DEMLab User Guide

A. The basics

DEMLab is an open-source discrete code, developed in a matlab environment for pedagogical purposes. Its structure is very classical, but it allows for a number of evolutions, such as the implementation of new contact laws.

The main program is "DEMLAB.m". It makes a call to a given .m data file (for example "User_Simulation_Data.m"), and then runs the discrete modelling of the selected problem. Hence, this program should not be modified (apart from the name of the .m data file that is to be called).

The file "User_Simulation_Data.m" (another name may be used, if properly called from "DEMLAB.m") contains all the data necessary for the definition of a discrete problem. The following variables have to be defined:

1. Location and possible restarting

-"**Directory**" (string of characters): the name of the directory that will be used for the storage of the results and of the plots.

-"**Restart_Simulation**" (1 or 2): should be equal to 1 if the simulation is completely new, or to 2 if the user only wants to restart the computations from an existing saved state. In the case 1, all the following variables are useful (except "**Restarting_File**"), otherwise the only important variable will be "**Restarting_File**", and the other ones will be ignored.

-"**Restarting_File**" (string of characters): the full path of the file from which the computations will restart, if the variable "**Restart_Simulation**" has been set to 2. Otherwise, the variable is ignored.

2. Creating balls

-"**Balls_Origin**" (1-column cell variable, with a number of rows equal to the number of sets of balls to be introduced in the simulation): each value in this cell corresponds to a given set, and should be equal to 1 if the balls are to be generated automatically by the code, equal to 2 if they are to be defined by hand, or equal to 3 if they are to be taken from an existing simulation.

-"**Balls_Sets**" (1-column cell variable with a number of rows equal to the number of sets of balls to be introduced in the simulation ; each cell containing a 1-line array of dimensions 1*5): a given line is only accounted for if "**Balls_Origin**" has been set to 1 for this line. Then, each line of this cell defines a set of balls that has to be generated by the code, and contains the following variables:

$$[N_{balls}, R_{min}, R_{max}, Unit\_weight, Material\_number]$$

with N_{balls} the desired number of balls in the set, R_{min} and R_{max} the smallest and largest radiuses in the samples (the radiuses are sampled uniformly in the interval), *Unit_weight* the unit weight of the material composing the balls, and *Material_number* a given integer corresponding to the material of the set (it will be used later on when defining the contact laws between materials). N_{sets} such lines have to be written, if N_{sets} sets are desired (most often, one is enough)

-"**Xvertices**" (cell of dimensions N_{sets}*1): only accounted for if "**Balls_Origin**" has been set to 1. Each cell I_{set} contains the x-coordinates of the segments defining a region into which the set of balls I_{set} is to be generated. More precisely, a given cell of this variable contains an array of dimensions 1*($N_{vertices}$+1), with the following variables :

$$[x_{vertice1}, x_{vertice2}, ..., x_{verticen}, x_{vertice1}]$$

-"**Yvertices**" (cell of dimensions N_{sets}*1): only accounted for if "**Balls_Origin**" has been set to 1. Each cell I_{set} contains the y-coordinates of the segments defining a region into which the set of balls I_{set} is to be generated. More precisely, a given cell of this variable contains an array of dimensions 1*($N_{vertices}$+1), with the following variables :

$$[y_{vertice1}, y_{vertice2}, ..., y_{verticen}, y_{vertice1}]$$

The points defined by the variables "**Xvertices**" and "**Yvertices**" should form a proper polygonal region into which the balls are to be generated (and that should be true for any set defined in "**Balls_Sets**".

-"**Balls**" (1-column cell variable, with a number of rows equal to the number of sets of balls to be introduced in the simulation ; each cell contains an array of dimensions N_{balls}*5): a given line is only accounted for if "**Balls_Origin**" has been set to 2 for this set. This array defines manually the properties of the balls (it should only be used for very small samples). Each line defines a given ball, with the following variables:

[*Radius, X_coordinate, Y_coordinate, Unit_weight, Material_number*]

-"**Balls_File**" (1-column cell variable, with a number of rows equal to the number of sets of balls to be introduced in the simulation ; each cell contains a string of characters): a given line is only accounted for if "**Balls_Origin**" has been set to 3 for this set. This string contains the full path of the .m file from which the balls of this set are to be loaded.

3. Creating walls

-"**Walls_Origin**" (1 or 2): should be equal to 1 if the walls are to be defined by hand, or equal to 2 if they are to be taken from an existing simulation.

-"**Walls**" (array of dimensions N_{walls}*5): only accounted for if "**Walls_Origin**" has been set to 1. Each line of the array defines a given wall, with the following variables :

[X_{origin}, Y_{origin}, $X_{extremity}$, $Y_{extremity}$, *Material_Number*]

-"**Walls_File**" (string of characters): only accounted for if "**Walls_Origin**" has been set to 2. This string contains the full path of the .m file from which the walls are to be loaded.

4. Physics of the simulation

-"**LAWS**" (cell of dimensions N_{laws}*4): Defines the type and the parameters of the different contact laws used in the simulation. Each line of this cell variable contains 4 arrays, which altogether define a given interaction. The content of these arrays is :

{[*Material_1*], [*Material_2*], ["*Name_of_the_law*"], [*Parameters_of_the_law*]}

In these arrays, *Material_1* and *Material_2* are integers (corresponding to material numbers assigned to walls and/or balls in the previous stages), "*Name_of_the_law*" is a string of characters giving the name of the chosen law for these two materials, and *Parameters_of_the_law* is an array with the chosen set of parameters. Details on the available contact laws and corresponding parameters are available further in this user guide.

-"**GRAVITY**" (array of dimensions 1*2): A simple array providing the value of the gravity field along x and y, with the syntax :

[*Gravity_along_x, Gravity_along_y*]

5. Numerical parameters

-"**Time_ini**" (real number): the time at the beginning of the simulation

-"**Time_fin**" (real number): the time at the end of the simulation

-"**TIME_STEP**" (real number): the time step used in the Verlet scheme

-"**Neighbours_Parameter**" (real number): parameter used for the detection of proximity: is considered as a neighbour of a given grain i any particle j for which the separation distance is smaller than **Neighbours_Parameter***R_{max}, where R_{max} is the radius of the largest grain of the sample.

-"**Period_Neighbours**" (integer): number of time steps between two successive updates of the neighbours lists.

-"**Period_Monitoring**" (integer): number of time steps between two successive monitoring of the simulation (details on user-defined monitoring are provided further in this document).

-"**Period_Save**" (integer): number of time steps between two successive back-up of the state of the simulation on the hard-drive.

6. Display options

-"**Display_Time_Step**" (0 or 1): should be set to 1 if the number and duration of the time steps are to be displayed, and 0 otherwise. It is advisable to choose 0 if the time steps are very short, since this display might be quite time-consuming.

-"**Period_Plot**" (integer): number of time steps between two successive updates of the graphic plot of the simulation.

-"**Period_Print**" (integer): number of time steps between two successive back-up of the graphic plot of the simulation on the hard-drive.

-"**Plot_Type**" (1, 2, 3, or 4): indicates the choice of the type of graphic plot:
1. Balls are represented by hollow circles, no chain forces
2. Balls are represented by discs with a color code, no chain force
3. Balls are represented by hollow circles, chain forces are plotted with a color code
4. Balls are represented by discs with a color code, chain forces are plotted with a color code

-"**Rotation_Marker**" (0 or 1): should be equal to 1 is a rotational marker is to be plotted for each ball, and to 0 otherwise

-"**Colored_Quantity_Discs**" (integer between 0 and 4): defines the quantity represented by the colors of the plotted discs:
0. Neutral grey
1. Norm of the velocity of the ball
2. Norm of the angular velocity of the ball
3. Radius of the ball
4. Index (i.e. number) of the ball

-"**Discs_Quantity_Range**" (array of dimensions 1*2): defines the extreme values for the color scale of the discs, with the following syntax :

[*Minimum_value_in_blue, Maximum_value_in_red*]

-"**Colored_Quantity_Contacts**" (integer between 0 and 1): defines the quantity represented by the colors of the force chains:
0. Neutral grey
1. Norm of the corresponding contact force

-"**Contacts_Quantity_Range**" (array of dimensions 1*2): defines the extreme values for the color scale of the force chains, with the following syntax :

[*Minimum_value_in_blue, Maximum_value_in_red*]

-"**Maximum_Line_Width**" (real number): defines the width of the force chains corresponding to the largest value of the scale chosen for the contact forces (i.e. *Maximum_value_in_blue*).

-"**Plot_Axes**" (array of dimensions 1*4): defines the plotting window, with the following variables :

$[X_{min}, X_{max}, Y_{min}, Y_{max}]$

-"**Print_Resolution**" (integer): resolution of the hard-drive back-up of the plots, expressed in dots-per-inch.

B. Advanced controls

1. Updating walls

Any wall in the simulation can have a controlled motion, either in a predefined way or as a function of the forces it is submitted to. This motion is controlled by the means of the coordinates [X_{origin}, Y_{origin}] of the origin and [$X_{extremity}$, $Y_{extremity}$] of the extremity of each wall. Such coordinates are updated at each time-step following the user-defined commands coded in the m-file "User_Update_Walls.m". This function takes as input arguments the following variables :

-**WALLS** : current coordinates of the walls
-**WALLS_CONTROL** : some variable that may be needed to transfer information from a function to another (often an empty variable with no use at all)
-**FORCES_ON_WALLS** : list of the forces applied to each wall
-**RESULTING_FORCES_ON_WALLS** : resulting normal an tangential forces applied to each wall, and corresponding resulting torque.
-**TIME** : current time
-**TIME_STEP** : simulation time step

The detailed content of each of these variables is detailed further in this document. The output variables of the m-file are :

-**NEWWALLS** : updated coordinates of the walls
-**WALLS_CONTROL** : some variable (possibly updated) that may be needed to transfer information from a function to another (often an empty variable with no use at all)

Hence, the objective of this routine is to receive the current positions of the walls, and to update it (if needed), based on the current time (if their trajectory was decided from the beginning) and/or on the forces they are submitted to (if their trajectory is force-dependant). This is done in a coding zone available for users, into which the variable **NEWWALLS** should be appropriately modified. More specifically, each line of this array corresponds to a given wall of the system, and the following columns are to be modified :

-column 1: X_{origin}
-column 2: Y_{origin}
-column 5: $X_{extremity}$
-column 6: $Y_{extremity}$

A typical used of the variable **WALLS_CONTROL** is the case of servo-controlling of a wall for force-controlled boundary conditions. A proper feedback requires to smooth the force applied to the walls along time, which requires the knowledge of some past states of the system, and not only its current state. **WALLS_CONTROL** is then used to store (for example) the n last values of the force applied to the walls.

The last part of the m-file should not be modified by the user. It consists in automatically computing the velocity of the wall, based on the user-defined position update.

2. Setting initial conditions

The m-file "User_Initial_Conditions.m" is called only once by the master program, just prior to starting the simulation. It receives the totality of the variables of the problem as input variables, and delivers the same variables to the main program after possible modifications by the user. The detail of the content of all the variables of the problem is described further in this document.

3. Monitoring the simulation

A real-time monitoring of the simulation is possible at each time-step, using the m-file "User_Monitoring.m". This technique is to be used if some interesting data are to be stored at much smaller intervals than the back-up period : these data may then be stored without the need to write on the hard drive at high frequency. On the other hand, if there is no need to store data at high frequency, it is advisable to ignore this routine and to run a post-processing on the stored mat-files after the main simulation to limit its time-cost.

The "User_Monitoring.m" routine takes as input variables the totality of the variables of the problem at each time step. Monitoring is performed at a given period called **Period_Monitoring**, and the routine starts by checking that a monitoring should actually be done at this time step. If not, then it is ignored. A monitoring consists in adding a new line to the variable **MONITORED_VARIABLES**, which is an array into which data is stored along time. The first two columns of this array are respectively the time step and the time, and the other possible columns contain the data stored at this time step.

Basically, the role of the programmer is to choose what kind of data he/she wants to store every **Period_Monitoring** time steps, and to send it to the proper column in the line-array **NewStored**, which has as many columns as the array **MONITORED_VARIABLES**. The last part of the routine (which should not be modified) consists in adding **NewStored** at the end of **MONITORED_VARIABLES**.

4. Postprocessing

The program "User_Post_Process.m" is only to be used after the simulation. It operates on a desired set of mat-files (defined by the two integers in the array **Files_Indices**) that were saved every **Period_Save** time steps during the simulation. This program makes it possible to print the state of the system corresponding to each saved file (by having the variable **Print** set to 1 and by choosing the appropriate graphic options). It may also be used to run any user-defined post-processing of the data at each saved iteration.

C. Implemented contact laws

Currently, four contact laws are implemented in DEMLab.

1. Standard DEM explicit law

This law corresponds to the most classical one used in explicit DEM, with all the required parameters for a simple analysis. The keyword is **"standard"**, and the parameters are:

$$[k_n, k_t, \mu, \gamma_n]$$

-k_n : Normal stiffness
-k_t : Tangential stiffness
-μ : Friction coefficient
-γ_n : Normal damping coefficient

2. Simple cohesive law

This law defines a maximum attractive and shear force with a chosen distance of action. The keyword is **"Cohesion"**, and the parameters are:

$$[k_n, k_t, \gamma_n, F, d]$$

-F : Maximum attractive force applied between any pair of close enough particles
-d : Distance of action of the attractive force

3. Cohesive bonds

This law introduces a maximum attractive force with a chosen distance of action, which corresponds to a cohesive bond. At the initial state, before the beginning of the simulation, all the pairs of particles that are within a certain distance are considered as bonded (and an attractive force exist for such pairs). Then, during the simulation, these bonds may break if the grains get separated by a large enough distance. If so, these bonds will never be created again, and completely disappear. The contact law switches to **"Standard"**. This is the only difference with the previous law, for which the bonds are created again if the particles get close enough a second time. The keyword is **"Breakage"**, and the parameters are:

$$[k_n, k_t, \mu, \gamma_n, F, d]$$

4. Rolling resistance

This law superimposes to the standard law a rolling resistance. This resistance may be applied in two different ways. The first one is related to rolling damping, and corresponds to a torque opposing rolling, with a value proportional to the rolling friction. The second one is related to rolling friction, and requires a rolling stiffness (i.e. a torsional spring opposing a torque proportional to the rolling) and a threshold (i.e. a maximum value of this opposing torque, proportional to the normal force in the contact). The keyword is "**Rolling**", and the parameters are:

$$[k_n, k_t, \mu, \gamma_n, k_r, \mu_r, \gamma_r]$$

-k_r : Rolling stiffness
-μ_r : Rolling friction coefficient
-γ_r : Rolling damping coefficient

D. Data structures

-"**ACCELERATIONS**" (array of dimensions Nballs*3): contains the accelerations of the discs at the current time step, with the following variables in each line:

[*Acceleration_along_x, Acceleration_along_y, Angular_acceleration*]

-"**BALLS**" (array of dimensions Nballs*5): contains the properties of the discs, with the following variables in each line:

[*Radius, Material_number, Density, Mass, Rotational_Inertia*]

-"**CONTACTS**" (array of dimensions Ncontacts*17): contains the properties of the contacts existing at the current time step, with the following variables in each line:

[*First_contactor, Second_contactor, Type_of_second_contactor, X_contact, Y_contact, X_normal_vector, Y_normal_vector, X_tangential_vector, Y_tangential_vector, Normal_interpenetration, Normal_interpenetration_derivative, Tangential_spring_extension, Tengential_spring_extension_derivative, Rotational_spring_extension, Rotational_spring_extension_derivative, Contact_law_number, Contact_status*]

First_contactor and *Second_contactor* are the two contacting body. The first one is always a ball, the second one may be a ball or a wall (the variables *Type_of_second_contactor* will respectively take the value 1 or 2). The variable *Contact_status* indicates if a contact is open (0), slipping (1), adhesive in the Coulomb sense (2), bonded and not failed yet (3).

-"**FORCES_ON_BALLS**" (cell of dimensions Nballs*1): each line of this cell-variable corresponds to a given ball and contains an array of dimensions (Ncontacts_on_ball*9), with the following structure in each line:

[*Second_contactor, Type_of_second_contactor, X_contact, Y_contact, X_force, Y_force, Torque, Normal_Force, Tangential_Force*]

-"**FORCES_ON_WALLS**" (cell of dimensions Nwalls*1): each line of this cell-variable corresponds to a given wall and contains an array of dimensions (Ncontacts_on_wall*9), with the following structure in each line:

[*Second_contactor, Type_of_second_contactor, X_contact, Y_contact, X_force, Y_force, Torque, Normal_Force, Tangential_Force*]

-"**GRAVITY**" (array of dimensions 1*2): describes the field of gravity:

[*Gravity_along_X, Gravity_along_Y*]

-"**LAWS**" (cell of dimensions Nlaws*4): each line of this cell variable contains arrays which describes the interaction law between two materials, with the following structure:

{[*Material_number_1*], [*Material_number_2*], [*Law_name*], [*Law_parameters*]}

-"**MONITORED_VARIABLES**" (array of dimensions Nmonitorings*Nvariables): this array contains the data that has been stored so far by the procedures coded by the user in the User_Monitoring.m program. Each line correspinds to a given monitoring, with the following variables:

[*Time_step_number, Time, User_variable_1, User_variable_2, ..., User_variable_n*]

-"**NEIGHBOURS**" (cell of dimensions Nballs*2): Each line of this cell variable corresponds to a given ball of the simulation. The first cell of such line contains a column vector with the list of the neighboring balls, and the second cell contains a column vector with the list of the neighboring walls.

-"**POSITIONS**" (array of dimensions Nballs*3): contains the coordinates of the discs at the current time step, with the following variables in each line:

[*Position_along_x, Position_along_y, Angular_position*]

-"**RESULTING_FORCES_ON_BALLS**" (array of dimensions Nballs*3): contains the summary of the forces applied on each disc at the current time step, with the following variables in each line:

[*Force_along_x, Force_along_y, Torque*]

-"**RESULTING_FORCES_ON_WALLS**" (array of dimensions Nwalls*3): contains the summary of the forces applied on each wall at the current time step, with the following variables in each line:

[*Force_along_x, Force_along_y, Torque*]

-"**VELOCITIES**" (array of dimensions Nballs*3): contains the velocities of the discs at the current time step, with the following variables in each line:

[*Velocity_along_x*, *Velocity_along_y*, *Angular_velocity*]

-"**VELOCITIES_HALF**" (array of dimensions Nballs*3): contains the velocities of the discs at the next half time step, with the following variables in each line:

[*Velocity_along_x*, *Velocity_along_y*, *Angular_velocity*]

-"**WALLS**" (array of dimensions Nwalls*9): contains the kinematics of the walls at the current time step, with the following variables in each line:

[*X_origin*, *Y_origin*, *X_velocity_origin*, *Y_velocity_origin*, *X_extremity*, *Y_extremity*, *X_velocity_extremity*, *Y_velocity_extremity*, *Material_number*]