Marcin Słupek

Multiscale modelling first report

1. Introduction

This report describes functionalities of implemented grain growth simulation algorithms and analyses its accuracy compared to real life microstructures.

2. Used technologies

Programming language:

Scala – language that combines object oriented and functional programming. There is a lot of libraries. Fully compatible with Java which includes multiplatform purpose. Good performance and community support. Easy syntax and powerful standard library.

Graphic library libgdx – lightweight library focused on OpenGl. Gives tools for rendering textures, UI creation and controls.

3. Graphic user interface description

Figure 1 shows GUI, which functionalities are decribed below:

Figure 1 Graphic User Interface

1) – simulation space 2D size width x height
2) – amount of nucleons to be placed
3) – clear simulation space
4) – calculate single iteration of simulation
5) – change size of simulation space
6) – start simulation

7) – neighborhood type selection
8) – boundary condition type selection
9) – microstructure import and export to file
10) – file type for import/export
11) – clears space with grain boundaries left
12) – select if want to leave selected grains borders only
13) – add specified amount and shape of inclusions, if simulation has not started yet place them randomly, else place on grain boundaries only
14) – inclusions shape selection
15) – if checked changes behavior of simulation, and applies specified rule probability
16) – selects simulation type for new growth (substructure or dualphase)
17) – leaves selected grains and applies conditions for new growth.

4.1. Simple grain growth

First step was to implement simple grain growth algorithm with Moore neighbourhood. It can be useful to generate single phase microstructures such as copper (Figure 2). Generated structure has 100 initial nucleons with 300x300 simulation space size (Figure 3)

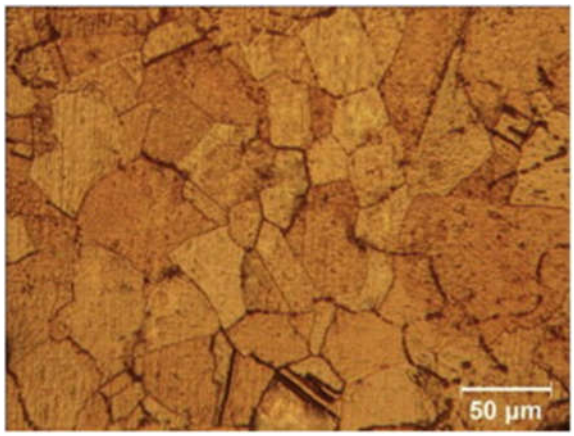

Figure 2 Microstructure of copper, https://www.sciencedirect.com/science/article/abs/pii/S0921509312012804 (11.12.2018)

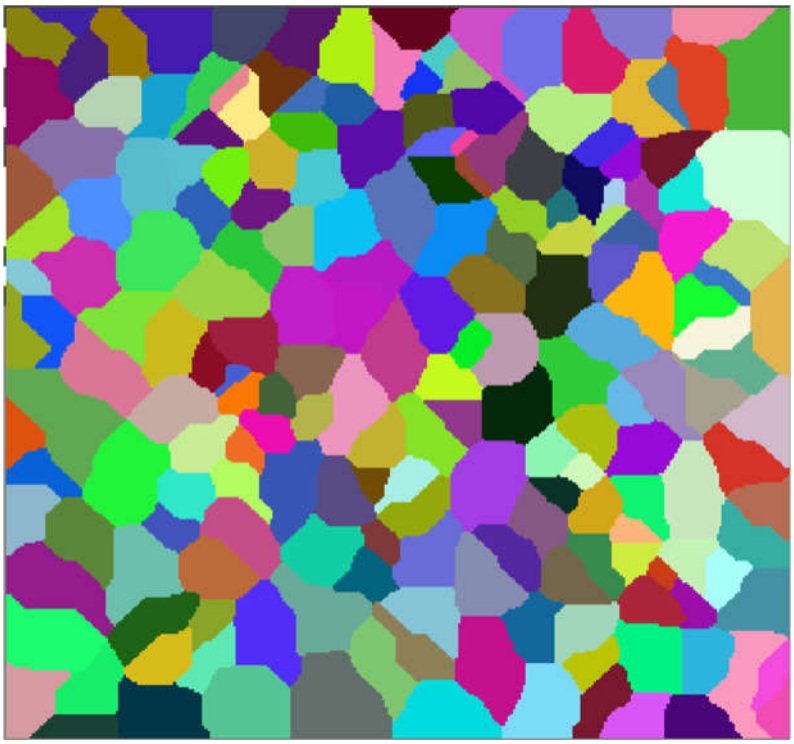

Figure 3 Result of naive growth simulation algorithm.

4.2. Microstructure export and import. Next step was to implement functionalities of importing and exporting generated microstructure to image file and text file. Figure 4 shows example content of generated file.

```
300,300
0,0,135
0,1,135
0,2,135
0,3,135
0,4,135
0,5,135
0,6,135
0,7,135
0,8,135
0,9,135
0,10,135
0,11,135
0,12,135
0,13,135
0,14,135
0,15,135
0,16,135
```

Figure 4 Content of generated text file. First row contains simulation space size. Any other row contains information aboput specific cel (grain id, x coordinate, y coordinate).

4.3. Third classes included modification to the algorithm which added inclusion to be places before (anywhere) or after simulation (on grain boundary). Figure 5 shows simulation output, that resembles microstructure of low carbon steel (Figure 6).

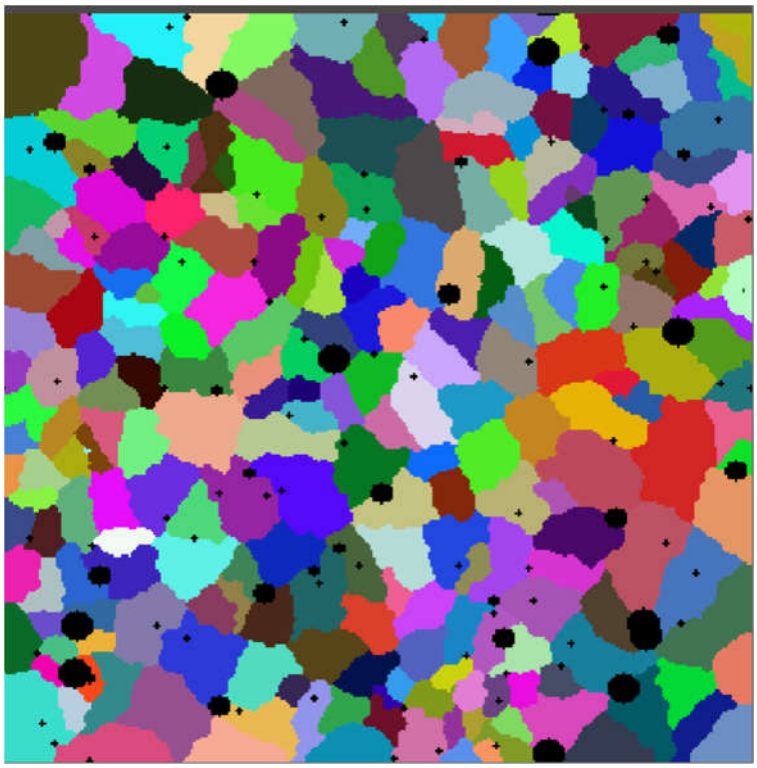

Figure 5 Outcome of simulation with 300x300 space, Moore Neighborhood, 30 inclusion of size 3 before simulation, 10 inclusion of size 10 and size 7 after simulation.

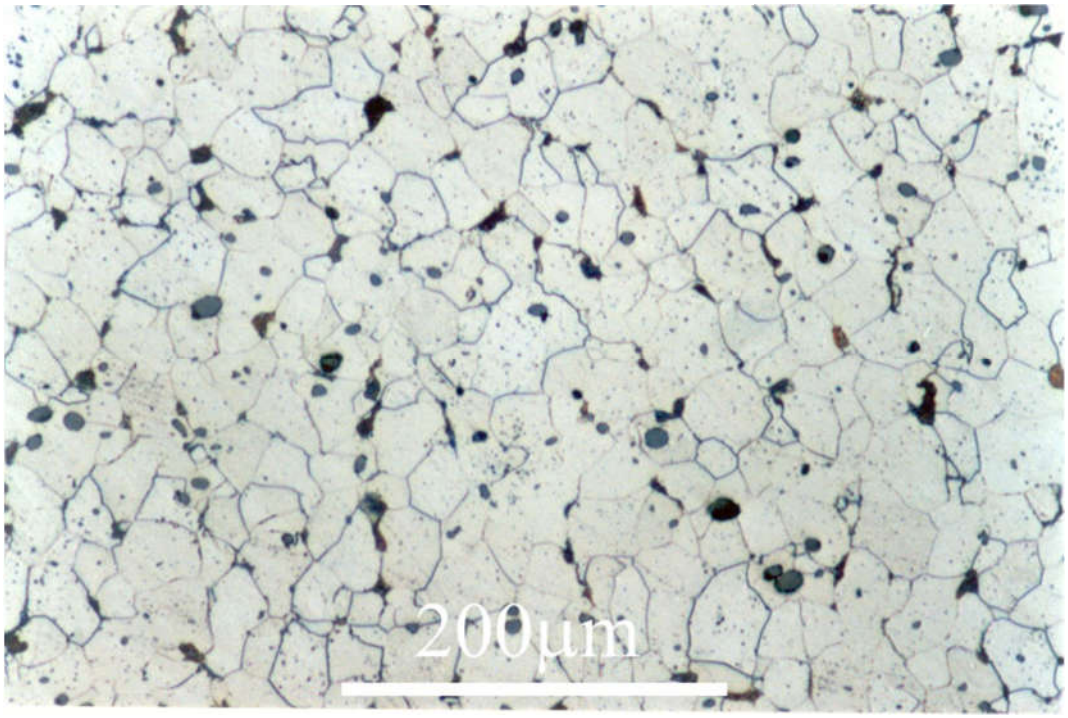

Figure 6 Microstructure of low carbon steel, http://core.materials.ac.uk/search/detail.php?id=1243 (10.12.2018)

4.4. Next class task was to implement alternative grain growth rules to gain some control over curvature of grains. Figure 7 presents image of generated microstructure with mentioned functionality.

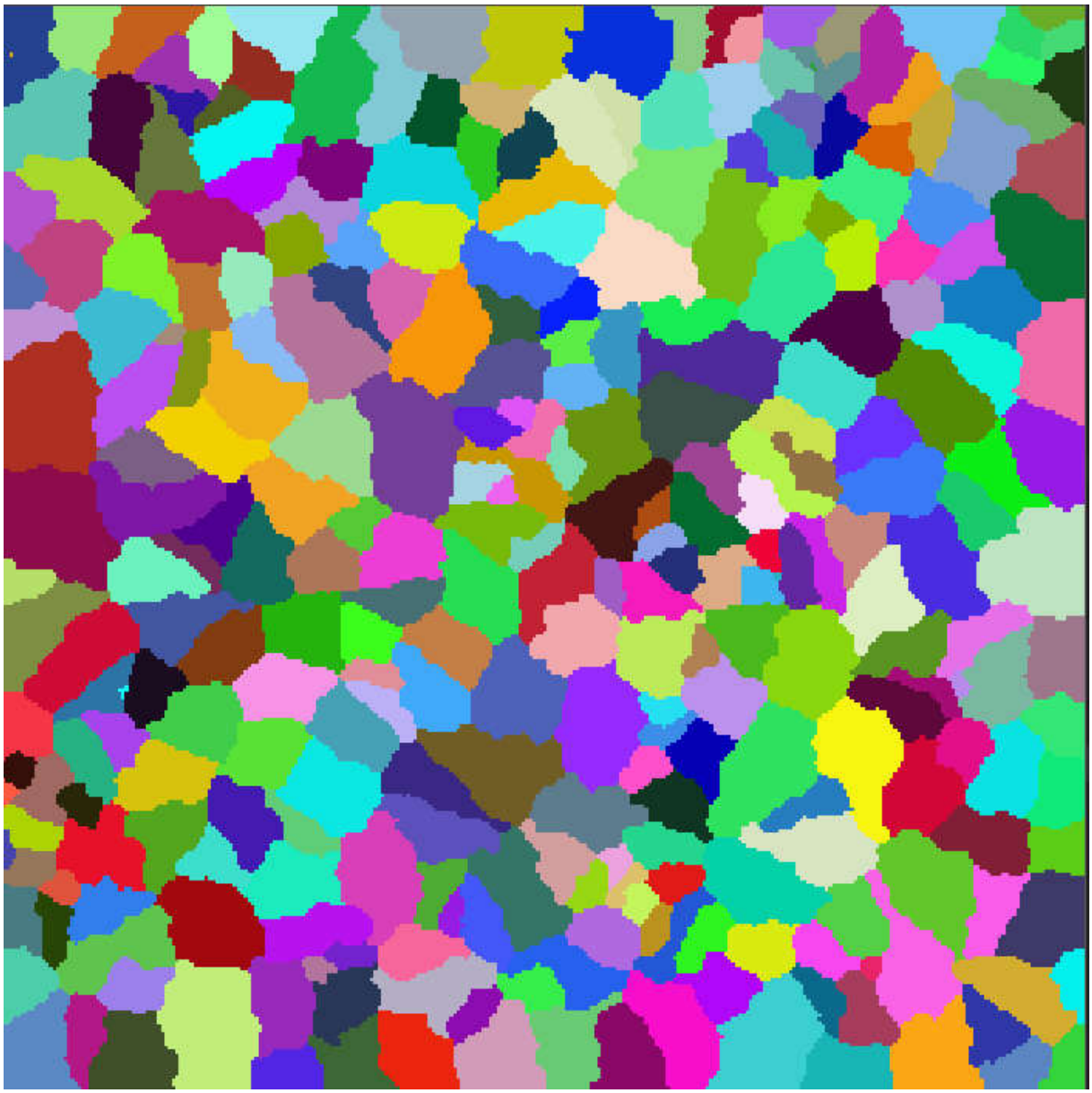

Figure 7 Alternative grain growth result with 300x300 simulationn space, 10% probability of rule 4.

4.5. On next class application was extended with second step of simulation which is supposed to generate dual phase microstructure. Figure 8 presents output of 2 step simulation. This method can generate microstructures resembling dual phase steel (Figure 9).

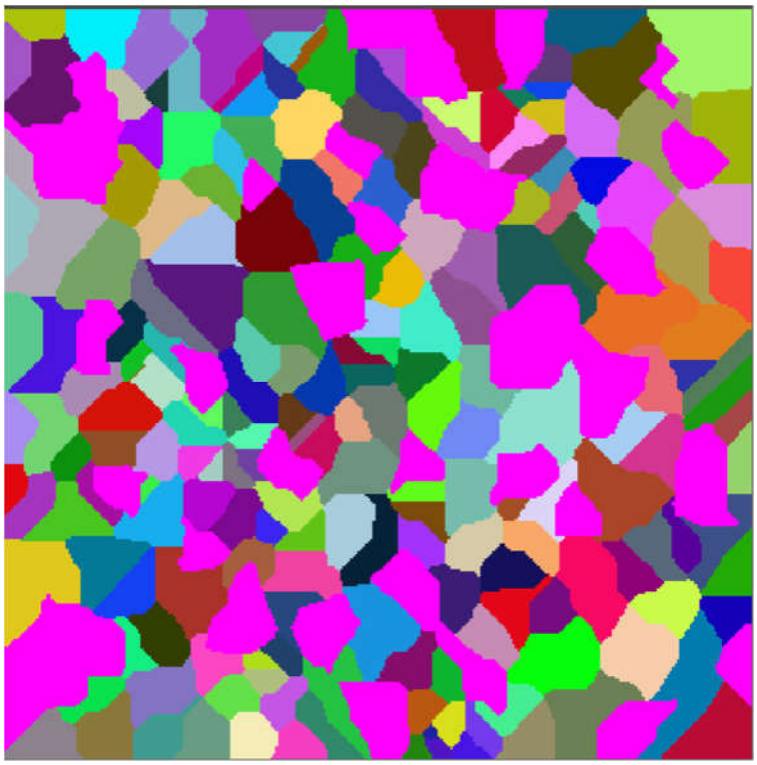

Figure 8 Generated dual phase microstructure with 300x300 simulation space, and dual phase as method for step 2.

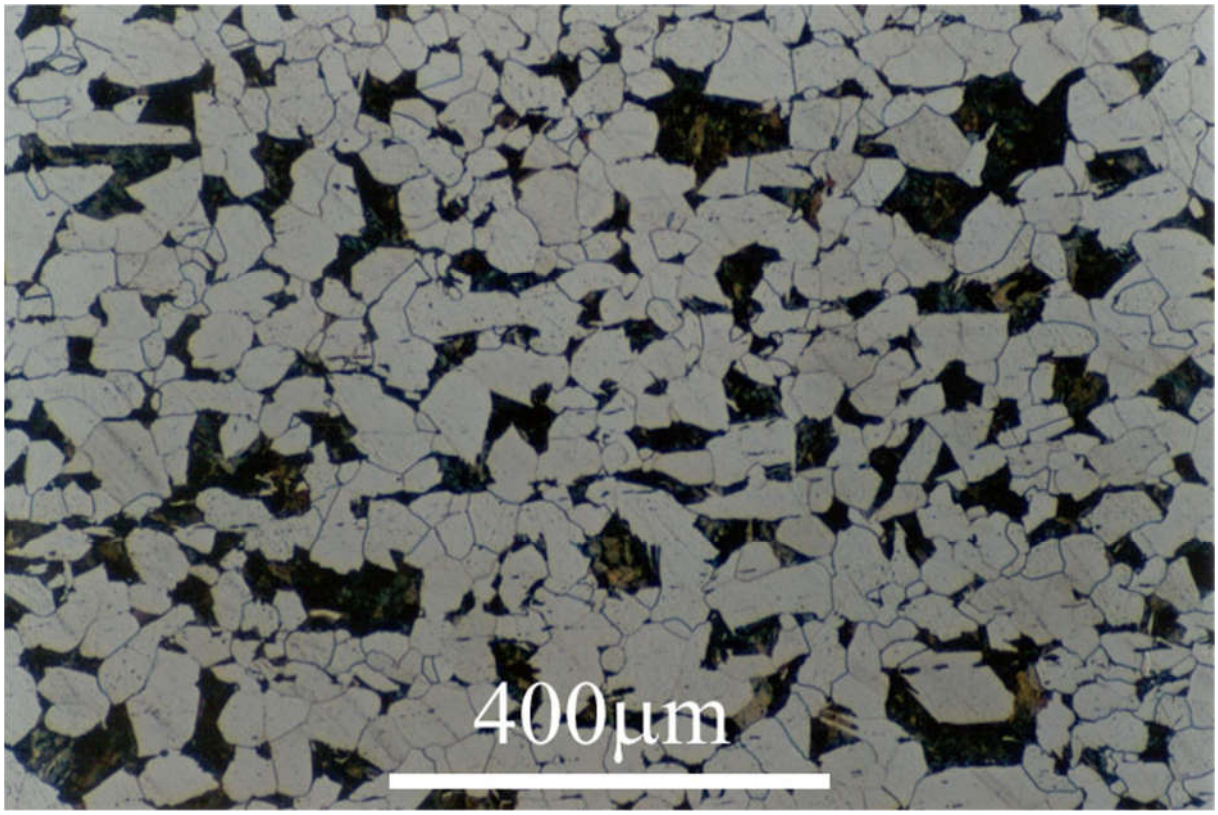

Figure 9 Microstructure of dual-phase steel, http://www.phase-trans.msm.cam.ac.uk/2001/adi/cast.iron.html (11.12.2018)

4.6. This microstructure was generated by using every earlier mentioned algorithm. Result of simulation created microstructure (figure 10) which is similar to ductile cast iron shown on figure 11

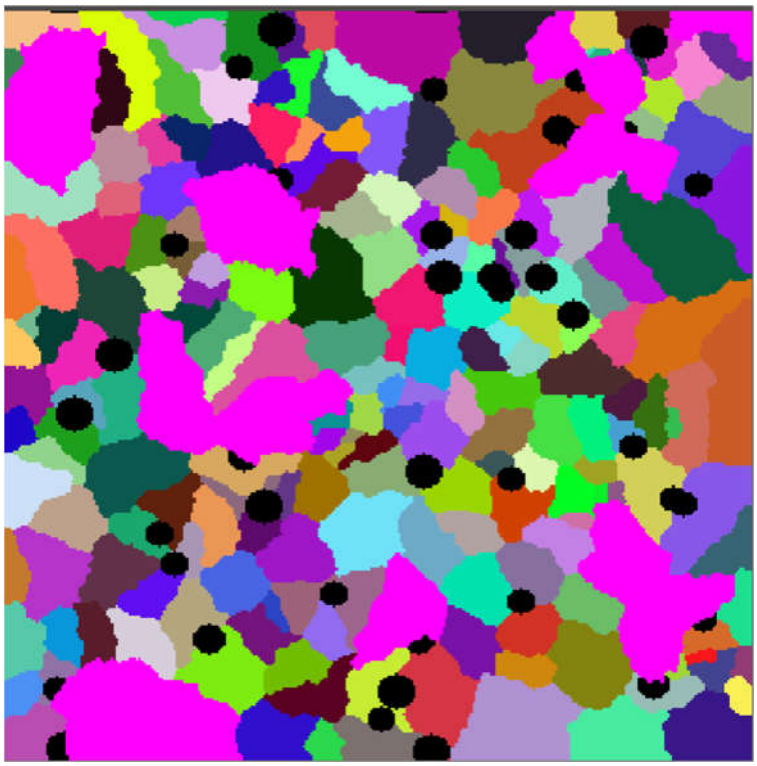

Figure 10 Microstructure generate by using 300x300 simulation space, with 10% rule 4 probability, 10 inclusions of size 10 and 7 after simulation and dual phase as second step.

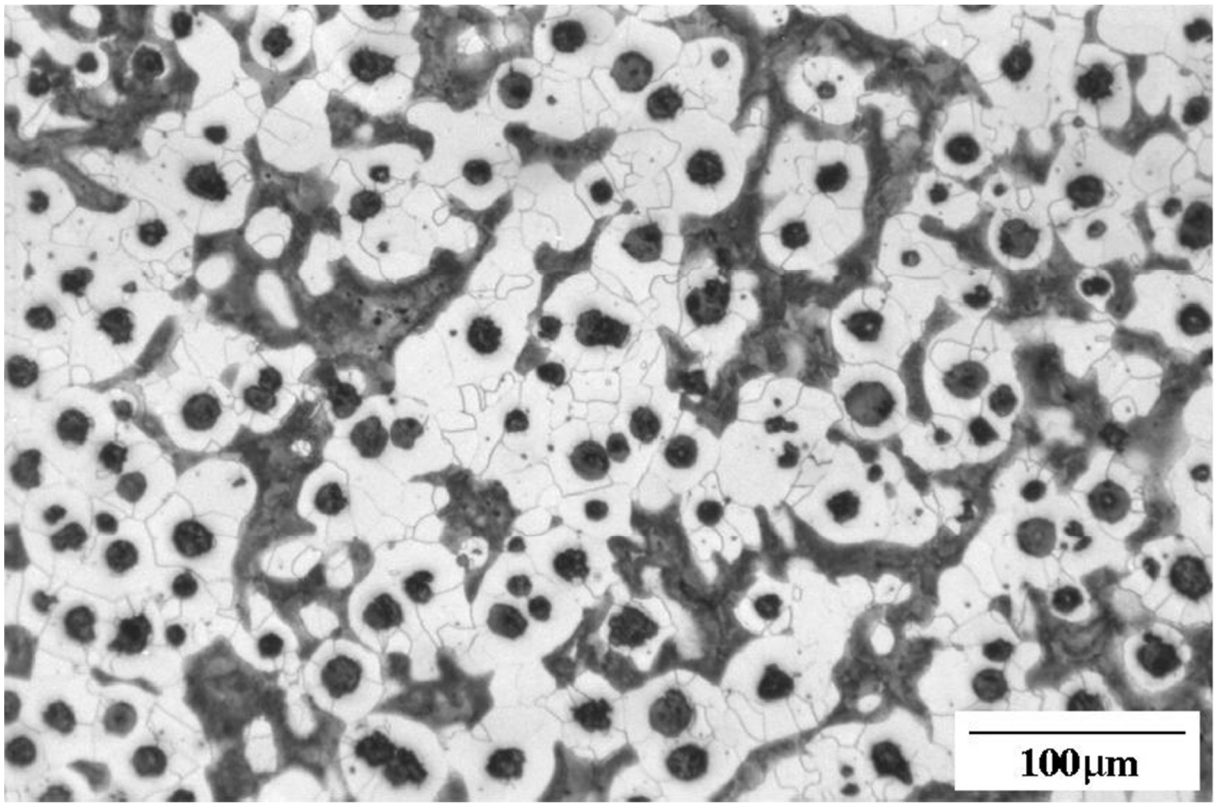

Figure 11 Microstructure of ductile cast iron,http://www.phase-trans.msm.cam.ac.uk/2001/adi/cast.iron.html (11.12.2018)

5. Conclusion

Implemented algorithms are capable of generating very accurate representation of various microstructures. Properly implemented application runs with good performance. If user gives accurate parameters to simulation it looks similar to microstructure photos. On the other hand size of simulation is limited. Processor and memory usage grows very fast with increased dimensions of simulation.