PyCMG

Release 1.0

PyCMG developers

Dec 29, 2021

CONTENTS

PyCMG is an open source software package for generating virtual concrete mesostructures. It is implemented in pure Python.

CHAPTER

ONE

INSTALLATION

1.1 Installation

- Install a python IDE or an editor (PyCharm, Spyder etc. https://realpython.com/python-ides-code-editors-guide/)
- Download the latest version of PyCMG from Github: https://github.com/jtimo/pycmg
- Install missing third-party packages required for PyCMG using the IDE specific package manager or equivalent tool.
- That is it ! You are good to go.
- Choose our pre-built concrete specification in the examples folder to generate your first mesostructure.
- Advanced users can use PyCMG to generate custom mesostructures (see tutorial).

If you have questions, drop us an email at jithender.timothy[at]tum.de.

1.2 License

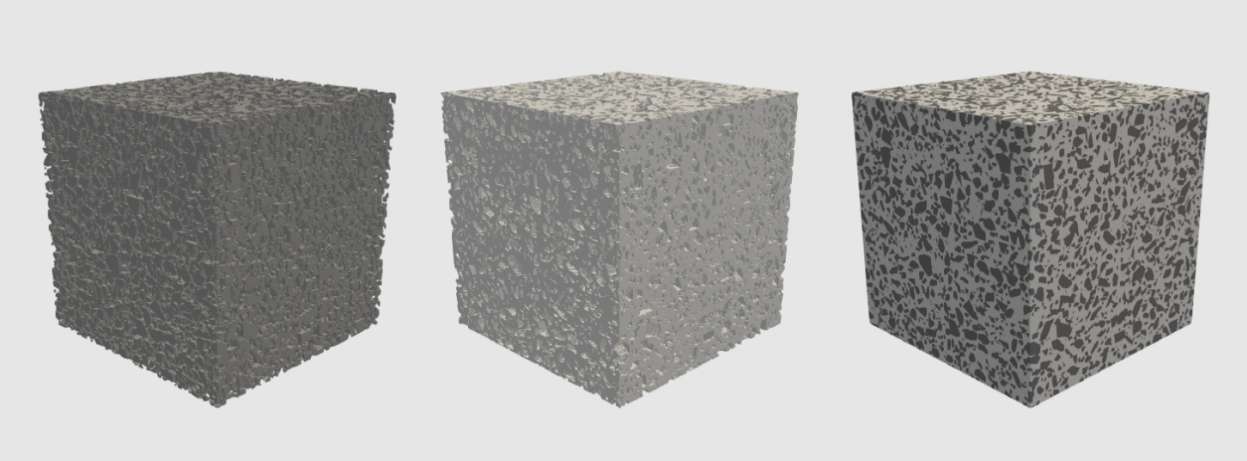
CHAPTER

TWO

TUTORIAL

2.1 Generating an AB8 Mesostructure using an aggregate size distribution

In order to generate a concrete mesostructure with a certain aggregate size distribution, we first need to specify the total aggregate volume fraction in the concrete material and then the aggregate size distribution. First create a csv file containing the aggregate size distribution.



The csv file should contain atleast two columns. The first column lists the aggregate sizes while the second column the corresponding volume-fraction. This volume fraction is relative to the total volume fraction of the aggregates in the material. Here is the text file with a distribution for AB8 standard concrete.

Table 1: An example csv file for AB8 concrete

a	vf_max
2	0.134042553
3	0.134042553
4	0.075886525
4	0.075886525
5	0.075886525
5	0.075886525
6	0.075886525
6	0.075886525
7	0.046099291
7	0.046099291
7	0.046099291
8	0.045138889
8	0.045138889
8	0.048020095

please do not forget to add the correct header.

Having specified the aggregate size distribution and given the total aggregate volume fraction, we have to option to provide the 'average shape' of aggregate in terms of ellipsoidal radii. The first value must be 1 and the second and third values are scaling factors that will scale the radii in the other dimensions with respect to the first dimension.

we can generate the required mesostructure as follows:

```
#IMPORT THE NECESSARY LIBRARIES
from generate_mesostructure import Mesostructure
from configuration import Configuration
from visualization import export_data, visualize_sections

#SPECIFY THE TOTAL AGGREGATE VOLUME FRACTION AND AGGREGATE SIZE DISTRIBUTION
my_configuration = Configuration(vf_max_assembly=0.3, average_shape = [1, 0.5, 0.5])
my_configuration.load_inclusions(conf_csv='AB8_CMG.csv')
```

Now, we can specify the size of the REV-Mesostructure and add our configuration

```
my_mesostructure = Mesostructure(mesostructure_size=[200, 200, 200])
my_mesostructure.add_configuration(my_configuration)
```

Note, we can add multiple configurations (multiple standards) with various sizes. Here we just have one configuration.

Now we can generate the required concrete mesostructure, visualize and export the data using the following three lines.

```
#GENERATE
my_synthetic_microstructure =my_mesostructure.assemble_sra()
# VISUALIZE
visualize_sections(my_synthetic_microstructure, 2)
#EXPORT
export_data(my_synthetic_microstructure, 'vtk', 'mesostructure.vti')
```

That is it. We have generated our first mesostructure.

2.2 Generating a fully customized AB8 Mesostructure

Concrete Mesostructure in voxel format is generated by assembling inclusions of given size onto the main assembly. Matrix (Cement) in the mesostructure is represented by voxel number 0 and inclusions (Aggregate) by 1, coating (ITZ) is represented by 2. The process of mesostructure generation involves first generating inclusion of given input details and then assembling onto the main mesostructure. To specify the concrete inclusion details such as size, volume fraction etc. a Configuration object has to be first created. This object is then loaded onto the Mesostructure object in which when the assembly method is called, the inclusions are generated by importing details from the loaded Configuration and assembled onto the mesostructure. Inputs are provided for a family of inclusions. Based on the maximum volume fraction specified for the family, required number inclusions in that family is estimated and assembled.

To generate the concrete mesostructure using CMG, following are the instructions. Import the required libraries as given in the following lines:

```
from generate_mesostructure import Mesostructure
from configuration import Configuration
from visualization import export_data, visualize_sections
import numpy as np
```

Firstly, Configuration object has to be created with maximum volume fraction for the current configuration as the input

```
my_configuration = Configuration(vf_max_assembly=0.4)
```

Followed this, inclusions input details in csv file have to be loaded to the configuration as below

```
my_configuration.load_inclusions(conf_csv=AB8_CMG_full.csv')
```

Following input details must be/can be specified for the inclusion family to load to the Configuration object.

1. a: diameter of the inclusion along axis-1, default 10
2. b: diameter of the inclusion along axis-2, default b=a*average_shape[1]
3. c: diameter of the inclusion along axis-3, default c=a*average_shape[2]
4. vf_max: Maximum volume fraction for the current inclusion family.
5. coat: True/False for coating provision on the inclusion, default False
6. t_coat: thickness of coating on the inclusion, default 0
7. space: True/False for providing spacing between inclusions in the assembly, default False
8. t_space : thickness of the spacing on the inclusion (similar to coating), default 0

Note: vf_max is relative to the maximum volume fraction of the inclusion given for the entire assembly. For example, for the mesostructure assembly, if the maximum volume fraction is 0.5, then total maximum volume fraction of all inclusion families (all rows in input csv file) should be equal to 1 since it is relative to the maximum volume fraction of the assembly.

An example csv input file template is given below.

Table 2: An example csv file for AB8 concrete

a	b	c	vf_max	n_cuts	coat	t_coat	space	t_space	concave	width	depth
2	2	3	0.13404253	153	FALSE	2	FALSE	2	FALSE	0	0.1
3	5	3	0.13404253	153	FALSE	2	FALSE	2	FALSE	0	0.1
4	3	4	0.07588652	125	FALSE	2	FALSE	2	FALSE	0	0.1
4	6	4	0.07588652	225	FALSE	2	FALSE	2	FALSE	0	0.1
5	3	3	0.07588652	325	FALSE	2	FALSE	0	FALSE	3	0.1
5	3	3	0.07588652	325	FALSE	2	FALSE	0	FALSE	3	0.1
6	4	3	0.07588652	325	FALSE	2	FALSE	0	FALSE	3	0.1
6	4	4	0.07588652	125	FALSE	2	FALSE	0	FALSE	3	0.1
7	12	5	0.04609929	191	FALSE	2	FALSE	0	FALSE	3	0.1
7	12	8	0.04609929	191	FALSE	2	FALSE	0	FALSE	3	0.2
7	12	8	0.04609929	191	FALSE	2	FALSE	0	FALSE	5	0.2
8	4	8	0.04513889	189	FALSE	2	FALSE	2	FALSE	5	0.2
8	5	8	0.04513889	289	FALSE	0	FALSE	2	FALSE	5	0.2
8	5	8	0.04802005	295	FALSE	0	FALSE	2	FALSE	5	0.2

Column header name should be same as the input names given above, but position of the columns can be changed and non-mandatory input columns can be removed.

Followed with this, Mesostructure object is created with mesostructure size and configuration as input. Default mesostructure size is 100,100,100. Default resolution is 1,1,1. The resolution relates the voxel dimensions and the phyiscal dimensions. For e.g. if the resolution is 0.5, 0.5, 0.5, and the mesostructure is 50, 50, 50 mm then each mm3 is equal to 2x2x2 voxels. So, it is not mandatory to load this value. Also configuration can be separately added by using 'add_configuration' method

```
my_mesostructure = Mesostructure(mesostructure_size=[200,200,200]),
my_configuration, resolution = [0.5, 0.5, 0.5]
```

or

```
my_mesostructure = Mesostructure(mesostructure_size=[200,200,200])
resolution = [0.5, 0.5, 0.5]
my_mesostructure.add_configuration(my_configuration)
```

Finally, assembly of the inclusions as per the details given in the configuration is done using CMG Semi-Random Assembly (SRA) algorithm. This algorithm assembles the inclusions at random locations, but with CMG optimization. SRA algorithm is called as follows:

```
asmbly.assemble_sra()
```

The algorithm tries to assemble the inclusions till maximum volume fraction is achieved. Since the assembly is at random points, it becomes difficult to fit the inclusions into the mesostructure as the packing density increases. So, assembly time increases with the given maximum volume fraction. To terminate the process after some time, a parameter attempt_max is given which gives a limit on how many failed attempts to assemble the inclusion can be made. This parameter can also be given as input to the method if required. Also, there is a threshold value beyond which the algorithm shifts from completely random assembly to semi-random to accommodate more inclusions. This threshold value can also be given as input. More the threshold value, more random the assembly is (more slow!).

```
my_synthetic_microstructure = asmbly.assemble_SRA(attempt_max=500000, threshold=50)
```

here my_synthetic_microstructure is the voxel representation of the mesostructure (3D array, int). Mesostructure can be exported to different types of output files for either visualization or some other analysis by using following code

```
exportData(data= my_synthetic_microstructure, export_type='vtk', fileName='mesostructure.
↪vti')
```

or

```
exportData(data= my_synthetic_microstructure,
     export_type='csv', fileName='mesostructure.csv')
```

vtk, csv, npy, npz, txt export types are allowed. Please note that the export type and extension in the file name/location should be consistent (eg. vtk-vti, csv-csv, npy-npy etc.) One can also visualize sections of the mesostructure using following code with the mesostructure and number of input slices in each direction as arguments. Default value for argument slices is 3.

```
visualizeSections(my_synthetic_microstructure, slices=5)
```

Here is the complete code to generate a concrete mesostructure for AB8 standard:

```
from generate_mesostructure import Mesostructure
from configuration import Configuration
from visualization import export_data, visualize_sections

my_configuration = Configuration(vf_max_assembly=0.3)
my_configuration.load_inclusions(conf_csv='AB8_CMG.csv')
my_configuration.sort_inclusions()
my_mesostructure = Mesostructure(mesostructure_size=[200, 200, 200])
resolution = [0.5, 0.5, 0.5]
my_mesostructure.add_configuration(my_configuration)
my_synthetic_microstructure =my_mesostructure.assemble_sra()
visualize_sections(my_synthetic_microstructure, 2)
export_data(my_synthetic_microstructure, 'vtk', 'mesostructure.vti')
```

CHAPTER

THREE

DOCUMENTATION

3.1 Configuring the Mesostructure

class configuration.**Configuration**(*vf_max_assembly=0.3*, *average_shape=False*)

Bases: object

Provides methods for configuring the geometrical and topological parameters of the mesostructure.

load_inclusions(*conf_csv=None*)

Parameters **conf_csv** – string (with .csv extension), Location of the csv file which has aggregate parameters.

Table 1: An example csv file for concrete

a	b	c	vf_max	n_cuts	coat	t_coat	space	t_space	con-cave	width	depth
2	2	3	0.13404255	53	FALSE	2	FALSE	2	FALSE	0	0.1
3	5	3	0.13404255	53	FALSE	2	FALSE	2	FALSE	0	0.1
4	3	4	0.07588625	25	FALSE	2	FALSE	2	FALSE	0	0.1
4	6	4	0.07588625	25	FALSE	2	FALSE	2	FALSE	0	0.1
5	3	3	0.07588625	25	FALSE	2	FALSE	0	FALSE	3	0.1
5	3	3	0.07588625	25	FALSE	2	FALSE	0	FALSE	3	0.1
6	4	3	0.07588625	25	FALSE	2	FALSE	0	FALSE	3	0.1
6	4	4	0.07588625	25	FALSE	2	FALSE	0	FALSE	3	0.1
7	12	5	0.04609929	91	FALSE	2	FALSE	0	FALSE	3	0.1
7	12	8	0.04609929	91	FALSE	2	FALSE	0	FALSE	3	0.2
7	12	8	0.04609929	91	FALSE	2	FALSE	0	FALSE	5	0.2
8	4	8	0.04513889	89	FALSE	2	FALSE	2	FALSE	5	0.2
8	5	8	0.04513889	89	FALSE	0	FALSE	2	FALSE	5	0.2
8	5	8	0.04802009	95	FALSE	0	FALSE	2	FALSE	5	0.2

Note: The header of the parameters in the csv file should be as follows:

- a: diameter of the inclusion along direction-1 in actual units (mm/cm etc.).
- b: diameter of the inclusion along direction-2 in actual units (mm/cm etc.).
- c: diameter of the inclusion along direction-3 in actual units (mm/cm etc.).
- n_cuts: Number of faces/cuts for the polyhedron shaped aggregates.
- concave: Yes/No. Provision for concave depressions on the aggregates.

- n_concave: Number of concave depressions on each aggregate surface in actual units (mm/cm etc.).
- depth: A parameter which determines depth of the concave depression on the aggregate surface. Values should be between 0 to 1 (0 lowest, 1 highest).
- width: A parameter which determines width of the concave depression on the aggregate surface.
- coat: Yes/No. Provision for the coating on the aggregate surface.
- t_coat: Thickness of the coating on the aggregate surface in actual units (mm/cm etc.).
- space: Yes/No. Provision for the spacing on the aggregate surface. Spacing is like a coat on top of the aggregate which provides minimum gap between each inclusion in the mesostructure.
- t_space: Thickness of the spacing on the aggregate surface in actual units (mm/cm etc.).
- vf_max: Maximum volume fraction of each sized aggregates (value between 0 to 1).

Parameters

- **conf_header** – If not csv, then a header with parameter names as given above and corresponding array of values have to be loaded.
- **conf_values** – In not csv, then values corresponding to the header have to be loaded.
- **conf_dict** – If not csv and conf_header & conf_values, inputs can be also given through a dictionary.

class configuration.**InclusionFamily**(*average_shape=False*, *Id=None*, *inclusion_list=[]*, *vf_max=1*, *a=10*, *b=0*, *c=0*, *n_cuts=10*, *concave=False*, *n_concave=0*, *depth=0*, *width=0*, *coat=False*, *t_coat=0*, *space=False*, *t_space=0*, *x=0*, *y=0*, *z=0*, *kwargs=None*)

Bases: `object`

This class is for family of inclusions

Parameters

- **average_shape** (*array of size (3),float*) – Aspect ration of the inclusion along all three axes (value between 0-1).
- **Id** (*int, default: None*) – Id of the inclusion family.
- **inclusion_list** (*array/list (1D)*) – Gives list of inclusions belonging to the current family.
- **vf_max** (*float, value between 0 to 1, default:1.0*) – Maximum volume fraction of the inclusion family.
- **a** (*float, default:10*) – Diameter of the inclusion along direction-1 in actual units (mm/cm).
- **b** (*float, default:b=a* average_shape[1]*) – Diameter of the inclusion along direction-2 in actual units (mm/cm).
- **c** (*float, default:c=a* average_shape[2]*) – Diameter of the inclusion along direction-3 in actual units (mm/cm).
- **n_cuts** (*int, default:10*) – Number of faces of the irregular polyhedron.
- **concave** (*bool, True/False, default:False*) – Boolean for concave depression on inclusion surface.
- **n_concave** (*int, default:0*) – Number of concave depressions on the inclusion surface.

- **depth** (*float, value between 0 to 1, default:0*) – Parameter which determines depth of the concave depression from the inclusion surface.
- **width** (*float, default:0*) – Parameter which determines width of the concave depression on the inclusion surface.
- **coat** (*bool, True/False, default:False*) – Boolean for coat on inclusion.
- **t_coat** (*float, default:0*) – Thickness of the coating in actual units (mm/cm).
- **space** (*bool, True/False, default:False*) – Boolean for space which determines gap between inclusions in micro/mesostructure.
- **t_space** (*flaot, default:0*) – Thickness of the spacing in actual units (mm/cm). Voxel value for the coat.
- **kwargs** (*Other parameters, default:None*) –

generate_inclusion()

This method generates an inclusion

Returns Object of class `smg_inclusion.Inclusion`.

set_resolution(*resolution*)

3.2 Generating the Mesostructure

class generate_mesostructure.**Mesostructure**(*mesostructure_size=[100, 100, 100]*, *configuration=None*, *resolution=False*)

Bases: `object`

This class generates micro/mesostructure by assembling the inclusion/aggregates on to the main micro/mesostructure

Parameters

- **mesostructure_size** (*array of size (3), type int, default:[100,100, 100]*) – Size of the mesostructure 3D matrix.
- **configuration** (*Configuration object*) – Configuration object which provides details about the aggregate type and size distribution for assembly.
- **default** (*resolution array of size (3), type float,*) – resolution of the mesostructure (resolution for the voxel format)

add_configuration(*configuration*)

Add inclusion configuration to the assembly.

Parameters **configuration** (*Configuration object*) – Configuration object which provides details about the aggregate type and size distribution for assembly.

assemble_sra(*attempt_max=500000*, *threshold=50*, *iter_limit=10*)

Assemble aggregates/pores onto the mesostructure 3D matrix using Semi-Random Assembly (SRA) algorithm.

Parameters

- **attempt_max** (*int, default:500000*) – Maximum number of unsuccessfull assembly attempts before temrinating the assembly algorithm.
- **threshold** (*int, default:50*) – Number of unsuccessfull attempts after which the algorithm shifts to SRA (alorithm type-2) from RSA (algorithm type-1)

- **iter_limit** (*int, default:10*) – Number of unsuccessfull attempts to try with the same particle/aggregate orientation before switching to another random orientation.

Returns **mat_meso** – Mesostructure 3D array with aggregates/pores/particles assembled inside.

Return type 3D array of type int

3.3 Aggregate Design

class inclusion.**Polyhedron**(*a*, *b*, *c*, *coat*, *t_coat*, *space*, *t_space*, *n_cuts*, *concave*, *n_concave*, *depth*, *width*, *vox_inc*, *vox_coat*, *vox_space*)

Bases: object

Class for irregular polyhedron

Parameters

- **a** (*float, default:0*) – Diameter of the inclusion along direction-1 in voxel units
- **b** (*float, default: b=a*) – Diameter of the inclusion along direction-2 in voxel units
- **c** (*float, default: c=a*) – Diameter of the inclusion along direction-3 in voxel units
- **n_cuts** (*int, default:20*) – Number of faces for the irregular polyhedron.
- **coat** (*bool, default:False*) – Coating on inclusion, True/False.
- **t_coat** (*float, default:0*) – Thickness of coating on the polyhedron in voxel units.
- **space** (*bool, default:False*) – Space (like coating) on the polyhedron, True/False. This spacing creates gap between inclusion when assembled in the main meso/microstructure.
- **t_space** (*float, default:0*) – Thickness of spacing on the polyhedron in voxel units.
- **concave** (*bool, default: False*) – Provision to apply coating on the polyhedron, True/False.
- **n_concave** (*int, default:0*) – Number of concave depressions on the polyhedron surface.
- **depth** (*float, default:0*) – Parameter to determine depth of concave depression on the polyhedron surface (from 0 to 1)
- **width** (*float, default:0*) – Parameter to determine width of concave depression on the inclusion in voxel units

compute_vox_volume()
Compute voxel volume of the inclusion

generate_inclusion_matrix()
generate polyhedron inclusion

generate_new_inclusion()
Instantiate new polyhedron object and generates the matrix

3.4 Visualization and Export

visualization.**export_data**(*data*, *export_type='vtk'*, *fileName='mesostructure.vti'*)
The function exports data in the 3D array to the given export type (ex. vtk, npy, npz, csv, txt etc.).

Parameters

- **data** (*3D array of size NXNXN, type int*) – Micro/Mesostructure/Inclusion in voxel format.
- **export_type** (*str.*) – Export type (vtk/csv/txt/npy/npz)
- **fileName** (*str.*) – File location and file name with proper extension (./. . . /fileName.csv for export_type='csv')

visualization.**visualize_sections**(*matrix*, *slices=2*)
Visualize 2D sections of 3D matrix. Given number of sections (slices) are generated in each direction (xy,xz,yz)

visualization.**vti_tail**(*f*)

visualization.**write_vti_format**(*filename*, *phaseScalar*)
e.g write_vti_format('model101.vti', mcrt)

CHAPTER

FOUR

DEVELOPERS

- Jithender J. Timothy (lead developer and maintains the code)

The following have contributed to the development of this package:

- Vijaya Holla (programmed the original MATLAB code, optimized the Python version),
- Giao Vu (specification for concrete),
- Fabian Diewald (data for validation),
- Ketaki Godbole (converting MATLAB code to Python)

4.1 Funding

This software is partially supported by the German Research Foundation (DFG) within the framework of the sub-project FOR CODA.

CHAPTER

FIVE

PUBLICATIONS

1. Holla, V., Vu, G., Timothy, J. J., Diewald, F., Gehlen, C., & Meschke, G. (2021). Computational Generation of Virtual Concrete Mesostructures. Materials, 14(14), 3782.
2. Vu, G., Diewald, F., Timothy, J. J., Gehlen, C., & Meschke, G. (2021). Reduced Order Multiscale Simulation of Diffuse Damage in Concrete. Materials, 14(14), 3830.
3. Finger, C., Saydak, L., Vu, G., Timothy, J. J., Meschke, G., & Saenger, E. H. (2021). Sensitivity of Ultrasonic Coda Wave Interferometry to Material Damage—Observations from a Virtual Concrete Lab. Materials, 14(14), 4033.
4. Vu, G., Timothy, J. J., Singh, D. S., Saydak, L. A., Saenger, E. H., & Meschke, G. (2021). Numerical Simulation-Based Damage Identification in Concrete. Modelling, 2(3), 355-369.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX