script can directly be executed in the Abaqus® preprocessor to re-create a solid geometric model of the particulate microstructure, which can then be further processed, meshed and analysed.

By selecting the 'Voxel mesh' button, grid-based meshing is activated and a regular hexahedral mesh ('voxel mesh') of the particulate microstructure is generated. The number of elements on each edge of the cubical computational domain (default setting is 50) and the Abaqus® element type (default setting is 'C3D8' [1]) can be specified via a dialog box, cf. Fig. 3. The resulting element sets representing the spherical particles and the inter-particle space are stored in a Python-based input script entitled 'Abq_voxel_mesh.py'. This script can directly be executed in the Abaqus® preprocessor to re-create the voxel mesh of the discretised particulate microstructure.

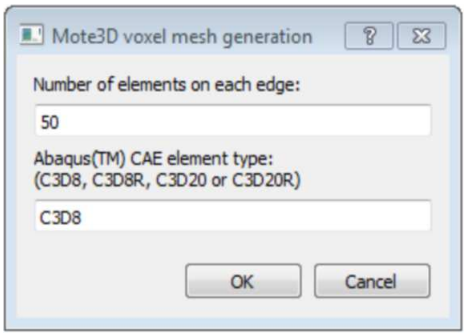


Figure 3: Dialog box for voxel mesh parameter specification.

The grid-based meshing approach is particularly useful if the generated particulate microstructure is too complex for meshing with tetrahedral elements. It assures periodicity of the resulting voxel mesh, rendering it suitable for the application of periodic boundary conditions via nodal constraint equations.

By pressing the 'Cancel' button in the dialog box shown in Fig. 2, no additional output data is written.

5 Modelling approach

MOTE3D generates particulate microstructure models with periodic boundaries by randomly positioning spherical particles in a cubical computational domain. The algorithm underlying particle positioning in MOTE3D is, in brief, based on a modified Random Sequential Addition algorithm [4]: within the predefined computational domain, a random point $\boldsymbol{p}_i$, which represents the centre of a spherical particle with radius r_i, is generated. Then, another spherical particle with radius r_j is added at a second random point $\boldsymbol{p}_j$. If the distance between the two particles is smaller than the product of the particle overlap factor c and the sum of their radii, i.e.

$$\|\boldsymbol{p}_i - \boldsymbol{p}_j\| < c\,(r_i + r_j)\,, \tag{1}$$

the position $\boldsymbol{p}_j$ of the latter particle is rejected and a new point $\boldsymbol{p}_k$ is chosen at random. If the distance is sufficiently large, the centre $\boldsymbol{p}_j$ of the newly added particle is translated along the shortest path towards its nearest neighbour $\boldsymbol{p}_i$ to match the predefined particle overlap factor c. The sequential addition of new particles continues until the predefined number of particles n has been placed inside the computational domain or the maximum number of positioning trials has been reached. The result is a set $\mathcal{P}$ containing the random centre coordinates

$\boldsymbol{p}_v$ and corresponding radii r_v of n spherical particles,

$$\mathcal{P} = \left\{ \boldsymbol{p}_v \in \mathbb{R}^3 \,\middle|\, \|\boldsymbol{p}_v - \boldsymbol{p}_w\| \geq c\,(r_v + r_w)\,;\; v, w = 1, ..., n;\; v \neq w \right\}. \tag{2}$$

With the outlined approach, random configurations of overlapping ($c < 1$) or non-overlapping spherical particles ($c \geq 1$) can be generated. Geometric periodicity of the boundaries is achieved by mapping particles protruding from the cubical computational domain to opposite faces. In addition, a spherical particle is deliberately placed on each vertex of the cubical computational domain in order to facilitate the assignment of boundary conditions for finite-element analyses.

6 References

[1] Dassault Systèmes, 2016, Abaqus® FEA.

[2] J.W. Eaton, D. Bateman, S. Hauberg and R. Wehbring, 2015, GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations.

[3] G. van Rossum and J. de Boer, 1991, Interactively Testing Remote Servers Using the Python Programming Language. CWI Quarterly 4(4), 283–303.

[4] B. Widom, 1966, Random Sequential Addition of Hard Spheres to a Volume. J. Chem. Phys. 44(10), 3888–3894.