SCARLET-1.0 : user manual

Author: Eduardo Rossi

Version 1.0 – update 2-12-2020

1. Installation

SCARLET-1.0 package contains two subfolders:

- "*source code*": all the Matlab files are inside this folder.
- "*STL examples*": it contains a collection of STL shapes that the user can employ to create virtual aggregates

No setup or installation is required. However the user must open Matlab at same level of the folder "*source code*" (check the Matlab path).

2. How to run SCARLET-1.0

SCARLET-1.0 is a user-friendly package that can be run just typing few commands in the Matlab Command Window. The two main codes are "*fromStlToSpheres.m*", a preliminary code to translate STL files into their sphere-equivalent structure, and "*SCARLET_v1.m*", the true engine of the package that builds virtual aggregates and produces their STL representation.

2.1 Converting STL shapes into their sphere equivalent structure: fromStlToSpheres.m

The conversion of a 3D shape into its sphere equivalent representation is done by means of the function *fromStlToSpheres.m*. It reads as follows:

```
>>[spheres_struct,fv]=fromStlToSpheres(stl_file,N_spheres,N_iter_max_per_sp
here, barycenterMode, light_mode)
```

Where:

- `stl_file`: the file path where the STL file is contained (the file path is defined respect to the main SCARLET-1.0 folder).

- `N_spheres`: the number of spheres aggregated around the central one.

- `N_iter_max_per_sphere`: number of iterations to be done before placing each single sphere (higher values mean spheres with larger diameters).

- `barycenterMode`: a Boolean variable where "1" means placing the first sphere in the barycenter of the STL shape. "0" means the contrary.

- `light_mode`: a Boolean variable where "1" means to discard the inner spheres and to keep only the ones that are not surrounded by others. This option must be used carefully, since it can speed up the code but also provide a not accurate description of the intersections depending on the complexity of the shape.

- `spheres_struct`: output structure that contains all the information about the spheres used to represent the STL shape, such as the coordinates of the center and radius.

- `fv`: output structure that contains the faces and vertices of the original STL file.

For example, let us assume that we want to create the sphere equivalent representation of the ellipse contained inside the folder "STL_examples". In the Matlab Command Window it is sufficient to type:

```
>>[sph_ellipse,fv_ellipse]=fromStlToSpheres('../STL_examples/ellipse.stl',
100,10,1,0);
```

The output of the function will be the two structures "*sph*" and "*fv*" (they appear as new structures in the workspace) and three plots showing the original STL file (Fig. 1a), the sphere equivalent object (Fig. 1b) and the vertices of the original STL file (Fig. 1c).

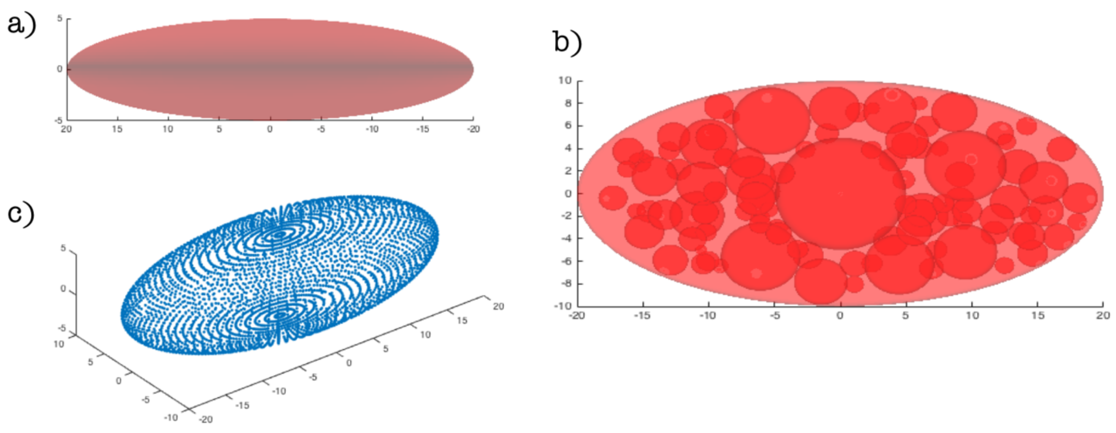

Figure 1 Figures generated after the execution of fromStlToSpheres.m: (a) original STL file; (b) STL file filled with N non overlapping spheres; (c) Vertexes used to represent the original STL file.

The creation of the sphere equivalent structure is mandatory for SCARLET-1.0. However, once that the *sph* and *fv* structures have been created, they can be saved and reused for a different simulation. It is worth noticing that the request of *N* multiple shapes inside the virtual aggregate requires the repetition of *fromStlToSpheres.m N* times.

2.2 Creation of the 3D virtual aggregate: SCARLET_v1.m

The creation of 3D virtual aggregates as well as the STL output of the aggregate is done by means of the function *SCARLET_v1.m*. This function requires the creation of a dedicated input structure and some additional internal settings, as illustrated in the following with an example.

Creation of the input structure. Let us assume that with *fromStlToSpheres.m* we created the sphere equivalent representation of two different STL files: "ellipse.stl" and "particle.stl", each of them contained inside the folder "STL_examples". Let us also assume that we named the outputs of each representation as "fv_particle", "sph_particle", "fv_ellipse" and "sph_ellipse".

```
>>[sph_ellipse,fv_ellipse]=fromStlToSpheres('../STL_examples/ellipse.stl',
100,10,1,0);
```

```
>>[sph_particle,fv_particle]=fromStlToSpheres('../STL_examples/particle.stl',
100,10,1,0);
```

Let us assume that we want to create an aggregate with the "particle.stl" at the center and 10 ellipses as a coating. In the command window we type:

```
>>input_struct(1).fv = fv_particle;
>>input_struct(1).sphere_struct = sph_particle;
>>input_struct(2).fv = fv_ellipse;
>>input_struct(2).sphere_struct = sph_ellipse;
```

Important! The first STL file in the new structure will be the core of the virtual aggregate and all the shapes in position from 2 to N will be used for the coating.

Internal settings. The internal settings for a single simulation are defined from line 26 to line 66 in the main function `SCARLET_v1`. Here

`closet.core_size_mu:` size of the core (in micron)
`closet.core_density:` density of the particles (in kg m^{-3})
`closet.N_particles:` number of particles in the coating
`closet.vector_coat_sizes_mu:` vector with the sizes of the coating particles (in micron). The i-th particle contained in the vector is scaled in order to set its maximum dimension equal to the size contained at the position i-th in the vector.
`closet.displace_in_order:` Boolean variable to place the sizes contained inside closet.vector_coat_sizes_mu in sequence (1) or randomly (0)
`closet.cone_aperture_degree:` aperture of the solid angle of investigation (Ω in degrees)
`closet.N_raysXSA:` number or rays (N_r) used within each solid angle Ω
`closet.N_Euler_triplets:` number or rotations (N_o) used within each solid angle Ω
`closet.origin_in_the_CM:` Boolean variable to select the origin of the solid angle of investigation in the center of mass of the aggregate (1) or inside on of the already placed particles (0)
`closet.alpha_transparency:` transparency of the figures (default value: 1)
`closet.view_angle_1:` azimuth camera angle (default value: 35)
`closet.view_angle_2:` altitude camera angle (default value: 42)
`closet.translation_iter_max:` maximum number of iterations before placing a new coating particle (default value: 1000)
`closet.delta:` constant for the coarse inward movement (default value: 0.05)
`closet.delta_2:` constant for the fine outward movement (default value: 0.0005)
`closet.n_points4Feret:` number of point used to calculate the Feret diameter of a shape (default value: 1000)

Important! The units used in the inward and outward movement of the particle (i.e. closet.delta and closet.delta_2) are expressed as a fraction of the particle size. Each iteration moves the object of a quantity equal to "size * closet.delta * i"

Let assume that we want to create a virtual aggregate with 20 rays for each solid angle of aperture 90 with 10 rotations. Once we created the input structure (as shown before) we simply set the internal settings as desired and type the following command:

```
>> [output_st] = SCARLET_v1(input_struct);
```

The output of the code is made of

- The output structure `output_st`: inside this structure there are two sub-structures: `packing_info` and `aggregate`. `Packing_info` contains the the external volume of the aggregate, the internal volume, the porosity, the density, the size and the mass. `Aggregate` contains the vertices and the faces of each component of the aggregate.

- three figures showing: the virtual aggregate (Fig. 2a), the estimated external volume using the convex hull approximation (Fig. 2b), the external points used for the definition of the external volume (Fig. 2c);

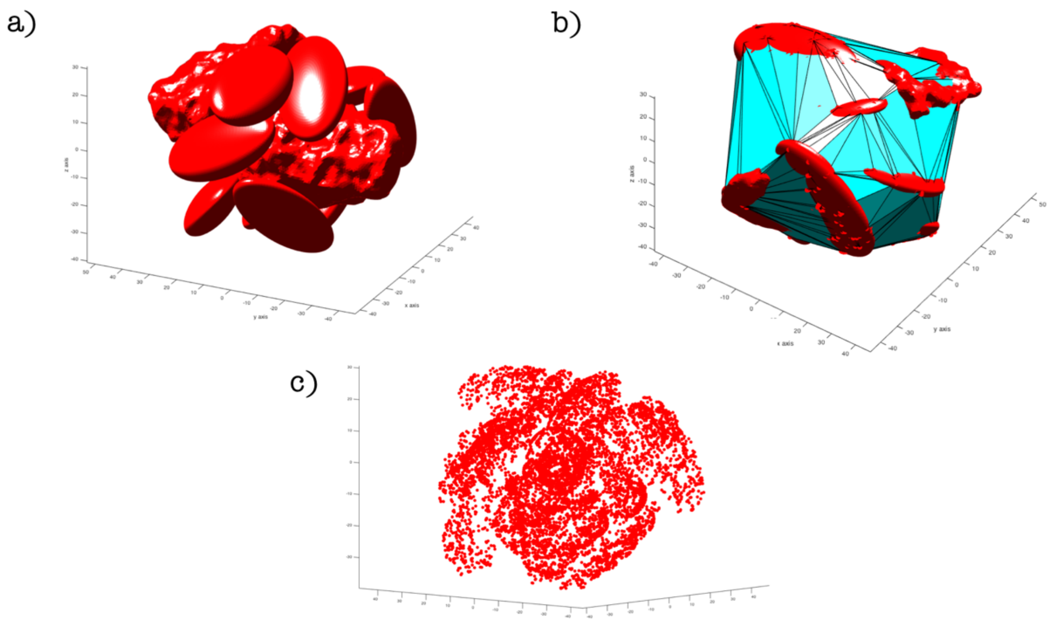

Figure 2 Main figures generated by SCARLET_v1: (a) the virtual aggregate; (b) the external convex hull used by the software to calculated the external volume; (c) the external points used to define the external convex hull.

- the STL file of the virtual aggregate, denominated by default: "`SCARLET_new_stl.stl" (that will be generated inside the same folder "source code")`.

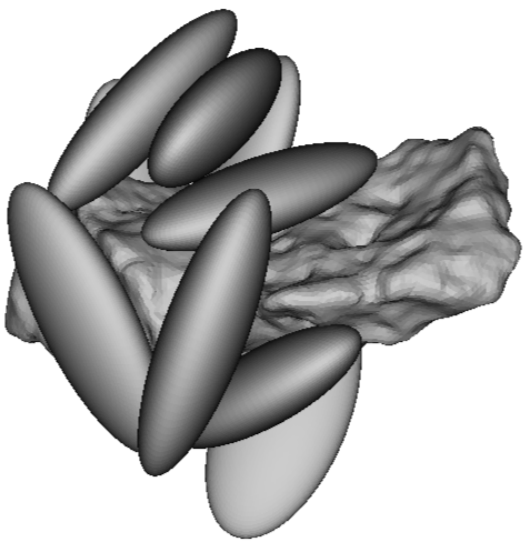

Figure 3 New generated STL file of the virtual aggregate, named by default "SCARLET_new_stl.stl"

3. Copyright and license

SCARLET-1.0 uses few internal Matlab subroutines downloaded from the MathWorks file exchange. The list of them and the license is reported in the following: