USER'S GUIDE

# Homogenizator MT 2.0: Composites with cracks

**Effective stiffness estimation according to Mori-Tanaka scheme, mortars containing coated and/or uncoated spherical inclusions and penny-shaped cracks caused by shrinkage**

*Author:*
Václav NEŽERKA

June 6, 2013

The program was developed for fast and user-friendly homogenization using Mori-Tanaka scheme, considering spherical inclusions, voids and spherical inclusions with coating. It also evaluates the crack density in the RVE due to shrinkage cracking between aggregates, which is commonly present in cementitious materials. If you encounter any problems or bugs when using the program, please don't hesitate and contact the author at `nezerka.v@seznam.cz`. The author would like to thank his supervisor, Jan Zeman, PhD., who provided him with fruitful consultations during the software development. This work was supported by the grant no. SGS13/034/OHK1/1T/11.

# 1 Program Structure

The program is started using the file main.m in the main folder. The program structure is following (see Fig.1):

- Input file processing script can be found in the folder "files" and the user needn't open this folder.
- The functions needed for a calculation can be found in the folder "functions".
- Folder "input" is of the biggest importance for the user, since all the input data should be stored there. The program starts searching for the input files right in this folder.
- The script and functions needed for the evaluation of crack density within the RVE can be found in the folder "CrackCounter3D", this contains the C++ scripts for the placement of particles and calculation of the crack density — these files have to be compiled for every other system than Win 7 (64-bit) in the MATLAB environment by the command `mex f_cpp_placeParticles.cpp` and `mex f_cpp_countCracks.cpp`.

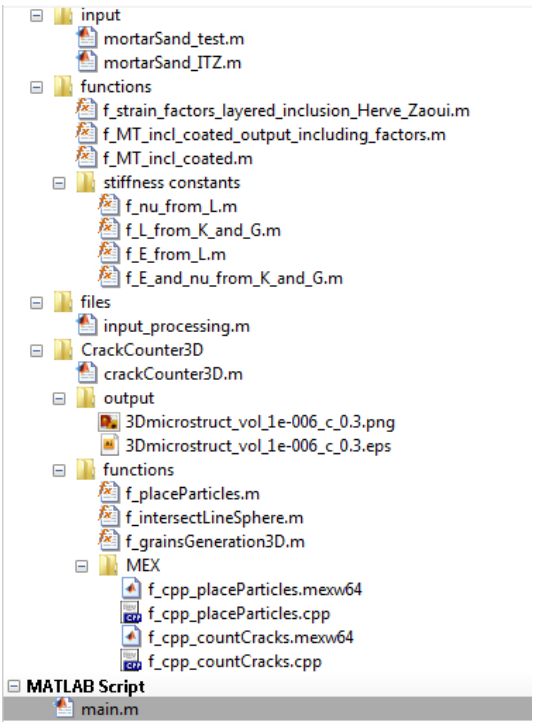


Figure 1: Program data structure

# 2 Input File Format

The input file must be in a format as in the sample file `input\_materials\mortarSand_test.m`, Every phase is characterized by several data, according to its type: matrix, coated inclusion, coating, uncoated inclusion, and voids (see Fig.2).

```
input_name = 'composite material 1';

%% Input parameters for individual phases

% matrix:
materials(1).name = 'matrix';
materials(1).code = 'matrix';

materials(1).m = 4; % mass portion in the mixture [kg]
materials(1).rho = 1900; % density [kg/m^3]
materials(1).E = 2000e6; % Young's modulus [Pa]
materials(1).nu = 0.25; % Poisson's ratio [-]

% brick:
materials(2).name = 'coated inclusion 1';
materials(2).code = 'coated';
materials(2).ID = 1; % it must be in agreement with a corresponding coating

materials(2).m = 4; % mass portion in the mixture [kg]
materials(2).rho = 1600; % density [kg/m^3]
materials(2).E = 2400e6; % Young's modulus [Pa]
materials(2).nu = 0.17; % Poisson's ratio [-]

materials(2).GC.frac = [0.063,0.125;
                        0.125,0.25;
                        0.25,0.5;
                        0.5,1;
                        1,2;
                        2,4]*1e-3; % grading cuve - fractions (diameter) [mm]
materials(2).GC.amount = [6;
                          24;
                          45;
                          60;
                          66;
                          150]; % number of grains from each interval(mass, volume) [-]

materials(2).n_in_fraction = 2; % number of fractions within one fraction interval

% sand:
materials(3).name = 'uncoated inclusion';
materials(3).code = 'uncoated';

materials(3).m = 4; % mass portion in the mixture [kg]
materials(3).rho = 2600; % density [kg/m^3]
materials(3).E = 70000e6; % Young's modulus [Pa]
materials(3).nu = 0.17; % Poisson's ratio [-]

% voids:
materials(4).name = 'voids';
materials(4).code = 'voids';

materials(4).volume = 35; % volume [%]
materials(4).E = 1e-9; % Young's modulus [Pa]
materials(4).nu = 0.25; % Poisson's ratio [-]

% C-S-H gel:
materials(5).name = 'coating of coated inclusion';
materials(5).code = 'coating';
materials(5).ID = 1; % it must be in agreement with a corresponding coated particle

materials(5).rho = 2000; % density [kg/m^3]
materials(5).E = 22000e6; % Young's modulus [Pa]
materials(5).nu = 0.2; % Poisson's ratio [-]
materials(5).thickness = 20e-6; % thickness of coating [m]
materials(5).p_i = 20; % percentage of inclusion consumed by coating when coating is created;
materials(5).p_m = 50; % percentage of matrix consumed by coating when coating is created;
materials(5).p_v = 30; % percentage of voids consumed by coating when coating is created;
```

Figure 2: Format of an input file

# 3 Calculation of Effective Material Properties

The effective material properties are calculated according to Mori-Tanaka scheme. The program is able to calculate the crack density parameter, based on the given criteria, in particular critical distance between particles for the formation of shrinkage-induced crack between them. This can be obtained for instance from a numerical study, see Figure 3.

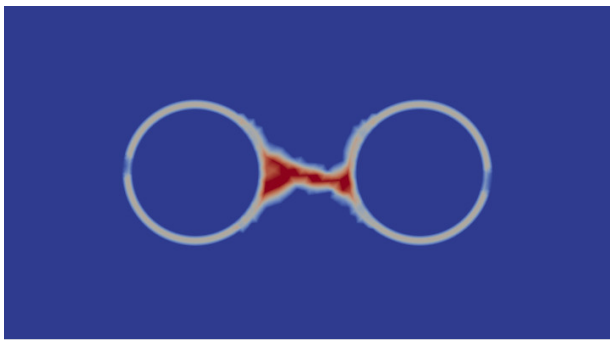

Figure 3: Shrinkage induced cracking between aggregates

If there is a critical ratio of a gap between the neighboring aggregates to their mean diameter $l_{\text{crit}} = (d_1 + d_2)/2L$, where $d_1$ and $d_2$ are the diameters of two neighboring particles, and $L$ is the distance between them, the simulation of particle distribution within a sufficiently large RVE can provide the information about the crack density parameter, $f$. This is accomplished by the *CrackCounter3D*.

Knowing the crack density parameter, the reduction of the matrix stiffness is accomplished by an additional compliance tensor, represented by its volumetric and deviatoric parts, $H_{\text{V}}$ and $H_{\text{D}}$, given by

$$H_{\text{V}} = \frac{f}{E^{(0)}} \frac{16(1-(\nu^{(0)})^2)(10-3\nu^{(0)})}{45(2-\nu^{(0)})} \quad \text{and} \quad H_{\text{D}} = \frac{f}{E^{(0)}} \frac{32(1-(\nu^{(0)})^2)(5-\nu^{(0)})}{45(2-\nu^{(0)})} \tag{1}$$

corresponding to the scheme of dilute distribution, which appears to be a reasonable approximation if the cracks are randomly oriented. The uncracked matrix is represented in the Equations 1 by its Young's modulus, $E^{(0)}$, and Poisson's ratio, $\nu^{(0)}$. The effective Young's and shear moduli of the cracked matrix can be then determined as

$$E^{(0)}_{\text{cr}} = \left[\frac{1}{E^{(0)}} + H_{\text{V}}\right]^{-1} \quad \text{and} \quad G^{(0)}_{\text{cr}} = \left[\frac{2(1+\nu^{(0)})}{E^{(0)}} + H_{\text{D}}\right]^{-1} \tag{2}$$

After the homogenization of cracks and matrix, the M-T scheme is employed for the determination of the effective mortar stiffness.

# 4 Program Settings

Figure 4 shows the options available in the program. These are:

- The *run* command determines what input file is to be loaded.
- *cubeSizeRVE* determines the size of the RVE for the purpose of crack density calculation, recommended $10\times$ maximum aggregate diameter.

```
%% Calculation

% input materials
run mortarSand_test

% sample dimensions
cubeSizeRVE = 30e-3;       % size of cubical RVE
minDistBetweenGrains = 0;  % minimum distance between grains
periodicBC = true;         % true = grains can intersect the RVE boundary

% utilization of C++ (MEX) functions (faster, no plotting !!! , only periodic BC available)
C_functions = true; % true = utilizes MEX function written in C++ for placement of grains
                    % false = utilizes only matlab (.m) functions

% setting the cracking criteria
criticalRatio = 0.7; % critical ratio of gap between aggregates to their mean diameter

% microstructure plotting options
figureOut = false; % true = plots the generated microstructure, false = no plotting
saveFigure = false; % true = saves the figure
uncrackedComparison = true; % true = compares with uncracked sample, false = no comparison
```

Figure 4: Settings available in main.m

- *minDistBetweenGrains* determines the minimum distance between neighboring grains.
- *periodicBC* determines whether the grains can cross the RVE boundary (true) or not (false)
- *C_functions* determines whether the C-language written functions shall be used, these must be compiled using the *mex* command prior to calculation, when Windows 7 (64bit) system is not used. The utilization of C-functions makes the calculation approximately 300× faster.
- *criticalRatio* represents the critical ratio of a gap between the neighboring aggregates to their mean diameter, see Section 3.
- *figureOut* determines whether the simulated particles shall be plotted. The plotting is quite memory demanding and time consuming. Example of the RVE image can be seen in Figure 5.
- *saveFigure* determines whether the plotted figure shall be saved, which also takes some time and memory. The figures are exported in *.png and *.eps format to the folder `\CrackCounter\output\`.
- *uncrackedComparison* determines whether the results are compared with the results of composite without consideration of shrinkage cracking.

After the calculation is complete, the results are printed in the MATLAB command window (console). The user should be provided with the data as depicted in Figure 6.

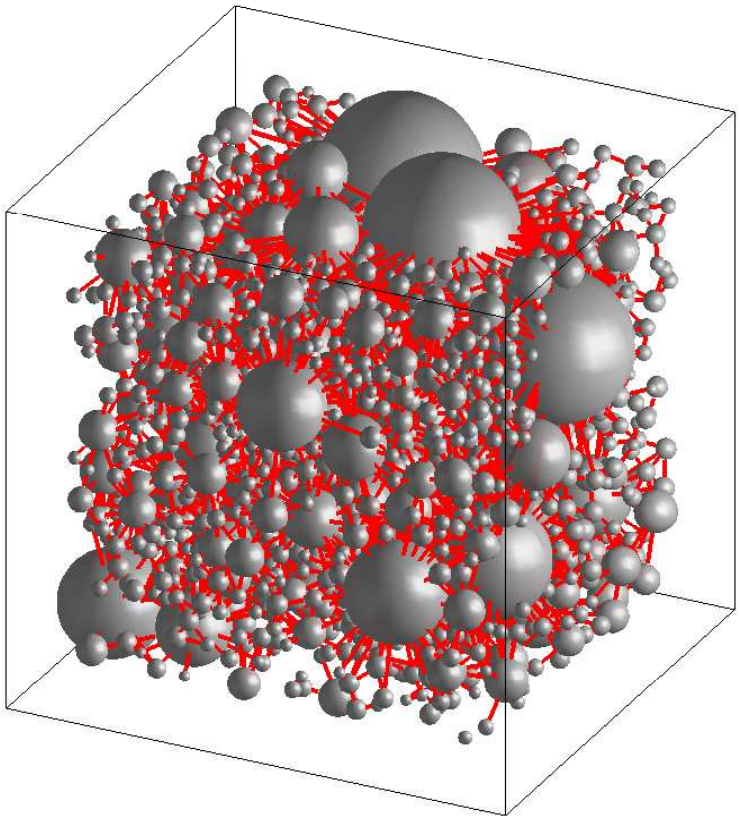

Figure 5: Simulated microstructure: random distribution of particles and shrinkage induced cracks (red)

```
MATERIAL: mortarSand
Mix is composed of matrix and sand grains.
---------------------------------
Running the Crack Counter 3D - simulation of microstructure
31679 particles were generated, placement of grains...
100% particles placed, calculation of crack density...
Crack density is 0.024.
Elapsed time is 15.542211 seconds.
---------------------------------
The Young's modulus of matrix was reduced by cracks from 6900.00 to 6621.87 MPa
-- uncracked matrix configuration --
E_eff = 8561.0 MPa, nu_eff = 0.242
-- cracked matrix configuration --
E_eff = 8227.1 MPa, nu_eff = 0.216
```

Figure 6: Program output