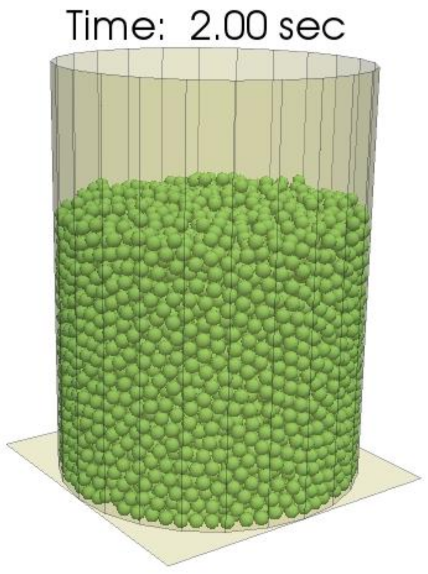
cemfDEM® tutorial

One: Packing of mono-sized particles



A discrete element method simulation to created a packed bed of mono-sized spheres

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

Document history

Revision	Description	Date
Rev1	The final version is prepared.	Dec - 08 - 2020
Rev0	The first version is drafted.	Dec - 01 - 2020

Table of Contents

1. Problem definition

In this simulation, we create a random packed bed of particles in a cylindrical container with the diameter of 15 cm and length of 20 cm. To create this bed, 8000 7-mm spherical particles with the density of 2500 kg/m^3 are used. In the followings, we describe how to set-up a simulation case using the cemfDEM program.

2. Simulation setup

Download the simulation setup files from the github repository of the code or copy them from "./tutorials/packingMono" folder into the main root of the source code. Some code lines of main.f90 are shown in Listing 1. Setting-up the code consists of some steps that are described below.

2.1. Options and general settings

Options and settings (in lines 23-32) are specified using `DEM_opt` of type `DEMS_Options` (defined in g_Prtcl_DefaultValues.f90 file). This object will be used when initializing the DEM system in the program. This section is explained in the followings:

- Line 23: the gravity acceleration is defined, which is -9.8 m/s^2 in y direction.
- Line 24: the frequency of saving results (particle and geometry) as the output files is specified. Based on this line, the program saves the results in output files once per 1000 time steps.
- Line 25: this line defines an optional run name for the simulation. Name of all output files and log-file are preceded by the run name.
- Line 26, the result directory is set to "Results". All output files and log-file are saved in the result directory. Please note that in all the simulations, the result directory is "Results", although this can be changed by the user. It should be further noted that the result directory should be created manually by the user and the program does not create it if it does not exist.
- Line 27: format of the output files is specified. Here, VTK and Tecplot® formats are set. The VTK format can be opened and processed by Paraview®.
- Line 28: the contact torque model is set in this line. At the moment, there are two torque models implemented in the program, constant torque model and variable torque model, which are selected by `CTM_ConstantTorque` and `CTM_VariableTorque`, respectively.

- Line 29: the contact force and its associated friction models are selected in this line. There are four different types of such a combination in the program: linear model with limited tangential overlap, linear model with non-limited tangential overlap, non-linear model with limited tangential overlap and non-linear with non-limited tangential overlap, which can be selected by `CFT_LSD_l`, `CFT_LSD_nl`, `CFT_nLin_l` or `CFT_nLin_nl`, respectively.
- Line 30: the contact search method is selected in this line. There are two contact search methods for non-sized systems, NBS and Munjinza, which are selected by `CSM_NBS` and `CSM_NBS_Munjiza`, respectively. Note that these models also can be used for polydisperse systems. Moreover, two contact search methods are developed based on hierarchical method which can be selected by `CSM_NBS_Hrchl` and `CSM_NBS_Munjiza_Hrchl` for polydisperse systems.
- Lines 31 and 32: methods of integration of linear and rotational equations of motion are specified. Here, the third order Adams-Bashforth combined with the fourth order Adams-Moulton method (predictor-corrector type) is selected. A list of 15 different integration methods which are implemented in the program can be found in the file g_Prtcl_DefaultValues.f90.

Listing 1: some code lines of file main.f90 for packing of mono-sized particles simulation

```
20  !//// Initial settings
21
22      ! options and settings for DEM simulation
23      DEM_opt%gravity   = real3( 0.0, -9.8 , 0.0 )
24      DEM_opt%SaveFreq  = 1000
25      DEM_opt%RunName   = "PackingMono"
26      DEM_opt%Res_Dir   = "./Results/"
27      DEM_opt%OutputFileType = OP_Type_VTK +  OP_Type_Tec
28      DEM_opt%CT_model  = CTM_ConstantTorque
29      DEM_opt%CF_Type   = CFT_LSD_l
30      DEM_opt%CS_Method = CSM_NBS_Munjiza !_Hrchl !
31      DEM_opt%PI_Method = PIM_AB3AM4
32      DEM_opt%PRI_Method= PIM_AB3AM4
33
34      !// properties of particles and walls
35      Y = 1000000.0_RK ! Young's modulus
36      pr = 0.23_RK     ! Poisson's ratio
37      call prop%set_prop(2500.0_RK, Y , Y/(2*(1+pr)), pr , 0.0_RK)
38
39      ! mono-sized particles, 8000 particles with size of 7 mm.
40      allocate(PSD)
41      PSD  = PS_Distribution( 8000 , PSD_Uniform, 1 , 0.007_RK, 0.007_RK )
42
43      ! associates particle size distribution with property
```

```
44      allocate(PSDP)
45      PSDP = PSD_Property( PSD , prop )
46      deallocate(PSD)
47
48  !//// main components of DEM system
49
50      !//// geometry
51      allocate(geom)
52      call ProgramDefinedGeometry(geom)
53      !//// Property
54      allocate( Property )
55      call Property%ParticleProperty( PSDP )    ! particles
56      call Property%WallProperty(1 , (/prop/) ) ! walls
57      call Property%PP_BinaryProp(DEM_opt, 0.2_RK, 0.1_RK, 0.8_RK, 0.8_RK) ! binary pp
58      call Property%PW_BinaryProp(DEM_opt, 0.3_RK, 0.1_RK, 0.8_RK, 0.8_RK) ! binary pw
59      !//// DEM system with particle insertion
60      ! simulation domain
61      minDomain = real3(-0.08, -0.01,  -0.08)
62      maxDomain = real3( 0.08 , 0.21 ,  0.08)
63      ! insertion plane
64      p1 = real3(-0.05, 0.18, -0.05)
65      p2 = real3(-0.05, 0.18,  0.05)
66      p3 = real3( 0.05, 0.18,  0.05)
67      p4 = real3( 0.05, 0.18, -0.05)
68      res = insPlane%CreateWall_nv(p4,p3,p2,p1)
69
70      ! initializes DEM system, dt = 0.00001 s, insert in 130 k iterations with initial
71  velocity of 0.2 m/s
72      call DEM%Initialize( 0.00001_RK, PSDP, insPlane, 130000 , 0.2_RK ,geom, Property,
    minDomain, maxDomain, DEM_opt )
73
74  !//// iteration loop for 2.5  seconds
75      call DEM%iterate(250000)
```

2.2. Particles and properties

A property set is created in lines 35-37 and will be used for particles and walls (different properties for particles and walls can be specified, if required). The Young's modulus, Poisson's ratio and density of material are set to 1,000,000 Pa, 0.23 and 2500 kg/m3, respectively.

In line 41, 8000 particles of the size 0.007 m are created in the `PSD` object. In line 45, the particles are associated with the property set in the `PSDP` object. Using the property set and the diameter of particles, mass and inertia of particles are calculated.

Main components of the DEM system are: geometry, physical properties of particles and walls, and particles information. In line 52, the geometry of the problem is created. This will be explained later in this tutorial. Properties of wall and particles are specified in lines 54 - 58. Specifying the properties has four steps which should be followed in sequence as shown in these lines. The particle properties are first specified by invoking `ParticleProperty` procedure in line 55. The wall properties are specified using a vector of property sets. In this problem, we consider similar property type for all walls and hence one property set should be considered here. Properties including dynamic and rolling friction factors and coefficients of normal and tangential restitution are the binary properties that should be defined for particle-particle and particle-wall contacts. These are specified in lines 57 and 58. The arguments of these procedures are `DEM_Options`, dynamic friction, rolling friction, coefficient of normal restitution and coefficient of tangential restitution, respectively.

One way to create the DEM system from its main components (particles, geometry and properties) is to specify an insertion plane for particles through which particles are inserted into the simulation domain with a pre-defined rate and velocity. In Line 72, the DEM system is initialized using its main component and an insertion plane (`insPlane`). Based on this statement, particles (8000 particles) are inserted into the simulation domain in 130,000 time steps with an initial velocity of 0.2 m/s. The time step for integration is 0.00001 s. After initializing the DEM system, the program is ready to iterate. Line 75 performs 250,000 iterations which is equivalent to 2.5 seconds of simulation. During this period, particles are inserted into the simulation domain in the first 1.3 seconds and they are allowed to settle down in the rest of the simulation. The interface of Initialize procedure is as follows:

Listing 2: Interface of Initialize procedure defined for DEMSystem type.

```fortran
subroutine Initialize(this, dt , Particles, Ins_Plane, num_steps, ins_vel ,Geom, Property, minDomain, maxDomain , DEM_opt )
    implicit none
    class(DEMSystem)                         this
    real(RK),                intent(in):: dt          ! time step
    class(PSD_Property),     intent(in):: Particles   ! Particles to be inserted
    class(PlaneWall),        intent(in):: Ins_Plane   ! Insertion plane
    integer(IK),             intent(in):: num_steps   ! number of steps for particle insertion
    real(RK),                intent(in):: ins_vel     ! insertion velocity of particles
    class(Geometry),pointer,intent(in):: Geom        ! Geometry object
    class(PhysicalProperty),pointer,intent(in):: Property  ! Property object
    type(real3),             intent(in):: minDomain, maxDomain   ! corner points of simulation domain
    type(DEMS_Options),      intent(in):: DEM_opt               ! DEM options
```

```
dt          : time step for integration
Particles   : object that contains particle data (diameter, property type etc.)
Ins_Plane   : insertion plane
Num_steps   : number of time steps during which particles should be inserted
Ins_vel     : insertion velocity of particles, the direction of velocity vector is normal
vector of the insertion plane.
Geom        : geometry object which contains information of all walls
Property    : property object which contains properties of particles and walls
minDomain, maxDomain: the corner points of the hexagonal simulation domain for DEM system
DEM_opt     : contains general settings for setting-up different parts of DEM system
and adjusting the output behavior of the program.
```

2.3. Geometry

The container is an empty cylinder with one open end. Based on the features of this program, this container can be created using an empty cylindrical shell and a plane at the bottom. The geometry is created by calling `ProgramDefinedGeometry` subroutine defined in file `ProgramDefinedGeometry.f90`. Some code lines of this subroutine are shown in Listing 3. At line 17, a cylindrical shell with radius 7.5 cm and length of 20 cm is created. The axis of this cylinder starts at point `(0, 0, 0)` and ends at point `(0, 0.2,0)`. Program decomposes the cylinder shell into some plane walls. In this command, the number of divisions is 24 and the `user_id` and property type of this shell are 1.

Listing 3: some code lines of file ProgramDefinedGeometry.f90 for creating the cylindrical container.

```fortran
14      ! Creates a cylindrical shell with radius 7.5 cm and length of 20 cm whose main axis is y-axis.
15      ! the property type is 1
16      ! the user_id of this shell wall is 1
17      res =  cyl%CreateCylinder( 0.075_RK, 0.075_RK, p_line( real3(0.0, 0.0, 0.0), real3(0.0,0.2,0.0) ),24, 1, 1 )
18      call Geom%add_Cylinder( cyl )
19
20      ! a plane with width of 15 cm with normal vector of (0,1,0).
21      ! this plane is placed at the bottom of the cylinder shell.
22      ! the property type is 1
23      ! the user_id of this plane is 1
24      p1 = real3( -0.075,   0, -0.075)
25      p2 = real3( -0.075,   0,  0.075)
26      p3 = real3(  0.075,   0,  0.075)
27      p4 = real3(  0.075,   0, -0.075)
28      call Geom%add_PlaneWall( p1, p2, p3, p4, 1, 1 )
```

The interface for the `CreateCylinder` procedure is as follows:

Listing 4: Interface of the CreateCylinder procedure defined in the g_CylinderWall.f90 file.

```fortran
function CreateCylinder (this, rad1, rad2, axis, nw, user_id, prop_type, both) result (res)

    implicit none
    class(CylinderWall) this
    real(RK),         intent(in) :: rad1, rad2 ! radii of the cylinder
    type(p_line),     intent(in) :: axis       ! axis of the cylinder
    integer(IK),      intent(in) :: nw         ! number of divisions
    integer(IK),      intent(in) :: user_id, prop_type
    logical,optional,intent(in) :: both       ! both side active status
    logical res

end function
```

```
rad1, rad2 : end points radii of the cylinder
axis       : cylinder axis with start and end points
nw         : number of divisions in the cylinder shell (increasing this number
results in a smoother surface for the shell)
user_id    : an user-defined id for the wall
prop_type  : property type of the wall
both       : determines if both sides of the wall is active (true)
```

A plane wall is created using four non-collinear points which should lie on a plane. The active side of the wall is determined using right-hand role. In lines 24 to 28 of Listing 3, the points are defined and the plane wall is created with `user_id` and property type 1. The interface for `add_PlaneWall` procedure is as follows:

Program Listing 5: Interface of Initialize procedure defined for DEMSystem type.

```fortran
subroutine add_PlaneWall(this, p1, p2, p3, p4 , user_id ,prop_type, both  )

    implicit none
    class(Geometry) this
    type(real3),intent(in) :: p1, p2, p3, p4 ! corner points
    integer(IK),intent(in) :: user_id, prop_type
    logical,intent(in)     :: both           ! both side active status
    optional both

end subroutine
```

```
p1, p2, p3, p4 : corner points of the plane
user_id        : an user-defined id for the wall
prop_type      : property type of the wall
both           : determines if both sides of the wall is active (true)
```

When creating walls, an important parameter is the property type (it starts from one). This shows the property set number that is considered for that wall. In this example, only one property type is considered for two walls. Therefore, the property type for two walls was set to 1. The number of property types of the wall, when creating the geometry, should be equal to the number of property sets passed to `WallProperty` procedure in line 56 of Program Listing 1.

3. Running the program (on Ubuntu)

Before building the code and running it, make sure that all the required files and folders are in the main folder of the code, as shown in Figure 1. The folder “src” contains all source code files of the program. The folder “Results” is there for output files. The program saves all output files in this folder. Set-up files of simulation, `main.f90`, `ProgramDefinedGeometry.f90` and `User_Mark.f90`, located in the main root of the program, are necessary for all simulations. The file makefile is provided to facilitate compiling and building the program. For building and running the program, change the current directory to the main folder of the code in the terminal and enter the following command:

```
> make
```

This compiles all the required source codes and creates the executable file in the current directory. The name of executable file is `cemfDEM` by default, although it can be changed to other valid names by user in the makefile. To run the simulation, enter the following command:

```
> ./cemfDEM
```

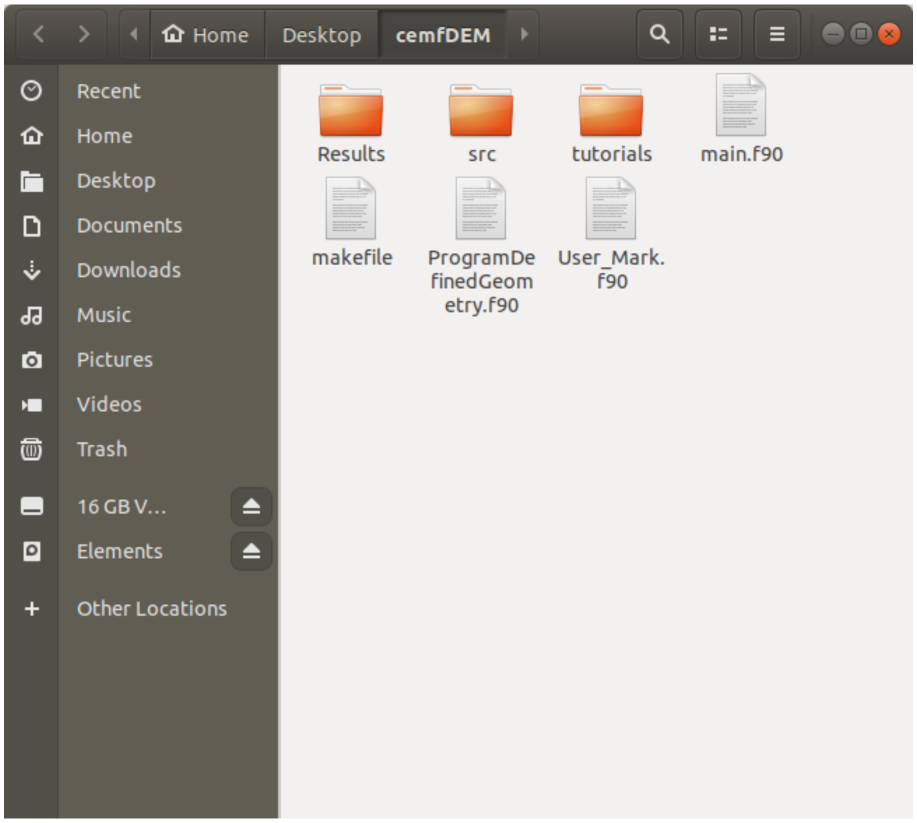


Figure 1: The main folder of program when is ready to be built and run

Depending on the computational resources available on your PC, the execution may take between several minutes to some hours. The results of this simulation are visualized using ParaView®. Three snapshots of the container during filling process are shown in Figure 2.

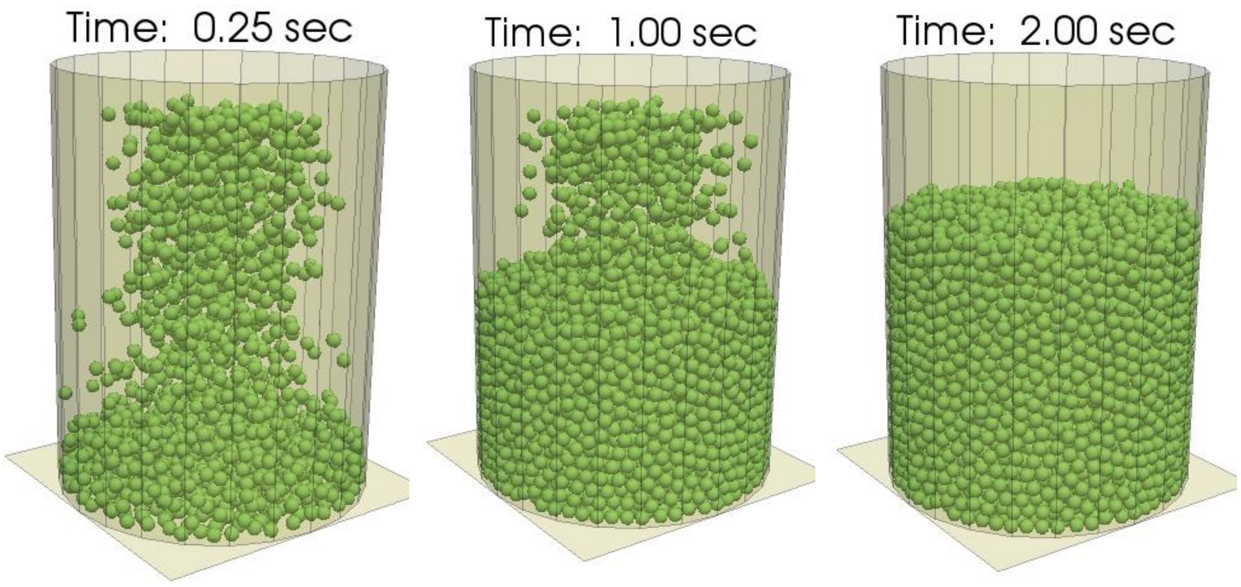


Figure 2: Packing of spherical particles

Hints
For cleaning the build and all intermediate files, the following command should be used: $> make clean
All the output files in “Results” folder can be removed by the following command: $> make vtkClean It cleans all the files with vtk and plt suffix in the current and “Results” directories.
To change the compiler for building the code follow these steps: • Open the makefile • Find the line that starts with “FortC” and   ○ Set “FortC = gfortran-4.9” or “FortC = gfortran-7.5” for gnu compiler   ○ Set “FortC = ifort” for Intel Fortran compiler • Save the makefile and re-build the program.