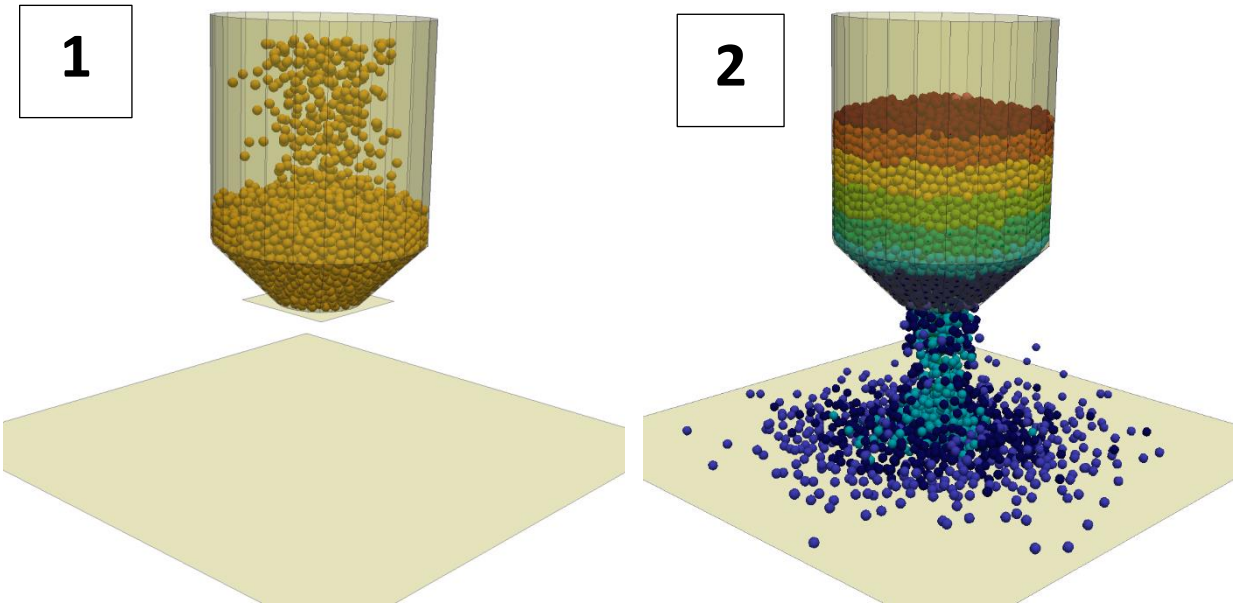
cemfDEM® tutorial

Three: Conical Hopper



Flow pattern of particles in the discharge of a conical hopper using DEM

Compatible with	gfortran-4.9, intel Fortran 2013, intel Fortran 2015, gfortran-7.5

Author

Hamidreza Norouzi

Amirkabir University of Technology

Center of Engineering and Multiscale Modeling of Fluid Flow

Extra consideration:

- This document is developed to teach how to use cemfDEM® software. The document has gone under several reviews to reduce any possible errors, though it may still have some. We will be glad to receive your comments on the content and error reports through this address: h.norouzi@aut.ac.ir

Document history

Revision	Description	Date
Rev1	The final version was prepared.	Dec - 15 - 2020
Rev0	The first version was drafted.	Dec - 14 - 2020

Table of Contents

1. Problem definition

The flow of spherical particles in a conical hopper with the cone angle 45° is studied. Density and size of particles are 2500 kg/m^3 and 9 mm, respectively. Diameter of the hopper is 20 cm and diameter of the exit orifice is 8 cm. A wide, flat wide plate is located 14 cm below the exit orifice on which a pile of discharged particles is formed. The simulation consists of two stages: the first stage is filling of hopper in which the exit gate is closed, the second stage is discharging the hopper in which the exit gate is removed and particles are allowed to exit the hopper under the gravity.

Note

Prerequisites: If you are not familiar with the code setup, you are recommended to read the first tutorial (Tutorial one: Packing of Mono-sized Particles), to learn how to define particles and their properties in the simulation.

2. Simulation setup

Download the simulation setup files from the github repository of the code or copy them from "./tutorials/conicalHopper" folder into the main root of the source code. Here, you first learn how to define the geometry of the conical hopper and the plate underneath. Then you will learn how to set-up charging particle, tagging particles, and discharging steps.

2.1. Geometry

Some code lines of file ProgramDefinedGeometry.f90 are shown in Program Listing 1. The following components comprise the geometry in the simulation:

- Cylindrical shell (line 14): The ends radii are 0.1 m and axis end points are (0, 0, 0) and (0, 0, 0.2).

- Conical section (line 18): The radius of one end is 0.04 m (exit orifice) and the other end is 0.1 m. The end points of cone axis are (0, 0, -0.059) and (0, 0, 0).
- The exit gate (line 27): This is a plane which blocks the orifice. This wall is removed during the discharge stage.
- Flat plate for pile formation (line 34): This is a horizontal, square plane with edge length of 0.5 m which is located 0.14 m under the exit orifice.

Program Listing 1: Some code lines of file ProgramDefinedGeometry.f90 for creating the conical hopper with exit gate and flat plate for pile formation.

```
13      !// shell of the hopper, radius = 0.1 m
14      res =  cyl%CreateCylinder( 0.1_RK, 0.1_RK, p_line( real3(0.0, 0.0, 0.0),
    real3(0.0, 0.0, 0.2) ),24, 1, 1 )
15      call Geom%add_Cylinder( cyl )
16
17      !// Conical section of the hopper with a 8-cm exit orifice
18      res =  cyl%CreateCylinder( 0.04_RK, 0.1_RK, p_line( real3(0.0, 0.0, -0.059),
    real3(0.0,0.0,0.0) ),24, 1, 1 )
19      call Geom%add_Cylinder( cyl )
20
21      !// the gate to block the exit orifice, this will be removed after packing stage
22      !// the user_id of this wall is 2. This id will be used to remove the wall later
    in the simulation
23      p1 = real3(-0.05, -0.05, -0.059);
24      p2 = real3( 0.05, -0.05, -0.059);
25      p3 = real3( 0.05,  0.05, -0.059);
26      p4 = real3(-0.05,  0.05, -0.059);
27      call Geom%add_PlaneWall( p1, p2, p3, p4, 2, 1, .true. )
28
29      !// flat plate under the hopper to collect particles for pile formation
30      p1 = real3(-0.25, -0.25, -0.2);
31      p2 = real3( 0.25, -0.25, -0.2);
32      p3 = real3( 0.25,  0.25, -0.2);
33      p4 = real3(-0.25,  0.25, -0.2);
34      call Geom%add_PlaneWall( p1, p2, p3, p4, 3, 1, .true. )
35
```

2.2. Filling and discharge

Some code lines of main.f90 are shown in Program Listing 2. Up to line 79, all the necessary components of the DEM system are created (they are not explained here for the sake of brevity).

In line 79, the DEM system with insertion plane is created. The time step is 0.00001 s and number of time steps for inserting particles is 150,000. This means that all particles are inserted within 1.5 s real time. The simulation has two main stages: filling and discharge.

Filling stage: In line 83, program iterates for about 2 seconds. This includes 1.5 s for particle insertion and 0.5 s to allow particle completely settle down under the gravity. To track the flow pattern of particles, we mark them layer by layer based on their vertical position in the hopper. This is done by invoking User_prtclMark method in line 86 (this will be explained later).

Discharge stage: The exit gate is removed in line 94 and the discharge stage is started. Program iterates for 10 s to allow particle exit the hopper under the gravity.

Program Listing 2: some code lines of main.f90 for conical hopper simulation

```
76      // . . . some code lines . . .
77
78      ! initializes DEM system, dt = 0.00001, particles are inserted in 150k
    iterations with initial velocity of 0.8 m/s
79      call DEM%Initialize( 0.00001_RK, PSDP, insPlane ,150000, 0.8_RK ,geom, Property,
    minDomain, maxDomain, DEM_opt )
80
81
82  !//// iteration loop for 1.99999  seconds
83      call DEM%iterate(199999)
84
85  !// marks particle layers
86      call DEM%User_prtclMark()
87
88  !// iterates one more time step
89      call DEM%iterate(1)
90
91
92  !/// removes the exit gate
93  !/// the user_id of the exit gate is 2.
94      call geom%delete_wall(2)
95
96
97  !/// iteration for 10 seconds
98      call DEM%iterate(1000000)
99
```

2.3. User_prtclMark procedure

Invoking User_prtclMark procedure from DEMSystem class, it invokes the following procedure which is defined in file User_Mark.f90 for every particle in the system. This function passes id, flag (the state of particle: being in domain, not inserted, or deleted), property type of particle and position of particle to this function and returns an integer value which is recorded (and saved) as a mark/tag for that particle until the next call of User_prtclMark. Every time that the program saves the results in an output file, the mark/tag of each particle is also saved (as usr_mark in vtk file) besides other variables of the particles. In Program Listing 3, the User_Mark function is illustrated. This function calculates the mark of particle based on the vertical position of particles. We marked particles in layers with thickness of 2.5 cm. The bottom part of the hopper is 0.059 m.

Program Listing 3: User_Mark function defined in User_Mark.f90

```
01  function User_Mark( id, flag, ptype, dpos ) result (mark)
02      use g_TypeDef
03      implicit none
04
05      integer(IK),intent(in):: id, flag, ptype
06      type(real4),intent(in):: dpos
07      integer(IK) mark
08
09      !// locals
10      real minx, maxx, dx
11      integer(IK) numx
12
13      ! marks layers of particles based on their z-coordinate
14      dx = 0.025
15      mark = (dpos%z + 0.059)/dx
16
17      return
18
19  end function
```

3. Running simulation (on Ubuntu)

Before building and running the simulation, make sure that all the required files and folders are in the main folder of the program. Then enter the following command in the terminal:

```
> make
> ./cemfDEM
```

This builds the executable file (if no error occurs) and then runs the simulation. Depending on the computational resources available on your PC, the execution may take between several minutes to some hours. Results of this simulation are visualized using ParaView®. Snapshots of the hopper during filling and discharge stages are shown in Figure 1.

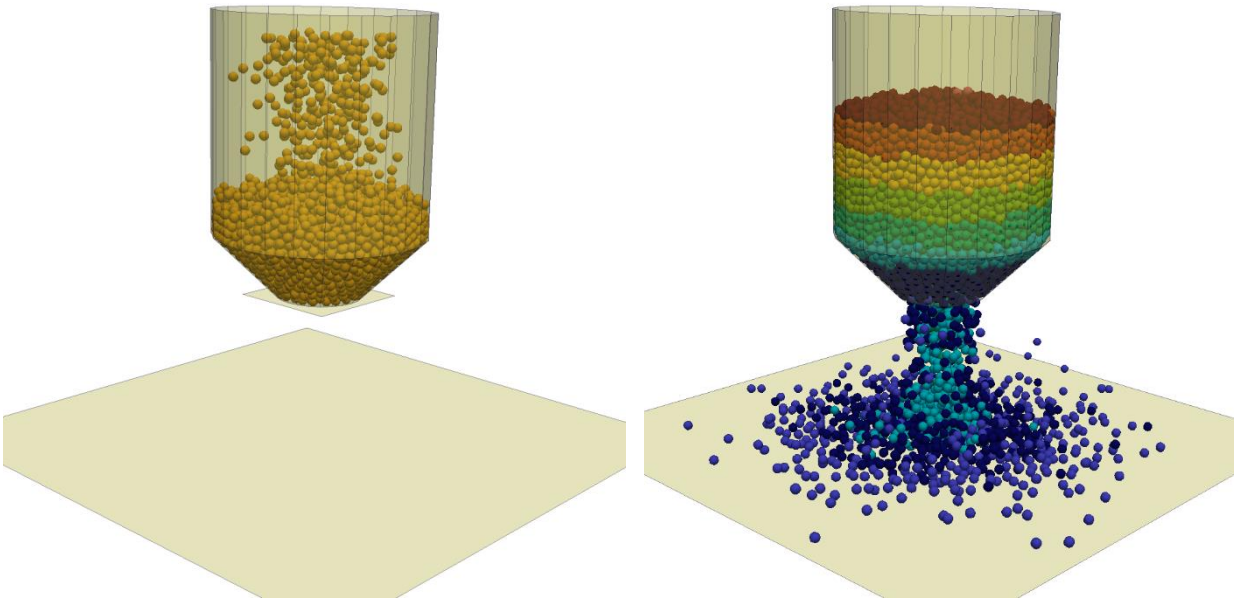


Figure 1: Snapshots of hopper during the filling stage (left) and discharge stage after removing the exit gate (right). In the discharge stage, particles are marked in layers.