USER MANUAL for GRAPE v1
(MATLAB version)

Introduction

GRAPE is a code specifically designed to compute the gravitational field of irregular shapes, described in terms of STL file. It has been realized for teaching purposes. It is sufficient to modify and save the "input.inp" file present in the folder and then type "output_structure = GRAPE_v1" in the MATLAB command window. The speed of the algorithm depends on the number of spheres used to fill the STL file and the shape. The code is parallelized in the computation of the gravitational field at the probe points.

Caveats

1. *Orbit evaluation*: the main purpose of the code is to calculate and show the gravitational acceleration around an irregular shape. In addition, GRAPE can also calculate the orbit of a probe around the irregular shape (considered fixed in space). However the code is not optimized for this task. Moreover, the use of "griddata" in the interpolation process and the strong dependence of the orbit on the initial and boundary conditions can result in a large computational time. We suggest to activate the Boolean value of the code for the evaluation of the orbit only in the case of a perfect knowledge of MATLAB and of the problem under analysis. Alternatively, the user can directly download the gravitational field calculated by the package (available in the output structure) and use it in her/his own algorithm for orbit evaluation.

2. *Number of vertices in a STL files*: the algorithm uses the function "inpolyhedron" to verify if a point is inside a triangulated surface and the vertices of the triangulation to calculate the radius of the new generated sphere. In case of a STL surface with a reduced number of vertices the algorithm will generate a non-accurate sphere composite representation of the actual shape. We suggest to only use STL files with a large number of vertices.

3. *STL format*: binary STL files should be preferred instead of the ASCII format. In case of a STL file saved in ASCII format, it can be easily converted in binary using a specific software or one of the free online converters available on the internet.

Input file

The user must modify the following input file and save it.
Then type "output_structure = GRAPE_v1" in the MATLAB command window.

```
rosetta2.stl    % stl_file_name (with stl extension)
  50            % Number of spheres to use to fill the STL shape
  5             % Number of iterations needed to place each single sphere
  1e16          % Total mass of the object [kg]
  4000    % Maximum size of the object [m]
```

```
50          % Number of probes per latus (i.e. NxNxN)
10000           % Maximum x coordinate for probes [m]
-10000          % Minimum x coordinate for probes [m]
10000           % Maximum y coordinate for probes [m]
-10000          % Minimum y coordinate for probes [m]
10000           % Maximum z coordinate for probes [m]
-10000          % Minimum z coordinate for probes [m]
10          % x slice coordinate [m] for Fig. 4
10          % y slice coordinate [m] for Fig. 5
10          % z slice coordinate [m] for Fig. 6
10          % x slice coordinate [m] for Fig. 7
10          % y slice coordinate [m] for Fig. 8
10          % z slice coordinate [m] for Fig. 9
 0          % Boolean variable to calculate the orbit (0 = no; 1 = yes)
-3000           % Initial condition on the position - x component (x0) [m]
3000        % Initial condition on the position - y component (y0) [m]
3000        % Initial condition on the position - z component (z0) [m]
15          % Initial condition on the velocity - x component (vx0) [m/s]
0           % Initial condition on the velocity - y component (vy0) [m/s]
0           % Initial condition on the velocity - z component (vz0) [m/s]
1000        % Final time of integration [s]
35          % Number of steps for the ODE
10          % Number of probes in the ODE
```

More details about the meaning of each line in the input file:

- "*rosetta2.stl % stl_file_name (with stl extension)*": here is where the STL file must be specified. It is important to have it in binary extension (otherwise it can create troubles with the memory space) and to specify the ".stl" extension. Some examples are reported in the folder.

- *"50 % Number of spheres to use to fill the STL shape"*: this is the number of spheres that are used to describe the irregular shape. We suggest to use a number within 50-1000 for computational time reasons. In case more computational time is permitted also numbers up to 10000 can be used.

- *"5 % Number of iterations needed to place each single sphere"*: this number specifies the number of iterations requested before placing a single sphere. If this number is higher, in general, the user will fill the STL file with sphere of larger radii. We suggest to set it up to 5.

- *"1e16 % Total mass of the object [kg]"*: this is the total mass of the actual body expressed in kilograms.

- *"4000 % Maximum size of the object [m]"*: this number expresses the maximum length of the real body. It is needed to scale the STL file object to the actual size desired.

- *"50 % Number of probes per latus (i.e. NxNxN)"*: this number specifies the number of point around the object where the gravitational field vector is computed. The user should imagine a rectangular box around the body where NxNxN points are placed. This number depends on the resolution required. But since it grows polynomially we suggest to have it in the range 20-80.

- *"10000 % Maximum x coordinate for probes*
  -10000 % Minimum x coordinate for probes
  10000 % Maximum y coordinate for probes
  -10000 % Minimum y coordinate for probes
  10000 % Maximum z coordinate for probes
  -10000 % Minimum z coordinate for probes"
  These six input lines define the size of the rectangular box around the body in which the probes will be located. The size is in meters. The size of the rectangular box depends on the object. We suggest to run several times the code to find the best desired configuration.

- *"10 % x slice coordinate [m] for Fig. 4*
  10 % y slice coordinate [m] for Fig. 5
  10 % z slice coordinate [m] for Fig. 6
  10 % x slice coordinate [m] for Fig. 7
  10 % y slice coordinate [m] for Fig. 8
  10 % z slice coordinate [m] for Fig. 9"
  These six input lines define the position of the slice for the 3D visualization.

- 0 % Boolean variable to calculate the orbit (0 = no; 1 = yes)
  This variable is usually set to "zero", which means that no orbit is evaluated. If it is set to "one", the code computes the orbit according to the initial conditions set in the following lines.

- -3000 % Initial condition on the position - x component (x0) [m]
  3000 % Initial condition on the position - y component (y0) [m]
  3000 % Initial condition on the position - z component (z0) [m]
  15 % Initial condition on the velocity - x component (vx0) [m/s]
  0 % Initial condition on the velocity - y component (vy0) [m/s]
  0 % Initial condition on the velocity - z component (vz0) [m/s]
  Initial conditions for the evaluation of the orbit. It is important to keep the initial point location within the maximum coordinates of the probes.

- 1000 % Final time of integration [s]
  The final ending time of the integration for the ODE

- 35 % Number of steps for the ODE
  Number of steps at which the ODE is evaluated between t0 = 0 and tmax.

- 10 % Number of probes in the ODE

This is the number of probes used for the interpolation of the value of the gravitational acceleration at a generic location within the ODE. Since the code is not optimized for the evaluation of the orbit, we suggest a reduced number of probes (between 10 and 20).

Output of the code

The output of the code is made of figures and a MATLAB structure.

- 9 figures (10, if the Boolean value for the orbit evaluation is set to 1) – representing the original STL file, the one filled with spheres and the gravity field module or direction - and a Matlab structure (data_probes). In the structure is present all the information about g_x, g_y, g_z that can be later used by the user to calculate the orbital motion around the body. In this structure *"x_rand"*, *"y_rand"* and *"z_rand"* represent the 3D location of the probes in the rectangular box around the STL object. The x, y, and z component of the gravitational acceleration is reported in the field *"g_x_point"*, *"g_y_point"* and *"g_z_point"* respectively.

- Output structure: the structure is made by one or three fields, depending on the Boolean value for the orbit evaluation.

    - data_probes (always present, regardless the Boolean value of the orbit):
        - x_min, x_max, y_min, y_max, z_min, z_max : minimum and maximum coordinate of the probes
        - x_rand, y_rand, z_rand: location of the points where the gravitational fiel is evaluated
        - g_x_point, g_y_point, g_z_point: x, y, z components of the gravitational acceleration vector
    - T_orbit, Y_orbit (present if the Boolean value of the orbit is one): vectors containing the time, location and velocity that form the orbit of the object

External subroutines used by the software and implemented by different authors

Function: stlVolume
Author: K. Suresh; suresh@engr.wisc.edu
Reference: Krishnan Suresh (2021). Volume of a surface triangulation (https://www.mathworks.com/matlabcentral/fileexchange/26982-volume-of-a-surface-triangulation), MATLAB Central File Exchange. Retrieved April 15, 2021.

Function: Inpolyhedron
Author: Sven Holcombe
Reference: Sven (2021). inpolyhedron - are points inside a triangulated volume? (https://www.mathworks.com/matlabcentral/fileexchange/37856-inpolyhedron-are-points-inside-a-triangulated-volume), MATLAB Central File Exchange. Retrieved April 15, 2021.