These codes are generated by Dr. Manish Kumar and Prof. Enrico Salvati which can be used as a computational model to simulate the presence of the secondary small phases present in the polycrystalline materials, e.g., inclusions, intermetallic, and many more. The provided codes can be used to generate a mathematical domain to resemble actual microstructures, which can be used in various micromechanical models. For instance, in crystal plasticity, phase transformations, failure mechanisms through Phase-Filed or Cohesive Zone Method, etc. The provided code is for the two-dimensional model but can be easily extended to the three-dimensional problem.

Contact email: Manish Kumar <manish.kumar@uniud.it>, Enrico Salvati <enrico.salvati@uniud.it>, Group website <https://simed.uniud.it/>

Two codes are provided:

1. model_sub.py => To generate 2D rectangular domain resembling a polycrystalline material that contains the primary and secondary small subdomains using the Python API of the Gmsh library.
2. simulation_sub.py => To import the generated model, assign the different material properties to the domain and subdomains, and simulate for the linear elastic isotropic material considering small-scale deformations using the Python API of the FEniCS library.

Conceptual details of mesh_sub.py

This code generates a 2D rectangular domain resembling a polycrystalline material that contains the primary and secondary small subdomains inside the primary subdomains, as shown in the figure below.

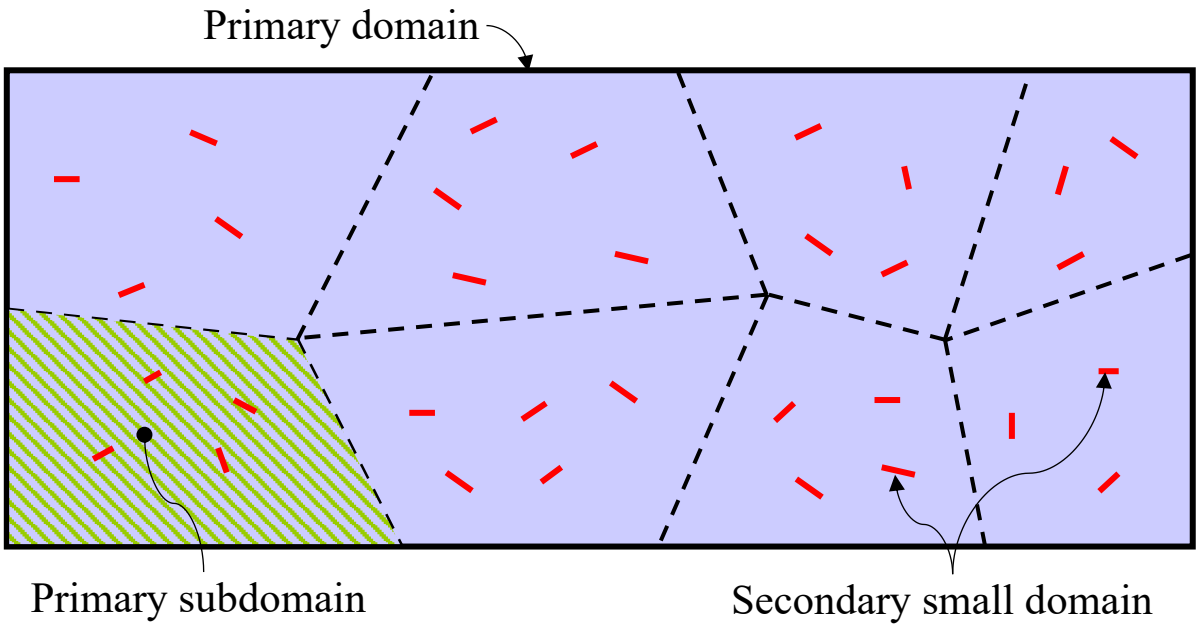

Figure 1: An illustration of the primary and secondary small subdomains in the primary domain.

The Python API of the Gmsh library is used for this purpose. The target is to produce a model in which the secondary small domains are randomly positioned and oriented in the primary subdomains. The secondary subdomains must be non-intersecting between themself. They also need to maintain the minimum distance from the boundaries of the primary domain and primary subdomains. The primary subdomains are created by defining the location of points and lines. The location of the secondary subdomain is obtained randomly and accepted only if it has a minimum distance from the primary domain's boundary. A minimum distance is checked with the previously accepted locations to make it non-self-intersecting. To make secondary subdomains non-intersecting with the boundaries of the primary subdomains, an offset polygon (shrunk polygon) is created, and checked the location of the secondary subdomain is within any of the offset polygons. If the location satisfies all the conditions, it is saved along with the track in which primary subdomain it lies. Later, the secondary subdomain is created and rotated by a random angle using the rotate operation of Gmsh. Eventually, the planer surfaces are created and meshed. The created entities are also converted into physical groups that can quickly transfer to the FEniCS for the simulation. More details about implementation can be found in the provided code.

Convert mesh into the appropriate format to import in the FEniCS

The generated model can be converted into .xml format using the command `dolfin-convert inputmesh.msh outputmesh.xml`. This conversion generates the three files: `outputmesh.xml`, `outputmesh_physical_region.xml`, and `outputmesh_facet_region.xml`. The `outputmesh.xml` file contains the mesh data of the model, the `outputmesh_physical_region.xml` file contains the tags for primary and secondary subdomains, and the `outputmesh_facet_region.xml` file has tags for all the domain and subdomain boundaries. All the features are extracted as physical entities by converting the model in this manner, which can be directly imported into FEniCS to apply different material properties. To use this method of model conversion, `.msh` has to be in the old version as implemented in the

provided code. Before importing the model files in the FEniCS environment, the file outputmesh_physical_region.xml must modify. The line number 3 in the original file is,

```
<mesh_function type="uint" dim="2" size="244674">
```

That must be edited as,

```
<mesh_function type="double" dim="2" size="244674">
```

Assigning the material properties in FEniCS

The Python API of the FEniCS library is used for this purpose. To assign the different material properties cpp interface of FEniCS is implemented in which material properties can be assigned based on the subdomain's tags. To assign the boundary conditions to the subdomain boundary, the following command can be used (not included in the provided code),

```
mesh = Mesh("outputmesh.xml")
boundaries_sub = MeshFunction("size_t", mesh, " outputmesh_facet_region.xml")
V_sub = FunctionSpace(mesh, 'CG', 1)
tag_number = 10
bc_sub = [DirichletBC(V_sub, Constant(1.0), boundaries_sub, tag_number)]
```

More details about implementation can be found in the provided code. In the provided code, a cantilever (as shown in Figure 2) is considered whose right edge is constrained, and point displacement is applied near the left edge at a point. The main purpose of this code is to present the assignment of the different material properties.

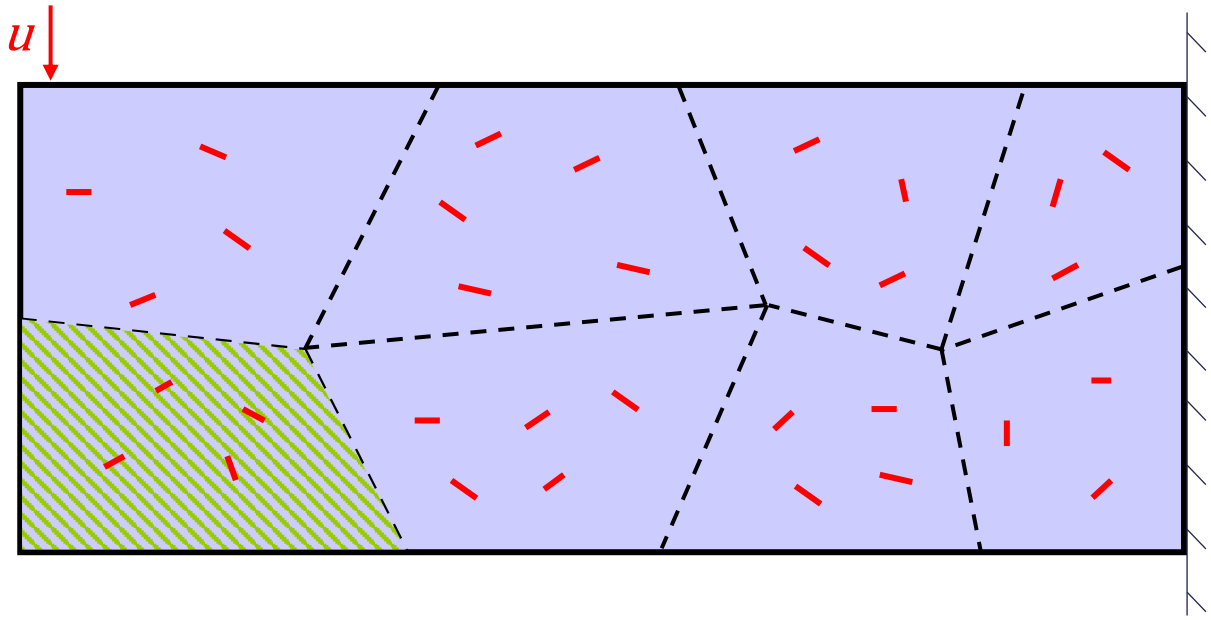

Figure 2: The representation of the considered cantilever for the simulation in simulation_sub.py.

For the reference purpose the generated model with model_sub.py and material assignment with code simulation_sub.py are presented in Figure 3 and 4.

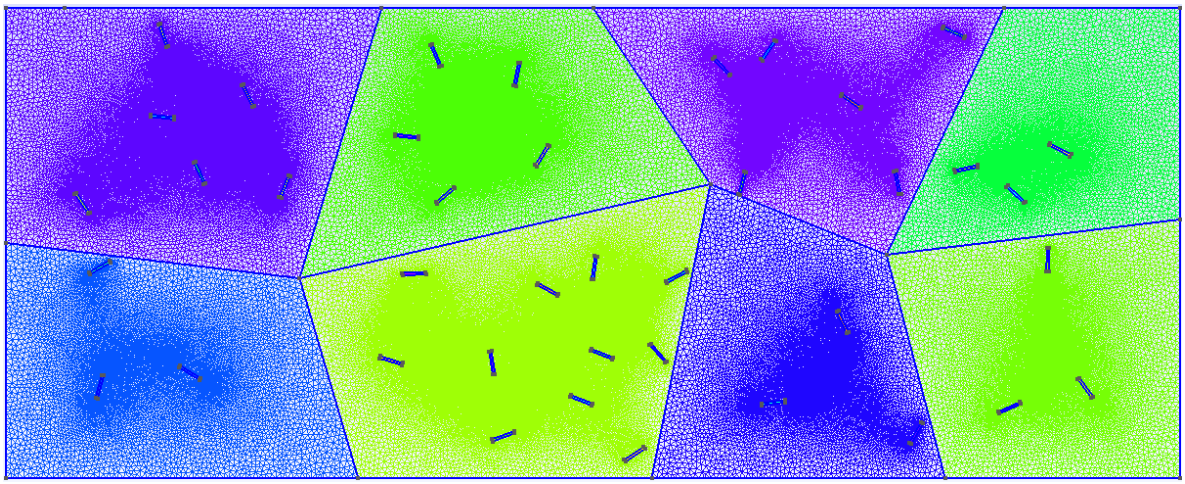

Figure 3: The computational model generated with the code model_sub.py.

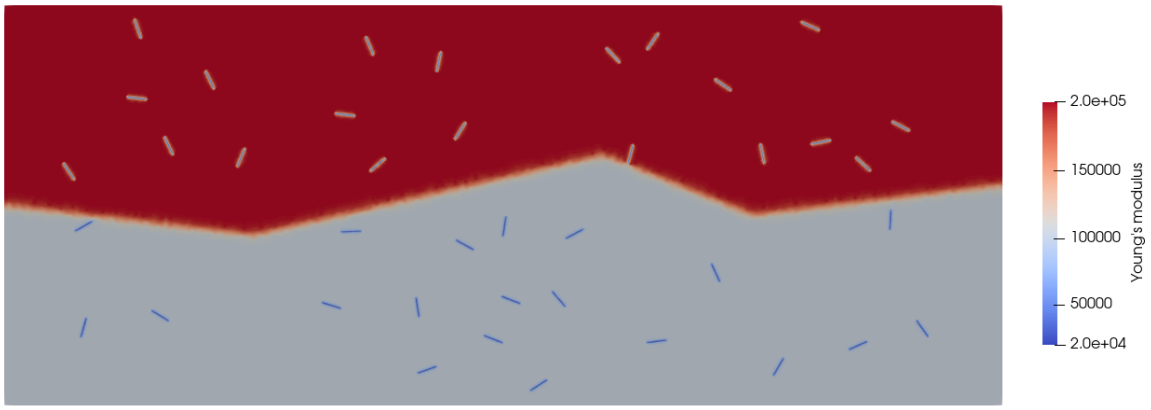

Figure 4: The contour plot of the material assignment by the code simulation_sub.py.