MBuilder Manual

Toolkit for 3D Synthetic Microstructure Generation

Joseph C. Tucker

May 25, 2011

Contents

Chapter 1

Introduction

Microstructure Builder (MBuilder) is a strategy to construct simulated 3D polycrystalline materials. The input is typically grain size and shape data as obtained from orthogonal images (optical or SEM) or 3D datasets. The output is a 3D voxel structure that matches the size and shape statistics provided at input. The voxel structures can be used directly as input to Monte Carlo simulations or can be converted to mesh structures for use in FE structural analysis.

The goal of MBuilder is to provide a simple, flexible, and modular tool to construct 3D synthetic microstructures. Simple in that the package requires minimal inputs and requires no coding. Flexible in that it can produce a wide range of microstructures, including: 2D, mono-disperse, multimodal, twinned, coarse, refined, and "fat"-tailed. Modular in that it accepts standard statistical parameters as inputs and outputs formats used for surface meshing, FFT, and peridynamics. Other considerations are portability (MBuilder has been tested on Mac OS, Linux/Unix, and Windows (Cygwin)).

1.1 History of MBuilder

MBuilder started as a collaboration between David Saylor at Carnegie Mellon University (CMU) and Joe Fridy at the Alcoa Technical Center, with help from Tony Rollett, Bassem El-Dasher and Kyung-Jun Kung (all at CMU). It was supported by the Mesoscale Interface Mapping Project (MIMP) under the NSF-supported Materials Research Science and Engineering Center at CMU (mimp.materials.cmu.edu). Various individuals have contributed to MBuilder over the years, including Chris Roberts, Abhijit Brahme, Sukbin Lee and Steve Sintay. Programs that have supported it include the Computational Materials Science Network (CMSN), DARPA under the SIPS program and the Commonwealth of Pennsylvania.

1.2 What MBuilder does

In many ways MBuilder is a turnkey, with a few caveats. If your goal is to produce a synthesized material, i.e., a material based off statistics from EBSD or 3D datasets, MBuilder is the tool for you. If you are making a simple theoretical material with no exotic grain size distribution, MBuilder will certainly be helpful. If you want to create many instantiations of a microstructure, MBuilder can do that.

1.3 What MBuilder doesn't

MBuilder is not a black box. And it requires some thought and expectation for what you are looking to achieve. If you just throw in random inputs, chances are MBuilder won't work. MBuilder is currently serial and with that has size limitations. Much effort has been made to have the MBuilder experience involve little to no learning curve.

1.4 Software Requirements

The user need not be proficient in any programming language. MBuilder is operated by text control files. The basic software requirements include:

- c, c++, and fortran (77,95) compilers
- boost
- cmake
- git (or just download *.tgz)
- paraview (for visualization)

1.5 Terms of use

MBuilder is made freely available for anyone performing non-profit scientific research; those interested in using MBuilder for other purposes should contact us first. If you use MBuilder in your research, please spread the word. Send us modifications that you would like to see incorporated into the package and give us feedback!

Chapter 2

Getting started with MBuilder

The following sections provide the necessary information for new users to obtain and set up MBuilder. This involves downloading a source code archive, unpacking it, and configuring. Developers or those interested in maintaining an up-to-date version of the code should consider checking out a copy from the git repository.

2.1 Download

The MBuilder source code is hosted online at MatForge. From the main MatForge site, users should navigate to the Microstructure Builder home page and choose the appropriate links under the section *Download*, as these links are set up to point to the latest MBuilder release. Archives containing the MBuilder source code are provided in the usual Linux tarball convention (with file extension `.tgz`).

Alternatively, a user may choose to check out a copy from the git repository (it is public read-only). This requires the git version control software to be installed on the target machine, as well as a working internet connection. From the command line type:

```
git init
git pull ssh://code.matforge.org/mbuilder
```

If the checkout is successful, then there is no need to perform the steps for installation described below, and the user should move on to setup (unless using Windows).

Having a local copy of MBuilder makes it simple to keep your code up-to-date. From the root folder, simply type:

```
git pull ssh://code.matforge.org/mbuilder
```

to update a working copy with the latest version of the MBuilder source code.

2.2 Installation

After an appropriate source code archive is obtained as described above, the next step is to install MBuilder. This should be as simple as unpacking the archive. The following paragraphs provide platform specific instructions for a local installation.

Mac OS Local installation for Mac users should simply involve unpacking the archive. After downloading the archive file, move it to the directory where you want MBuilder to reside, making sure that you have read access to the file as well as write access to the directory. Then issue a command to unpack the archive:

```
tar zxf mbuilder_20May11.tgz
```

This will unpack the contents of the archive folder. Next type:

```
ls
```

which should indicate that folders such as `Source/`, `Scripts/`, etc. have been created. If either command fails or the folders are not created, check the `tar` documentation.

Linux/Unix Linux users will follow much of the same procedure as Mac users, so it is advisable to read the previous section on Mac OS installation.

Windows For those who insist on using Windows, it is still possible to use MBuilder. The preferred way to do this is to use the cygwin environment. To use cygwin with MBuilder, it is necessary that appropriate packages, these are:

- `gcc` (the GNU compiler)
- `make` (the GNU make utility)
- automake
- boost
- cmake
- git (if you want)

These are optional packages that must be chosen manually during installation. If cygwin has been installed properly, MBuilder may be installed by following the steps described above for Mas OS installation.

2.3 Setup

Once MBuilder has been installed, from the MBuilder root directory type:

```
mkdir Build
```

```
cd Build
```

This will create a Build directory to separate the executables from the source code and enter this Build directory. Next, type:

```
ccmake ../
```

which launches the cmake utility. To configure your makefile enter:

```
c
```

If there were no errors then a generate option will appear. Note: Depending on the method of boost installation, the Linux user will likely have to specify the path of the boost include directory by entering:

```
t
```

Then scroll down to the boost include directory section and enter the absolute path to boost. Then attempt to configure again. Note: The cygwin user will need to install g95 (fortran 95 compiler) then enter the absolute path to g95 into the cmake utility in the same fashion as the boost path described above. If the configuration was successful, a generation option will appear, type:

```
g
```

If the cmake procedure was successful, a `Makefile` should reside in your Build directory. If this is the case, type:

```
make
```

The screen should display progress feedback as the source compiles and links. If the make procedure succeeded, a Bin directory will be created in your Build directory. Navigate to the Bin directory, which should contain executables such as `mbuilder`, `singlerun`, etc. Then type:

```
./RUN_FIRST
```

which will place all of the control files, temporary folders, and output directories in the correct location.

2.4 Support

MBuilder is not commercial code and there are no guarantees or claims, stated or implied, pertaining to its fitness for any purpose. MBuilder is intended solely for use in non-profit scientific research. In spite of this, the MBuilder team is devoted to producing a quality product that addresses the needs of the scientific community. Please do not hesitate to contact our development team with any questions or suggestions. Contact information can be found on the MBuilder page at MatForge.

Chapter 3

MBuilder tutorials

The following sections present a few short tutorials on running MBuilder. The user will need the use of a basic text editor such as emacs, vi, aquamacs, gedit. etc.

3.1 Quick tutorial

Because every good interface tutorial starts with a "Hello World!" example program, we'll do the equivalent of that with MBuilder – generate a 3D synthetic structure with a mono-disperse grain size distribution.

Within your Bin directory use a text editor to open `singlerun_controlfile.txt`. If using emacs, the command will be:

```
emacs singlerun_controlfile.txt
```

The `singlerun_controlfile.txt` should look something like this:

```
scale 1
dimension 100 100 100
overlap 1.05
periodic false
burn_fraction 0.0
ellipsoid_mean 0.08
ellipsoid_stdev 0.0
min_grainsize 0.0
lognormal_threshold 100
num_twins 0
MC_steps 0
```

A description of the parameters is provided in a later tutorial, but for now, adjust all the parameters to read as above. Next, open `ellipsoid_controlfile.txt` in the text editor of your choice, which should look like this:

```
x_incline_min 0
x_incline_max 360
y_incline_min 0
y_incline_max 360
z_incline_min 0
z_incline_max 360
BoA 1
CoA 1
axes_randomizer 0
mode_2_frac 0.0
A2oA 1.0
B2oA2 1
C2oA2 1
```

Again, a description of these parameters will be given later, just make yours look like this. At this point we are ready to run MBuilder. Execute:

```
./singlerun
```

which will begin MBuilder. During the process, lots of feedback is produced on the screen. When the program is done, the command prompt will return indicating that MBuilder has completed. Since this is the quick tutorial, we'll save the output hierarchy and file types for later and provide you with some visual satisfaction of your achievements. Open Paraview then <File>/<Open>. Your 3D microstructure will be located according to this path: `MBUILDER_ROOT_DIR/Build/Bin/OUTPUT_FILES/new_output_MMDDYY_v#` and is called `grains-renum.vti`. Select it, then <Apply> which should make the outline of your 100x100x100 cube appear. Then the dropdown tab displaying <Outline>, change to <Surface> which should create a white cube. Finally, the dropdown tab displaying <Solid Color>, change to <ID>. You should now be looking at your 3D synthetic microstructure.

3.2 Single Run

The option used in the quick tutorial was single run. The alternative described in the next section is the feedback loop. The single run option executes exactly one iteration of mbuilder. It is governed by two control files: `singlerun_controlfile.txt` and `ellipsoid_controlfile.txt`. Let's start with the former.

The `singlerun_controlfile.txt` contains 11 parameters which are described here:

scale The scale parameter changes the voxel-to-micron (or whatever unit you like) ratio. Setting the scale to one makes the voxel-to-micron ratio = 1. Unless the user is importing fitted ellipsoids from EBSD or 3D data, the scale parameter should always be set to one and other means are used to fit large or small grain sizes into a workable domain for MBuilder.

dimension The dimension parameter dictates and length, width, and height (in voxels) of your volume element. The user must use some common sense in determining their grain size inputs relative to the dimensions of the simulation domain. As mentioned earlier, MBuilder can only generate structures up to a certain size. That maximum size is influenced by many of the other parameters. If the user's goal is to generate the largest structure possible, the quick tutorial settings is their best shot. The largest structure successfully generated with MBuilder was among 300x300x300. However, the if user is seeking a grain size distribution with a large standard deviation or with a small mean grain size (say < 6 voxels (sphere equivalent radius)) then the maximum dimension will shrink considerably.

overlap The overlap parameter is used in seeding the ellipsoids into the simulation domain. The default value is set to 1.05 which allows slight overlap which is beneficial for space filling since the goal is to turn the ellipsoids into grains. The algorithms in MBuilder are optimized for an overlap value of 1.05, but if the user absolutely needs to attempt for more faceted grains, the value could be increased. Note that more faceted grains result from lower standard deviations.

periodic Periodicity is used in MBuilder to simulate bulk characteristics. If periodicity is enforced, ellipsoids and grains are allowed to be continuous over boundaries, this includes all the placement, grid, and growth algorithms. The converse is true for not enforcing periodicity. The result of enforcing periodicity is usually more "island" grains or grains with only one nearest neighbor. Setting periodic to true is usually done for purely theoretical structures that are not modeling a real material. It is recommending that periodicity not be enforced when attempting to synthesize actual materials.

burn_fraction The burn fraction parameter determines what percentage of the voxels with be filled with grain IDs in the final structure. If a number less than one is chosen, then voxels with remain unoccupied. This feature can be used to represent secondary phases, voids, particles, microconstituents, etc. If the user wants a dense structure, set the burn fraction equal to 1.

ellipsoid_mean The ellipsoid mean parameter is the μ (log-normal mean) used to generate a log-normally distributed random variable which is used to define the semi-axes of the ellipsoids. To give the user some expectation, think $\mu < 0.05$ gives thousands of grains and $\mu > 0.05$ gives hundreds of grains (for a 100x100x100 domain). This of course will be affected by the other input parameters.

ellipsoid_stdev The ellipsoid standard deviation parameter is the σ (log-normal standard deviation) used to generate a log-normally distributed random variable which is used to define the semi-axes of the ellipsoids. As a general rule, $2\sigma < \mu$ and $\sigma < 0.04$ if the user hopes to impose any sort of realistic minimum threshold. Log-normal standard deviations that violate these rules are certainly "exotic" grain size distributions and MBuilder will have trouble producing a structure with the desired statistics, if at all.

min_grainsize The minimum grain size sets a threshold on the small grains in the simulation domain. The algorithm will "erode" smaller grains away, replacing their ID with its majority nearest neighbor parent grain. Think of this parameter as the minimum allowable semi-axis of an ellipsoid. So a reasonable value is 0.015 for a log-normal mean of 0.075, e.g.

lognormal_threshold The log-normal threshold sets a truncation on the log-normal distribution. This applies to the upper tail (min_grainsize affects the lower tail). As opposed to "eroding" the large grains away, this algorithm prevents ellipsoids larger than the truncation from being placed in pre-processing. If the user wishes to use no truncation, enter a large value such as 100. A reasonable value is 0.15 for a log-normal mean of 0.075, e.g.

num_twins This determines the number of twins inserted into the structure. Note that if the user attempts to insert too many twins (say >500, again depends), they will break the code. A twin will never be inserted into a grain with a sphere equivalent radius < 5 voxels, so if the user has a mean < 0.05 it is not recommended to attempt to many twin insertions. The twins are inserted randomly. One of the four variants of <111> is chosen at random and the corresponding grain orientation for the twin is calculated by rotating the orientation of the parent grain 60° about that <111> variant.

MC_steps The number of Monte Carlo steps used in the Potts Model. This parameter is used to provide more "realistic" grain boundary curvature. A good starting value = 10. Note that it will be periodic or non-periodic based on the periodicity selection. The user should be warned that if they are trying to embody anisotropy (e.g. rolling), this is quickly lost in the Potts model.

`ellipsoid_controlfile.txt` is identical for both the single run and feedback loop options. Its contents are described here:

x_incline_min The minimum inclination angle of the ellipsoid semi-axis in the x-direction (in degrees). This value is most commonly set to zero unless an off-axis constant is desired, e.g. 45 degrees.

x_incline_max The maximum inclination angle of the ellipsoid semi-axis in the x-direction. If the user is seeking a random axis orientation distribution, the value should be set to 360 paired with x_incline_min set to 0, accordingly with the other two directions.

y_incline_min Same as x in the y-direction.

y_incline_max Same as x in the y-direction.

z_incline_min Same as x in the z-direction.

z_incline_max Same as x in the z-direction.

BoA b over a is the ratio of the semi-axis b over the semi-axis a. In, MBuilder, one log-normally distributed random variable is generated per ellipsoid and all three semi-axes are defined from it. More specifically, the semi-axis a is first defined then b and c are based on the ratios defined in the control file. For instance, if the user wants elongated grains, there are three main variants: oblate (cigars), prolate (pancakes), and scalene. For "cigars", the user will set BoA > 1. For "pancakes", the user will set BoA < 1. For scalene, the user will change the next parameter CoA and BoA so that all three semi-axes are unequal.

CoA c over a is the ratio of the semi-axis c over the semi-axis a. All the same rules apply as for BoA.

mode_2_frac This parameter sets the number fraction of a second mode for a bimodal distribution. If the user wants only one mode then set it to 0.0. If the user e.g. wants 30% second mode then set mode_2_frac = 0.3. The following parameters control the second mode ellipsoids.

axes_randomizer This parameter is used the introduce "play" into the semi-axes. This is a cleanup feature. For instance, if the user is generating an equiaxed structure, then they may want to set this parameter to 10 in order to make the final structure more visually pleasing. This allows the axes to vary ±10%. It is not recommended to set the axis randomizer greater than 20 because this will start to distort the grain size distribution.

A2oA This is the ratio of a semi-axis a of the second mode to the semi-axis a of the first mode. A value of one means that they are equal. If the user wants larger grains in the second mode they should set A2oA > 1.

B2oA2 Same as BoA but for the second mode.

C2oA2 Same as CoA but for the second mode.

3.3 Feedback loop

The feedback loop iterates through "singlerun's" of MBuilder. The control file, `mbuilder_controlfile.txt` contains many of the same parameters as the control file for the single run control file `singlerun_controlfile.txt`. However, the input statistics now are targets for MBuilder to iteratively match. An itemized description of the contents of `mbuilder_controlfile.txt` is below:

scale Same as in `singlerun_controlfile.txt`.

dimension Same as in `singlerun_controlfile.txt`.

overlap Same as in `singlerun_controlfile.txt`.

periodic Same as in `singlerun_controlfile.txt`.

burn_fraction Same as in `singlerun_controlfile.txt`.

mean The mean is the first input that is a target for MBuilder to try and match. It is the mean sphere equivalent radius of the grain size distribution that the user wants. If mean is set to 7 e.g. MBuilder will iteratively attempt to match the grain size distribution to a mean sphere equivalent radius of 7 voxels. It is suggested that mean not be set less than 4 since this will produce grains too close to the resolution of voxels. On the other, end if the user is using a 100x100x100 domain, they should not set the mean greater than around 18 at most, since this size will approach a grain size equal to the simulation domain. These indicated values are extremes, to achieve satisfactory results, it is recommended to stay away from these limits if possible.

stdev Just like the mean parameter, stdev is the target sphere equivalent radius standard deviation target that MBuilder will try to match (in voxels). If the standard deviation is set to zero then the user should expect an equiaxed structure. The standard devation should not be set greater than 4 if the user expects to yield consistent results.

min_grainsize The minimum grain size represents the smallest grain (in sphere equivalent radius) that the final structure will contain. This parameter is an opportunity to set a limit on data collection, e.g. if it is set at 3.5, then MBuilder will remove any grain less than 180 voxels in volume. The user should not set this value too high unless they seek a matrix phase (see later)

max_grainsize The maximum grain size is the largest grain (in sphere equivalent radius) that the final structure will contain. This parameter is used to embody upper tail departure from a log-normal. If a log-normal distribution is predicting too many large grains, a threshold should be enforced to cutoff the top of the distribution. If the user desires no limit on maximum grain size, then this parameter should be less to something large, e.g. 100.

num_twins Same as in `singlerun_controlfile.txt`.

MC_step Same as in `singlerun_controlfile.txt`.

epsilon The tolerance that MBuilder will iteratively match the mean, standard deviation, and (possibly) skewness. If epsilon = 0.05 then it will attempt to match the aforementioned parameters to within 5%. It is not recommended to go below 0.05 because of the inherent randomness involved in MBuilder.

max_iters The maximum number of iterations that MBuilder will try until it gives up and outputs the final iteration.

min_suceess_iters The number of successful iterations MBuilder will continue until it reaches. This is useful if the user needs multiple instantiations for their models.

skew_on A logical true or false which decides whether MBuilder should attempt to match skewness in addition to the mean and standard deviation.

skew The value of skewness in sphere equivalent radius that MBuilder tries to match. Using this feature is effective in getting MBuilder to accurately match the tails of the grain size distribution. The user should be aware that using this function may result in MBuilder taking considerably longer to match all the desired statistics.

3.4 Elongated

Elongated grains are effective for representing rolled grain size distributions. This section displays the two control files needed to produce an elongated structure within the singlerun framework. Descriptions of the important control file alterations are provided along with suggestions for further changes if a slightly different result is desired.

To produce a generic elongated grain size distribution to embody things like rolling anisotropy, the two control file should look like this:

`singlerun_controlfile.txt`

```
scale 1
dimension 100 100 100
overlap 1.05
periodic true
burn_fraction 0.0
ellipsoid_mean 0.08
ellipsoid_stdev 0.0
min_grainsize 0.025
lognormal_threshold 100
num_twins 0
MC_steps 0
```

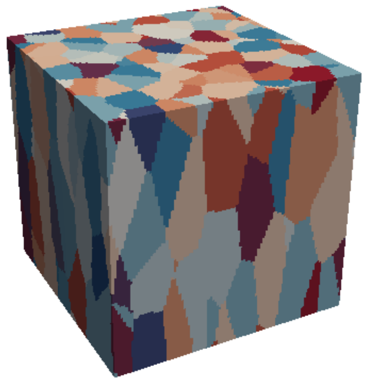
Notable here is the periodic is set true since it is assumed to be a purely theoretical structure. The mean is set to 0.08 which should end up with ~130 grains with the given set of parameters. Increase the mean for less grains, visa versa. An equiaxed grain size distribution is assumed in the non-elongated directions, so the standard deviation = 0.

`ellipsoid_controlfile.txt`

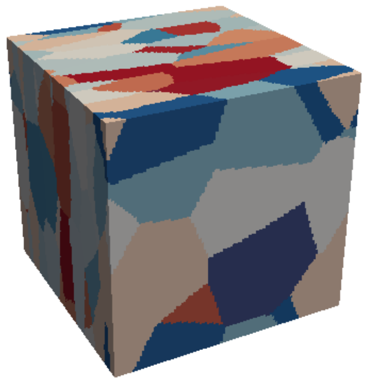
```
x_incline_min 0
x_incline_max 0
y_incline_min 0
```

```
y_incline_max 0
z_incline_min 0
z_incline_max 0
BoA 3
CoA 3
axes_randomizer 10
mode_2_frac 0.0
A2oA 1.0
B2oA2 1
C2oA2 1
```

All of the incline parameters are set to 0, making the axis orientation constant and on-axis. BoA = 3 which makes the b semi-axis three times greater than the other semi-axes which will produce the "cigar" ellipsoids and elongated grains. If the user wants more elongation, they should increase BoA. A picture of the elongated structure is shown here:



To produce "pancake" grains the same two control files are used, except CoA is set to 3. The structure should have ~50 grains and is shown here:



3.5 2D

Creating a 2-dimensional structure in MBuilder is as simple as setting one of the dimensions equal to 1. The pair of control files is shown below with explanations of the parameter selections where appropriate.

`singlerun_controlfile.txt`

```
scale 1
dimension 100 100 1
overlap 1.05
periodic false
burn_fraction 0.0
ellipsoid_mean 0.13
ellipsoid_stdev 0.0
min_grainsize 0.025
lognormal_threshold 100
num_twins 0
MC_steps 0
```

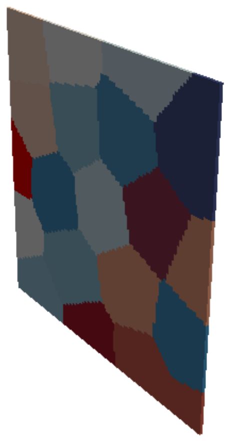
The mean is made relatively large here to better illustrate the 2D behavior.

`ellipsoid_controlfile.txt`

```
x_incline_min 0
x_incline_max 360
y_incline_min 0
y_incline_max 360
z_incline_min 0
z_incline_max 360
BoA 1
CoA 1
axes_randomizer 0
mode_2_frac 0.0
A2oA 1.0
B2oA2 1
C2oA2 1
```

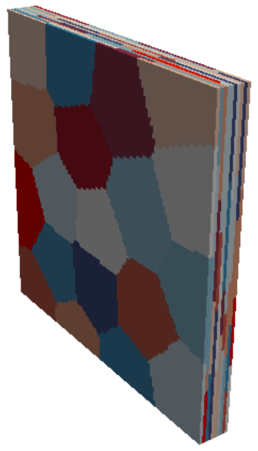
The axes randomizer is set to zero in order to get the most equiaxed grains as possible.

The result should have ~20 grains and is displayed here:



3.6 Layered

To produce a layered structure such as that seen below, a run of the 2D tutorial is done 10 times then the user must perform some surgery on the vti file. If the users are unfamiliar with the vti format, they should consult vtk/vti documentation.



3.7 Bimodal

The procedure to produce a bimodal distribution involves only changing parameters in `ellipsoid_controlfile.txt`. However, both control files used to generate the structure are shown below (within the singlerun framework).

`singlerun_controlfile.txt`

```
scale 1
dimension 100 100 100
overlap 1.05
```

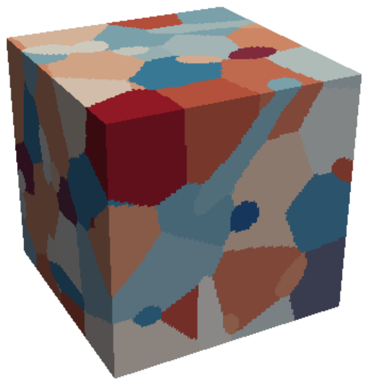
```
periodic false
burn_fraction 1.0
ellipsoid_mean 0.12
ellipsoid_stdev 0.0
min_grainsize 0.01
lognormal_threshold 100
num_twins 0
MC_steps 0
```

The minimum is made relatively small because the second mode is going to represent needle-like grains, making sure they will not be thresholded out. Periodicity is false so that the needle grains are not allowed to grow across boundaries, resulting in many island grains.

`ellipsoid_controlfile.txt`

```
x_incline_min 0
x_incline_max 360
y_incline_min 0
y_incline_max 360
z_incline_min 0
z_incline_max 360
BoA 1
CoA 1
axes_randomizer 10
mode_2_frac 0.7
A2oA 0.33333
B2oA2 18
C2oA2 1
```

Here, the second mode is set to a number fraction of 0.7 meaning that the structure will have 70% of the second mode. The first mode is equiaxed. The second mode has two semi-axes ~1/3 the length of the first mode semi-axes and the third semi-axis 5 times the length of the mode one semi-axes, giving a needle grain result.



3.8 Particles

Particles can be represented in MBuilder in two different ways. First, through using a bimodal distribution and making the second mode much smaller than the first and having that second mode represent the particles. An example is shown here:

`singlerun_controlfile.txt`

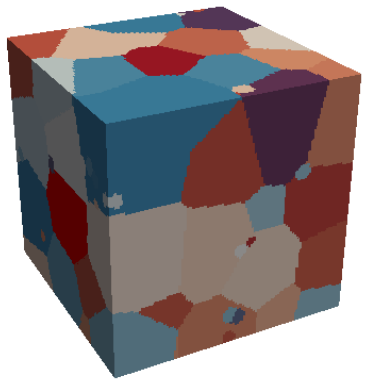
```
scale 1
dimension 100 100 100
overlap 1.05
periodic false
burn_fraction 1.0
ellipsoid_mean 0.14
ellipsoid_stdev 0.00
min_grainsize 0.0
lognormal_threshold 100
num_twins 0
MC_steps 0
```

The mean is made large so as to distinguish the smaller mode two. The minimum is set to zero since the second mode a approaching voxel resolution.

`ellipsoid_controlfile.txt`

```
x_incline_min 0
x_incline_max 360
y_incline_min 0
y_incline_max 360
z_incline_min 0
z_incline_max 360
BoA 1
CoA 1
axes_randomizer 0
mode_2_frac 0.5
A2oA 0.2
B2oA2 1
C2oA2 1
```

The number fraction is set to 50% since it is the volume of the second mode, it will be much smaller than the first.



The second way to represent particles is to use the burn fraction option in the control file to partially assign voxels with grain IDs. An example of this is:

`singlerun_controlfile.txt`

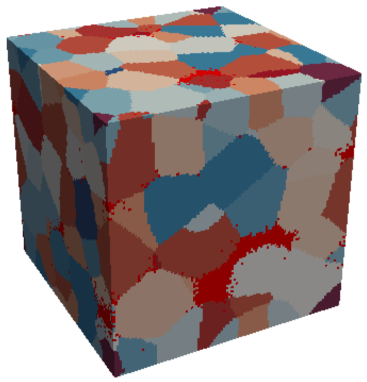
```
scale 1
dimension 100 100 100
overlap 1.05
periodic true
burn_fraction 0.95
ellipsoid_mean 0.12
ellipsoid_stdev 0.0
min_grainsize 0.0
lognormal_threshold 100
num_twins 0
MC_steps 0
```

The burn fraction is set to 0.95 so some of the voxel remain unassigned to represent particles.

`ellipsoid_controlfile.txt`

```
x_incline_min 0
x_incline_max 360
y_incline_min 0
y_incline_max 360
z_incline_min 0
z_incline_max 360
BoA 1
CoA 1
axes_randomizer 10
mode_2_frac 0.0
A2oA 1
B2oA2 1
```

```
C2oA2 1
```



3.9 Twins

Inserting twins is as simple as changing the num_twins parameter from zero to a value greater than zero. Twins are inserted randomly. Note that if the user attempts to insert an inordinate number of twins, MBuilder will not work.

singlerun_controlfile.txt

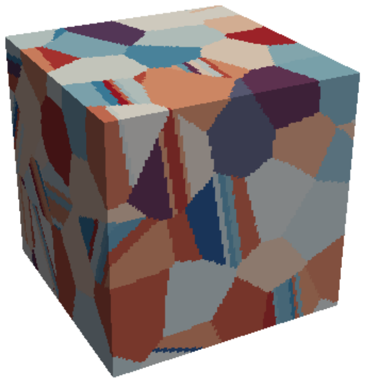
```
scale 1
dimension 100 100 100
overlap 1.05
periodic false
burn_fraction 1.0
ellipsoid_mean 0.12
ellipsoid_stdev 0.0
min_grainsize 0.025
lognormal_threshold 100
num_twins 100
MC_steps 0
```

The number is twins is set to 100. Periodic is false since it is assumed twins are inserted to represent a real material.

ellipsoid_controlfile.txt

```
x_incline_min 0
x_incline_max 360
y_incline_min 0
y_incline_max 360
z_incline_min 0
z_incline_max 360
BoA 1
```

```
CoA 1
axes_randomizer 10
mode_2_frac 0.0
A2oA 1
B2oA2 1
C2oA2 1
```



3.10 Skewness

Skewness is operated within the feedback loop framework. Simply set `skew_on = true` in `mbuilder_controlfile.txt` and provide a target skew value.

Example coming soon.

3.11 Truncation

As mentioned in the max_grainsize and lognormal_threshold parameter descriptions, truncating the upper tail of the log-normal distribution prevents too-large ellipsoids from being generated and turning into too-large grains. Here is a table to give perspective on this truncation value (for mean = 4.69, stdev = 2.37):

Truncation ln(...)	**% Lognormal Distribution**	**1 in ...**	**Largest Ellipsoid Sphere Equivalent Radius**
1.20	5.50×10^{-9}	1.82×10^{8}	20
1.18	2.79×10^{-7}	3.59×10^{6}	18
1.16	9.17×10^{-6}	109,107	16
1.14	1.92×10^{-4}	5203	14
1.12	2.52×10^{-3}	396	12
1.10	2.04×10^{-2}	49	10

The values in the first column belong to max_grainsize which is in `mbuilder_controlfile.txt`. To scale them into `singlerun_controlfile.txt` divide by 100.

3.12 Coarsened

Using the Potts Model to coarsen the microstructure is an effective ways to improve the grain boundary curvature. An effective starting point is 10 MC_steps which is what is demonstrated below with a before and after and accompanying control file (for the after instantiation).

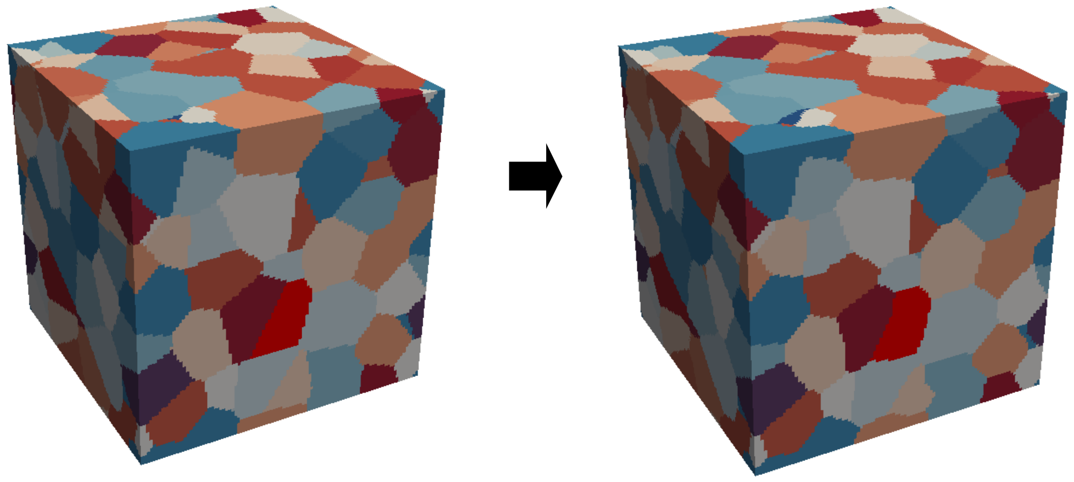
`singlerun_controlfile.txt`

```
scale 1
dimension 100 100 100
overlap 1.05
periodic true
burn_fraction 1.0
ellipsoid_mean 0.14
ellipsoid_stdev 0.0
min_grainsize 0.025
```

```
lognormal_threshold 100
num_twins 0
MC_steps 50
```

```
ellipsoid_controlfile.txt
```

```
x_incline_min 0
x_incline_max 360
y_incline_min 0
y_incline_max 360
z_incline_min 0
z_incline_max 360
BoA 1
CoA 1
axes_randomizer 0
mode_2_frac 0.0
A2oA 1
B2oA2 1
C2oA2 1
```



3.13 Matrix-Phase

Representing a matrix phase with MBuilder could find uses in trying to synthesize concrete with some size distribution of aggregate in a matrix phase. There are two different ways to create a matrix phase with MBuilder. The first way is appropriate for an aggregate with curvature. The second is for faceted aggregate.

The first method is similar to the process of producing particles by partially burning the voxel except, here an even larger fraction of the voxel will remain unassigned and will ultimately represent the matrix phase.

```
scale 1
dimension 100 100 100
overlap 1.05
periodic false
burn_fraction 0.7
ellipsoid_mean 0.10
ellipsoid_stdev 0.0
min_grainsize 0.0
lognormal_threshold 100
num_twins 0
MC_steps 0
```

```
x_incline_min 0
x_incline_max 360
y_incline_min 0
y_incline_max 360
z_incline_min 0
z_incline_max 360
BoA 1
CoA 1
axes_randomizer 10
mode_2_frac 0.0
A2oA 1
B2oA2 1
C2oA2 1
```

The second procedure involves setting a very high minimum threshold and turning a significant fraction of the grains into the matrix phase.

```
scale 1
dimension 100 100 100
overlap 1.05
periodic false
burn_fraction 1.0
ellipsoid_mean 0.10
ellipsoid_stdev 0.0
min_grainsize 0.14
lognormal_threshold 100
num_twins 0
MC_steps 0
```

```
x_incline_min 0
```

```
x_incline_max 360
y_incline_min 0
y_incline_max 360
z_incline_min 0
z_incline_max 360
BoA 1
CoA 1
axes_randomizer 0
mode_2_frac 0.0
A2oA 1
B2oA2 1
C2oA2 1
```

Chapter 4

File input and output

File input coming soon, e.g. reading in ellipses from orthogonal EBSD scans and reading in ellipsoids from 3D datasets.

After MBuilder has run to completion its output files are stored in the `OUTPUT_FILES` directory under a subdirectory denoted my the current date. What resides in this subdirectory will differ depending on whether it is from a run of the feedback loop or a single run.

If a single run was performed, the contents of the output subdirectory will be, going (more or less) in order of their incarnation:

`test.input` This file is the output of the ellipsoid seeding algorithm. Its contents are a multi-line ellipsoid description with a repeat looking like this:

```
0.997052  0.99861   0.964535
0.111992  0.118579  0.113325
-0.071101 0.393157  0.916718
-0.324986 -0.878027 0.351357
0.943042  -0.272939 0.190199
0.681863  0.172474  0.0946573
```

Line 1 defines the ellipsoid centroid within the simulation domain. The simulation domain is scaled to a maximum dimension of 1, so 100x100x100 becomes 1x1x1.

Line 2 are the three semi-axes.

Lines 3-5 are the unit vectors of the three Euler angles defining the axis tilt.

Line 6 is the crystallographic orientation. MBuilder assigns random orientations but there are existing codes which overlay texture.

`elliptical.cells` This file is the result of the ellipsoid optimization procedure. It will be a significant subset of `test.input`, and has the same format.

`Grains-Bulk-Edge-grains.ph.txt` This contains a number of statistics from the voxel image and looks something like this:

Grain	Bulk(0)/Edge(1)	Spin	Old-Spin	Vol	NN	Edge-Corr-Vol	X-edge	Y-edge	Z-edge
1	1	1	55	7847	8	62776	2	2	2
2	1	2	40	7023	7	28092	2	2	1
3	1	3	108	12686	10	50744	2	2	1
4	1	4	31	5718	5	22872	2	2	1
5	1	5	24	983	2	7864	2	2	2
6	1	6	101	11422	11	45688	2	2	1
7	0	7	71	12702	11	50808	0	0	0
8	1	8	45	3159	5	12636	2	1	2
9	0	9	63	5657	9	11314	0	0	0
10	1	10	92	5437	8	21748	2	1	2

Grain is the final grain ID.

Bulk(0)/Edge(1) takes a value of zero if the grain is entirely in the bulk and one if it touches at least one boundary.

Spin and **Old Spin** are previous renumberings of the structure.

Vol is the volume in voxels.

NN is the number of nearest neighbors.

Edge-Corr-Vol is an adjusted volume based on how much of the grain is in contact with a boundary.

X,Y,Z-edge take value of 0,1,2 if the grain is in contact with zero, one, or two of the x,y,z boundaries respectively.

`Summary-grains.txt` is as its name suggests, a summary. It looks like this:

```
110     Number of grains
          22     Number of bulk grains
   13.637769        Average Sphere Equiv. Radius
   14734.909        Average Area
   10695.863        Average Volume
   13.818182        Average no. of NN
   2.5935593        Average Volume/Size ratio
```

```
  0.19017474          Ratio of VS:<R>
```

`mRad.txt` This is a file that contains a single column of the radius over the mean radius (R/<R>)

`grains-renum.ph` This is the final image of the synthetic microstructure. It is written in rho major (x varies faster than y varies faster than z). This is the final structure that is read into the marching cubes surface meshing.

`grains-renum.vti` This is the visualization file of the final structure (written in rho major with some headers)

`singlerun_stats.txt` Which contains some statistics about the MBuilder run, it looks like, should be self-explanatory:

```
Mean: 12.7313
Lognormal Ellipsoid Mean: 0.12
Standard Deviation: 1.59333
Lognormal Ellipsoid Standard Deviation: 0
Skewness: -1.12729
Lognormal Minimum: 0.025
Lognormal Maximum: 100
```

If a feedback loop run was performed, the output files are mostly the same as the singlerun, but the subdirectory structure will be different. In the subdirectory denoted by the date will be subdirectories denoted by an iteration, e.g. `iter_4`. If `min_success_iter` > 1 then more than one subdirectory will exist on this level. Also located in this level are `mbuilder_iterations_input.txt` and `mbuilder_iterations_output.txt` examples of which are shown below, respectively:

```
Iteration  ellipsoid_mean  ellipsoid_stdev  min_grainsize  lognormal_threshold
    1        0.083500        0.016200           18             0.104000
    2        0.084466        0.018041           18             0.104000
    3        0.084466        0.012295           18             0.104000
    4        0.084466        0.014488           18             0.104000
```

```
Iteration     Mean    Mean Residual  Standard Deviation  Standard Deviation Residual
    1      8.446622    0.096622         1.804101                0.184101
    2      8.467809    0.117808         1.890905                0.270905
    3      8.468175    0.118175         1.452850                0.167150
    4      8.663630    0.313630         1.572661                0.047339
```

Within the iteration denoted subdirectory should be the same files found in the singlerun option, with one exception. Instead of `singlerun_stats.txt`, there will be a file called `residuals.txt.`

Chapter 5

Toolkit

Many of the standalone codes described in this section are bundled into the MBuilder distribution described up to this point, and in many ways make MBuilder what it is. This would be classified as the advanced section for those users who wish to reorder the steps implicitly called in MBuilder or to call some of the functions one at a time in attempt to obtain a very specific result.

Much more coming…