DEMPLA and OVAL: programs for analyzing particle assemblies and for simulating wave propagation and liquefaction with the Discrete Element Method

Matthew R. Kuhn,* kuhn@up.edu

April 13, 2026
DEMPLA version 0.2.5

1 Introduction

DEMPLA means "Discrete Element Method for Propagation and Liquefaction Analysis." The program includes the older code OVAL, which serves as the underlying back-engine for Discrete Element Method (DEM) simulations of individual particle assemblies or of the multiple assemblies along a one-dimensional soil column. The two programs, DEMPLA and OVAL, are included as two modes of the DEMPLA code. In Mode 1, the program OVAL (which is contained within the larger DEMPLA code) performs DEM analysis of a single assembly of particles. In the Mode 2, DEMPLA uses multiple DEM assemblies to analyze one-dimensional wave propagation within a soil column. The DEM is a method for simulating the motions and interactions of the individual particles in a granular material while the entire assembly is being deformed [1]. Perhaps most important, the code allows extracting global averages of such quantities as stress and fabric for the entire assembly as well as the micro-level quantities, such as particle movements and contact forces. The program is primarily intended for analyzing "dense" assemblies (i.e., jammed, as opposed to diffuse, assemblies) that are roughly rectangular in shape, such as in so-call *element tests*. In Mode 1, the code can also be used to create assemblies by compacting a diffuse assembly into a denser state. The program's capabilities in this mode are summarized in Section 3. In the Mode 1, the code runs the DEM code with a single computing thread; whereas, in Mode 2 the code can run in a parellel on multiple threads.

In Mode 1, the program handles both two- and three-dimensional simulations with particles that are either circles (2D), ellipses (2D), ovals (2D), spheres (3D), "nobby" clusters of circles (2D), "bumpy" clusters of spheres (3D), a special non-spherical "ovoid" particle (3D).

In Mode 2, the code uses 3D DEM assemblies as the representative volume elements (RVEs) within a one-dimensional soil column to analyze wave propagation and liquefaction

*Dept. of Civil Engineering, Donald P. Shiley School of Engineering, University of Portland, 5000 N. Willamette Blvd., Portland, OR 97203 USA, kuhn@up.edu.

in the column. In this mode, the code is a multi-phase, multi-scale rational method for modeling and predicting free-field wave propagation and for modeling the weakening and liquefaction of near-surface soils. The one-dimensional time-domain model of a soil column uses the discrete element method (DEM) to track stress and strain within a series of representative volume elements (RVEs), driven by seismic rock displacements at the column base. The RVE interaction are accomplished with a time-stepping finite-difference algorithm. The method applies Darcy's principle to resolve the momentum transfer between a soil's solid matrix and its interstitial pore fluid. Different algorithms are described for the dynamic period of seismic shaking and for the post-shaking consolidation period. The method can analyze numerous conditions and phenomena, including site-specific amplification, down-slope movement of sloping ground, dissolution or cavitation of air in the pore fluid, and drainage that is concurrent with shaking. Several refinements of the DEM are described for realistically simulating soil behavior and for solving a range of propagation and liquefaction problems, including the poromechanic stiffness of the pore fluid and the pressure-dependent drained stiffness of the grain matrix. The model is applied to the conditions of four sets of well-documented centrifuge studies. The verification results are favorable and highlight the importance of the pore fluid conditions, such as the amount of dissolved air within the pore water. In Mode 2, the several DEM algorithm can be run with multiple computing threads to analyze the many DEM assemblies (RVEs) in parallelized fashion.

In Mode 2, other than being restricted to one dimension, the method is quite general and allows one to model and understand diverse conditions and phenomena: (1) three-dimensional motions of rock and soil, propagating as both p-waves and s-waves; (2) nearly arbitrary stress and strain paths during ground shaking; (3) sloping ground surface with down-slope movement; (4) multiple stratigraphic soil layers; (5) sub-surface water table or complete submergence of the site; (6) sloping water table with down-dip seepage forces; (7) onset and depth of liquefaction; (8) saturated soil, dry soil, or quasi-saturated soil with entrained air at a specified saturation; (9) dissolution or cavitation of entrained air; (10) effect of saturation on wave propagation and liquefaction; (11) effect of drainage that is concurrent with shaking; (12) pore fluid with a specified viscosity; (13) site-specific amplification of surface accelerations relative to those of the rock base; (14) pressure-dependence of wave speed and the slowing of waves as they approach the surface; (15) dilation and the coupling of s-waves and p-waves; (16) voids redistribution and development of a water film beneath less permeable layers; (17) softening of soil during shaking, with a slowing of wave speeds and shifting of frequency content due to build-up of pore fluid pressures; (18) post-shaking consolidation and settlement; (19) post-consolidation reshaking and post-triggering behavior; and (20) relative rise of the water table during and after shaking. The authors has not fully investigated most of the possibilities.

The code is freely available on GitHub. If you find opportunities to improve the code or if you find errors, please let me know or fork the code and augment or correct the code yourself. The author compiles the Fortran code on BSD Unix and Gnu Linux systems, although the code can likely be compiled on windows systems, too.

Besides the source code, the GitHub repository also contains sample assemblies, example problems, and Octave code for analyzing results.

of the License, or (at your option) any later version. These programs are distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place—Suite 330, Boston, MA 02111-1307, USA.

2 Contents

3 Capabilities and limitations

3.1 Capabilities and limitations of the code

The its current version the code has the following capabilities and limitations:

1. In Mode 1, DEMPLA can perform discrete element analysis on assemblies containing any one of the following particle types: circles (2D), ellipses (2D), ovals (2D), spheres (3D), "nobby" clusters of circles (2D), "bumpy" clusters of spheres (3D), and non-spherical "ovoids" (3D).

2. In Mode 2, only the 3D shapes can be used.

3. At present, DEMPLA works best with flat periodic boundaries on all sides of a two- or three-dimensional assembly. For two-dimensional assemblies (in Mode 1), it can also use flexible boundaries; with both two- or three-dimensional assemblies, it can use rigid boundaries. It can only analyze assemblies with parallel boundaries (i.e. assemblies of 2D parallelogram and 3D parallelepiped shapes).

4. The code is not intended for boundaries of arbitrary shape or for arbitrary boundary conditions.

5. The code is intended for dense (jammed) particle assemblies that are slowly deformed at quasi-static rates. Although the code can also model rapid flows and diffuse assemblies and can create a dense assembly by densifying an initially sparse assembly, these purposes are not a primary concern in the code's design.

6. In Mode 1, DEMPLA can create output files of both the macro-results and the micro-results for the DEM assemblies. The macro-result files include information on stress, deformation, pore pressure, etc. The micro-result files give information on particle positions, particle orientations, contact forces, and system topology.

7. In Mode 2, the program can create the same output files of macro-results for each of the DEM assemblies that comprise the RVEs of the soil column, as well as average assembly values of both the grain matrix and pore fluid for all DEM assemblies along the height of the soil column.

8. DEMPLA can employ a number of contact models, including a simple contact force mechanism consisting of linear springs (both normal and tangential) and a frictional slider. The normal and tangential spring constants can be specified independently (Sections 11.1.22 and 11.1.23). The frictional slider can be "turned off" (Sections 11.1.24 and 11.1.25).

9. DEMPLA also includes more advanced contact models: Hertz-Mindlin models of sphere-sphere contacts, cone-cone contacts, and contacts of intermediate shape (surfaces of revolution of power-form contours). With these models, the grains' shear modulus, Poisson ratio, and friction coefficient must be specified.

10. The code can analyze assemblies with a poromechanic description of the pore fluid. As such, the code directly computes the pore fluid pressure as well as the inter-granular (effective) stress, and it can adjust the inflow of pore fluid as well as the deformation of the grain matrix.

11. Available poromechanic models include dry assemblies, saturated assemblies, and quasi-saturated assemblies in which gas is presents as entrained bubble that do not form a continuous network within the pore space separate from the pore liquid.

12. The following pore fluid characteristics can be specified: saturation, bubble size or number density, gas solubility, surface tension, liquid bulk modulus, etc.

13. DEMPLA includes the option of a modified DEM algorithm for self-regulating and maintaining the quasi-static nature of a simulation (Sections 11.1.2, 13.3.5, 13.3.15, and 19).

14. The code includes the following types of damping: translational mass (global) damping, rotational mass (global) damping, contact damping (refer to [1]), and the damping method of Potyondy and Cundall. Each may be independently specified or "turned off" (Sections 11.1.28–11.1.30).

15. Arbitrary sequences of stress-controlled and strain-controlled (and arbitrary combinations of the two) steps can be modeled. DEMPLA includes a robust servo-control algorithm for controlling the boundary stresses. The deformation (or stress) path is supplied by the user in a series of steps, which specify in a component-by-component manner either the deformation rate tensor or stress rate tensor. The specified stress or deformation rate components may be mixed. See Section 11.2.

16. With stress-control, either the effective stresses or total stresses can be controlled in arbitrary combinations.

17. When a pore fluid is modeled, either the fluid pressure or the fluid inflow can be controlled. This capability is necessary for analysis wave propagation and liquefaction in Mode 2.

18. DEMPLA can create binary "restart" files that allow a new simulation to begin at the exact ending condition of a previous simulation. These restart files include all of the position, velocity, and contact information that allow the new run to begin where the previous run had stopped. See Sections 11.1.7, 11.1.8, and 11.2.9.

19. The code can assign initial random velocities to the particles in the assembly (Sections 11.1.31 and 11.1.32).

20. As an option, DEMPLA can prevent particle rotations, particle motions, or both.

21. In Mode 2 DEMPLA is limited to a one-dimensional column of soil. Other than this restriction, DEMPLA is quite general and allows one to model and understand diverse conditions and phenomena:

- three-dimensional motions of rock and soil, propagating as both p-waves and s-waves;
- nearly arbitrary stress and strain paths during ground shaking;
- sloping ground surface with down-slope movement;
- multiple stratigraphic soil layers;
- sub-surface water table or complete submergence of the site;
- sloping water table with down-dip seepage forces;
- onset and depth of liquefaction;
- saturated soil, dry soil, or quasi-saturated soil with entrained air at a specified saturation;
- dissolution or cavitation of entrained air;
- effect of saturation on wave propagation and liquefaction;
- effect of drainage that is concurrent with shaking;
- pore fluid with a specified viscosity;
- site-specific amplification of surface accelerations relative to those of the rock base;
- when using one of the Hertz-type contact models, pressure-dependence of wave speed and the slowing of waves as they approach the surface;
- dilation and the coupling of s-waves and p-waves;
- voids redistribution and development of a water film beneath less permeable layers;
- softening of soil during shaking, with a slowing of wave speeds and shifting of frequency content due to build-up of pore fluid pressure;
- post-shaking consolidation and settlement;
- post-consolidation reshaking and post-triggering behavior; and
- the relative rise of the water table during and after shaking.

3.2 Things to do

Please help write this section.

4 Availability and installation

The code is written in a rather inelegant dialect of Fortran 77. The entire program and other files can be downloaded from the GitHub repository, including the source code, this documentation, sample assemblies, and example problems. Source code is freely available under terms of the open source GNU General Public License (GPL), version 2.

The program can be run in two modes, with each mode requiring several input components:

1. In Mode 1 (stand-along simulation of a single particle assembly), the following input files are required:

   - The compiled executable file of DEMPLA, created as described below in an unparallelized form.
   - A RunFile that specifies the contact model, the poromechanic model (if any), the general conditions of the simulation, and the sequence of stress-strain control steps to be performed by the simulation.
   - A StartFile that gives the size of the assembly, the numbers of particles in the assembly, and the locations, sizes and orientations of all particles in the assembly.

2. In Mode 2 (wave propagation and liquefaction analysis of a one-dimensional soil column), several input files are required. Many of these files must be arranged within a particular directory (folder) structures, described later.

   - The compiled executable file of DEMPLA, as described below. The source code should be compile with the OpenMP parallelization library, as described below. Shell variables should also be set to permit multi-thread running of DEMPLA, as described below.
   - A GFile that specifies the number of RVEs (DEM assemblies) in the soil column, the height of the column, the number of stratigraphic soil layers, water depth, and other general characteristics of the simulation.
   - One LFile for each of the stratigraphic soil layers, giving the RunFile and StartFile for setting up and running the DEM algorithm for the stratigraphic layer.
   - A RunFile for each stratigraphic layer that specifies the contact model, the poromechanic model, the general conditions of the simulation, and the sequence of stress-strain control steps to be performed in setting up the initial assemblies in the layer.
   - A StartFile for each stratigraphic layer that gives the size of the assembly, the number of particles in the assembly, and the locations, sizes and orientations of all particles in the assembly.
   - A MotionFile that gives the displacement sequence (in three directions) at the base of the soil column, thus specifying the base seismic motion.

4.1 Compiling Dempla on Linux systems

The following steps are required to create a compiled executable DEMPLA file and to set up multi-thread running of the code (Mode 2, only). If you are only running the program in Mode 1 (for a single assembly), only steps 1 and 2 are required, and steps 3 through 6 can be ignored. The author has compiled the code with both the `gfortran` and `ifort` compilers. If you compile with other compilers, please augment the descriptions below with your own experience. Before compiling, you should read Section 4.4 to be sure that you are compiling for a sufficient number of particles, number of RVEs, number of stratigraphic layers, etc.

1. Download or pull the following three files from the GitHub repository:

   - `dempla-X.X.XX.f`, the latest version of the main DEMPLA source code.
   - `common-dempla-X.X.XX.f`, the corresponding file of the common blocks used throughout the DEMPLA code.
   - `param-dempla-X.X.XX.f`, the corresponding file of parameters that sets the sizes of arrays in the DEMPLA code.

   These three files should be placed in the same build directory (folder). For example, the three files could be placed in a `source/` directory, as in the GitHub repository.

2. DEMPLA can be parallelized by compiling with the proper OpenMP flag, which for the gcc `gfortran` and Intel `ifort` compilers are as follows:
   `gfortran`: `-fopenmp`
   `ifort`: `-qopenmp`
   To compile with optimization in the Linux shell, move to the directory (folder) with the source files (e.g. `/dempla/source/`), and enter the following command (because the compile commands are cumbersome, you might consider creating an `alias` within the Linux `.bashrc` configuration file):

```
gfortran -std=gnu -O3 -mtune=native \
                  -mcmodel=large \
                  -fopenmp \
                  -o dempla.exe \
                  dempla-X.X.XX.f
```

   or

```
ifort -ipo -O3 -no-prec-div -fp-model fast=2 -xHost
      -fpconstant \
      -mcmodel=large \
      -qopenmp -heap-arrays \
      -o dempla.exe \
      dempla-X.X.XX.f
```

   The options `-std=gnu -O3 -mtune=native` and `-ipo -O3 -no-prec-div -fp-model fast=2 -xHost` produce optimized code.

   The options `-fltconsistency -fpconstant` improve floating point precision.

   Option `-mcmodel=large` is necessary for large compiled arrays (as might be needed when there are millions of particles or when there are thousands of particles but with the contact history parameter `mlistJ` being exceptionally large).

   The options `-fopenmp` and `-qopenmp` specifiy compiling with the OpenMP library. These options are not required with Mode 1 simulations.

   Option `-heap-arrays` places temporary arrays that are used within subroutiens (i.e., arrays that are treated as private and take new values with each threaded instance) onto the heap instead of the stack.

   Finally, option `-o dempla.exe` specifies the name of the executable file.

To preclude parallelization and always run the code as a single thread (for example, with Mode 1), simply recompile dempla without the `-fopenmp` or `-qopenmp` flags.

Before compiling DEMPLA, you will need to consider the values of various parameters that specify the sizes of arrays. These parameters are described and given in the file `param-dempla-X.X.XX.f`. The most important parameters are `mp` (the maximum number of particles in a DEM assembly) and `mrve` (the maximum number of DEM assemblies (RVEs) in the soil column, when running in Mode 2). For example, if you are analyzing a soil column and wish to have 30 DEM assemblies in the column, you will need to set `mrve=30` in the file `param-dempla-X.X.XX.f` before compiling.

Another important parameter is `mlistJ` (also see Section 4.4). If an advanced Hertz–Mindlin contact model is used (i.e., a Jäger model, with `imodel=6, 7` or `9`, see Section 11.1.12), then `mlistJ` gives the length of arrays that store the contacts' histories. If made too small, DEMPLA will crash with an error message advising a larger value of `mlistJ`. If another, simpler contact model is used, `mlistJ` can be set equal to 1, and a much smaller executable file will be created, possibly avoiding memory overloads for smaller computers.

3. In order for parallelization to work within a Linux shell (i.e., a terminal window), you should specify the number of threads to be used during runtime inside of the shell. The number of threads is given by the shell environment parameter `OMP_NUM_THREADS`. Within a Linux bash shell:
   ```
   export OMP_NUM_THREADS=4
   ```
   which would enable parallelized runs with 4 threads. The authors typically uses 10 threads on a 28-thread Xeon processor. This step must be done within *every* shell (terminal) within which the program is run, or it can be included in the `.bashrc` file or in another shell initialization script that can be sourced when needed. Note that if no value of `OMP_NUM_THREADS` is set, then the code will possibly run with all possible threads on the CPU, which might not be desirable.

   You can query the current value of the environment parameter with this command:
   ```
   echo $OMP_NUM_THREADS
   ```
   To reset the `OMP_NUM_THREADS` parameter, within a Linux bash shell:
   ```
   unset OMP_NUM_THREADS
   ```
   To query the number of threads available on the computer's hardware:
   ```
   lscpu | grep -E '^Thread|^Core|^Socket|^CPU\('
   ```
   See this guide for servers with multiple sockets, as with Xeon and EPYC processors: https://software.intel.com/en-us/forums/intel-moderncode-for-parallel-architectures/topic/739878.

   Instead of setting the parameter `OMP_NUM_THREADS` in the shell, the parameter can also be assigned at runtime. For example, if the executable code is `dempla.exe`, the following can be run in the bash shell:
   ```
   OMP_NUM_THREADS=4 ./dempla.exe
   ```
   when 4 threads are desired.

4. For parallelized code, you might need to increase the stack size to accommodate the multiple threads. For example,

```
export OMP_STACKSIZE=80m
```
The default size of `OMP_STACKSIZE` is usually small, 1m or 2m, and can lead to segmentation fault errors during runtime. You might need to try different sizes, until an adequate stack size is found, but without exhausting the machine memory. See this explanation:
https://stackoverflow.com/ questions/13264274/ why-segmentation-fault-is-happening-in-this-openmp-code

The parameter can also be assigned at runtime. In the bash shell:
```
OMP_NUM_THREADS=4 OMP_STACKSIZE=80m ./dempla.exe
```

5. One must provide sufficient memory for parallel execution. The `gfortran` and `ifort` compilers create executable programs that use stack memory instead of static memory. The limit on stack memory size will need to be increased. With Linux, use the following shell command:
```
ulimit -s unlimited
```
This command must be entered in any shell that will run the executable program, or it can be included in the .bashrc or other shell initialization script. Without this command, you will likely receive a segmentation fault error.

6. Finally, because of the many niggling steps that are required to prepare a Linux shell to run parallelized DEMPLA, you might consider placing all of these commands in a single Linux bash script file that can be sourced within a terminal. I use the following file `dempla.bsh` for setting up the use of DEMPLA in a bash shell:
```
#!/bin/bash
#
ulimit -s unlimited
export OMP_STACKSIZE=80m
export OMP_NUM_THREADS=10
```
but you will want to customize your own script and place it somewhere within your shell's `$PATH`.

To run the script in a shell, it must be sourced(!!) within the shell:
`source dempla.bsh` This must be done in each shell within which DEMPLA is run (or the same commands could be placed in the `.bashrc` file, so that they are automatically run when a new shell is opened. Note, be sure to source the `.bashrc` file itself, after it has been changed.).

4.2 macOS systems

Please help to write this section.

4.3 Windows systems

Please help to write this section.

4.4 Source code parameters

The source file `param-dempla-X.X.XX.f` contains Fortran parameters that give the sizes of various arrays in the code. For example, the parameter `mp` is currently set at `12000`, which sets the maximum number of particles in the DEM assemblies. The actual number of particles in an assembly is the input value `np`, given in the StartFile (usually a DFile), as explained in Section 12.1.2. However, the value of `mp` is a hard limit on `np`. If your assembly contains more particles than `mp` (that is, if `np`>`mp`), then DEMPLA will crash. If `np`>`mp`, then change the value of `mp` in the `param-dempla-X.X.XX.f` file, and then recompile the code (Section 4.1).

Other parameters include the following:

- `lc1`: the maximum number of segments of the deformation–stress path that are given in a RunFile (see Section 11.2).
- `mrve`: the maximum number of RVEs in the soil column of a Mode 2 wave propagation analysis (see `nRVEs`, Section 7.1.3).
- `mlayer`: the maximum number of stragigraphic soil layers in a Mode 2 wave propagation simulation (see `nLayers`, Section 7.1.4).
- `mRock`: the maximum length of the MFile that contains the base motion of a Mode 2 wave propagation analysis (see Section 9).
- `mlistJ`: when the contact model is a full Jager-type model (`imodel`= 6, 7, and 9, see Section 11.1.12), a complete history of each contact's history must be maintained, so that the friction force can be computed in the next DEM time step. `mlistJ` gives the lengths of the arrays that are used for maintaining this history. When the lengths of these arrays are exceeded, the DEMPLA will crash, with this error message:

  ```
  Insufficient space in array listJ in subroutin Jager3D
  ```

  This error usually occurs during a quiescent period of deformation for the DEM assembly. (When the assembly is being actively deformed, then the history of a contact will be frequently reset). This problem can usually be solved by shortening or removing the quiescent segment of the deformation–stress history (see Section 11.2).

  If this fails, then increase the parameter `mlistJ` in the source file `param-dempla-X.X.XX.f`, and then recompile.

There are many other pararameters, which are listed in the `param-dempla-X.X.XX.f` file.

5 Helpful/essential utilities

Please help to write this section.

6 Running Dempla

DEMPLA is *not* yet an interactive program with a graphical user interface. At present, it runs only in a batch mode.

You would normally run DEMPLA (in either of the two modes) from within a terminal (e.g. an Linux xterm window, the MacOS terminal utility, or DOS/Windows console) with the following command at the system prompt:

```
<path><dempla_executable_name>
```

where `<path>` is the path to your executable file with its `<dempla_executable_name>` (e.g. `dempla.exe` or whatever name or symbolic link is given to the file). For example,

```
~/dempla/source/dempla.exe
```

(Also, with Linux, if the executable is in the current directory, you may need to use the path "`./`", if the current directory (`./`) is not included in the environment variable `$PATH`, as specified in the user's `/.bashrc` file.) Instead of explicitly providing the `<path>`, you may wish to create a link (shortcut) to the `dempla` executable file (say, `dempla.exe`), place the file it in a directory in your system's search `$PATH`, or modify your system's search `$PATH` to include the directory that contains the `dempla` executable file.

After starting the `dempla` command, the verison of DEMPLA will be shown, and you will be queried for the run mode:

```
Running Dempla, version  dempla-0.2.6.f

Mode: single-assembly (1) or wave propagation (2):
```

After entering the mode, you will be queried for various names and information, which will depend on the mode that was selected.

6.1 Running Dempla in Mode 1 — analyzing single assemblies

In Mode 1 (`imode=1`, single-assembly mode), you will be queried for the names of two files,

```
Name of the RunFile:

Name of the StartFile:
```

and the simulation will proceed to run. We will henceforth refer to these two files with the generic names "RunFile" and "StartFile", which are described in Section 11 and Section 12. These two files must be created before running DEMPLA. The file names of the RunFile and StartFile must conform to Unix conventions (no spaces, asterisks, question marks, etc.) The RunFile must reside within the directory from which the DEMPLA is run. The StartFile can be located in another directory, in which case, you must enter the full path of the StartFile.

The output will normally be written to a set of files, whose names will contain the core RunFile name. The various output files are listed in Table 1 and will be described later in the documentation. In Mode 1, the output files are all placed within the directory in which DEMPLA is run.

As an example, the output file `A<RunFile>.txt` will be created, noting that the name of this output file contains the core RunFile name. The file is created in the same director from which the DEMPLA program is run. This "`A`" file is a text file that can be imported into a spreadsheet, such as Excel or Gnumeric to view the average stresses and deformations of the assembly (note, *imported*, not opened). The file can also be read into Octave or Matlab with a utility function.

Before describing the detailed contents of the RunFile and the StartFile, you should know that the RunFile is an ASCII (text) file that describes how the simulation is to be run: boundary deformation rates, boundary stress rates, friction coefficients, spring constants, etc. (see Section 11). A StartFile gives the number of particles, particle type, and the initial particle arrangement (sizes, positions, etc., see Section 12). A StartFile may be of either ASCII or binary type (Table 1).

6.2 Running Dempla in Mode 2 — wave propagation analysis

Before you can run DEMPLA in Mode 2 (i.e. to simulate wave propagation), you must create a special directory (folder) structure that will hold the various input files and output files. This directory structure must be created inside of the directory in which DEMPLA is being run (note that you can create a sybolic link, symlink, to the executable `dempla.exe` file within this directory. With Linux, `ln -s <path to dempla.exe> dempla.exe`). Three of these directories and their contents are as follows:

- A directory named `MotionInput`. This directory contains the input motion files, which give the base (rock) motion for the simulation. A MotionFile is a text file, having the format described in Section 9. The directory `MotionInput` can either be a directory with that name or a symbolic link (with the name `MotionInput`) targeting a directory that contains the MotionFile. The directory `MotionInput` can contain multiple MotionFiles, for running different simulations.

- A directory named `RunFiles` that contains all of the RunFiles that are used in the simulation. Each stratigraphic layer will have a RunFile. Each RunFile should reside inside this directory (folder) named `RunFiles`. A single RunFile is a text file with contents and format described in Section 11. The RunFile describes the conditions under which the assemblies within the layer will be initialized and run: contact models, friction coefficients, spring constants, poromechanic fluid properties, etc. The name of a RunFilemust conform to Unix conventions (case-sensitive, no spaces). The directory `RunFiles` can either be a directory with that name or a symbolic link (with the name `RunFiles`) targeting the directory that contains the RunFiles.

- A directory named `StartFiles` that contains all of the StartFiles that are used in the simulation. A StartFile is a text file with contents and format described in Section 12. Each StartFile is a text file (a DFile for Mode 2 simulations) that gives information about the particle assembly of a stratigraphic layer at the start of a simulation: assembly size, number of particles, and particle sizes, locations, and orientations. Each stratigraphic layer will have a StartFile. Each StartFile should reside inside the folder `StartFiles`. The name of a StartFilemust conform to Unix conventions (case-sensitive, no spaces). The directory `StartFiles` can either be a directory with that name or a symbolic link (with the name `StartFiles`) targeting a directory that contains the StartFiles.

The above three directories have a general role, since many simulations can reference these same directories, with the simulations being run inside of the directory that contains the

three directories (i.e., the directory in which DEMPLA is being run). As such, multiple MotionFiles, RunFiles, and StartFiles can be contained in their respective directories (folders).

Besides the three directories, you must also create a BaseName directory (not necessarily with the name "`BaseName`") that will contain input information for (and all output from) simulations on a particular soil column, noting that different input motions can be applied to the same soil column (that is, the same BaseName directory can contain the input and output for multiple base motions that are run with the same soil column). The name of the BaseName directory which must conform with Unix naming conventions (case-sensitive, no spaces).

You will run a simulation from within a main directory within your directory structure, say the directory "`MySimulations`", for example. The BaseName directory should be created inside of this directory. You will likely be running simulations on different soil columns, and you must create a BaseName directory for each of the soil columns, inside of the directory from which you are running the simulations (for example, inside of your "`MySimulations`" folder). Within the main directory, you must create access to the three directories that were described above: the `MotionInput` directory, the `RunFiles` directory, and the `StartFiles` directory. The three directories can be inside the main directory, or alternatively, you can create this access with symbolic links (symlinks). With Linux, the latter is done with the following:

```
cd <path to main directory from which you will run simulations>
ln -s <path to MotionInput> MotionInput
ln -s <path to RunFiles> RunFiles
ln -s <path to StartFiles> StartFiles
```

The main directory must also contain the BaseName directories (and its input files and sub-directories), with one BaseName directory for each soil column, as described in the next paragraph.

After creating the BaseName directory, you must place some input files (GFiles and LFiles) within the BaseName directory:

1. *The remainder of this paragraph in brackets [] does not apply for* DEMPLA *versions 0.2.5 and higher, since the three sub-directories are created when* DEMPLA *is run. For these versions of* DEMPLA*, the three sub-directories are not required before running* DEMPLA. [The BaseName directory must contain three sub-directories with the following names: `02_LayerSetup`, `03_RVESetup`, and `04_RunOutput`. These three sub-directories will be used to hold pre-processing files and the output files that are created by the simulations.]

   While running DEMPLA, various output files will be placed inside the sub-directory `04_RunOutput`, and these output files will be given names that contain the BaseName and the name of the MotionFile, so that the output sub-directory can contain output files from multiple simulations, each with a different name. The various output files contain pore fluid pressures, soil displacements, effective stresses, etc. The range, format, and content of the various output files are described in Section 14.

2. The BaseName directory must contain a file that gives general information about the

soil column: the number of DEM assemblies (RVEs) along the height, the number of stratigraphic layers, the column height, water depth, etc. This ColumnFile must have the name G<BaseName><suffix>, noting the "G" prefix. As such, we refer to this file with the interchangeable generic names ColumnFile and GFile. The format and content of GFiles are described in Section 7. The name of the GFile can include an optional suffix (several simulations can be run with the same BaseName and input motion, but can differ by the pore fluid saturation, and these variations can be distinguished with the suffix). See the input prompt for the suffix.

3. The BaseName directory must also contain files that give information about each of the stratigraphic soil layers that are within the soil column. One LayerFile must be created for each of the soil layers (note that the water table can be within the thickness of a soil layer, but only one LayerFile is required for the layer). The LayerFile must have the name L<XXXX>_BaseName. The name must begin with letter "L", followed by a 4-digit number (beginning with 0001 for the top soil layer at the ground surface, and increasing with 0002, 0003, etc. in increasing depth, to the bottom layer). The number is followed by an underscore "_" and the BaseName.

   We will refer to this file with the interchangeable generic names LayerFile and LFile. The format and content of LFiles are described in Section 8. The LFile gives the thickness of the layer, void ratio, etc. Most important, the LFile also gives the names of the RunFile and StartFile for the layer.

   The RunFile for a layer provides general information about the DEM assembly that represents the particular stratigraphic layer. The RunFile must be contained in the RunFiles directory (i.e., one of the three general sub-directories, described earlier). A RunFile is a text file with contents and format described in Section 11. The RunFile describes the conditions under which the assemblies within the layer will be initialized and run: contact models, friction coefficients, spring constants, poromechanic fluid properties, etc. The names of RunFiles must conform to Unix conventions.

   The StartFile for a layer gives information about the size of the DEM assembly that represents the soil layer, along with the shapes, locations, sizes, and orientations of all of the particles in the assembly. A StartFile is a text file with contents and format described in Section 12. Each stratigraphic layer will have a StartFile. Each StartFile should reside inside the folder StartFiles. The names of StartFiles must conform to Unix conventions (case-sensitive, no spaces).

To run DEMPLA in Mode 2 (multi-assembly wave propagation), start the program and enter 2 as the mode (i.e. imode=2). You will be queried for the following three names,

```
BaseName of the simulation (main folder name):

Suffix of variation (if none, press return):

MotionFile (name of rock motion file):
```

and the simulation will proceed to run.

File name	Type	Function	Sections
A<RunFile>.txt	text	macro-results for spreadsheets	13.1
B<RunFile>	text	macro-results for text editors	13.2
C<RunFile>	binary	restart StartFile	11.1.7
C?<RunFile>	binary	restart StartFile	11.2.9
D<RunFile>	text	StartFile	12
F[1-4]?<RunFile>	text	micro-results for post-analyses	13.5
Results_<etc.>	text	results of Mode 2 simulations	14
? – a letter that corresponds to the deformation-stress path in which the file was created (Section 11.2.9			

Table 1: DEMPLA output files

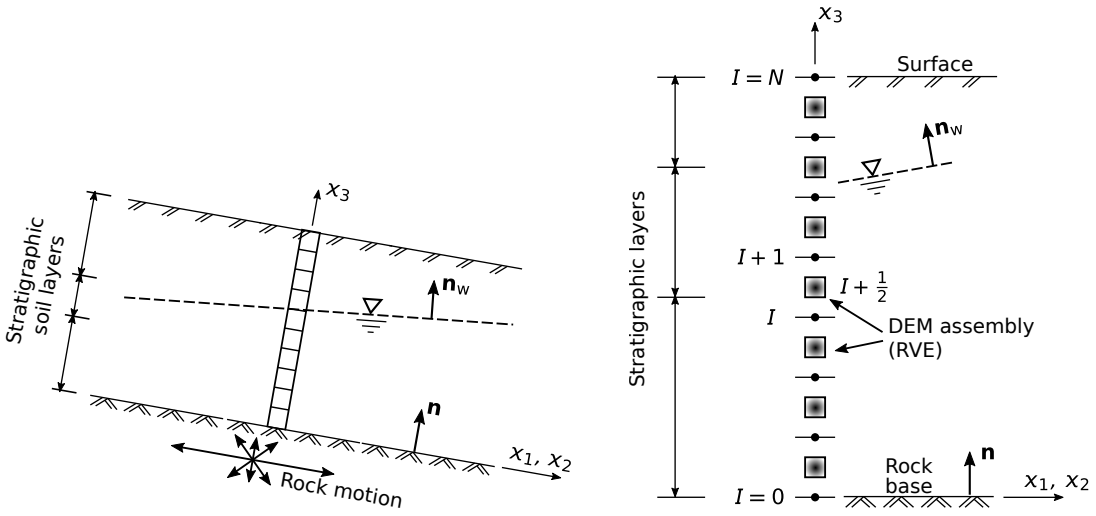


Figure 1: Soil column in Mode 2. RVEs and Nodes are number from the bottom upward. The stratigraphic layers are numbered from the top (ground surface) downward.

7 GFiles (Column files)

GFiles (or ColumnFiles) describe the general conditions of the one-dimensional soil column that is being used for simulating wave propagation and liquefaction. If you are running DEMPLA in Mode 1 (to analyze a single granular element, not an entire soil column), you can skip this section — a GFile is not needed.

A schematic soil column is shown in Fig. 1. Every simulation of wave propagation requires a single GFile, which must have a name of the form G<BaseName><suffix>, as described on page 20. This file must be placed inside the directory BaseName, which must be located in the parent directory from which DEMPLA is run. A GFile is a formatted ASCII text file (*not* a Word file!), which means that the input information must be placed within certain rows and columns (or column ranges). Sample GFiles can be downloaded from the GitHub site, and an example of a GFile is shown in Fig. 2. I suggest that you use this file as a template for creating your own GFiles.

```
The file of general information for this simulation
    108          : gvers       | version of this file type
     17          : nRVEs       | number of RVEs in the soil column, integer
      1          : nLayers     | number of stratigraphic layers
      0          : iABfile     | whether to create A & B files in dempla run
      1          : iPreProcess | is it necessary to do pre-processing
      0.483d0    : BaseDepth   | depth of rock below ground surface, m
      1.000d0    : WaterDepth  | depth of water below ground surface, m
      8.00d-3    : visc_w      | viscocity of water, Pa*s
     18.5d-6     : visc_a      | viscocity of air, Pa*s
    998.23d0     : rho_w       | water density, kg/m^3
      1.18d0     : rho_a       | air density, kg/m^3 (free air above ground)
    196.140d0    : grav        | acceleration of gravity, m/s^2
      1.290d0    : centrifuge  | intertial mass factor for centrifuges
      1.0d-6     : ddef_target | strain-step during a dempla DEM step
      1          : nDempla_out | interval for dempla results
     20          : nDEM_out    | interval for DEM results during dempla
      0.50d0     : C_factor    | adjustment to the dempla Courant number
      0.         : Direct_soil | ground slope direction (CCW from x1 toward x2)
      0.         : Dip_soil    | ground and rock dip
      0.         : Direct_water| water table slope direction
      0.         : Dip_water   | water table dip
    200          : nTiltSteps  | DEM time steps to create slope
    200          : nTiltEquil  | DEM time steps to equilibrate slope
      0.001d0    : t_PostShake | post-shaking quiescent period of no rock motion
    200.d0       : t_Setl      | post-shaking settlement (consolidation) time

************ Additional Layer Information ************
   lSat =   a special saturation when S_o=1?  (0,1,2)
   xSat =   adjustment to p_wcav when S_o=1
   k_geot = hydraulic conductivity, m/s (Q = k*i*A)

lSat|   xSat   |  k_geot |iCap|   xCap  |
----|---------|---------|----|---------|
  0   0.       0.0034d-2   0       0.
```

Figure 2: An example GFile (column file). Because there is only one layer (`nLayers=1`), the file ends with a single line of additional information about the one layer.

A GFile file is arranged in three parts:

1. a general information section consisting of 26 lines, including a comment line (see Section 7.1). The data in these lines have the formats of either `i16` or `f16.7`, beginning in the first column of each line. That is, each line contains a single input value at the beginning of the beginning 16 columns of the line.

2. eight spacer lines of comments. These lines are ignored.

3. for each layer, one line of additional information is given about the pore fluid and permeability of the layer (see Section 7.2) The GFile should include one of these lines per layer, (i.e., per LFile).

Although the format specifier `f16.7` is used, Fortran allows the input data to be in either fixed (`F`) or exponential (`E` or `D`) formats with any number of significant digits, *provided that the data fits within the first 16 columns of each line.*

7.1 GFile: General information section

7.1.1 `comment` (\)

The first line of a GFile is ignored. Put some useful information here.

7.1.2 `gvers` (i16)

An integer number giving the version of the GFile. This variable is included so that current files will be back-compatible with future versions of the DEMPLA code. The current version is 108, and `108` should be place in the second row of the GFile.

7.1.3 `nRVEs` (i16)

The integer number of DEM assemblies (RVEs) in the soil column (integer N in Fig. 1). In the paper [2], this is the value N. For the simulations in the paper, `nRVEs` varied from 17 to 20. RVEs are numbered from 1 to `nRVEs`, from the bottom upward. Nodes are numbered from 0 to `nRVEs`, from the bottom upward.

The value of `nRVEs` must not exceed the parameter `mrve` in the source code file `param-dempla-0.X.X.f`, where "`X`" is the version number. The current value is 20. If you need an `nRVEs` greater than 20, then you must change the parameter file and recompile the code (Section 4.1).

7.1.4 `nLayers` (i16)

The number of stratigraphic layers within the soil column (Fig. 1). Note that one LFile is required for each stratigraphic layer (see Section 8). Layers are numbered from 1 to `nLayers`, from the top downward.

The value of `nLayers` must not exceed the parameter `mLayer` in the source code file `param-dempla-0.X.X.f`, where "`X`" is the version number. The current value is 100. If you need an `nLayers` greater than 100, then you must change the parameter file and recompile the code (Section 4.1).

7.1.5 `iABfile` (i16)

When running DEMPLA, the program creates a number of output files: AFiles (Section 13.1), BFiles (Section 13.2), FFiles (Section 13.5), and ResultsFiles (Section 14). The AFiles and BFiles require more storage space than other files, since one AFile and one BFile are created for each RVE. To conserve disk space, `iABfile` is set to `0`, and AFiles and BFiles are not created. With any other integer value, the files will be created.

7.1.6 `iPreProcess` (i16)

Before running the DEMPLA algorithm, some preprocessing is required:

- The program must use the information of each stratigraphic layer to create the particle contact list and other information for the progenitor DEM assembly of the layer. The StartFile information about each layer is stored as DFiles for the layers (one for each layer), since the StartFiles are likely DFiles (Section 12).
- The program must consolidate the progenitor DEM assemblies of the layers to create the RVEs within the layer. These DEM assemblies (RVEs) are consolidated to the initial conditions of geostatic and hydrostatic pressures at the various RVE points. These pre-processed assemblies are stored as CFiles for later use at the start of the wave propagation algorithm.
- The program must determine the stiffness moduli of each DEM assembly, so that the wave propagation time step can be established (i.e., the time step Δt in [2]). These moduli are stored as a file in the directory `03_RVESetup`, so that they can be retrieved later.

These pre-processing steps can take a lot of time, and they must only be done once for a given soil column (for example, once created, the same initial soil column can be subjected to a number of input motions. If the pre-processing steps have already been performed, then a value for `iPreProcess` of `0` will skip the pre-processing of future runs. Any other integer value of `iPreProcess` will cause the pre-processing to be performed. The pre-processing must be performed at least once.

7.1.7 `BaseDepth` (f16.7)

The height of the soil column.

7.1.8 `WaterDepth` (f16.7)

The depth of water below the ground surface. For a dry soil soil column, give a value of `WaterDepth` that is greater than `BaseDepth`. For a submerged soil column, give a negative value of `WaterDepth`. For a water table between the ground surface and the rock base, give a positive value between 0 and the `BaseDepth`.

7.1.9 visc_w (f16.7)

Viscosity of the pore fluid (usually water, 0.0010 Pa·s) for soil below the water table. This viscosity and the geotechnical hydraulic conductivity `k_geot` are used to calculate the water permeabilty having units of $m^2s^{-1}Pa^{-1}$ that is used in the Biot equations. Note that in some of the examples of centrifuge studies, the water was chemically treated, so that the viscosity of water was much greater than that of untreated water (the latter being about 1 mPa·s).

7.1.10 visc_a (f16.7)

Viscosity of the pore fluid above the water table (usually air, about 18.5×10^{-6} Pa·s). This viscosity and the geotechnical hydraulic conductivity `k_geot` are used to calculate the air permeabilty having units of $m^2s^{-1}Pa^{-1}$ that is used in the Biot equations.

7.1.11 rho_w (f16.7)

Density of the pore fluid (usually water, 998 kg/m^3) for soil below the water table.

7.1.12 rho_a (f16.7)

Density of the pore fluid (usually air, about 1.18 kg/m^3) for soil above the water table.

7.1.13 grav (f16.7)

Acceleration of gravity (about 9.807 m/s^2). Note that in some of the examples of centrifuge studies, the acceleration is greater than that of gravity.

7.1.14 centrifuge (f16.7)

This value is usually 1.0. However, in centrifuge studies, the mass of the centrifuge box must be included with the soil mass. In this case, the input `centrifuge` is multiplied by the soil density to account for the box mass.

7.1.15 ddef_target (f16.7)

When running the wave propagation algorithm, time is advanced in increments Δt (see [2]), and strains within the DEM assemblies are advanced by an increment $\Delta\varepsilon$ during Δt. However, when running the DEM algorithm for each RVE, the increment $\Delta\varepsilon$ can be achieved in several DEM time steps, so that DEM strain is advanced in smaller increments. These smaller strain increments may be required to maintain stable and quasi-static conditions when running the DEM algorithm. The input `ddef_target` gives the maximum DEM strain increment, so that the simulations remain stable. What is a suitable value of `ddef_target`? For typical geostatic stresses, a value between 1×10^{-7} and 2×10^{-6} seems to maintain stable simulations and minimal viscous dissipation, while reducing run-times.

7.1.16 nDempla_out (i16)

The results of a DEMPLA simulation are given as output in various ResultsFiles. The integer input nDempla_out gives the frequency of output in these files. A value of 1 means that the results of every Δt step is written to the ResultsFiles, a value of 3 means that the results are written for every third Δt.

7.1.17 nDEM_out (i16)

Every time step Δt of the DEMPLA wave propagation algorithm can involve several DEM time steps. The integer input nDEM_out gives the frequency of reporting the DEM output in the AFiles and BFiles.

7.1.18 C_factor (f16.7)

Because the dynamic problem solves a wave equation, increment Δt must be sufficiently small so that the solution does not outpace the wave equation's characteristic velocities. The Courant–Friedrichs-Lewy (CFL) condition requires the Courant number, $C = c\Delta t/\Delta x$, is less than 1, where c is the characteristic wave speed. After initial consolidation but before shaking, various moduli Q are measured with small strain-probes so that the corresponding wave speeds can be estimated. The moduli include the shear moduli in the x_1 and x_2 directions, the drained uniaxial confined compression modulus, and the undrained (Biot's closed system) uniaxial confined compression modulus. The largest of these moduli among all RVEs is used to estimate the fastest (local) wave speed $\sqrt{Q/\rho}$, which is used to compute the increment Δt for all RVEs.

Within the DEMPLA code, a base Courant number of 0.75 is applied to $\Delta x/\sqrt{Q/\rho}$ to compute Δt, which is found to achieve marginal stability of the simulation. An input C_factor less than 1 reduces Δt below the base value and can provide more stable simulations. A value of 0.5 is used in most of the example calculations.

7.1.19 Direct_soil (f16.7)

The two input values Direct_soil and Dip_soil give the direction and amount of slope of the ground surface (the complementary values Direct_water and Dip_water give the direction and amount of slope of the water table). The two values define the direction **n** in Fig. 1. The input Direct_soil is the direction (in degrees) of the (downward) slope (dip) of the soil surface (and of the rock base), measured counter-clockwise from the x_1 direction (see [2]). Direction Direct_soil is perpendicular to the strike direction.

7.1.20 Dip_soil (f16.7)

The downward slope (in degrees) of the soil surface (and of the rock base).

7.1.21 Direct_water (f16.7)

The two input values Direct_water and Dip_water give the direction and amount of slope of the water table. The input Direct_water is the direction (in degrees) of the (downward)

slope of the water surface, measured counter-clockwise from the x_1 direction (see [2]). The two values define the direction $\mathbf{n}_w$ in Fig. 1.

7.1.22 `Dip_water` (f16.7)

The downward slope (in degrees) of the water table. Note that when the water table is above the soil surface (i.e., the surface is submerged), the water slope must be zero.

7.1.23 `nTiltSteps` (f16.7)

When the ground surface is sloping, the DEM assemblies must be slowly sheared so that the (sloping) geostatic condition is attained. The input `nTiltSteps` gives the number of DEM time steps that are used to apply the geostatic shear stress.

7.1.24 `nTiltEquil` (f16.7)

When the ground surface is sloping, the DEM assemblies must be slowly sheared so that the (sloping) geostatic condition is attained. After the assemblies are sheared, the input number `nTiltEquil` of DEM time steps is used for the assemblies to equilibrate.

7.1.25 `t_PostShake` (f16.7)

The input values of `t_PostShake` and `t_Setl` specify actions after the shaking is finished (i.e., after the full input base motion in the MotionFile has complete, see Section 9). After shaking, the input value of `t_PostShake` gives the duration (time) of a quiescent period in which the dynamic algorithm of DEMPLAis implemented with zero base motion.

7.1.26 `t_Setl` (f16.7)

After the shaking is finished and after the quiescent period of duration `t_PostShake` is finished, the input `t_Setl` gives the duration (time) in which the consolidation algorithm is conducted (see [2] for the distinction of shaking and consolidation algorithms).

7.2 GFile: Layer options section

The final section of a GFile gives additional information about each layer (for example, the layer's permeability). This part of the GFile must have one line per stratigraphic layer (i.e., the number of layers `nLayers` and the number of LFiles, Section 7.1.4). The lines are arranged sequentially, with each line specifying a single layer, starting with the top layer and ending with the bottom layer. Each line contains 5 fields, arranged and formatted as follows:

```
format(i4,1x,     lSat
f9.6,1x,          xSat
f9.6,1x,          k_geot
i4,1x,            iCap
f9.6)             xCap
```

Note that the input fields are separated with the blank character (`1x`) and are arranged horizontally *on a single line*. Each line defines a single stratigraphic layer. Each layer must also have its own LFile. Remember, the first line (first stratigraphic layer) is for soil at the ground surface; the next line is for soil below the first stratigraphic layer; etc.

For a better understanding of the input `lSat`, `xSat`, `iCap`, and `xCap`, one is referred to [3] for guidance related to poromechanics models used in DEMPLA.

7.2.1 `lSat (i4, 1x)`

The input values `lSat` and `xSat` only apply when the layer is initially saturated (i.e., with no gas bubbles) and the poromechanic model `iporo=3` is being used (see Section 11.1.20). In this situation, the values `lSat` and `xSat` allow you to specify the amount of dissolved gas in the pore water. The amount of dissolved gas can be set with the value of `p_wcav` in the RunFile (see Section 11.1.48), but with `lSat` and `xSat`, you can change these values at the start of the wave propagation algorithm. The input `lSat` tells the program whether to use a special saturation condition for the layer, over-riding `p_wcav`.

`lSat=0` No special treatment, and the value of `xSat` is ignored. When `iporo=3`, use the values of `S_o`, `p_o`, and `p_wcav` in the RunFile of this layer (see Sections 11.1.41, 11.1.38, and 11.1.48). The RunFile for the layer should be located in the `RunFiles` sub-directory (page 19), and the name of this RunFile should be given in the LFile for the layer (Section 8.7).

`lSat=1` When `iporo=3` and `S_o`=1.0, then the pore water pressure `p_o` will be reset to the hydrostatic pressure at the end of consolidation, and `p_wcav` (i.e., the relative pressure at which disolved gas will cavitate and bubbles will form, as in Section 11.1.48) will be reset to this water pressure minus the value of `xSat` (elsewhere in the code, pressure `u_wcav`), which should have a value greater than 0. When `iporo=3` but `S_o` < 1.0, then `xSat` is ignored and the program uses the values of `S_o`, `p_o`, and `p_wcav` in the RunFile.

`lSat=2` When `iporo=3` and `S_o=1.0`, then `p_o` will be reset to the water pressure at the end of consolidation, and `p_wcav` (elsewhere in the code, pressure `u_wcav`) will be reset to this water pressure times the value of `xSat`, which should have a value between 0 and 1. When `iporo=3` but `S_o` < 1.0, then use the values of `S_o`, `p_o`, and `p_wcav` in the RunFile of this layer in the RunFiles sub-directory. Note that cavitation is disallowed unless the following values are included in the layer RunFile: `iporo=3`, `S_o`=1.0, `N_o` is non-zero (Section 11.1.46), and `Hcc` is non-zero (Section 11.1.43).

7.2.2 `xSat (f9.6, 1x)`

When `lSat` is 1 or 2 and `S_o` $= 1$, the value of `xSat` is used for adjusting `p_wcav` (elsewhere in the code, pressure `u_wcav`), which is the value of the water pressure, below which cavitation occurs.

`lSat=0` When `lSat`=0, no special treatment, and `xSat` is ignored.

`lSat=1` When `lSat=1`, and `iporo=3` and `S_o`=1.0, then `p_o` will be reset to the hydrostatic water pressure at the end of consolidation, and `p_wcav` will be reset to the hydrostatic pressure minus the value of `xSat` (which should be greater than 0). When `iporo=3` but `S_o`<1.0, then DEMPLA will ignore `xSat` and use the values of `S_o`, `p_o`, and `p_wcav` in the RunFile of the layer. For example, when `lSat`=1, `iporo`=3, and `S_o`=1, and `xSat` = 10, then the pore water is under-saturated with air, and cavitation will occur when the water pressure in the RVE assembly is reduced to 10Pa below the hydrostatic pressure.

`lSat=2` When `lSat=2`, and `iporo=3` and `S_o`=1.0, then `p_o` will be reset to the hydrostatic water pressure at the end of consolidation, and `p_wcav` will be set to the hydrostatic pressure times the value of `xSat`, which should be a value between 0 and 1. When `iporo=3` but `S_o`<1.0, then DEMPLA will ignore `xSat` and use the values of `S_o`, `p_o`, and `p_wcav` in the RunFile of the layer. For example, when `lSat`=1, `iporo`=3, and `S_o`=2, and `xSat` = 0.90, then the pore water is under-saturated with air, and cavitation will occur when the water pressure in the RVE assembly is reduced to 90% of the (absolute) hydrostatic pressure.

7.2.3 `k_geot (f9.6, 1x)`

The value of hydraulic conductivity for the stratigraphic layer. This is the standard geotechnical permeability with units of m/s.

7.2.4 `iCap (i4, 1x)`

The integer `iCap` specifies whether to cap the pore water pressure (relative to the hydrostatic pressure), perhaps to prevent pressure surges due to bubble collapse.

`iCap=0` No cap is placed on the pore water pressure, and `xCap` is ignored.

`iCap=1` The pore water pressure is capped to a pressure of `xCap` above the hydrostatic pressure.

7.2.5 `xCap (f9.6, 1x)`

When `iCap`=1, then the pore water pressure is capped to a pressure of `xCap` above the hydrostatic pressure.

8 LFiles (Layer files)

These files give basic information about stratigraphic soil layers. If you are running DEMPLA in Mode 1, you can skip this section — LFiles are not needed. Each layer requires a corresponding LFile. As explain in item 3 on page 21, the LFiles must have a specific name and must be located in a specific directory.The LayerFile must have the name `L<XXXX>_`BaseName. The name begins with letter "`L`", followed by a 4-digit number (beginning with `0001` for the top soil layer, and increasing with `0002`, `0003`, etc. to the bottom layer). The number is followed by an underscore "`_`" and the BaseName (but no suffix). An LFile is an ASCII

```
This file gives information for a single stratigraphic soil layer
    102          : lvers           | version of this file type
      2          : Isotropic       | consolidated isotropic(0,1), anisotropic(2)
     20.d0       : LayerThickness  | thickness of this single layer, m
     0.563d0     : VoidRatio       | void ratio
     2.67d0      : G_s             | specific gravity of soil solids
Prep_B_frict_040_Isotropic_b_Gen1.5_Aalph_100_dry
DEquil_Template_Frict_0.05_DBumpies-10648-6-06-08-064-c_Frict_0.05_LRforce_-0.20e-4-Int10-1-Nevada-D50
```

Figure 3: An example LFile (layer file). Among other information, the file gives the name of the RunFile for the layer and the StartFile of the initial particle arrangement.

text formatted input file (*not* a Word file!), which means that the input information must be placed within certain rows and columns (or column ranges). Sample LFiles can be downloaded from the GitHub site, and an example of an LFile is shown in Fig. 3. I suggest that you use this file as a template for creating your own LFiles.

An LFile contains 8 lines, described in the eight subsections below. Except for the first line, the beginning of each line contains the value a particular DEMPLA input variable. The lines give the value of the following variables, with the corresponding format.

8.1 comment (\)

The first line of a LFile is ignored. Put some useful information here.

8.2 lvers (i16)

An integer number giving the version of the GFile. This variable is included so that current files will be back-compatible with future versions of the DEMPLA code. The current version is 102, and 102 should be place in the second row of the LFile.

8.3 Isotropic (i16)

When DEMPLA does the preprocessing to prepare the soil column, the RVEs in each layer are consolidated to the geostatic and hydrostatic conditions of the RVE levels. Each layer begins as a single progenitor DEM assembly (i.e., a single DFile), which is consolidated to the particular conditions of each RVE within the layer. This initial consolidation can be performed isotropically or anisotropically.

`Isotropic=0,1` A value of 0 or 1 means that the layer will be consolidated isotropically by increasing (or decreasing) the normal stresses σ_{11}, σ_{22}, and σ_{33} at equal rates, until the target geostatic conditions are attained. Note that the shearing stresses σ_{13} and σ_{23} will also be increased (or decreased), in the case of a sloping ground surface or a sloping water table.

`Isotropic=2` The layer will be consolidated anisotropically by maintaining zero strain in the lateral directions, while increasing (or decreasing) the normal stress σ_{33}. Note that the shearing stresses σ_{13} and σ_{23} will also be increased (or decreased), in the case of a sloping ground surface or a sloping water table.

8.4 LayerThickness (f16.7)

The thickness of the stratigraphic soil layer. The sum of the thicknesses of all soil layers in the soil column must exceed the height of the column.

8.5 VoidRatio (f16.7)

The presumed initial void ratio of the stratigraphic soil layer. This void ratio might be the same as the void ratio of the DEM assembly, but the presumed void ratio will be used for computing the soil density. The presumed void ratio will be adjusted while running the DEMPLA algorithm, in accordance with the volume change of the DEM assemblies.

8.6 G_s (f16.7)

The specific gravity of the soil grains, which is used to find the soil density (note that the density of water is given in the GFile, Section 7.1.11).

8.7 RunFileLayer (a400)

The name of the RunFile for the layer. This RunFile will be used to prepare the DEM assemblies for the RVEs: the contact model and properties, the poromechanic model and properties, and other run variables. The name of `RunFileLayer` file must conform to Linux naming conventions of files, case-sensitive with no spaces. The file with this name must be located within the `RunFiles` folder within the base folder for the simulation. The contents and format of RunFiles are described in Section 11.

8.8 StartFileLayer (a400)

The name of the StartFile for the layer. This StartFile is a DFile that contains information about the assembly of particles: number of particles, shape category of the particles, sizes of particle, and locations and orientations of particles. The name of `StartFileLayer` must conform to Linux conventions, case-sensitive with no spaces. The file with this name must be located within the `StartFiles` folder within the base directory from which the simulation is run.

Although the OVAL engine in DEMPLA accommodates several different boundary types (Section 10), only D-type StartFiles (i.e., DFiles) with periodic boundaries are currently permitted with Mode 2 (wave propagation) analyses. Mode 2 analyses can only be conducted in three dimensions, and only 3-d assemblies are permitted. The contents and format of D-type StartFiles are described in Section 12. When running DEMPLA, the deformation gradient is described by an upper-triangular matrix (Section 11.2.1). The initial deformation gradient is the 3×3 identity matrix.

9 MFiles (Motion files)

An MFile or MotionFile gives the displacement history of the soil column's base (i.e. rock, or at the depth at which a motion was actually recorded). MFiles are only used in Mode 2

(wave-propagation) DEM analyses. MFiles are not used with Mode 1 (single-assembly) DEM analyses. Displacements are given in the two lateral directions x_1 and x_2 (parallel to the ground and rock surfaces) and the direction x_3 that is perpendicular to the ground and rock surfaces (vertical for non-sloping ground). The displacements must begin with zero displacements in all three directions. I recommend the end of the displacement history (i.e. the last line in the MFile) is such that the final velocity is zero or nearly zero. The first line of the file gives the time increment between displacement steps (this time increment will likely be different than the Δt of the wave propagation algorithm). The first line is following by lines of the displacement history (the number of lines are limited by the parameter `mRock` in the source code file `param-dempla-X.X.XX.f`, currently set at 50,000).

9.1 `dtRock (*)`

The time increment (seconds) between rock displacement values. Use double precision format (e.g. `5.000000d-04`). This value will likely differ from Δt in the wave propagation algorithm (see [2]).

9.2 `dtRock (xRock(1,i), xRock(2,i), xRock(3,i))`

Rock displacements (meters) in the three directions. Use three double precision values (e.g. `5.000000d-04`), separated by zeros. The x_3 direction is perpendicular to the (possibly sloping) rock base.

10 Boundary Types

DEMPLA (and before it, OVAL) is primarily intended for element studies of using rectangular (2D) and box (3D) assemblies of particles. During a simulation, the boundaries (sides) are moved to produce prescribed rates of strain or rates of stress, as described in Section 11.2. The boundaries themselves can be of several types.

10.1 Periodic boundaries

The default boundaries are periodic. These are the easiest boundaries to use, and they can be used with either dense or sparse assemblies. Moreover, some of the other types of boundaries are created by starting with an assembly having periodic boundaries and then replacing the periodic boundaries with the another boundary type.

For Mode 2 (wave propagation) analyses, only three-dimensional assemblies with periodic boundaries are permitted.

10.2 Tight-fitting particle boundaries

This type of boundary can only be created with 2D assemblies, and it is created by beginning with a non-sparse (at least, moderately dense) assembly having periodic boundaries. The process of creating a tight-fitting particle boundary involves finding the particle graph of the assembly (i.e., finding the topological arrangement of the contacts) and then identifying the string of contacting particles that surround the assembly. These particles become

the boundary particles, which will fit tightly against (i.e., will be in contact with) the interior particles. After the periodic boundaries are "broken" and replaced with tight-fitting boundaries, the boundary particles will not likely be in equilibrium, so a period of several hundred time steps should be included to allow the assembly to equilibrate with its new boundaries. Once periodic boundaries are replaced with tight-fitting particle boundaries, the periodic boundaries can not be retrieved. Tight-fitting boundaries can be placed on the left and right sides (with periodic boundaries remaining top and bottom), on the top and bottom (with periodic boundaries remaining left and right), or on all four sides of the assembly. The intended combination of boundaries is specified with the `iflexc` input variable (Section 11.2.10).

Several types of stress or strain control are available with tight-fitting boundaries:

- Stress control (`iflexc = x1, 1x,` or `11`). The stress (actually, the stress rate) can be controlled with the `icontr=1` and `defrat` at the desired rate (Section 11.2.1 and Section 11.2.1). For example, if tight-fitting boundaries are created on the left and right sides, the stress σ_{11} is applied against the two sides, and the rate of this stress can be controlled. In this same example, the other stress components (σ_{12}, σ_{21}, and σ_{22}) are also applied on the left and right sides, but only their original (not current) values are applied (those stresses present when the tight-fitting boundary was created). This approach prevents "hydro-fracturing" of the side boundaries if the assembly is being compressed vertically. Stress-controlled tight-fitting boundaries approximate the membrane-type conditions that are commonly used in soil testing. The boundary stress is applied to imaginary boundary element: the branch vectors that connect the centers of the boundary particles.

- Displacement control with free rotation (`iflexc = x2, 2x,` or `22`). The particles along a tight-boundary are constrained to translate at a rate in accord with the prescribed strain rates (Section 11.2.1 and Section 11.2.3). The boundary particles are allowed to rotate. Boundary forces are applied at the centers of the boundary particles.

- Displacement control with free rotation (`iflexc = x3, 3x,` or `33`). The particles along a tight-boundary are constrained to translate at a rate in accord with the prescribed strain rates (Section 11.2.1 and Section 11.2.3). The boundary particles constrained to rotate in accord with the prescribed rotation rate (the Eulerian rate that corresponds to $\frac{1}{2}F_{12}$ in Section 11.2.1. Boundary forces are applied at the centers of the boundary particles.

- Displacement control with friction and free rotation (`iflexc = x4, 4x,` or `44`). Suppose that tight-fitting boundaries have been created on the left and right sides, and periodic boundaries remain on the top and bottom (`iflexc = 40`). With this type of control, particles along the left and right sides are constrained to translate horizontally at a rate in accord with the prescribed horizontal strain rates F_{11} and F_{12} (Section 11.2.1 and Section 11.2.3). The side particles are free to translate vertically, but only if they overcome the side friction coefficient prescribed by `frictw` (Section 11.1.25). The boundary particles are allowed to rotate. Boundary forces are applied at the centers of the boundary particles.

- Displacement control with friction and free rotation (`iflexc = x5, 5x,` or `55`). Suppose that tight-fitting boundaries have been created on the left and right sides, and periodic boundaries remain on the top and bottom (`iflexc = 40`). With this type of control, particles along the left and right sides are constrained to translate horizontally at a rate in accord with the prescribed horizontal strain rates F_{11} and F_{12} (Section 11.2.1 and Section 11.2.3). The side particles are free to translate vertically, but only if they overcome the side friction coefficient prescribed by `frictw` (Section 11.1.25). The boundary particles constrained to rotate in accord with the prescribed rotation rate (the Eulerian rate that corresponds to $\frac{1}{2}F_{12}$ in Section 11.2.1). Boundary forces are applied at the centers of the boundary particles.

10.3 Rigid-flat boundaries

This type of boundary can be created with either 2D or 3D assemblies. The boundary is produced by giving an input value for `iflexc` of `9`, `90`, or `99` in the first line of the deformation-stress path section of a RunFile (see Section 11.2.10). A pair of rigid-flat boundaries (one each on opposite sides of the assembly) can coexist with periodic boundaries on the other sides. The size of the assembly (the box dimensions) are input with the dimensions `xcell` (Section 12.1.2 and Section 12.1.3). Unlike with tight-fitting boundaries (Section 10.2), particles interact with rigid-flat boundaries at the particle surfaces, instead of at the particle centers. Stresses and strains can be controlled with rigid-flat boundaries, just as with other types of boundaries (Section 11.2.1 through Section 11.2.7).

10.4 External-particle boundaries

These boundaries are created by surrounding the assembly with a set of external particles, which confine the interior particles. This type of boundary can be created with either 2D or 3D assemblies. The boundary is produced by giving an input value of greater than one to the integer `nplatn` (Section 11.1.13). For example, if `nplatn=4`, then you will be queried to give the names of four files that provide information about each of the four sets of boundary particles—their positions, radii, etc.—as described below. Stresses and strains can be controlled with rigid-flat boundaries, just as with other types of boundaries (Section 11.2.1 through Section 11.2.7). Note that the boundary particles can only be circles (2D) or spheres (3D).

A file that provides information about a set of boundary particles contains four lines that give general information about the particles followed lines that provide information on each particle. The content of each line is described in the following subsections (Fortran free format is used, with integer or double precision type corresponding to the leading letter of the variable name).

10.4.1 `ipvers`

Set this value to 1. It gives a version number for the file, in the event that future changes are made to the format of these files.

10.4.2 ixfix(1),ixfix(2),ixfix(3),ithfix(1),ithfix(2),ithfix(3)

The manner in which the boundary particles are constrained in their motions. Values of either 0 or 1 (unconstrained or constrained, respectively) are assigned to the three directions of translation, (**ixfix**), and the three directions of rotation, (**ithfix**). Note that when translation is constrained in a particular direction, then the boundary particle move in accord with the prescribed deformation rate F_{ij} (Section 11.2.1).

10.4.3 idirec

The "direction" of the boundary. For example, if a set of boundary particles are one the left (i.e., x_1) side of a 2D assembly, then **idirec** is set to 1. If a set of boundary particles are one the top or bottom of a 2D assembly, then **idirec** is set to 2 (i.e., x_2). This feature is necessary to enable the control of stress within these boundaries.

10.4.4 nplt

The number of particles in the boundary file.

10.4.5 rad,xp(1),xp(2),xp(3)

The radius and position of a particle center, with one particle per line of input.

11 RunFiles for Dempla and Oval

RunFiles give general information about the conditions of the DEM analysis. RunFiles are required for either Mode 1 (single assembly) or Mode 2 (wave propagation) analysis. The ASCII text file is a formatted input file, which means that the input information must be placed within certain rows and columns (or column ranges). Sample RunFiles can be downloaded from the web site, and an example of a RunFile is shown in Figs. 4 and 5, which give the upper and lower parts of the file. This particular RunFile has the following name:
`Prep_B_frict_040_Isotropic_b_Gen1.5_Aalph_100_nobubble`
and the file is located in the **RunFiles** directory (folder) of the GitHub repository This and other sample files are available from this repository. I suggest that you use these files as templates for creating your own RunFiles.

The RunFile file name must conform to Linux conventions, with no spaces, asterisks, question marks, and other characters. The RunFile file name will be used for assigning names to various output files (Section 6 and Table 1). On Windows systems, you may want to give the RunFile name a **.txt** extension so that it will be more properly treated with word processors such as Word or Word Pad.

The content and format is describe below. Note, however, that the current format is not backward-compatible with earlier versions of Oval. A RunFile file is arranged in two parts:

1. a general information section consisting of the following 29 lines (see Section 11.1):
    - a single title line.

```
Heading with useful information about this particular simulation
       1         : algori   | the algorithm for advancing the particle positions (1 or 2)
       6         : ivers    | add extra input lines (0)
       0         : ncownt   | fequency for updating non-periodic boundaries (0)
       0         : iout(2)  | output files with avg. def. and gradients in layers (0)
       0         : iout(3)  | output files with avg. stresses within layers (0)
       1         : istart   | type of file defining the initial configuration (1 or 3)
       0         : iend     | type of file to be created at end of the run (0, 1, or 3)
       0         : idef     | reference configuration for reporting deformations (0)
     500         : iupdtm   | max. no. of time steps between linked-list updates
       0         : icirct   | compute and regularly update the particle graph (0)
       9         : imodel   | contact model (linear, Hertz, etc.) (0 1, 5, 6, 7, or 9)
       0         : nplatn   | number of additional D-files with boundary particles (0)
       5         : nloop1   | minimum number of iteration loops when algori=2
       0         : iexact   | don't use the same mass for every particle (0 or 1)
       0         : isub     | number of submerged particles (0)
       1         : idamp    | standard or Potyondy/Cundall damping (0, 1, 2, or 3)
       0         : iheat    | temperature-dependent model (0)
       0         : icoef    | enable changing the friction coefficient during a run (0)
       3         : iporo    | poromechanic fluid model (0, 1, 3, or 4)
       0         : ifree(11)| a free input integer.  Placeholder for future versions
       0         : ifree(12)| a free input integer.  Placeholder for future versions
       0         : ifree(13)| a free input integer.  Placeholder for future versions
       0         : ifree(14)| a free input integer.  Placeholder for future versions
       0         : ifree(15)| a free input integer.  Placeholder for future versions
       0         : ifree(16)| a free input integer.  Placeholder for future versions
       0         : ifree(17)| a free input integer.  Placeholder for future versions
       0         : ifree(18)| a free input integer.  Placeholder for future versions
```

Figure 4: An example RunFile – the upper 28 lines.

```
  29.d9        : kn       | normal contact stiffness (force/indentation)
   0.15        : kratio   | ratio of tangential/normal contact stiffnesses
   0.40        : frict    | coefficient of friction at particle contacts
   0.          : frictw   | coefficient of friction between particles and wall
  -1.          : rho      | the mass density of the particle material
   0.250       : sep      | threshold separation during the near-neighbor searches
   0.03        : pcrit(1) | viscosity coefficient for translational body damping
   0.05        : pcrit(2) | viscosity coefficient for rotational body damping
   0.          : pcrit(3) | viscosity coefficient for contact damping
   0.64        : xseed    | seed for assigning random initial velocities (when motion=1)
   0.          : rmsvel   | rms initial velocity (when motion=1)
   0.25        : pdif     | parameter for reducing Jager memory demand (when imodel>=6)
   0.          : tmax     | maximum time for the simulation run
 100.          : A_1      | shape factor for conical or general contact asperity profile
  -1.          : dt       | time increment
   0.          : gravty(1)| gravity in x_1 direction. Depracated. (0.)
   0.          : gravty(2)| gravity in x_2 direction. Depracated. (0.)
   0.          : gravty(3)| gravity in x_3 direction. Depracated. (0.)
   0.          : p_o      | initial fluid pressure of pore liquid relative to atm. pressure
  31.8d9       : K_s      | bulk modulus of grain bodies (when iporo = 1, 3, or 4)
   2.2d9       : K_f      | bulk modulus of pore fluid (when iporo = 1 or 3)
   1.00d0      : S_o      | initial fluid saturation of pore fluid at p_o (iporo=3)
 100.d3        : p_atm    | the reference atmospheric pressure (iporo=3)
   0.          : Hcc      | Henry's coefficient of solubility of pore gas (iporo=3)
  72.75d-3     : gamm     | surface tension of pore bubbles (iporo=3)
   0.0001d0    : D_o      | initial bubble diameter at p_o and S_o (iporo=3)
   2.d13       : N_o      | number/density bubbles per unit initial pore volume(iporo=3)
2337.d0        : p_vap    | water vapor pressure as abs. pressure (iporo=3)
   0.          : p_wcav   | cavitation pressure relative to atm. pressure (iporo=3)
   0.          : rfree(15)| placeholder
   0.          : rfree(16)| placeholder
   0.          : rfree(17)| placeholder
   0.          : rfree(18)| placeholder
   1.5         : palpha   | alpha power in contact profile (when imodel=9)
                                                                                        imicro
    ************  Deformation-Stress Path Segments  **********            krotat          iflexc |            iplot
  icontp (100000)   (10000)   (1000)    (100)      (10)      (1)                |          idump  | |ibodyf    ipts2 |
icontr |  rate_11 | rate_22 | rate_33 | rate_12 | rate_13 | rate_23    vrate  igoal   finalv  ipts |  | | |  defdot  |   |
------|-|---------|---------|---------|---------|---------|---------|---------|--|-|---------|----|-|--|-|--|------|----|--|
111000 1  0.000e+0  0.000e+0  0.000e+0  0.000e+0  0.000e+0  0.000e+0  0.000e+0 70 0     4000.  100 0  0 0  0     0.    0  0
```

Figure 5: An example RunFile – the lower lines.

- a series of 27 formatted lines that provide integer input.
- a series of 34 formatted lines that provide floating point double-precision (8-byte, `real*8`) input.

2. five spacer lines of comments.

3. a deformation-stress path that consists of a series of formatted lines that describe each phase (segment) of the deformation-stress control path (see Section 11.2). For simulations with a poromechanic (pore fluid) model, these deformation-stress path also includes control of either the pore fluid pressure or the inflow rate of pore fluid. The program currently accepts up to 49999 segments, although this limit can be changed with the source code parameter `lc1` in the source `common` file.

The contents of the first part, detailed in the next section, are contained in a series of 62 lines, each with a single input value at the beginning of a line. The third part, which specifies the deformation-stress path, is detailed in Section 11.2, page 50. Although the format specifier `f16.7` is used, Fortran allows the input data to be in either fixed (`F`) or exponential (`E` or `D`) formats with any number of significant digits, *provided that the data fits within the first 16 columns each line.*

Note that many input items in the first part are deprecated and abandoned from earlier versions, and others are place-holders for as-yet undefined options. That is, over twenty of the items in the first part are ignored. A 0 (for integers) or a 0. (for floating-point) can be placed as the input values for these items.

11.1 RunFile: General information section

11.1.1 `title` (`a80`)

The `title` could include, perhaps, information on the nature of the simulation for your future reference. At present, the variable `title` is not used within the program, nor is it echoed to any of the output files. Use this line for your own purpose.

11.1.2 `algori` (`i16`)

The program can be run with either of two DEM algorithms:

`algori=1` The conventional DEM algorithm (refer to [1]). The algorithm uses an implicit integration scheme. At present, this is the most robust of the two algorithms.

`algori=2` A new algorithm that the author has developed to self-monitor the progress of an intended pseudo-static simulation. With the standard algorithm (`algori=1`), the otherwise natural imbalance of forces on the particles can become excessive, particularly if the loading rate is too rapid. With the new algorithm (`algori=2`), several time steps are cycled within each deformation step. The cycling continues until the average force imbalance on a particle is within a threshold limit which constitutes a *near-equilibrium criteria.* At present the threshold is a particle force imbalance less than 1% of the average contact force magnitude (variable `chiavg.lt.chimax`). The number of cycles is currently

programmed to be no less than 3 and no more than 101. See Sections 13.3.5, 13.3.9, and 13.3.15 for more information on the threshold limits that define the near-equilibrium criteria.

I recommend using `algori=1` with sparse assemblies (for example, if you are consolidating a gaseous assembly into a dense one) or when you are trying to capture the true dynamics of a deformation process (vibration studies, flow studies, etc.); but I recommend using either `algori=1` or `algori=2` for pseudo-static simulations with dense assemblies. For wave propagation analyses (i.e. Mode 2 DEMPLA), I have only used `algori=1`, so I can not recommend its use for this mode.

11.1.3 `ivers` (i16)

An integer number giving the version of the RunFile. This variable is included so that current files will be back-compatible with future versions of the DEMPLA code. The current version is 6, and the integer `6` should be place in the second row of the RunFile.

11.1.4 `ncownt` (i16)

When a flexible, tight-fitting boundary is used (Sections 10.2 and 11.2.10), it must be periodically updated, as the topology of the assembly is constantly changing. `ncownt` gives the frequency of updating the boundary. For example, if `ncownt=1`, the boundary is updated after every time step; if `ncownt=10`, the boundary is updated after every tenth step. When a flexible boundary is not being used, the input value of `ncownt` is ignored. When DEMPLA is being run in Mode 2 (wave propagation analysis), only periodic boundaries are allowed, and `ncownt` is ignored. If `ncownt=0` a flexible boundary is not being used, and the boundary will never be updated.

11.1.5 `iout(2)` (i16)

Currently not supported. Use zero (integer 0).

11.1.6 `iout(3)` (i16)

Currently not supported. Use zero (integer 0).

11.1.7 `istart` (i16)

The type of StartFile that will be used. The program supports two formats of StartFiles, which give the number of particles, particle type, and the initial particle arrangement (sizes, positions, etc.).

`istart=1` The initial particle arrangement will be given in a `D<RunFile>` file, henceforth referred to as a "D-file" (Section 12). This file is a text (ASCII) file (very portable).

`istart=2` This type of StartFile has been abandoned. Only `istart=1` and `istart=3` are currently supported.

`istart=3` The entire initial state will be given in a C<RunFile> file, or "C-file". This binary "restart" file allows the current simulation to begin at the exact condition that was "dumped" at the end of a previous simulation. The restart file includes all positions, velocities, and contact information that allow the new run to begin where a previous run had left off. Note that with D- and E-files, the simulation will begin with the particles having zero velocities (or perhaps randomly assigned velocities, Section 11.1.32), and there will be no history of the contact forces. With restart C-files, the simulation will begin with velocities and forces carried over from a previous run. Also see Sections 11.1.8 and 11.2.9.

11.1.8 `iend` (i16)

The type of StartFile that will be created at the end of the simulation. The file that is created can later be used as the initial condition of a future simulation (see Sections 6 and 11.1.7).

`iend=0` No file will be created at the end of the simulation.

`iend=1` An ASCII D<RunFile> file will be created, containing the final particle arrangement. See Section 12.

`iend=2` This type of StartFile has been abandoned. Only `istart=1` and `istart=3` are currently supported.

`iend=3` A binary C<RunFile> file will be created, containing the entire end state of the simulation. This "dump" file can be used as a "restart" file to begin a future simulation at the exact ending state of the current simulation. Also see Sections 11.1.7 and 11.2.9.

11.1.9 `idef` (i16)

The value of `idef` determines the reference configuration of the assembly. `idef` is only of consequence when the StartFile is binary-type, restart C-file, and `idef` does not affect the results when the StartFile is a D-file.

A value `idef=0` is recommended for wave propagation analyses (Mode 2 Dempla), so that a consistent initial (reference) state is used throughout the output for the simulation.

`idef=0` When a C-file is being used to begin the simulation, then the reference configuration is carried over from the simulation from which the C-file was created. Deformations and deformation rates are relative to this older, carried-over configuration. That is, the strains that are reported as output are relative to an older configuration.

`idef=1` When a C-file is being used to begin the simulation, then the reference configuration is taken as the start of the current simulation. A zero-strain condition is reported at the start of the simulation. When a C-file is being used to start the simulation, the strains will be different at the start of the simulation than

the strains that were reported at the end of the older simulation from which the C-file was created. Because of this difference, the results of a "restarted" simulation will differ from that of a simulation that continues to run past the restart point. The option `idef=1` is of no consequence when a D-file or E-file is being used to begin the simulation.

11.1.10 `iupdtm` (i16)

The frequency of updates to the linked list of near-neighbor particle pairs. You don't have to be too concerned about its value, as the program automatically determines if a more frequent update is required. A value of between 50 and 500 should be fine, but larger values will lead to somewhat faster computations. See Section 11.1.27.

11.1.11 `icirct` (i16)

Currently not supported. Use zero (integer 0).

11.1.12 `imodel` (i16)

The contact model.

`imodel=0` A linear contact model with friction (see Sections 11.1.22, 11.1.23, and 11.1.24). The contact model requires the following parameters: the normal indentation-stiffness (`kn`, Section 11.1.22), the ratio of tangential and normal stiffnesses ν (`kratio`, Section 11.1.23), and the friction coefficient μ (`frict` Section 11.1.24).

`imodel=5` A Hertz-Mindlin contact model with friction. (see Sections 11.1.22, 11.1.23, and 11.1.24). The model is one of two spheres in contact, with each sphere having isotropic elastic properties, but with surface friction between the particles. The tangential force model is a simple modified-Mindlin model in which tangential stiffness is a function of the normal force. The contact model requires the following parameters: the material's shear modulus G (`kn`, Section 11.1.22), the material's Poisson ratio ν (`kratio`, Section 11.1.23), and the friction coefficient μ (`frict` Section 11.1.24). The radius of curvature of the contact surfaces are assumed as follows: for spheres, the mean of the two sphere's diameters; for sphere clusters (bumpy shapes), the mean of the two particles' contacting spheres; and for ovoids, the mean of the mean size of the two particles.

`imodel=6` A Jäger contact model with friction [4, 5]. The Jäger contact is a generalized Cattaneo-Midlin-Deresiewicz contact for arbitrary sequences of loading and unloading in a three-dimensional setting. In this sense, it is far superior to the simple modified-Mindlin `imodel=5` model, which can only handle a single reversal in loading direction. Refer to Sections 11.1.22, 11.1.23, 11.1.24, and 11.1.33 for other input value that are required with the Jäger contact. The radius of curvature of the contact surfaces are assumed as follows: for spheres, the mean of the two sphere's diameters; for sphere clusters (bumpy

shapes), the mean of the two particles' contacting spheres; and for ovoids, the mean of the mean size of the two particles.

Moreover, because a full description of the Cattaneo-Midlin-Deresiewicz contact requires the full three-dimensional history of tangential and normal contact motions, you will also need to adjust the value of the parameter `mlistJ`, which sets the size of arrays that store the history, in the `param-dempla-X.X.XX.f` file and in the `subroutine Jager3D` and other locations within the `dempla-X.X.XX.f` file. Parameter `mlistJ` has likely been set to a very low value in these locations, because a large value greatly in increases the size of the executable Dempla file. The value will need to be much larger when the Jäger model is used, and you should change `mlistJ` to, say, 500 or larger. The memory demand can be somewhat reduced with the input parameter `pdif`, Section 11.1.33.

`imodel=7` A Hertz-type contact model with friction, but with a cone-to-cone contact between particles, representing pointed asperities (see [6]). As with the Jäger contact model of `imodel=6`, the contact algorithm accounts for the full three-dimensional history of contact motions. In addition to the parameters required with `imodel=6`, the cone-to-cone model requires the angle of the cone tips (`A_1`, Section `sec:A1`).

`imodel=9` A Hertz-type contact model with friction, but with a general power-law profile of the asperity shapes between particles (see [6]). The author has used this contact model to to produce an assembly shear modulus $\bar{G}$ that increases with mean effective stress $\bar{p}$, roughly in the proportion $\bar{G} \propto \bar{p}^{\alpha}$, with α near 0.50. As with the Jäger contact model of `imodel=6`, the contact algorithm accounts for the full three-dimensional history of contact motions. In addition to the parameters required with `imodel=6`, the cone-to-cone model requires the angle of the cone tips (`A_1`, Section `sec:A1`) and the exponent of the contact power-law profile (`palpha`, Section `sec:palpha`).

11.1.13 `nplatn` (i16)

Currently not supported. Use zero (integer 0).

11.1.14 `nloop1` (i16)

The minimum number of iteration time steps per deformation step. This value is only used when `algori=2`. If `nloop1` is zero and `algori=2`, a value of 3 is assigned to `nloop1`. If `algori=1`, then the input value of `nloop1` is ignored.

11.1.15 `iexact` (i16)

To reduce quasi-static run-times, I usually use a common mass for all particles (`iexact=0`). To use the actual mass of individual particles, use `iexact=1`. The latter case may be buggy.

11.1.16 `isub` (i16)

Currently not supported. Use zero (integer 0).

11.1.17 `idamp` (i16)

The type of damping:

`idamp=0,1` Viscous damping. This damping is described by three parameters: translational damping proportional to a particle's velocity relative to the mean field (`pcrit(1)`, Section 11.1.28), rotational damping proportional to a particle's rotational velocity relative to the mean field vorticity (`pcrit(2)`, Section 11.1.29), and contact damping proportional to the relative contact velocities of two particles (`pcrit(3)`, Section 11.1.30). The damping parameters are a fraction of critical damping (typically in the range of 0.02–0.10).

`idamp=2` Potyondy–Cundall damping [7], in which the damping force is a fraction of a particle's force imbalance. This damping is described by two parameters: the fraction used with force imbalance (`pcrit(1)`, Section 11.1.28), and the fraction used with moment imbalance (`pcrit(2)`, Section 11.1.29).

`idamp=3` Same as `idamp=1`, except that contact damping is not applied when one of a particle's contacts is sliding.

11.1.18 `iheat` (i16)

Currently not supported. Use zero (integer 0).

11.1.19 `iheat` (i16)

Currently not supported. Use zero (integer 0).

11.1.20 `iporo` (i16)

The poromechanic model that is used for computing pore fluid pressure (and the total stress) or the inflow of pore fluid.

`iporo=0` No poromechanic model is used. Only intergranular (effective) stress is computed.

`iporo=1` The pore space is saturated with the pore fluid, and not bubbles, surface tension, or dissolved gas is considered. The model requires two parameters: the bulk modulus of the solid grains (`K_s`, Section 11.1.39), the bulk modulus of the pore fluid (`K_f`, Section 11.1.40), and the initial pore fluid pressure (`p_o`, Section 11.1.38).

`iporo=3` Model of pore fluid compressibility for a pore liquid and entrained gas (the quasi-saturated condition). One is referred to [3] for guidance related to

poromechanics models used in DEMPLA. The model accounts for dissolution of pore gas, surface tension of gas bubbles, and vapor pressure of the pore liquid. Many parameters are required for this model: the bulk modulus of the solid grains (`K_s`, Section 11.1.39), the bulk modulus of the pore fluid (`K_f`, Section 11.1.40), initial fluid pressure of pore liquid relative to atmospheric pressure (`p_o`, Section 11.1.38), initial fluid saturation of pore space at pressure `p_o` (`S_o`, Section 11.1.41), the reference atmospheric pressure (`p_atm`, Section 11.1.42), Henry's coefficient of pore the liquid/gas (`Hcc`, Section 11.1.43), surface tension of pore bubbles (`gamm`, Section 11.1.44), initial diameter of gas bubbles at `p_o` and `S_o` (`D_o`, Section 11.1.45, note that `N_o` is alternatively given), number density of bubbles per unit initial of pore volume at `p_o` and `S_o` (`N_o`, Section 11.1.46, note that `D_o` is alternatively given), pore liquid vapor pressure (`p_vap`, Section 11.1.47, note that `p_vap` is an absolute pressure), and cavitation pressure below atmospheric pressure for under-saturated dissolved gas (`p_wcav`, Section 11.1.48).

`iporo=4` Model of pore fluid compressibility for a pore space filled with an ideal gas (e.g., dry sand). Three parameters are required for this model: the bulk modulus of the solid grains (`K_s`, Section 11.1.39), initial fluid pressure of pore liquid relative to atmospheric pressure (`p_o`, Section 11.1.38), and the reference atmospheric pressure (`p_atm`, Section 11.1.42).

11.1.21 `ifree(11)--ifree(18) (i16)`

Currently not supported. These eight rows are place-holders for future features. Use zeros for all eight inputs (integer 0s).

11.1.22 `kn` or `G f16.7`

With linear contacts (`imodel=1`), this input variable is the linear (spring) contact stiffness for determining the contact forces normal to contact surfaces. This stiffness is multiplied by the indentation at the particle contacts (i.e., half the overlap between two particles) to compute the normal contact force. As a result, the contact stiffness relative to the particle separation (overlap) is `kn`/2, so that the DEMPLA stiffness value will be half of that used in most DEM codes.

With Hertz-Mindlin contacts and Jäger contacts (`imodel=5`, `imodel=6`, or `imodel=9`), this input variable is the shear modulus G of the particles' material. For quartz, G is about `29.d9` Pa.

Although the format specifier `f16.7` is used, Fortran allows the input data to be in either fixed (`F`) or exponential (`E` or `D`) formats with any number of significant digits, provided that the data fits within the field width of 12 characters.

11.1.23 `kratio (f16.7)`

With linear contacts (`imodel=1`), this input variable is the ratio of two contact stiffnesses: the tangential stiffness divided by the normal stiffness.

With Hertz-Mindlin contacts and Jäger contacts (`imodel=5`, `imodel=6`, or `imodel=9`), this input variable is the Poisson ratio of the particles' material. For quartz, ν is about `0.15`.

11.1.24 `frict` (f16.7)

The friction coefficient between particles.

`frict=0.` The contacts will be frictionless—friction will be "turned off." I sometimes use this mechanism to help densify a loose assembly.

`frict>0.` The contacts will be frictional with the given coefficient of friction.

11.1.25 `frictw` (f16.7)

The friction coefficient between particles and boundary walls (or boundary particles). See Section 10. This input value is ignored with periodic boundaries.

11.1.26 `rho` (f16.7)

The mass density of the particle material. See Section 11.1.36 on input `dt` for options to automatically assign a value of `rho`.

11.1.27 `search` (f16.7)

The threshold distance between two particles that will place them into a linked list of near-neighbors (i.e., a list of candidate contacts). To reduce runtime, the subroutine that assembles the near-neighbor linked list is only occasionally called. The actual contact detection process, which is repeated with each time step, is only applied to this candidate list of near-neighbors (also see Section 11.1.10 on the input `iupdtm`). The threshold distance is equal to the dimensionless `search` value multiplied by the minimum particle radius (or half of the particle half-width, for nonspherical particles). Larger values of `search` slow the contact detection process within every time step, since it will increase the length of the list of candidate contacts, most of which will not actually be in contact. Larger values of `search`, however, will mean less frequent construction of the linked list of near-neighbors, a relatively slow process that requires a more exhaustive search for candidate contacts. Values of `search` between 0.20 and 0.80 seem to be appropriate.

11.1.28 `pcrit(1)` (f16.7)

A dimensionless damping coefficient, which will be applied to the translational velocities of the particles. For viscous damping (`idamp=0,1,3`), this coefficient represents a fraction of the critical damping $2\sqrt{mk}$, and the resulting viscous force is applied as a body force. When periodic boundaries are being used, viscous damping is only applied to the particle velocities that are measured relative to the mean-field velocity. With viscous damping, you will probably want to experiment with different values, with due attention to such performance parameters as `chi1`, `chi2`, `chi3`, `chi4`, and `psi` (pages 69–70).

With Potyondy–Cundall damping (`idamp=2`), `pcrit(1)` is the fraction of the out-of-balance particle force that is applied as a counter-acting damping force.

11.1.29 `pcrit(2)` (f16.7)

A dimensionless coefficient of damping, applied to the rotational velocities of a particle or to the out-of-balance torque on a particle (see the previous Section 11.1.29).

11.1.30 `pcrit(3)` (f16.7)

A dimensionless coefficient of viscosity, applied to the contact velocities of any pair of particles that are touching. This viscous force is applied as a contact force. The contact viscosity is "turned off" whenever frictional sliding occurs.

11.1.31 `xseed` (f16.7)

A seed for the random number generator. It is used for assigning initial random velocities to the particles. The seed is only used when `rmsvel>0`. See Section 11.1.32.

11.1.32 `rmsvel` (f16.7)

The average (root mean square) random particle velocity, assigned at the beginning of the simulation. Non-zero velocities can only be assigned when `algori=1` (Section 11.1.2). When a `rmsvel` is assigned, the particles are also given an initial angular velocity, on average about 10% of `rmsvel` divided by the mean particle radius (rather arbitrary, but this choice resides in "`subroutine init`" as `rotfac = 0.10d0`. Although velocities are randomly assigned, care is taken to assure that the initial momentum and angular momentum of the entire assembly is zero.

`rmsvel=0.` Do not assign initial random velocities to the particles. If `istart=1` or `istart=2`, the simulation will begin with the particles in an initially stationary state. When `istart=3`, the particle velocities will be carried over from a previous run regardless of the value of `rmsvel`.

`rmsvel>0.` A random velocity will be assigned to each particle, with the average (root mean square) random particle velocity equal to `rmsvel`. I sometimes use this feature to help densify an assembly by applying an artificial "vibration" technique. This option has no effect when the simulation is begun with a binary restart file (`istart=3`), since the velocities are carried over from a previous run. This option is only available when `algori=1` (Section 11.1.2).

11.1.33 `pdif` (f16.7)

When a Jäger contact model is being used (with `imodel=6`, see Section 11.1.12), this parameter can be used to reduce the memory demand of the model. It should be set to a value between 0 and twice the friction coefficient `frict`. When a Jäger contact model is not being used, with `imodel` not equal to 6, then this input value is ignored.

11.1.34 `tmax` (f16.7)

This input value sets a limit on the maximum time (or time steps) of the simulation. With some stress-control boundary conditions, DEMPLA will keep running until an input target stress has been reached (if ever). In some situations, you may want set a limit on the number of time steps for the simulation (for example, with cyclic loading, when the number of cycles to failure is not known beforehand). When `tmax` is 0 or negative, `tmax` will be ignored (i.e., with no control on the maximum length of the simulation).

11.1.35 `A_1` (f16.7)

Shape factor for conical or general contact asperity profiles, when `imodel=7 or 9` (see Section 11.1.12). This parameter is the "A_α" in [8] and [6].

11.1.36 `dt` (f16.7)

The time step. The program will provide feedback on whether your time step is too large and will recommend a maximum time step, so you can just guess a trial time step and then adjust it later.

Several other options are available for establishing a time step, depending on the combined values of `dt` and `rho` (Section 11.1.26):

`dt>0, rho>0` If appropriate, your assigned values are used. DEMPLA will provide feedback on your assigned time step and the maximum recommended time step. (See the example output in Section 15.) If your time step exceeds this maximum, then DEMPLA will stop.

`dt=0, rho>0` The time step will be automatically set to a recommended time step. Your input density `rho` will be used.

`dt>0, rho=0` Your input time step will be used. An efficient density will be set to accommodate the input time step.

`dt=0, rho=0` The time step will be set to 1, and an efficient density will be set to accommodate this time step.

`dt<0, rho<0` The time step will be set to 1, and an efficient density will be set to accommodate this time step. This density will be revisited and adjusted after every time step.

11.1.37 `gravty(1)--gravty(3)` (f16.7)

Currently not supported. These three rows are place-holders for future features. Despite the name, this is not the acceleration of gravity that is used in the wave propagation algorithm. Use zeros for all three rows (floating point 0s).

11.1.38 p_o (f16.7)

Initial fluid pressure of pore liquid relative to atmospheric pressure (the atmospheric pressure is specified with p_atm, see Section 11.1.42). The input value is ignored unless the input iporo is 1, 3, or 4 (see Section 11.1.20). For this input parameter and for the next ten parameters, one is referred to [3] for guidance related to poromechanics models used in DEMPLA.

11.1.39 K_s (f16.7)

Bulk modulus of the grain bodies. For quartz, this is about 31.8d9 Pa. The input value is ignored unless the input iporo is 1, 3, or 4 (see Section 11.1.20).

11.1.40 K_f (f16.7)

Bulk modulus of the pore liquid. For water, this is about 2.2d9 Pa. The input value is ignored unless the input iporo is 1 or 3 (see Section 11.1.20).

11.1.41 S_o (f16.7)

The initial saturation of the pore fluid, at fluid pressure p_o (see Section 11.1.38) For dry soil, $S_o = 0.$; for fully saturated soil, $S_o = 1.0d0$; and for quasi-saturated soil, S_o is between 0.9 and 1. The input value is ignored unless the input iporo is 3 (see Section 11.1.20).

11.1.42 p_atm (f16.7)

Atmospheric pressure. Usually about 100.d3 Pa. The input value is ignored unless the input iporo is 3 (see Section 11.1.20).

11.1.43 Hcc (f16.7)

Henry's (dimensionless) coefficient of gas solubility in the pore liquid. For air in water, about 0.0187d0; for CO_2 in water, about 0.8780d0. The input value is ignored unless the input iporo is 3 (see Section 11.1.20). However, with iporo=3 and S_o=1, the formation of gas bubbles can be disallowed by setting Hcc=0.

11.1.44 gamm (f16.7)

Surface tension of the pore gas and liquid. For air and water, about 72.75d-3 N/m. The input value is ignored unless the input iporo is 3 (see Section 11.1.20).

11.1.45 D_o (f16.7)

Initial bubble diameter at pressure p_o and saturation S_o (see Sections 11.1.38 and 11.1.41). The input value is ignored unless the input iporo is 3 (see Section 11.1.20). When given, it should be about the ten percentile particle size (the size D_{10} in geotechnical parlance) or smaller. Note that there is redundancy in supplying the values of both D_o and N_o (see Section 11.1.46), so the following hierarchy is followed:

- D_o is only used when S_o<1.0 and N_o=0. In this case, D_o is used for computing N_o.
- When, S_o=1.0, then N_o is used and must be supplied. In this case, the emerging bubbles will have a diameter computed with N_o.
- N_o is used instead of D_o, when N_o is not zero. In this case, N_o is used for computing D_o.

11.1.46 N_o (f16.7)

The number density of bubbles per unit initial pore volume at pressure p_o and saturation S_o. For pore fluid that is initially saturated (Sø=1), any emerging bubbles (during cavitation) will have the number density N_o. The input value is ignored unless the input iporo is 3 (see Section 11.1.20). Note that there is redundancy in supplying the values of both D_o and N_o, so the hierarchy given in Section 11.1.45 is used to resolve conflicts.

11.1.47 p_vap (f16.7)

Vapor pressure of the pore liquid, expressed as an absolute pressure. For water, about 2337.d0 Pa. The input value is ignored unless the input iporo is 3 (see Section 11.1.20).

11.1.48 p_wcav (f16.7)

When when S_o=1, we can give a (non-zero) water pressure below atmospheric pressure p_atm at which bubbles will cavitate from the pore liquid. Parameter p_wcav is the water pressure below atmospheric pressure at which bubbles will emerge. For example, if S_o=1 and p_wcav=0, then the pore liquid is fully saturate with dissolved gas, and bubbles will appear when the pressure dips below atmospheric pressure. As another example, if S_o=1 and p_wcav=10 kPa, then the pore liquid is not fully saturated with dissolved gas, and bubbles will not appear until the pressure dips below the atmospheric pressure minus 10 kPa. Parameter p_wcav is a measure of the amount of dissolved gas within the pore liquid. The input value is ignored unless the input iporo is 3 (see Section 11.1.20).

11.1.49 rfree(15)--rfree(18) (f16.7)

Currently not supported. These eight rows are place-holders for future features. Use zeros for all eight inputs (floating point 0.s).

11.1.50 palpha (f16.7)

Shape factor for general contact asperity profiles, when imodel=9 (see Section 11.1.12). This parameter is the "α" in [8] and [6].

11.2 RunFile: Deformation-stress path section

The final section of a RunFile describes the manner in which either stresses or deformations are to be advanced. This section of the RunFile begins with five comment lines that are ignored by the program. These five lines are followed by a series of input lines, with each

line specifying its *segment* of the desired deformation-stress path. The lines are arranged sequentially (from the first segment to the last), with each line specifying a single segment of the deformation-stress path. Besides giving deformation-stress path information, these lines also determine the duration of each segment and specify supplementary actions to be taken at either the beginning or end of the segment. Each line contains 18 fields, arranged and formatted as follows:

```
format(
       i6, 1x,    icontr
       i6, 1x,    icontp
       f9.6, 1x,  defrat(1) (component 11)
       f9.6, 1x,  defrat(2) (component 22)
       f9.6, 1x,  defrat(3) (component 33)
       f9.6, 1x,  defrat(4) (component 12)
       f9.6, 1x,  defrat(5) (component 13)
       f9.6, 1x,  defrat(6) (component 23)
       f9.6, 1x,  defrat(8) (pressure or inflow)
       i2,1x,     igoal
       i1,1x,     krotat
       f9.6, 1x,  finalv
       i4,1x,     ipts
       i1,1x,     idump
       i2,1x,     iflexc
       i1,1x,     imicro
       i2,1x,     ibodyf
       f9.6, 1x,  defdot
       i4,1x,     ipts2
       i2         iplot
      )
```

Note that the input fields are separated with the blank character (`1x`) and are arranged horizontally *on a single line*. Each line defines a single *segment* of the deformation-stress path. The content of each input field is detailed in the remainder of this section. I suggest that you use the sample RunFiles on the GitHub site as a template for creating your own RunFiles.

11.2.1 `icontr (i6, 1x)`

The input variable `icontr` specifies the type of stress or deformation control. It does so by specifying six "components" of control.

The deformed shape of the assembly is described by an upper-triangular deformation gradient tensor $\mathbf{F}$ which we place alongside the corresponding components of the symmetric Cauchy stress tensor σ:

$$\mathbf{F} = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ 0 & F_{22} & F_{23} \\ 0 & 0 & F_{33} \end{bmatrix} \quad \sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \quad \texttt{icontr} \rightarrow \begin{bmatrix} 1 & 4 & 5 \\ & 2 & 6 \\ & & 3 \end{bmatrix} \tag{1}$$

where tensile stresses are positive. Note that for the diagonal stress components (σ_{11}, σ_{22}, or σ_{33}), we can control either the effective stress σ' or the total stress σ (with the tensile-positive convention, $\sigma_{ii} = \sigma'_{ii} - p$).

In the simplest form of `icontr`, you must control either the stress rate or the deformation rate for each of the six independent components (labeled 1 to 6 in the third matrix). For example, you could control the following combination of stress and deformation rates: $\dot{\sigma}_{11}$, $\dot{\sigma}_{22}$, $\dot{F}_{33}$, $\dot{F}_{12}$, $\dot{\sigma}_{13}$, and $\dot{F}_{23}$. In this simplest form, you must control either the stress rate or the deformation rate with each component; but you may *not* control *both* the stress and deformation rates of the same component, such as both $\dot{\sigma}_{13}$ and $\dot{F}_{13}$.

The input variable `icontr` specifies whether the deformation rate or the stress rate will be controlled for each of the six components. A "`0`" specifies deformation-control; a "`1`" specifies stress-control (specifically, the control of the intergranular *effective* stress). For example, an `icontr` value of `110010` would control the rates $\dot{\sigma}'_{11}$, $\dot{\sigma}'_{22}$, $\dot{F}_{33}$, $\dot{F}_{12}$, $\dot{\sigma}_{13}$, and $\dot{F}_{23}$, where the six digits of `icontr` are arranged in the order of components 11, 22, 33, 12, 13, and 23 (refer to the third matrix above for this mapping). In this example, note that the effective stresses $\dot{\sigma}'_{11}$ and $\dot{\sigma}'_{22}$ are being controlled The rates themselves are specified with `defrat` (Section 11.2.3).

Besides controlling deformation and/or stress with `icontr`, DEMPLA also allows control of either the inflow of pore fluid or the fluid pressure, with parameter `icontp`. Control of inflow or pressure can be used when `iporo` is 1, 3, or 4. See Section 11.2.2.

DEMPLA permits some other forms of stress and deformation control.

- Total stress control: instead of using a "1", use a "6" to control the total stress of components σ_{11}, σ_{22}, or σ_{33}.

- *Volume control*: when one or more of the first three digits of `icontr` is a "3", the corresponding deformations will be continually adjusted so that a given rate of volume change is maintained. The rate of volume change is given by the corresponding input variable `defrat` (Section 11.2.3). The volume rate is the computed as $d(F_{11}F_{22}F_{33})/dt$. The deformation rate is maintained by adjusting deformations of those components with the "3". For example if `icontr=133000`, then the 1st value of `defrat` gives the rate of stress change $\dot{\sigma}_{11}$; the 4th, 5th, and 6th values of `defrat` give the deformation rates $\dot{F}_{12}$, $\dot{\sigma}_{13}$, and $\dot{F}_{23}$; and the 2nd value of `defrat` gives the rate of volume change. In this situation, $\dot{F}_{22}$ and $\dot{F}_{33}$ will be continually adjusted to maintain that rate of volume change.

- *Mean stress control*: when one or more of the first three digits of `icontr` is a "2", the corresponding deformations will be continually adjusted so that a given rate of mean stress is maintained. For example if `icontr=022000`, then the 1st value of `defrat` gives the rate of deformation $\dot{F}_{11}$; the 4th, 5th, and 6th values of `defrat` give the deformation rates $\dot{F}_{12}$, $\dot{\sigma}_{13}$, and $\dot{F}_{23}$; and the 2nd value of `defrat` gives the rate of change of the mean stress. In this situation, $\dot{F}_{22}$ and $\dot{F}_{33}$ will be continually adjusted to maintain that rate of mean stress.

- *Deviator stress control*: when a single digit among the first three digits of `icontr` is a "4", the corresponding deformation will be continually adjusted so that a given rate of deviator stress is maintained. For example if `icontr=433000`, then the 1st value of

`defrat` gives the rate of change of the deviator stress $q = d(\sigma_{11} - \max(\sigma_{22}, \sigma_{33}))/dt$, noting that the stress components are usually negative. In this example, the 4th, 5th, and 6th values of `defrat` give the deformation rates $\dot{F}_{12}$, $\dot{\sigma}_{13}$, and $\dot{F}_{23}$. The 2nd value of `defrat` gives the rate of volume change, and $\dot{F}_{22}$ and $\dot{F}_{33}$ will be continually adjusted to maintain this rate of volume change.

11.2.2 `icontp (i1, 1x))`

When a poromechanic model is being used for the pore fluid (i.e., when `iporo` = 1, 3, or 4), then you must control the inflow rate of pore fluid (`icontp=0`) or the rate of the pore fluid pressure (`icontp=1`). The inflow rate is fluid strain (the negative of the divergence of the seepage velocity, rather than the discharge velocity). The input value is ignored unless the input `iporo` is 1, 3, or 4 (see Section 11.1.20).

11.2.3 `defrat(1-6) (6(f9.6, 1x))`

These are the six rates of either deformation or stress, as specified by `icontr` (Section 11.2.1). A deformation rate corresponds to a component of the rate of change of the deformation gradient, $\dot{\mathbf{F}}$. The stress rates are the rates of change of components of the Cauchy stress tensor. The stress rate can be the rate of the effective stress (when the corresponding digit in `icontr` is "0") or the rate of the total stress (when the corresponding digit in `icontr` is "6") Note that compressive stress is negative, and that there is not distinction between effective and total stress with the shearing components of `defrat(4)`, `defrat(5)`, and `defrat(6)`. At present, there is no provision for rotational springs at the particle contacts, so that the computed stress tensor components are very nearly symmetric ($\sigma_{ij} \approx \sigma_{ji}$, within the numerical precision of the computations), and, of course, there will be no couple stresses.

11.2.4 `defrat(8) (f9.6, 1x)`

When `icontp=0`, then `defrat(8)` gives the rate of fluid inflow. When `icontp=1`, then `defrat(8)` gives the rate of change of the fluid pressure. For example, for undrained conditions, `icontp=0` and `defrat(8)=0.` For drained conditions, in which the back-pressure is controlled and maintained constant, `icontp=1` and `defrat(8)=0.`

11.2.5 `igoal (i2, 1x)`

The input variables `igoal` and `finalv` determine the duration of the deformation-stress segment (see also Section 11.2.7). The duration is determined by monitoring just one of the six components of either stress or deformation (11, 22, 33, 12, 13, and 23) and whether that component has stepped across a given threshold value. The threshold value is specified with the input variable `finalv` (Section 11.2.7). The variable `igoal` is a 2-digit integer. The first digit is from `1` to `8`. Except when it is `7`, the first digit specifies which component will be monitored Note that with a poromechanic model (`iporo=1, 3, or 4`), a digit `8` corresponds to either inflow or pressure being monitored, to determine the duration of the deformation-stress segment. The second digit is either `0`, `1`, `2`, or `4`. When the digit is `0`, the deformation threshold is monitored; when the digit is `1`, the effective stress threshold is

monitored; when the digit is 2, the total stress threshold is monitored; and when the digit is 4, a deviator stress threshold is monitored. The exception to this scheme is when the first digit is 7, which specifies that the control segment will run for a fixed period of time. Several examples follow.

`igoal=51` The effective stress component σ'_{13} (first digit) will be monitored (the fifth component, 13, of stress, noting that the effective and total stresses are the same for shearing components). The particular deformation-stress control segment will finish when σ_{13} crosses the input threshold `finalv` from either above or below. For example, if the shear stress σ_{13} is 103.77 at the beginning of the control segment and `finalv` is 50.0, then the control segment will be finished when σ_{13} is reduced to 50.0 or less.

`igoal=32` The stress component σ_{33} (first digit) will be monitored (the third component, 33, of stress). The particular deformation-stress control segment will finish when the total stress σ_{33} crosses the input threshold `finalv` from either above or below.

`igoal=20` The deformation component F_{22} (first digit) will be monitored (the second component, 22, of deformation). If the deformation F_{22} is 0.745 at the beginning of the control segment and `finalv` is 0.800, then the control segment will be finished when F_{22} has increased to 0.800 or greater.

`igoal=81` The poromechanic component (first digit, 8) will be monitored. The second digit is "1", so the pressure will be monitored, and the deformation-stress segment will finish when the pressure crosses over `finalv`. If the second digit was "1", then the segment would end when the accumulated inflow crosses over `finalv`.

`igoal=70` The control segment will proceed for a given duration of time, as specified with `finalv`. The same result is achieved regardless of the second digit (74, 78, etc.). I almost always use this scheme to specify the duration of a control segment. For example, if `igoal`=70, `finalv`=2.500, and `dt`=0.25, then the control segment will be exited after 10 time steps ($10 \times 0.250 = 2.500$).

`igoal=14` The deviator stress corresponding to stress 11 will be monitored: the stress $q = \sigma_{11} - \max(\sigma_{22}, \sigma_{33})$. The particular deformation-stress control segment will finish when this deviator stress crosses the input threshold `finalv` from either above or below.

When assigning values to `finalv`, the stresses σ_{11}, σ_{22}, and σ_{33} are usually negative (compressive).

It is can sometimes be difficult to foresee the direction in which a particular stress (or deformation) component will be moving, and so the specified component may actually move further away from the anticipated target `finalv`. For this reason, I usually just use a value of 70 for `igoal`.

11.2.6 krotat (i1, 1x)

Currently not supported. Use zero (integer 0).

11.2.7 finalv (f9.6, 1x)

See the discussion of igoal, Section 11.2.5.

11.2.8 ipts (i4, 1x)

The program DEMPLA can create *lots* of output. Simulations may require many thousands of time steps. The input variable ipts allows you to choose the frequency at which results are appended to the output files. For example, when ipts is 50 then results will be output to the A- and B-files every 50th time step (Sections 13.1–13.4).

11.2.9 idump (i1, 1x)

As has already been mentioned in regard to the input variable iend, the program allows you to "dump" a binary "restart" file at the *end* of a control segment (Sections 11.1.7 and 11.1.8). This binary file can later be used to start a new simulation from the exact conditions that were present at the time the restart file was created. You can use the field idump to dump a restart file in the middle of a simulation. Also see Sections 11.1.7 and 11.1.8.

idump=0 Do not dump a binary restart file at the end of this control segment.

idump=1 Dump a binary restart file at the end of this control segment.

The name of the output file will have following form:

C?-<RunFile>

where the "?" is a three digit number (e.g., 007) that corresponds to the particular segment of the deformation-stress path at which the file was created.

11.2.10 iflexc (i2, 1x)

Currently not supported. Use zero (integer 0).

11.2.11 imicro (i1, 1x)

DEMPLA can create a set of data files for use in post-processing the micro-level behavior of the assembly. These data files are described in Section 13.5. They can be used as input to your own Matlab, c, Fortran, Octave, Scilab, or R data analysis programs.

11.2.12 ibodyf (i2, 1x)

Currently not supported. Use zero (integer 0).

11.2.13 defdot (f6.5, 1x)

Currently not supported. Use zero (integer 0).

11.2.14 ipts2 (i4, 1x)

Currently not supported. Use zero (integer 0).

11.2.15 iplot (i2, 1x)

Currently not supported. Use zero (integer 0).

12 StartFile: The initial particle arrangement

The StartFile provides the initial particle arrangement, including the particle sizes, shapes, orientations, and positions. There are two types of StartFiles, with the type indicated by the leading character of the StartFile name: `C` or `D`. Since only the DStartFile is readable as a text (ASCII) file, its contents will be described in this section. The nature and purpose of CStartFiles is described in Sections 6, 11.1.7, and 11.1.8.

Sample D-files can be found in the `StartFiles` directory of the GitHub DEMPLA repository, and descriptions of these assemblies are given in Section 16.

A D-file begins with either three or four lines that give general information about the assembly:

1. the particle shape (at present, the program does not allow the mixing of particle types within the same assembly). See Section 12.1.1.

2. the number of particles and the overall dimensions of the assembly (i.e. the distances between the periodic boundaries). See Section 12.1.2 and Fig. 9.

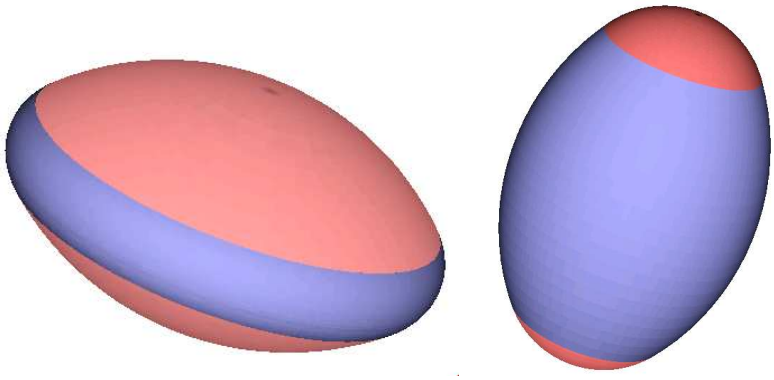
3. the shear offset distances (Section 12.1.3 and Fig. 9).

4. for oval and ovoid particles, an angle β that describes the manner in which circular arcs are spliced together to create oval and ovoid particles (Section 10). This angle must be given in *degrees*. This line of input is *only required for the oval and ovoid particle types*. Section 12.1.4 discusses limitations on the input value of β.

5. for bumpy clumps of spheres, the number of satellite spheres (Section 12.1.5) and the size of the central sphere and the relative size and offSet of the satellite spheres relative to the central sphere (Section 12.1.6).

These three (or four) lines are followed by information on each particle, with one line per particle. For example, the D-file for an assembly of 1002 circular particles might begin with

```
1
  1002  2.9899456327673043OE+01  2.9844688999321907OE+01  1.0000000000000000E+00
        0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  3.1345406371041857OE-01  1.7935286874155107OE+01  1.7926677757454375OE+01
  4.4375665469307011OE-01  1.2769811960928081OE+01  2.5276054823897219OE+01
etc.
```



(a) Oblate (b) Prolate

Figure 6: Ovoid particles.

The detailed contents of the first three or four lines are described in the following section. The remaining lines in the StartFile are described in Section 12.2.

12.1 Assembly data in D-StartFiles

12.1.1 `kshape (i1)`

The first column of the first input line should contain an integer of 1 through 7, which will designate the type of particle.

`kshape=1` circular (2D) disks

`kshape=2` oval (2D) disks. The ovals are composed of four circular arcs spliced together (Fig. 10). Although a more general variety of shapes can be formed from four (or more) arcs, the program currently supports only bi-symmetric ovals.

`kshape=3` elliptical (2D) disks. The code for this has not been recently tested and might not be stable.

`kshape=4` spheres (3D)

`kshape=5` ovoids (3D), a non-spherical shape that is composed of two spherical caps and a torus center (Section 6). The particle is smooth and strictly convex.

`kshape=6` nobbies (2D), a non-circular shape that is composed of clusters of circles (Fig. 7). The particle is neither smooth nor convex.

`kshape=7` bumpies (3D), a non-spherical shape that is composed of clusters of spheres (Fig. 8). The particle is neither smooth nor convex.

12.1.2 `np,xcell(1,1),xcell(2,2),xcell(3,3) (i6,3(1pd25.17))`

The number of particles, `np`, should appear within the first six columns, and the three dimensions of the assembly should be presented in double precision format, with 25 columns

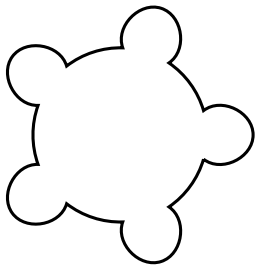


Figure 7: Nobby particle: `nobs=5`, `satrad=0.30`, and `cenrad=0.85`

per dimension (Fig. 9). These dimensions are simply the spacings between opposing periodic boundaries. The value of `xcell(3,3)` must be given, even with 3D assemblies (any value will work, since it will be ignored in the simulation).

12.1.3 `xcell(1,2),xcell(1,3),xcell(2,3) (6x,3(1pd25.17))`

After six (blank) columns, the three shear offsets should be given in double precision format, with 25 columns per offset (see the offset `xcell(1,2)` in Fig. 9.

12.1.4 `beta (1pd25.17)` — oval and ovoid particles only!

When 2D ovals or 3D ovoids are being used, this fourth line contains the splice angle, `beta`, in *degrees* (Fig. 10). The angle β must be greater than zero and no greater than 90°. The particle aspect ratio will be limited by your choice of β (Sections 12.2.2. The height/width ratio must lie within the following bounds:

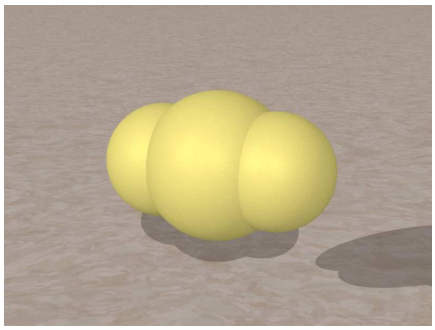
$$\frac{1-\cos\beta}{\sin\beta} < \alpha < \frac{\cos\beta}{1-\sin\beta} \,. \tag{2}$$

12.1.5 `nobs` or `nbumps (* integer)` — nobby (2D) and bumpy (3D) particles only!

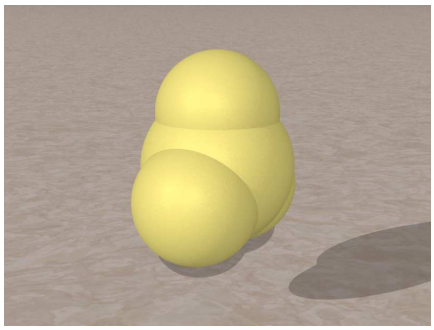
When 2D nobby particles are being used, this line contains the integer number of satellite circles (arcs) that surround a central circle (see Section 7). The input variable `nobs` gives the integer number of satellite circles. These circles are equally spaced around the central circles. The satellite circles are centered on a "circumscribing circle" of radius 1.0.

When 3D bumpy particles are being used, this line contains the integer number of satellite spheres that surround a central sphere (see Fig. 8). DEMPLA currently restricts `nbumps` to the following five values: 2, 3, 4, 6, or 8, with examples shown in Fig. 8. In each case, the satellite spheres of a bumpy particles will be centered on the surface of a "circumscribing sphere" (i.e., the satellite spheres are not necessarily centered on the surface of the central sphere). The component spheres of bumpy (3D) particles have the following arrangements:

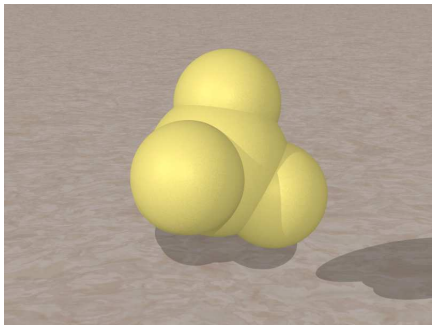
`nbumps=2` Two satellite spheres are place on opposite sides of the central sphere, diametrically opposed. The satellite spheres are centered on the surface of a "circumscribing sphere." See Fig. 8a.



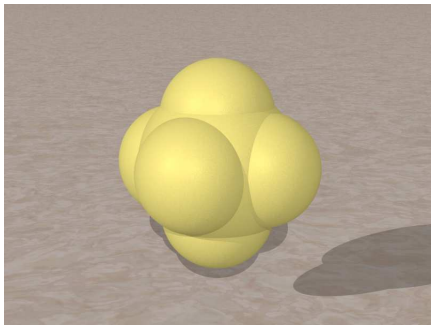
(a) `nbumps=2`, `satrad=0.7`, `cenrad=0.9`, `cirrad=0.75`



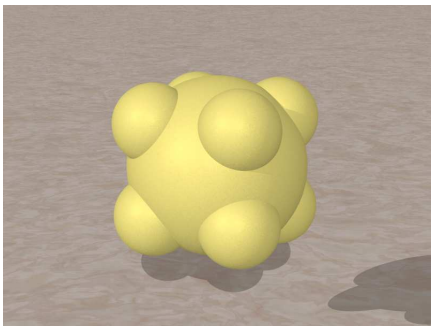
(b) `nbumps=3`, `satrad=0.8`, `cenrad=1.0`, `cirrad=0.7`



(c) `nbumps=4`, `satrad=0.6`, `cenrad=0.8`, `cirrad=0.7`



(d) `nbumps=6`, `satrad=0.7`, `cenrad=1.0`, `cirrad=0.6`



(e) `nbumps=8`, `satrad=0.4`, `cenrad=0.9`, `cirrad=0.8`

Figure 8: Bumpy particles.

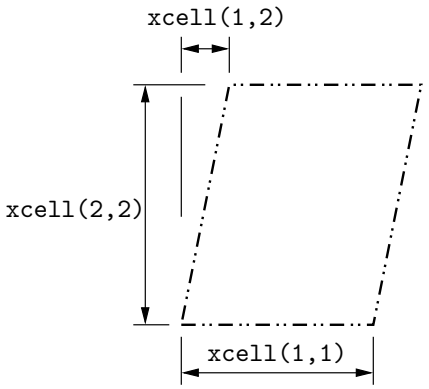


Figure 9: The `xcell(i,j)` dimensions for a 2D assembly

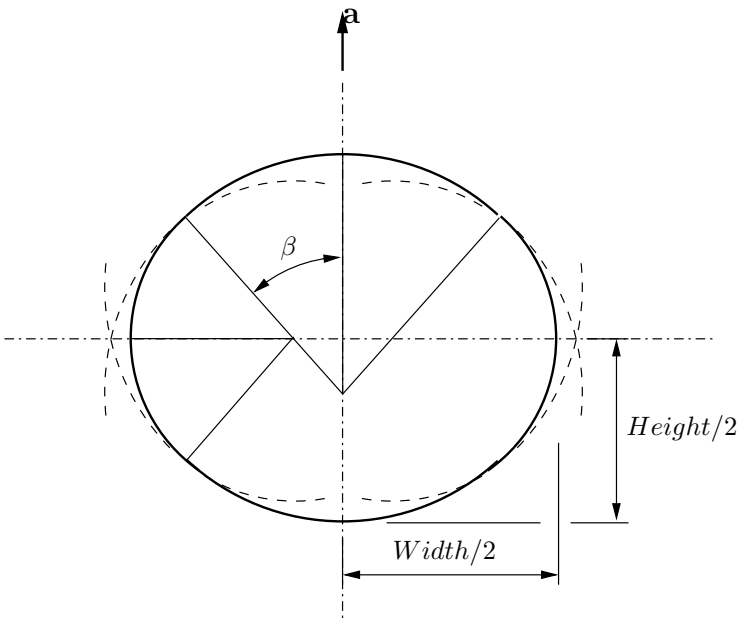


Figure 10: Geometry of an oval composed of four circular arcs.

`nbumps=3` Three satellite spheres are placed around the central sphere. The satellite spheres are centered on the vertices of an equilateral triangle that is circumscribed by the circumscribing sphere. That is, the satellite spheres are equally spaced along the equator of the circumscribing sphere. See Fig. 8b.

`nbumps=4` Four satellite spheres are placed around the central sphere. The satellite spheres are centered on the vertices of a tetrahedron that is circumscribed by the circumscribing sphere. That is, the satellite spheres are equally spaced around the full surface of the circumscribing sphere. See Fig. 8c.

`nbumps=6` Six satellite spheres are placed around the central sphere. The satellite spheres are centered on the vertices of a octahedron that is circumscribed by the circumscribing sphere. That is, the satellite spheres are equally spaced around the full surface of the circumscribing sphere. See Fig. 8d.

`nbumps=8` Eight satellite spheres are placed around the central sphere. The satellite spheres are centered on the vertices of a cube that is circumscribed by the circumscribing sphere. That is, the satellite spheres are equally spaced around the full surface of the circumscribing sphere. See Fig. 8e.

Regardless of whether nobbies or bumpies are being used, this line of input must be followed by a line that gives relative sizes of the circles or spheres that comprise a particle, as in the following section (Section 12.1.6).

12.1.6 `satrad, cenrad, cirrad` `(* double precision)` — nobby (2D) bumpy and (3D) particles only!

These values are only used with nobby (2D) bumpy (3D) particles, and they are placed on a single line in double-precision format separated with spaces.

Every particle in an assembly will have the same shape, but they can have different sizes. Their common shape is specified with the values of `nobs`, `nbumps`, `satrad`, `cenrad`, and `cirrad`. These values give a "base" shape and size for the particles. The actual size of each particle is equal to the common "base size" multiplied by an individual scaling radius for each particle (Sections 12.2.6 and 12.2.7).

The input variables `satrad` and `cenrad` are the base sizes of the satellite and central circles or spheres of the nobby (2D) and bumpy (3D) cluster particles. The input variable `cirrad` is the radius of circumscribing sphere, as described in Section 12.1.5. DEMPLA currently does not support `cirrad` for 2D nobby particles: the satellite circles are centered on a circumscribing circle of radius 1.0.

12.2 Particle data in D-StartFiles

Following the general information at the head of a DStartFile, information is given on each particle, with one line per particle. The arrangement of this data depends upon the type of particle.

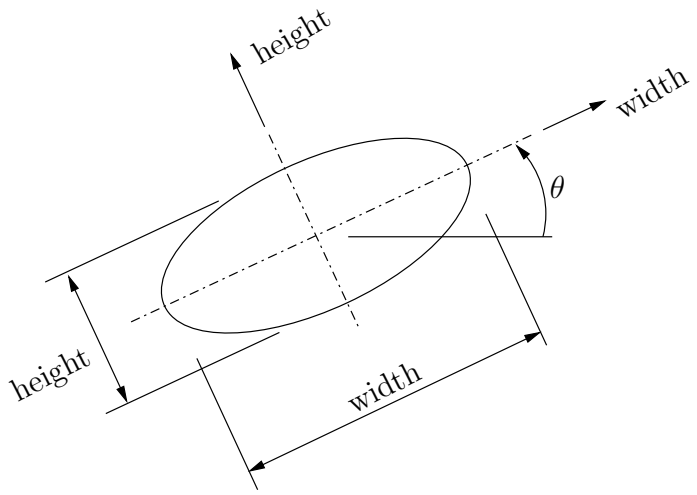


Figure 11: Orientation angle θ for elongated particles (2D ovals and ellipses).

12.2.1 Circle (disk) particle data

Three fields give the radius, the x_1–location, and the x_2–location of the particle. The location coordinates refer to the center of the particle. Following the general information at the head of a DStartFile, this information is given for each circle, with each circle beginning on a new line. The data items are listed below. Each data field is in `1pd25.17` format, with spaces or a new line between the fields. The data fields are given as double-precision values separated by spaces, one particle per line. The example input on page 56 shows two lines of data for circular particles.

- Radius of the circle.
- x_1–location of the circle's center.
- x_2–location of the circle's center.

12.2.2 Oval particle data

Five fields give the oval width, the ratio of height divided by the width, the x_1–location, the x_2–location, and the orientation angle (in *degrees*) of particle, measured counterclockwise from the x_1-direction (Fig. 11). The location coordinates refer to the center of the particle. Following the general information at the head of a DStartFile, this information is given for each oval, with each oval beginning on a new line. The data items are listed below. Each data item is given in `1pd25.17` format, with spaces or a new line between the fields. The data fields are given as double-precision values separated by spaces, one particle per line.

- Half-width of the oval. Note that the oval width may be greater or less than the height (with ellipses, the width must be greater than its height). The width is measured as shown in Fig. 11, at an angle of θ counterclockwise from the horizontal x_1 axis. As input, you should give one half of the full particle width.

- Ratio of the height divided by the width of the particle (Fig. 11). For ovals, the ratio may be greater or less than one.
- x_1–location of the oval's center.
- x_2–location of the oval's center.
- Orientation angle θ of the oval's width (in degrees, Fig. 11).

12.2.3 Ellipse particle data

The code for this has not been recently tested and might not be stable. Five fields give the major radius, the ratio of the minor radius divided by the major radius (a number greater than zero, but no greater than one), the x_1-location, the x_2-location, and the orientation angle (in *degrees*) of the major axis, measured counterclockwise from the x_1-direction (Fig. 11). The location coordinates refer to the center of the ellipse. Following the general information at the head of a DStartFile, this information is given for each ellipse, with each ellipse beginning on a new line. The data items are listed below. Each data item is given in `1pd25.17` format, with spaces or a new line between the fields. The data fields are given as double-precision values separated by spaces, one particle per line. At present, the ellipse width must be greater than its height, with a height-to-width ratio less than one.

- Major radius of the ellipse.
- Ratio of the minor radius divided by the major radius of the particle (Fig. 11). For ellipses, the ratio must be greater than zero, but no greater than one.
- x_1–location of the ellipse's center.
- x_2–location of the ellipse's center.
- Orientation angle θ of the ellipse's width (in degrees, Fig. 11).

12.2.4 Sphere particle data

Four fields give the radius, the x_1–location, the x_2–location, and the x_3–location of the particle. The location coordinates refer to the center of the particle. Following the general information at the head of a DStartFile, this information is given for each sphere, with each sphere beginning on a new line. The data items are listed below. Each data item is given in `1pd25.17` format, with spaces or a new line between the fields. The data fields are given as double-precision values separated by spaces, one particle per line.

- Radius of the sphere.
- x_1–location of the sphere's center.
- x_2–location of the sphere's center.
- x_3–location of the sphere's center.

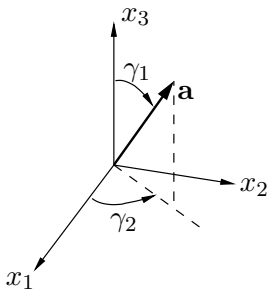


Figure 12: Orientation angles γ_1 and γ_2 for 3D ovoid particles. Vector **a** is the central axis of revolution of the ovoid (Fig. 10 and Fig. 11).

12.2.5 Ovoid particle data

Seven fields give the ovoid's revolved radius, the ratio of the axial radius divided by the revolved radius, the location coordinates, and the orientation angles γ_1 and γ_2 of the ovoid's axis of revolution, **a**. An ovoid is formed by rotating an oval (i.e. Fig. 10) about its central axis **a**. The location coordinates refer to the center of the particle. Following the general information at the head of a DStartFile, this information is given for each ovoid, with each ovoid beginning on a new line. The data items are listed below. Each data item is given in `1pd25.17` format, with spaces or a new line between the fields. The data fields are given as double-precision values separated by spaces, one particle per line.

- Half of the transverse (revolved, half) width of the ovoid. This half-width is measured perpendicular to the central axis of revolution, **a**.
- Ratio of the axial height divided by the transverse width. The ratio can be greater or less than one, but it must be greater than zero. Ratios less than one are oblate; ratios greater than one are prolate (Fig. 6). Note that large aspect ratios (or small inverse ratios) require much more computation time. I would not recommend using a ratio greater than 2 or less than 0.5.
- x_1–location of the ovoid's center.
- x_2–location of the ovoid's center.
- x_3–location of the ovoid's center.
- γ_1 orientation angle (in radians) of the ovoid's axis of revolution, **a** (Fig. 12). This angle should be no less than zero and no greater than 90°.
- γ_2 orientation angle (in radians) of the ovoid's axis of revolution **a** (Fig. 12).

12.2.6 Nobby (2D) particle data

Four fields give the scaling radius, the x_1–location, the x_2–location, and the orientation angle (in degrees) of the particle. The location coordinates refer to the center of the particle. Following the general information at the head of a DStartFile, this information is given for

each nobby, with each nobby beginning on a new line. The data items are listed below. Each data item is given in `1pd25.17` format, with spaces or a new line between the fields. The data fields are given as double-precision values separated by spaces, one particle per line.

- Scaling radius of the nobby particle. This scaling radius is multiplied by the base radii, `satrad` and `cenrad`, as described in Section 12.1.6. These products give the actual sizes of the radii that comprise the nobby particle.
- x_1–location of the nobby particle's center.
- x_2–location of the nobby particle's center.
- θ_3 orientation angle (in degrees) of the particle. The center of one of the satellite particles is located at angle θ_3, measured counter-clockwise from the x_1–direction. The other satellite circles are equally spaced at angle 360°/`nobs`.

12.2.7 Bumpy (3D) particle data

Eight fields give the scaling radius, the x_1–location, the x_2–location, the x_3–location, and a four-component unit quaternion of the particle's orientation. The location coordinates refer to the center of the particle. Following the general information at the head of a DStartFile, this information is given for each bumpy, with each bumpy beginning on a new line. The data items are listed below. Each data item is given in `1pd25.17` format, with spaces or a new line between the fields. The data fields are given as double-precision values separated by spaces, one particle per line.

- Scaling radius of the bumpy particle. This scaling radius is multiplied by the base radii (`satrad`, `cenrad`, and `cirrad`) as described in Section 12.1.6. These products give the actual sizes of the radii that comprise the nobby particle.
- x_1–location of the nobby particle's center.
- x_2–location of the nobby particle's center.
- x_3–location of the nobby particle's center.
- $\mathtt{Qp}_1$ orientation quaternion of the particle.
- $\mathtt{Qp}_2$ orientation quaternion of the particle.
- $\mathtt{Qp}_3$ orientation quaternion of the particle.
- $\mathtt{Qp}_4$ orientation quaternion of the particle.

13 Text output files from Dempla

While DEMPLA is running, output is periodically written to files. In Mode 1 (analysis of a single assembly), two output files are produced: an A-file and a B-file. In Mode 2 (wave propagation analysis), several output files with a `Results_` prefix give information about the multiple DEM assemblies (RVEs) that comprise the soil column. In addition, the Mode 2 parameter `iABfile` in the GFile specifies whether to create an A-file and a B-file for each DEM assembly (Section 7.1.5). The frequency (number of time steps between writing of output) is specified with the input variable `ipts` (Section 11.2.8). These two files contain macro-data, such as stress and deformation. A-files and B-files are described in Sections 13.1–13.4. You can also produce F-files, which will contain micro-level data. Because F-files can be quite large, each creation of these files must be triggered by the input value `imicro` in the RunFile (Section 11.2.11). F-files are described in Sections 13.5–13.7.

13.1 A-files: macro-data for spreadsheets

These files contain macro-data, such as stress, deformation, and fabric from a simulation. Mode 1 simulation will create a file named `A<RunFile>.txt`, which will be referred to as simply an "A-file". The file contains macro-data, such as stress, deformation, and fabric. In Mode 2, an A-file can be created for each DEM assembly, by setting the parameter `iABfile` equal to a non-zero integer in the GFile (Section 7.1.5). The frequency (number of time steps between writing of output) is specified with the RunFile variable `ipts` for Mode 1 (Section 11.2.8), and by the GFile variable `nDEM_out` in for Mode 2 (Section 7.1.17).

The fields in an A-file are separated by Tab characters, so that A-files can be imported into a spreadsheet (note: *imported* not opened). When importing, you will want to specify the columns as being "tab separated." The spreadsheet is headed with some general information about the simulation, followed by a history of time, strains, and stresses. The strains are reported with the simple measure

$$F_{ij} - \delta_{ij} \tag{3}$$

where F_{ij} are components of the deformation gradient tensor. Note that the shear deformations are reported as shear angles, like F_{12}, or twice the shear strain (a γ-strain, not an ε-strain). Stresses and strains are reported at the interval `ipts`, which may differ for each segment of the deformation-stress path (Sections 11.2 and 11.2.8).

For 2D assemblies, the A-file includes information on the assembly's particle graph [9, 10, 11]. This information includes the numbers of vertices (particles that are included in the particle graph), edges (contacts between particles), and face (void cells).

For both 2D and 3D assemblies, the A-file includes information on the *fabric tensor* for the assembly [12]. We will refer to the fabric tensor with the symbol $\mathbf{A}$, which is defined with a sum over contacts within the assembly. Note that in an A-file, this tensor $\mathbf{A}$ is unfortunately labeled as "`F`", with the components `F(1,1)`, `F(2,2)`, etc.

$$A_{ij} = \frac{1}{N} \sum_{k=1}^{N} n_i^k n_j^k \,, \tag{4}$$

where $\mathbf{n}^k$ is the (unit vector) direction of the kth branch vector (contact), and N is the number of contacts in the assembly. When computing (4), N includes only those contacts between particles that each have at least 2 contacts (2D assemblies) or 3 contacts (3D assemblies).

For both 2D and 3D assemblies, the A-file also includes information on the fabric tensor $\mathbf{A}^s$ of only those ("strong") contacts that carry a greater-than-average force. Note that in an A-file, this tensor $\mathbf{A}$ is unfortunately labeled as "`Fs`", with the components `Fs(1,1)`, `Fs(2,2)`, etc. This tensor is defined as a sum over the strong contacts within the assembly:

$$A_{ij}^s = \frac{1}{N} \sum_{k \in \mathcal{S}} n_i^k n_j^k \ , \tag{5}$$

Where the set of contacts $\mathcal{S}$ is a set of contacts k:

$$\mathcal{S} = \{k : |\mathbf{f}^k| > \overline{f}\} \tag{6}$$

that have a force magnitude $|\mathbf{f}^k|$ greater than average:

$$\overline{f} = \sqrt{\frac{\sum_{k=1}^{N} |\mathbf{f}^k|^2}{N}} \tag{7}$$

The A-file also reports the proportion v of strong contacts in the assembly.

An A-file contains several more columns of data. The labels of these columns explain their content, and many are given the same names as quantities that are described below for B-files (Section 13.2).

13.2 B-files: macro-data text files

These files contain macro-data, such as stress, deformation, and fabric from a simulation. A Mode 1 simulation will create a file named `B<RunFile>`, which will be referred to as simply an "B-file". The file contains macro-data, such as stress, deformation, and fabric. In Mode 2, a B-file can be created for each DEM assembly, by setting the parameter `iABfile` equal to a non-zero integer in the GFile (Section 7.1.5). The frequency (number of time steps between writing of output) is specified with the RunFile variable `ipts` for Mode 1 (Section 11.2.8), and by the GFile variable `nDEM_out` in for Mode 2 (Section 7.1.17).

The `B<RunFile>` (or just "B"-files) contain more information than A-files—not only stresses and deformations, but information that reflects the numerical performance of the simulation. For example, the B-file contains information that can characterize whether the simulation was nearly pseudo-static (`chi1`, `chi2`, `chi3`, `chi4`, and `knrgy`), the relative importance of viscous damping during the simulation (`chi3`, `chi4`, `viscbt`, and `viscct`), and whether the controlled stresses were maintained near their target values (`psi`). This information, although useful, is packed into a text (ASCII) file that can be somewhat difficult to read and understand. B-files are described in the following two sections.

13.3 B-files with 2D simulations

The first four lines in a B-file contain the following four pieces of information in the form of `format(i4,2x,a50,2x,a20)`:

- an integer that indicates the format (version) of the B-file (`i4` format). For example, whether the file is for a 2D or 3D simulation, since B-files 2D and 3D will contain different information.
- the name of the StartFile that was used for the simulation (Section 6 and Section 12)
- the version of the DEMPLA source code
- the RVE number (`i4.4` format). For Mode 1 simulation, this number is `0001`. For Mode 2 simulations, the RVE number begins with `0001` for the bottom-most RVE in the soil column to `nRVEs` for the top-most RVE (see Section 7.1.3).

These for lines are followed by the results of each output cycle, which occur at the frequency `ipts` for Mode 1 (see Section 11.2.8) or `nDEM_out` for Mode 2 (see Section 7.1.17).

For 2D simulations, the information for each output cycle is packed into just three lines, such as the following:

```
1.0320000E-04   5.948067E-05 -1.984000E-04  0.000000E+00  4.212188E-04     1623
  6.49E-04     -1.551593E+04 -2.071540E+04 -2.774535E+01  9.690910E+00 3.29E-07
  1.03E-03 0.00E+00 7.11E-04  6.519467E-03  5.346882E-02  2.655151E+00     3.00
```

The contents of these three lines correspond to the following variables (some of these are, in fact, arrays):

```
  timer          defout(1,1)   defout(2,2)   defout(1,2)    knrgy          ntacts
   chi1          stress(1,1)   stress(2,2)   stress(1,2)    pnrgy            psi
   chi2    chi3      chi4       viscbt        slidet         work1t         xloops
```

that are given in the following formats:

```
 1pe14.7,   1x,  1pe14.6,     1pe14.6,      1pe14.6,       1pe14.6,       i9,/,
 2x,1pe9.2, 4x,  1pe14.6,     1pe14.6,      1pe14.6,       1pe14.6,   1pe9.2,/,
 2x,1pe9.2, 1pe9.2, 1pe9.2,   1pe14.6,      1pe14.6,       1pe14.6,   0pf9.2
```

The various output values are now described.

13.3.1 `timer`

The accumulated time, which advances by an amount `dt` with each deformation step (Section 11.1.36).

13.3.2 `defout(i,j)`

The output values, `defout(i,j)`, are the difference between the deformation gradient matrix F_{ij} and the identity (Kronecker) matrix δ_{ij}, as in Eq. 3 on page 66. For example, the output

```
defout(1,1) =  0.001
defout(2,2) = -0.002
defout(1,2) =  0.003
```

for a 2D assembly corresponds to the following deformation gradient:

$$\mathbf{F} = \begin{bmatrix} 1.001 & 0.003 \\ 0 & 0.998 \end{bmatrix} \tag{8}$$

The deformation gradient $\mathbf{F}$ is referenced to the initial assembly (except when the simulation is started with a C-StartFile, in which case, $\mathbf{F}$ is carried over from a previous run).

13.3.3 `knrgy`

The kinetic energy of particle motions per unit of the assembly's original volume. The kinetic energy is computed from both translational and rotational velocities of the particles:

$$\frac{1}{\text{Initial volume}} \times \frac{1}{2} \sum_{\text{Particles}} \left(m\bar{\mathbf{v}}^2 + I\bar{\boldsymbol{\omega}}^2 \right) \ , \tag{9}$$

where m and I are the mass and the mass moment of inertia of a particle, and $\bar{\mathbf{v}}$ and $\bar{\boldsymbol{\omega}}$ are the average velocity and angular velocity of a particle (the averages of the velocities at the two times $t - dt/2$ and $t + dt/2$). The energy `knrgy` is not really meaningful when `algori=2` (see Section 11.1.2). Note that $F_{ij} = 0$ for $i > j$, as is the case in Fig. 9.

13.3.4 `ntacts`

The number of contacts. Here, a contact is shared by two particles. If you prefer to count a contact twice (once for each of the two particles, as in Table 5, then the number of contacts is, instead, `ntacts`×2.

13.3.5 `chi1`

One of four measures for determining whether the simulation is nearly pseudo-static. The property `chi1` is the average force imbalance on a particle divided by the average magnitude of a contact force. Small values signify that the particles were nearly in equilibrium during the deformation process. When `algori=2`, the variables `chi1` and `chi2` (Section 13.3.9) are used as a near-equilibrium criteria for controlling the pace at which the deformation is allowed to proceed (Section 11.1.2 and Section 13.3.15). If there are no contacts within the assembly, then `chi1` will be zero.

13.3.6 `stress(i,j)`

Components of the Cauchy stress tensor.

13.3.7 `pnrgy`

The elastic energy stored in the contact springs per unit of the assembly's original volume. For the simple linear contact mechanism, this energy is calculated as

$$\frac{1}{\text{Initial volume}} \times \frac{1}{2} \sum_{\text{Contacts}} \left[\frac{f_n^2}{k_n} + \frac{f_t^2}{k_t} \right] \ , \tag{10}$$

where f_n and f_t are the normal and tangential contact forces, and k_n and k_t are the normal and tangential contact stiffnesses at time t.

13.3.8 `psi`

This performance measure characterizes the fluctuation of the controlled stresses from their intended (target) values. When one or more digits in `icontr` are `1`'s, then a servo-control

algorithm will attempt to control certain components of the Cauchy stress and maintain them at target values (see Section 11.2.1). The property `psi` is the sum of the deviations of the controlled stress components from their target values divided by the mean stress (either 1/2 or 1/3 σ_{kk}). If you are not controlling any of the stress components, then `psi` will be zero.

13.3.9 `chi2`

This is another measure of whether the simulation is nearly pseudo-static (see `chi1`, Section 13.3.5). The value `chi2` is the average *moment* imbalance on a particle divided by both the average magnitude of a contact force and the average particle radius. Small values signify that particles remained nearly in equilibrium during a simulation.

13.3.10 `chi3`

A measure of whether viscous contact damping could have a significant effect on the reported stresses and deformations. Its value is the viscous force at an average contact divided by the average contact force.

13.3.11 `chi4`

A measure of whether viscous body damping could be having a significant effect on the reported stresses and deformations. Its value is the viscous force on an average particle divided by the average contact force.

13.3.12 `viscbt`

The energy expended in viscous body damping per unit of the assembly's original volume (cumulative since the beginning of the simulation). This quantity may not be meaningful when `algori=2` (see Section 11.1.2). The *increment* in `viscbt` for a time step dt is computed as follows:

$$\frac{1}{\text{Initial volume}} \times \sum_{\text{Particles}} \left[\eta^v \bar{\mathbf{v}} \cdot \bar{\mathbf{v}} dt + \eta^\omega \bar{\boldsymbol{\omega}} \cdot \bar{\boldsymbol{\omega}} dt\right] \ . \tag{11}$$

That is, the increment of work expended in body damping is equal the damping force ($\eta^v \mathbf{v}$) multiplied with the particle movement ($\mathbf{v}dt$). In this equation, η is the damping coefficient. The equation includes separate contributions of translational and rotational damping. The velocities $\bar{v}$ and $\bar{\boldsymbol{\omega}}$ are the averages for a particle at times $t - dt/2$ and $t + dt/2$.

13.3.13 `slidet`

The energy expended in frictional sliding per unit of the assembly's original volume (cumulative since the beginning of the simulation). The *increment* in `slidet` for a time step dt is computed as follows:

$$\frac{1}{\text{Initial volume}} \times \sum_{\substack{\text{Sliding} \\ \text{contacts}}} \left[\mu f^n \left|\bar{\mathbf{v}}^{\text{contact, } t}\right| dt\right] \ , \tag{12}$$

where μ is the friction coefficient `frict`, f^n is the normal contact force, and $\bar{\mathbf{v}}^{\text{contact},\,t}$ is the tangential contact deformation. The velocity $\bar{\mathbf{v}}^{\text{contact},\,t}$ is an average of the velocities at times $t - dt/2$ and $t + dt/2$.

13.3.14 `work1t`

The work done by the "boundary stresses" per unit of the assembly's original volume. It is computed from the work rate

$$\frac{\text{Current volume}}{\text{Initial volume}} \int \sigma_{ij} \dot{F}_{ik} F_{kj}\, dt \tag{13}$$

and is cumulative from the beginning of the simulation. In this equation, σ_{ij} is the Cauchy stress.

13.3.15 `xloops`

When `algori=2`, several time steps will occur within each deformation step (Section 11.1.2). The program self-monitors the number of cycles that are required to achieve a near-equilibrium condition before advancing the assembly deformations. The property `xloops` is the average number of cycles that were required.

The program currently limits the number of cycles per deformation step to between `nloop1` and 101 (Section 11.1.14). When `xloops` is consistently reported as 3, the near-equilibrium criteria was met in three or fewer loops (see Section 11.1.2). When `xloops` is 101, then the near-equilibrium criteria were likely not met even after the final cycle. The threshold, near-equilibrium criteria is currently defined as a value of 0.5(`chi1` + `chi2`) be less than 1%.

13.3.16 `viscct`

This quantity appears in the A-files, but not in the B-files. The quantity `viscct` is the energy expended in viscous contact damping per unit of the assembly's original volume (cumulative since the beginning of the simulation). This quantity may not be meaningful when `algori=2` (Section 11.1.2). The *increment* in `viscct` for a time step dt is computed as follows:

$$\frac{1}{\text{Initial volume}} \times \sum_{\text{Contacts}} \left[\eta^n \bar{\mathbf{v}}^{\text{contact},\,n} \cdot \bar{\mathbf{v}}^{\text{contact},\,n} + \eta^t \bar{\mathbf{v}}^{\text{contact},\,t} \cdot \bar{\mathbf{v}}^{\text{contact},\,t} \right] dt\,, \tag{14}$$

where η^n and η^t are the coefficients of contact normal and contact tangential damping, and $\bar{\mathbf{v}}^{\text{contact},\,n}$ and $\bar{\mathbf{v}}^{\text{contact},\,t}$ are the components of contact deformation velocities in the normal and tangential directions. That is, the contribution to `viscct` of a single contact is the contact damping force $\eta \bar{\mathbf{v}}^{\text{contact}}$ multiplied by the displacement increment $\bar{\mathbf{v}}^{\text{contact}} dt$.

13.4 B-files with 3D simulations

As with 2D simulations, the first four lines of a B-file contains a file-type identifier, the name of the StartFile, the version of the Dempla source code, and the RVE number (see

Section 13.3). These four lines are followed with results for each output cycle. With 3D simulations, the information for each output cycle is packed into five lines. However, with simulations with a poromechanic model (`iporo` = 1, 3, or 4), six lines are given. We first describe the five lines, such as the following:

```
2.6000000E-05   5.313745E-06 -5.000000E-05  0.000000E+00  6.161242E-05   5168
     9.65E-05  0.000000E+00  0.000000E+00  0.000000E+00  4.39854321784E+02
     1.13E-04 -5.684897E+05 -6.075969E+05 -5.778599E+05  2.12654326770E-02
     0.00E+00 -2.138418E+04  5.103270E+03 -1.097130E+04  2.57476543228E+01
     6.82E-05      5.10E-07  2.169865E-02  0.000000E+00         30.00
```

These contents correspond to the following variables (some of these are, in fact, arrays):

```
 timer         defout(1,1)   defout(2,2)   defout(3,3)    knrgy       ntacts
     chi1      defout(1,2)   defout(1,3)   defout(2,3)    pnrgy
     chi2      stress(1,1)   stress(2,2)   stress(3,3)    slidet
     chi3      stress(1,2)   stress(1,3)   stress(2,3)    work1t
     chi4           psi         viscbt       viscct       xloops
```

given in the following formats:

```
 1pe14.7,  1x,  1pe14.6,      1pe14.6,      1pe14.6,       1pe14.6,       i7,/,
 1pe15.2,       1pe14.6,      1pe14.6,      1pe14.6,       1pe19.11,         /,
 1pe15.2,       1pe14.6,      1pe14.6,      1pe14.6,       1pe19.11,         /,
 1pe15.2,       1pe14.6,      1pe14.6,      1pe14.6,       1pe19.11,         /,
 1pe15.2,       1pe14.2,      1pe14.6,      1pe14.6,       0pf14.2
```

These output fields were described in the previous section (13.3).

When a poromechanic model is used (`iporo` = 1, 3, or 4), then a sixth line is added with five items, for example:

```
      1.430419E+04  4.056670E-03  1.9403765E-02  1.8747376E-03 9.935762272E-01
```

with the following information:

```
      pfluid        defw          fpnrgy         spnrgy        S_now
```

with the following format:

```
   5x,1pe14.6,       1pe14.6,      1pe15.7,       1pe15.7,      1pe16.9
```

and these five poromechanic items are described below.

- `pfluid` is the pore fluid pressure relative to atmospheric pressure;
- `defw` is the inflow of fluid into the assembly per unit of original pore volume.
- `fpnrgy` is the potential energy of the fluid.
- `spnrgy` is the potential energy imparted to the grains by the fluid pressure.
- `S_now` is the fluid water saturation.

13.5 F-files: micro-data text files

F-files contain information on the positions of all particles and the status of all contact forces. In Mode 1, these files are created during an DEMPLA simulation by setting `imicro=1` within a deformation-stress segment of the RunFile (Section 11.2.11). With `imicro=1`, a set of F-files will be created at the *start* of the particular deformation-stress segment. As many as

	File name	Content
2D	Fa?<RunFile>	Assembly size
	Fb?<RunFile>	Particle data
	Fc?<RunFile>	Contact data
	Fd?<RunFile>	Void cell data
3D	Fa?<RunFile>	Assembly size
	Fb?<RunFile>	Particle data
	Fc?<RunFile>	Contact data

Table 2: Contents of the various F-files. Note that the "?" character in a file name is a 3-digit number (e.g., 005) that corresponds to the particular segment of the deformation-stress path in which the F-file was created (Section 11.2.11).

four separate F-files will be produced, with each containing a different type of information (Table 2). The files can be quite large: for an assembly of 1000 particles, a set of 2D F-files is about 300kBytes. The FFile s can be created when DEMPLA is run in Mode 2. Note that the "?" character in a file name (Table 2) is a 3-digit number (e.g., 005) that corresponds to the particular segment of the deformation-stress path in which the F-file was created (Section 11.2.11).

The contents of these text files are described in the following two sections. You will probably want to use a data analysis package to open, read, and analyze their data (e.g. Matlab, Octave, Scilab, R, etc.). The various F-files only contain information on the *status* of the assembly at a single instance; they do not provide velocities or other rates. If such rates are of interest, then you should use a RunFile that will produce two sets of F-files, with the two files separated by just a few time steps.

13.6 F-files for 2D assemblies

Four F-files are created at once (Table 2), and they are described in the following four subsections. F-files for 3D assemblies are described in Section 13.7. All F-files are created in `subroutine micro`.

13.6.1 Fa-files for 2D assemblies

These small files give the size of the assembly, the average deformation gradient relative to the start of the simulation, and other general information. The first thirteen lines are in the format `i3,/,1pe14.7,/, 9(3(1pe25.17),/),i2,/, 3(1pe17.9,/)`:

```
file_identifier
timer
xcell(1,1)  xcell(1,2)  xcell(1,3)
xcell(2,1)  xcell(2,2)  xcell(2,3)
xcell(3,1)  xcell(3,2)  xcell(3,3)
  def(1,1)    def(1,2)    def(1,3)
  def(2,1)    def(2,2)    def(2,3)
  def(3,1)    def(3,2)    def(3,3)
```

```
stress(1,1) stress(1,2) stress(1,3)
stress(2,1) stress(2,2) stress(2,3)
stress(3,1) stress(3,2) stress(3,3)
kshape
imodel
```

The `file-identifier` simply identifies the version of the F-files, in the event that the file content or format is modified at a later date. The cell dimension `xcell(i,j)` were illustrated in Fig. 9, page 60. The `def(i,j)` deformations are components of the deformation gradient **F**. For 2D assemblies, only four of the nine components `xcell`, `def`, and `stress` are meaningful. The meanings of `kshape` (particle shape) and `imodel` (contact model) are described in Section 12.1.1 and Section 11.1.12.

The next set of lines give information about the contact model, depending on the type of contact:

- If `imodel` is 0 or 1 (linear contacts), then the next two lines give the following information

  ```
  kn
  kratio
  ```

  as defined in Section 11.1.22 and Section 11.1.23.

- If `imodel` is 5 or 6 (Modified-Mindlin or Jaeger model), then the next two lines give the shear modulus G of the grains and the Poisson's ratio of the grains:

  ```
  G
  nu
  ```

  as defined in Section 11.1.22 and Section 11.1.23.

- If `imodel` is 7 (cone contacts), then the next three lines give the following information

  ```
  G
  nu
  A_1
  ```

  as defined in Section 11.1.22, Section 11.1.23, and Section 11.1.35.

- If `imodel` is 9 (general contact asperity), then the next four lines give the following information

  ```
  G
  nu
  A_1
  palpha
  ```

  as defined in Sections 11.1.22, 11.1.23, and 11.1.35, and 11.1.50.

The next line gives the friction coefficient

```
frict
```

as described in Section 11.1.24.

For ovoid, nobby, and bumpy particles, additional information is added at the bottom of the `Fa`-file:

- For 2D oval and 3D ovoid particles (`kshape=2` and `5`) (Fig. 10 and Fig. 6), the `Fa`-file ends with a line that gives the `beta` value (Section 12.1.4).
- For 2D nobby particles (`kshape=6`, Fig. 7, the `Fa`-file ends with three lines that give the `beta` values of these parameters: `nobs`, `cenrad`, and `satrad`.
- For 3D bumpy particles (`kshape=7`, Section 8, the `Fa`-file ends with four lines that give the `beta` values of these parameters: `nbumps`, `cenrad`, `satrad`, and `cirrad`.

13.6.2 Fb-files for 2D assemblies

These files contain information on the size and position of each particle. Each line gives data for a single particle, with the lines arranged by particle number (e.g. line number 3 is for particle 3). The format of each line is `i7,4(1pe17.9),1(1pe18.9)`, with the fields as follows:

Field 1	`hv`, a pointer to the DCEL (see Section 13.6.3)
Field 2	Particle half width (Fig. 10, page 60)
Field 3	Aspect ratio, Height/Width (Fig. 10, page 60)
Field 4	x_1 position
Field 5	x_2 position
Field 6	θ orientation in radians (Fig. 11, page 62)

Note that when `hv=0`, the particle is in contact with no other particles. The positions x_1 and x_2 correspond to the particle centers. See Section 12.2.

13.6.3 Fc-files for 2D assemblies of convex particles

These fields contain information on every contact within the assembly. The content of this file will differ for convex particles (circles, ellipses, and ovals) and non-convex particles (nobbies). This section applies to convex particles; non-convex particles are described in the next section.

The format of each line is `4(i7),2(i8),2(1pe17.9),5(1pe13.5)` (with Hertz-Mindlin contacts, the value of `Tstar` in format `1pe13.5` is included at the end of a line). The information in this file will allow you to reconstruct the entire topology of the 2D assembly (some data from the Fb- and Fc-files will also be needed) and to navigate within this topology. Doing this efficiently requires a rather ingenious data structure called a Doubly-Connected-Edge-List (DCEL). Although I plan to include a better description in a future edition of this document, for now you should refer to the text [13], which describes the DCEL and how to use it to navigate the topology of a 2D planar graph.

The first six fields in each line contain the following information on a contact: `V1`, `V2`, `F1`, `F2`, `P1`, and `P2`. As an example, Table 3 shows five rows and the first six columns in an Fc-file (Fig. 13).

- Row 3 in an F-file corresponds to contact 3 (see row 3 in Table 3).
- Contact 3 is between particle number 1 (`V1`) and particle number 225 (`V2`) (i.e., rows 1 and 225 in the corresponding Fb-file).

V1	V2	F1	F2	P1	P2
2233	1	1	4	6172	2
1	3985	1	2	3	6393
1	225	2	3	4	849
1	345	3	4	1	1260
690	2	5	8	2381	7

Table 3: An example DCEL table (see Fig. 13).

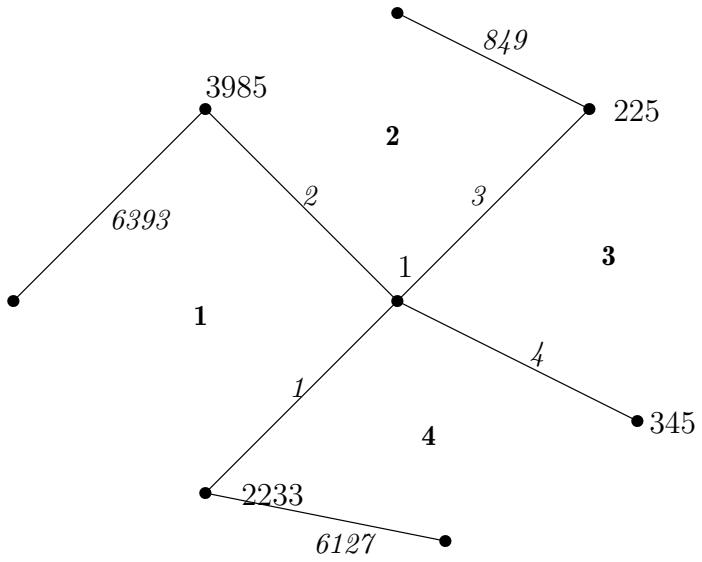


Figure 13: The particle graph associated with the DCEL of Table 3. The dots represent particle centers. Lines represent the contacts between particles.

- Void cell 2 (`F1`) lies on one side of this contact, and void cell 3 lies on its other side (`F2`).
- Pointer `P1` (= 4) points to a contact (row 4), which is also connected to `V1` (particle number 1) and lies directly clockwise around `V1` relative to the row 3 contact.
- Pointer `P2` (= 849) points to the contact (row 849) that is also connected to `V2` (= particle 225) and lies clockwise around `V2` relative to the third (row 3) contact.

The `hv` data in an Fb-files point to the starting (header) contact for a particle (Section 13.6.2). With this data structure, you will be able to identify all of the contacts that are connected to an arbitrary particle. You will also be able to identify all contacts and particles that lie around the perimeter of a polygonal void cell. That is, you can construct both the particle graph and its dual [9]. The `hf` data in an Fd-file points to the starting (header) contact for a void cell (Section 13.6.5).

The final seven columns in an Fc-file give the following data:

- `branch(1)` and `branch(2)`: the horizontal and vertical component of the branch vector that connects the centers of particles `V1` and `V2` (directed toward `V2`). Note that the format of these two fields is `1pe17.9`.
- `c_eta(1)` and `c_eta(2)`: the horizontal and vertical components of the unit normal vector, directed outward from the surface of particle `V1` at the contact location.
- `fnold1(1)`: the magnitude of the contact force component that acts normal to the contact surface—a positive value for compressive contact force.
- `ftold(1)` and `ftold(2)`: the horizontal and vertical components of the contact force tangential to the contact surface. The force acts upon particle `V1`.
- `Tstar`: For Hertz-Mindlin contacts only (`imodel=5`, (Section 11.1.12). See [14].

13.6.4 Fc-files for 2D assemblies of non-convex particles

With non-convex particles, such as nobbies, two particles can have multiple contacts between each other. The Fc-file gives information identifies the two particles and their component parts (satellite and central circles) that are touching. Each line in the Fc-file has format `4(i7),2(i8),2(1pe17.9)`, `i3,4(i7,i3,i3)` and contains the following information:

- The first six fields contain `V1`, `V2`, `F1`, `F2`, `P1`, and `P2`. These are described in Section 13.6.3.
- The next two fields contact `branch(1)` and `branch(2)` as described in Section 13.6.3.
- `ihit`: the number of contacts between these two particles.
- Four triples of integers. With non-convex particles, such as nobbies, two particles can have multiple contacts between each other. Each triple corresponds to one of these contacts. Each triple is composed of a pointer to a position within the list of an "Ff" file and two identifiers of the component circles that that are touching (one identifier

for the first particle `V1`, the other identifier for the second particle `V2`). The identifier is an integer ranging between 0 and `nobs`: the zero corresponding to the central circle, numbers 1 and above corresponding to the satellite circles (arcs). If a pair of particles has fewer than four contacts, then zeros will appear in the unused triples.

13.6.5 Fd-files for 2D assemblies

The lines of this file contain a single integer in `i7` format. The `hv` integer on each line is a pointer to the DCEL for a single void cell (Section 13.6.3). The lines are given in order (for example, line 14 gives the `hv` value for void cell number 14).

13.6.6 Ff-files for 2D assemblies

These files are only created for non-convex particles, such as nobbies. Each line in this file gives information on a single contact. Note that a pair of particles can share multiple contacts. As described in Section 13.6.4, each pair of contacting particles will be represented with a single line in an Fc-file. For non-convex particles, the Fc-file will identify the two particles (the locations of these particles are given in the Fb-file), the branch vector between the particles' centers, number of contacts between the two particles, and triple-integers that identify each of these contacts. The first integer in a triple points to a line in the Ff-file. The contents of the Ff-file depends on the contacts' displacement-force model (`imodel`, Section 11.1.12.

For nobby particles with a linear-frictional contact model (`imodel=0`), this line has format `4(1pe17.9)` and contains the following information about the contact:

- The two components of the contact's normal vector (directed outward from the first particle, `V1`).
- The normal force (compression is positive).
- The three components of the contact's tangential force (acting upon the first particle, `V1`).

For nobby particles with a simple Hertz-Mindlin model (`imodel=5`), this line has format `5(1pe17.9)` and contains the following information about the contact:

- The two components of the contact's normal vector (directed outward from the first particle, `V1`).
- The normal force (compression is positive).
- The three components of the contact's tangential force (acting upon the first particle, `V1`).
- `Tstar`.

For nobby particles with the Jäger contact model (`imodel=6`), this line has format `7(1pe17.9)` and contains the following information about the contact:

- The two components of the contact's normal vector (directed outward from the first particle, `V1`).
- The normal force (compression is positive).
- The three components of the contact's tangential force (acting upon the first particle, `V1`).

13.7 F-files for 3D assemblies

Four F-files are created together (Table 2, (Table 2), and they are described in the following three subsections: Fa-files (Section 13.7.1), Fb-files (Section 13.7.2), Fc-files (Section 13.7.3), and Ff-files (Section 13.7.4).

13.7.1 Fa-files for 3D assemblies

Same data as with 2D assemblies (Section 13.6.1).

13.7.2 Fb-files for 3D assemblies

These files contain information on the size and position of each particle. Each line gives data for a single particle, with the lines arranged by particle number (line number 3 in the file is for particle 3).

For 3D assemblies of spheres, the format of each line is `4(1pe17.9),3(1pe18.9e3)`, with the following fields for each sphere:

Field 1	radius
Field 2	x_1 position of the particle center
Field 3	x_2 position of the particle center
Field 4	x_3 position of the particle center
Field 5	θ_1 angular orientation of the particle (radians)
Field 6	θ_2 angular orientation of the particle (radians)
Field 7	θ_3 angular orientation of the particle (radians)

For 3D assemblies of ovoids, the format of each line is `7(1pe17.9),3(1pe18.9e3)`, with the following fields for each ovoid:

Field 1	transverse (revolved) half width (Section 12.2.5)
Field 2	aspect ratio: axial height / transverse width
Field 3	x_1 position of the particle center
Field 4	x_2 position of the particle center
Field 5	x_3 position of the particle center
Field 6	γ_1 orientation angle (in degrees) of the particle's axis (Fig. 12, Section 12.2.5)
Field 7	γ_2 orientation angle (in degrees) of the particle's axis (Fig. 12, Section 12.2.5)
Field 8	θ_1 angular orientation of the particle (radians)
Field 9	θ_2 angular orientation of the particle (radians)
Field 10	θ_3 angular orientation of the particle (radians)

For 3D assemblies of bumpies, the format of each line is `4(1pe17.9),4(1pe18.9e3)`, with the following fields for each ovoid:

Field 1	scaling radius (Section 12.1.6 and Section 12.2.7)
Field 2	x_1 position of the particle center
Field 3	x_2 position of the particle center
Field 4	x_3 position of the particle center
Field 5	1st component of the unit orientation quaternion
Field 6	2nd component of the unit orientation quaternion
Field 7	3rd component of the unit orientation quaternion
Field 8	4th component of the unit orientation quaternion

13.7.3 Fc-files for 3D assemblies

Each line in this file gives information on a single contact.

For 3D assemblies of spheres, the format of each line is `2i7,3(1pe17.9), 4(1pe13.5)`, with the following fields for each sphere (with Hertz-Mindlin contacts, the value of `Tstar` in format `1pe13.5` is included at the end of a line):

- `V1` and `V2`: the two particle numbers
- `branch(1)`, `branch(2)`, and `branch(3)`: the components of the branch vector that connects the centers of particle `V1` and particle `V2` (directed toward `V2`). Note that the format for these three fields is `1pe16.8`.
- `fnold1(1)`: the magnitude of the contact force component that acts normal to the contact surface—a positive value for compressive forces.
- `ftold(1)`, `ftold(2)`, and `ftold(3)`: the three components of the portion of the contact force that is tangential to the contact surface. The force acts upon particle `V1`.
- `Tstar`: For Hertz-Mindlin contacts only (`imodel=5`, Section 11.1.12). See [14].

For 3D assemblies of ovoids, the format of each line is `2i7,3(1pe17.9)`, `10(1pe13.5)`,with the following fields for each ovoid (with Hertz-Mindlin contacts, the value of `Tstar` in format `1pe13.5` is included at the end of a line):

- `V1` and `V2`: the two particle numbers.
- `branch(1)`, `branch(2)`, and `branch(3)`: the components of the branch vector that connects the centers of particle `V1` and particle `V2` (directed toward `V2`). Note that the format for these three fields is `1pe17.9`.
- `rx_i(1)`, `rx_i(2)`, and `rx_i(3)`: The components of a vector from the center of particle `V1` to the center of the contact point between the two particles.
- `fnold1(1)`: the magnitude of the contact force component that acts normal to the contact surface—a positive value for compressive contact force.
- `ftold(1)`, `ftold(2)`, and `ftold(3)`: the three components of the portion of the contact force that is tangential to the contact surface. The tangential force acts upon particle `V1`.
- `c_eta(1)`, `c_eta(2)`, and `c_eta(3)`: The three components of the unit vector that is the outward normal of particle `V1` at the contact point.
- `Tstar`: For Hertz-Mindlin contacts only (`imodel=5`, Section 11.1.12). See [14].

For 3D assemblies of bumpies, the format of each line is `2i7,3(1pe17.9)`, `i3,6(i7,i3,i3)`, with the following fields for each bumpy:

- `V1` and `V2`: the two particle numbers.
- `branch(1)`, `branch(2)`, and `branch(3)`: the components of the branch vector that connects the centers of particle `V1` and particle `V2` (directed toward `V2`). Note that the format for these three fields is `1pe17.9`.
- `ihit`: the number of contacts between these two particles.
- Six triples of integers. With non-convex particles, such as bumpies, two particles can have multiple contacts between each other. Each triple corresponds to one of these contacts. Each triple is composed of a pointer to a position within the list of an "Ff" file and two identifiers of the component spheres that that are touching (one identifier for the first particle `V1`, the other identifier for the second particle `V2`). The identifier is an integer ranging between 0 and `nbumps`: the zero corresponding to the central sphere, numbers 1 and above corresponding to the satellite spheres.

13.7.4 Ff-files for 3D assemblies

These files are only created for non-convex particles, such as bumpies. Each line in this file gives information on a single contact. Note that a pair of particles can share multiple contacts. As described in Section 13.7.3, Each pair of contacting particles will be represented with a single line in an Fc-file. For non-convex particles, the Fc-file will identify the two

particles (the locations of these particles are given in the Fb-file), the branch vector between the particles' centers, number of contacts between the two particles, and triple-integers that identify each of these contacts. The first integer in a triple points to a line in the Ff-file. The contents of the Ff-file depends on the contacts' displacement-force model (`imodel`, Section 11.1.12).

For bumpy particles with a linear-frictional contact model (`imodel=0`), this line has format `4(1pe17.9)` and contains the following information about the contact:

- The normal force (compression is positive).
- The three components of the contact's tangential force (acting upon the first particle, `V1`).

For bumpy particles with a simple Hertz-Mindlin model (`imodel=5`), this line has format `5(1pe17.9)` and contains the following information about the contact:

- The normal force (compression is positive).
- The three components of the contact's tangential force (acting upon the first particle, `V1`).
- `Tstar`.

For bumpy particles with the Jäger contact model (`imodel=6`), this line has format `7(1pe17.9)` and contains the following information about the contact:

- The three components of the contact's normal vector (directed outward from the first particle, `V1`).
- The normal force (compression is positive).
- The three components of the contact's tangential force (acting upon the first particle, `V1`).

14 Results files for wave propagation simulations

When DEMPLA is run in Mode 2, several files are created with results of a wave propagation simulation. These files include the following:

- A set of RunFiles, one for each RVE, which are created within DEMPLA (rather than by the user). Each of these RunFiles gives the sequence of stress-strain control for a single RVE (see Section 11 about RunFiles, particularly the section on deformation-stress paths). The files are placed in the `04_RunOutput` directory (folder), which is inside of the BaseName directory (see page 20). The file of a single RVE (i.e., DEM assembly) will have the following name:

  `I`<BaseName with suffix>`_Dempla_RVE_`<RVE number>`_`<MotionFile name>

  The first and third parts within "< >" are the two input items that are entered at the start of a DEMPLA wave propagation simulation (see the input description). The RVE number is in the format `i4.4`, so that the fourth RVE from the bottom of the soil column would have number 0004.

- A set of files, one for each RVE, that gives the screen output that would have been created for each RVE. This output for an RVE is not actually displayed on the screen, but is instead written to a file for viewing later. See Section 15 for a description of this screen output. The screen output files are placed in the `04_RunOutput` directory (folder), which is inside of the BaseName directory (see page 20). The file of a single RVE (i.e., DEM assembly) will have the following name:

  S<BaseName with suffix>_Dempla_RVE_<RVE number>_<MotionFile name>

  The first and third parts within "< >" are the two input items that are entered at the start of a DEMPLA wave propagation simulation (see the input description). The RVE number is in the format `i4.4`, so that the fourth RVE from the bottom of the soil column would have number 0004.

- A set of output AFiles and BFiles, one for each RVE. These files are only created if the `iABfile` parameter is set to 1 in the simulation's GFile. The files are placed in the `04_RunOutput` directory (folder), which is inside of the BaseName directory (see page 20). The files of a single RVE (i.e., DEM assembly) will have the following names:

  B<BaseName with suffix>_Dempla_RVE_<RVE number>_<MotionFile name>

  A<BaseName with suffix>_Dempla_RVE_<RVE number>_<MotionFile name>.txt

  The first and third parts within "< >" are the two input items that are entered at the start of a DEMPLA wave propagation simulation (see the input description). The RVE number is in the format `i4.4`, so that the fourth RVE from the bottom of the soil column would have number 0004.

- A set of "`Results`" files. These ResultsFiles will likely be the most useful ones for analyzing the results of DEMPLA Mode 2 simulations. Each file gives a single type of output (stress, pore pressure, saturation, etc.) for all of the RVEs or nodes during the full duration of the simulation. Each file is arranged in rows, with each row usually giving the particular result for all of the RVEs (or nodes) at a single time step during the simulation.

  The ResultsFiles are placed in the `04_RunOutput` directory (folder), which is inside of the BaseName directory (see page 20). The files of a simulation have the following names:

  Results_<Type>_<BaseName with suffix>_<MotionFile name>.txt

  The <Type> specifier is described below and in Table 4. The two parts within "< >" are the two input items that are entered at the start of a DEMPLA wave propagation simulation (see the input description).

  A Mode 2 simulation will create 23 of ResultsFiles, each with the type of data given in Table 4. All ResultsFiles begin with five lines, each with a single item of data:

  - `iResultsType` (format `i8`): An indicator of the number of columns that each output row will contain. This information will help in reading the ResultsFile during post-processing with programs such as Octave.
    - `iResultsType = 0`: each row contains `nRVEs`, N, data items.
    - `iResultsType = 1`: each row contains `nRVEs+1`, $N + 1$, data items.

Table 4: Contents of various ResultsFiles for Mode 2 simulations of wave propagation.

Type	Location	Columns	Format	Description
`u_1`	Nodes	$N+1$	`1pe18.10`	Displacement, u_1
`u_2`	Nodes	$N+1$	`1pe18.10`	Displacement, u_2
`u_3`	Nodes	$N+1$	`1pe18.10`	Displacement, u_3
`s_11`	RVEs	N	`1pe14.6`	Total normal stress, σ_{11}
`s_12`	RVEs	N	`1pe14.6`	Shear stress, σ_{12}
`s_13`	RVEs	N	`1pe14.6`	Shear stress, σ_{13}
`s_22`	RVEs	N	`1pe14.6`	Total normal stress, σ_{22}
`s_23`	RVEs	N	`1pe14.6`	Shear stress, σ_{23}
`s_33`	RVEs	N	`1pe14.6`	Total normal stress, σ_{33}
`w`	Nodes	$N+1$	`1pe18.10`	Pore fluid displacement in x_3 direction
`p`	RVEs	N	`1pe14.6`	Pore fluid pressure, p
`rho_soil`	Nodes	$N+1$	`1pe14.6`	Total soil density, ρ
`rho_fluid`	Nodes	$N+1$	`1pe14.6`	Total fluid density, ρ_f
`Porosity`	Nodes	$N+1$	`1pe14.6`	Soil porosity, n
`kperm`	Nodes	$N+1$	`1pe14.6`	Soil permeability, k
`sat`	Nodes	$N+1$	`1pe17.9`	Soil saturation, S
`water`	$*$	2	`1pe17.9`	Water depth and height from base, x_3
`time`	†	1	`1pe17.9`	Time from start of MotionFile motion
`I`	RVEs	N	`1pe17.9`	Inertia number of DEM assembly
`kn`	RVEs	N	`1pe17.9`	Avg. contact stiffness k_n
`chi`	RVEs	N	`1pe17.9`	Force imbalance in RVE
`mass`	RVEs	N	`1pe17.9`	Avg. particle mass in RVE
`steps`	RVEs	N	`1pe17.9`	Number of DEM steps in a Dempla step Δt

$*$ Two columns: water depth from the ground surface, and height above the rock base x_3
† One column: time since the start of the simulation (the beginning of the MotionFile)

    - * `iResultsType = 2`: each row contains one data item.
    - * `iResultsType = 3`: each row contains two data items.
  - `nRVEs` (format `i8`): the number of RVEs (DEM assemblies) N in the soil column. The value of `nRVEs` is given as input in a GFile (see Section 7.1.3).
  - `dt_ref` (format `1pe21.14`): the time step Δt for the Mode 2 wave propagation simulation. Note that this time step applies to the period defined by the MotionFile, and a scaled time step can apply during post-shaking consolidation.
  - `dx_rve` (format `1pe21.14`): the initial spacing Δx of the nodes (and RVEs) within the soil column. The total height of the soil column is `nRVEs` $\times$ `dx_rve`.
  - `nDempla_out` (format `i8`): the number of time steps Δt between output lines in the ResultsFile. The value of `nDempla_out` is given as input in a GFile (see Section 7.1.16).

15 Screen output from Dempla

As a simulation is running, information is printed to the screen, which can help to monitor the performance of the run. This information can, of course, alternatively be redirected from the screen to a file. The following is an example of the introductory information that might appear on the screen at the beginning of a simulation:

```
Program OVAL:  version            oval-0.5.41.f
c Matthew R. Kuhn 2001, Licensed under the GPL, version 2

The program was compiled under the following parameters:
   1) 2D or 3D problems. (mdim1=3)
   2) Circular, spherical, elliptical, oval, and ovoid particles. (mpiece=4*mp)
   3) A maximum of 10020 particles. (mp)
Errors will occur if your input data is otherwise (but you can always make
changes to the common-0.5.41 file and recompile).

  Name of the RunFile:
LoadComp                 <-- your input here
  Name of the StartFile:
Dsphere_1800             <-- your input here

 **** Warning ****.
 * The input value of rho was 0.
 * A mass will be automatically assigned.

 **** Warning ****.
 * The input value of dt was 0.
 * A time step will be automatically assigned.

  Your time step is:                1.00000E+00
  The maximum advised time step is: 1.25000E+00

  The assigned particle mass is:    9.76563E+00

  Spherical, 3D particles
```

```
  Number of particles      =    1800
  Initial void ratio       =  0.535828
  Initial solids fraction  =  0.651115
  Initial porosity         =  0.348885
  Volume of the cell       =  2.248245E+03

  Initial number of contacts       =    5130
  Average ratio of overlap/diameter =  3.186E-04
```

In this example, the names of the two input files are `Load1` and `Dcircls_1002`. The introductory information is followed by a table of diagnostic information that is periodically updated at the interval `ipts`, as specified in the RunFile (Section 11.2.8). This table will look something like the following:

```
Some diagnostic information during this run:
iout    timer istep nupd  ipt2 xloops   chi1      chi2       psi    sweep
   0  0.0000E+00  1    1  18108   1.0  0.00E+00  0.00E+00  0.00E+00  0.00
   1  2.0000E+00  1    1  18108  21.5  1.10E-02  7.73E-03  0.00E+00  0.00
   2  4.0000E+00  1    1  18108   3.0  9.81E-03  7.04E-03  0.00E+00  0.00
   3  6.0000E+00  1    1  18108   3.0  8.79E-03  6.34E-03  0.00E+00  0.00
   4  8.0000E+00  1    1  18108   3.0  7.98E-03  5.74E-03  0.00E+00  0.00
   5  1.0000E+01  2    1  18108   3.0  7.14E-03  5.15E-03  0.00E+00  0.00
   6  5.8000E+01  2    2  18109   2.9  2.42E-03  1.77E-03  3.68E-06  0.00
   7  1.0800E+02  2    2  18109   3.0  3.55E-04  3.77E-04  1.39E-06  0.00
   8  1.5800E+02  2    3  18109   3.0  2.68E-04  3.38E-04  1.17E-06  0.00
```

Most items in the table were described in Section 13.3. The integer `istep` is the current deformation-stress segment (from the RunFile, Section 11.2. The integer `nupdat` is the number of near-neighbor searches that have been performed (see Section 11.2). The integer `ipt2` is the current length of the near-neighbor linked list (Section 11.1.27). The value `xloops` is the average number of iteration loops (time steps) per deformation step. When `algori=1`, then `xloops` will always be one. The values , , and indicate the numeric performance of the run (see Sections 13.3.5, 13.3.9, and 13.3.8 and the other sections referenced from there). The value `sweep` is the average number of iterations per torus-torus contact when ovoids are being used.

16 Sample assemblies for Dempla

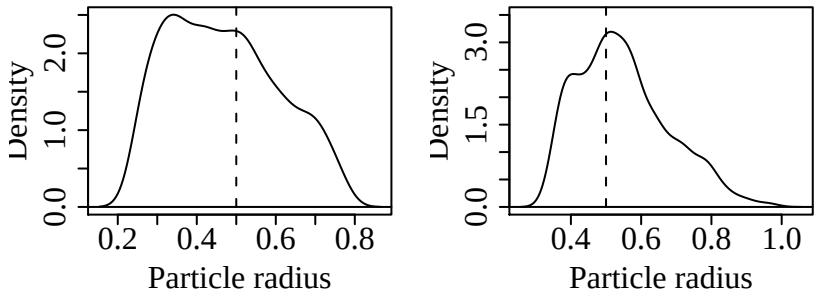
You can download sample StartFile assemblies from the GitHub repository, which are given in a D-file (text) format. The primary attributes of these assemblies are shown in Table 5. Each assembly is roughly square (or cubical) and with an isotropic fabric. They were created by isotropically compressing a sparse assembly with friction turned off.

The assemblies contain a range of particles sizes. The dimensions of the particles and the entire assemblies have been scaled so that the mean particle size D_{50} in each assembly is 1.00. (Here, we speak of the mean particle size D_{50} in the usual sense of geotechnical engineering: a "median" diameter that partitions the assembly into two sets of particles, so that each set has an equal cumulative mass.) A density plot (normalized histogram) of particle radii for both 2D and 3D assemblies is shown in Fig. 14. The histogram is centered on the median size $0.50D_{50}$. In Fig. 14a, the radii of non-circular 2D particles refers to their mean radii, (Height + Width)/4 (Fig. 11, Section 12.2). In Fig. 14b, the radii of non-spherical 3D particles refers to their mean radii, (Height + 2 · Width)/6 The distribution of

File Name	Particle Type	No. of Particles	Void Ratio	Coord. Number[5]	Dimensionless Overlap
`Dcircls_1002_2`[1]	circles	1002	0.18042	3.820	3.10×10^{-4}
`Dcircls_4008_2`[1]	circles	4008	0.17911	3.813	3.19×10^{-4}
`Dovals_1002_1d`	ovals	1002	0.18382	3.784	2.32×10^{-4}
`Dovals_1002_2d`	ovals	1002	0.12890	4.880	1.58×10^{-4}
`Dovals_4008_1d`	ovals	4008	0.18479	3.777	2.10×10^{-4}
`Dovals_4008_2d`	ovals	4008	0.12788	4.733	1.72×10^{-4}
`Dsphere_1800`	spheres	1800	0.53583	5.700	3.19×10^{-4}
`DOblate_1800`[2]	ovoids	1800	0.41520	8.214	1.65×10^{-4}
`DProlate_1800`[3]	ovoids	1800	0.41223	8.438	2.01×10^{-4}
`DObProlate_1800`[4]	ovoids	1800	0.41367	8.351	2.01×10^{-4}

[1]The assemblies `Dcircls_1002` and `Dcircls_4008` in OVAL version `0.4.0` have been replaced. These older assemblies were slightly anisotropic.
[2]Oblate ovoids with randomly assigned aspect ratios. The aspect ratio is uniformly distributed between 0.65 and 1.00.
[3]Prolate ovoids with randomly assigned aspect ratios. The aspect ratio is uniformly distributed between 1.00 and 1.60.
[4]A combination of oblate and prolate ovoids with randomly assigned aspect ratios. The aspect ratio is uniformly distributed between 0.65 and 1.60.
[5]The coordination number is computed as twice the number of contacts divided by the number of particles.

Table 5: Attributes of several sample assemblies



(a) 2D assemblies (b) 3D assemblies

Figure 14: Histograms of particle radii for 2D and 3D assemblies

aspect ratios for oval and ovoid assemblies are described in the footnotes of Table 5.

The void ratio is a measure of the packing density of a granular assembly (Table 5). The assemblies of circles, spheres, and ovoids are fairly dense. Both loose and dense assemblies of ovals are provided. The coordination numbers shown in Table 5 are computed as twice the number of contacts divided by the number of particles.

Among the attributes listed in Table 5, the dimensionless overlap probably has the greatest effect on the speed and performance of a simulation. The dimensionless overlaps, which are quite small, were computed by dividing the average overlap at the particle contacts by the mean particle size, D_{50}. Small overlaps more closely resemble those in real granular materials, which are often composed of hard granules. During simulations, however, small overlaps require slower deformation rates to assure the near quasi-static progression of particle rearrangements.

In addition to the files in Table 5, the website includes a directory `samples/startfiles/Series_circls_1002_` that contains 100 assemblies of 1002 circular particles. The particle size distribution in each assembly is the same as that of `Dcircls_1002_2` in Table 5 and Fig. 14a. Each assembly has exactly the same particle sizes and the same void ratio (=0.174257), but the assemblies have different particle arrangements. As a result of this difference, the files will have modest differences in the number of contact, dimensionless overlap, and initial stress. The assemblies were constructed with the same process, but the particle radii were shuffled among the particles before the diffuse assembly was compacted.

17 Example simulations

17.1 Mode 1 simulation of monotonic undrained triaxial compression of unsaturated sand

17.1.1 Introduction

This example is described in the author's paper [3], in Sections 3 and 3.2, with results depicted in Fig. 4a of that paper. All of the files for the simulation are in the GitHub folder `examples/CIUC_pq_unsat_loose`. The simulation is of a loose Toyoura Sand at a relative density of about 30%. The sand is consolidated to an isotropic stress of 100kPa. At this stress, the water saturation is 95%. The water pressure (back pressure) is increased to 270kPa while maintaining an effective stress of 100kPa. This achieves a 99.9% saturation (in separate simulations, discussed in Section 3.1 of the paper, the B-value was 0.42). The assembly is then monotonically loaded with undrained triaxial compression while maintaining constant total stress in the transverse directions.

In Section 17.2, a similar example is described, but for a cyclic CIUC test with an amplitude of 30kPa.

The input RunFile and StartFile are described in the next section. As background, the following is a summary of Section 3 of the paper [3].

DEM simulations were conducted to illustrate poroelastic effects in undrained loading. Several DEM assemblies were created with different densities. Only the simulation with a single assembly is included in the GitHub examples. The assemblies were composed of 10,600 sphere-clusters within a cubical box, with characteristics that were intended to

approximate Toyoura Sand. The particle sizes had a narrow range (uniformity coefficient $C_u = 1.9$) and a median size D_{50} of 0.19 mm. Because the distribution of particle sizes will affect the bulk properties of a soil, the sizes of the DEM particles were selected to approximate the size distribution of Toyoura Sand. Rather than using spherical particles, each particle was a composite cluster of seven spheres (see [6]). The complexity of these non-convex clusters produces assemblies with a more realistic void ratio, bulk strength, and bulk stiffness. These characteristics approximate those of Toyoura Sand, an achievement that is not possible with simple assemblies of spheres, as sphere assemblies are denser, weaker, and softer than most sands. Recently, the use of rotational springs at the contacts has been proposed as an expedient approach that uses spheres to produce a greater bulk strength (for example, [15]). Neither sphere particles nor rotational springs were used in the current study, due to the difficulty of achieving realistic results for the all three characteristics — void ratio, strength, and stiffness — even by manipulating the rotational resistance.

A series of 29 assemblies was created with an isotropic fabric and a range of densities, with void ratios of 0.596 to 0.984, roughly the e_{min} and e_{max} of Toyoura Sand. The 25 densest assemblies were created by compacting the same sparse assembly of 10,600 particles: a diffuse assembly with a solid fraction of 0.0274 and void ratio of 35.5. To simplify the compaction process, we began with particles having a mean diameter of D_{50} of 1.0 units and linear-frictional contacts with a unit spring stiffness (the particles' sizes would later be scaled and a more realistic contact model used, as explained below). In other studies, DEM assemblies are often compacted from such diffuse assemblies by assigning a small friction coefficient to the contacts, but we found that these measures could not produce assemblies as loose as the loosest Toyoura Sand. To produce more realistic densities, we used a friction coefficient of zero and included a long-range attractive force between particles that was engaged at inter-particle separations of $0.10D_{50}$ and nearer. This measure promoted the agglomeration of particles during the initial compaction and produced a loose assembly: a single assembly with a void ratio of 0.985. After producing this assembly, the attractive mechanism was removed from the contacts and the particles' sizes were scaled to those of Toyoura Sand, and further compaction was then performed with a linear-frictional contact model. Assemblies with a range of densities were created from this single initial assembly by using a range of friction coefficients (from 0.05 to 0.45): 25 assemblies were created with void ratios of 0.596 to 0.941. To create the four loosest assemblies, with void ratios of 0.950 to 0.984, we removing 1% to 4% of the particles, randomly chosen, from the one assembly with a void ratio of 0.941, in the manner of [16]. By removing a progressively larger number of particles from an already loose assembly, the void ratio was progressively increased with the four loosest assemblies, while roughly preserving the arrangements of the remaining particles. We noted, however, that although these four assemblies had larger void ratios as a consequence of the randomly dispersed large voids produced by the removal of particles, their undrained strengths were not significantly reduced by this particle-removal process, indicating that subtle fabric effects (other than density) are active in affecting undrained behavior.

The final cubic assemblies had dimensions in the range of 2.9 mm to 3.2 mm for the densest and loosest assemblies, respectively. The assemblies were bounded on all sides with periodic boundaries, a technique that produces a homogeneous fabric without the disruptive influence of boundary platens.

For the simulations with these assemblies, we used a contact mechanism that was more realistic than simple linear springs. The particles were assigned elastic properties that approximated those of quartz, with shear modulus $G_s = 29$ GPa, Poisson coefficient $\nu_s = 0.15$, and bulk modulus $K_s = 32$ GPa [17]. Because the evolution of water pressure during undrained loading depends upon the bulk stiffness of the soil skeleton and because this stiffness changes during loading, it is essential to use a realistic contact model.

A Hertzian Cattaneo–Mindlin model was used for the contacts, which was implemented with the Jäger algorithm to simulate the complex sequences of contact movements that occur during cyclic loading paths [5]. The stiffness modulus G of sand is known to depend upon the effective confining pressure p, with G proportional to p in the form $G \propto p^{\beta}$, with $\beta \approx 0.5$ (for Toyoura Sand, $\beta = 0.47$). Capturing this dependence is important in liquefaction studies, as the induced excess pore pressure reduces the effective confining pressure and alters the subsequent behavior. Simple linear springs do not produce a realistic pressure-dependence (β is about 0); the Hertz contact between spheres produces a β of about 0.39; and a Hertz contact between conical asperities produces a $\beta = 0.56$ [6]. For the current study, we adopted a Hertz contact with an asperity profile intermediate between spherical and conical: a blunt cone, whose shape is in the form of a power function with parameters α and A (see Kuhn et al. [6], with $\alpha = 1.5$ and $A_{1.5} = 180$). In separate DEMPLA simulations, the 29 assemblies exhibited a low-strain shear moduli G that varied with void ratio e and p in a manner that closely approximates Toyoura Sand, with $G \approx (70\,\text{MPa})(p/100\,\text{kPa})^{\beta}(2.17-e)^2/(1+e)$ and exponent $\beta = 0.47$, when measured at a shear strain of 0.0001 [18, 19]. The G_o value of 70MPa, the exponent β, and the value of 2.17 are closely approximated by the 29 assemblies. This agreement between simulations and experiments is not possible with sphere particles and linear contact springs.

We used an inter-particle friction coefficient of 0.45, yielding a bulk strength that was also similar to Toyoura Sand, with a critical state ratio q/p of about 1.22 and a critical state void ratio of about 0.86 for triaxial compression loading. Because both laboratory and DEM results are known to be sensitive to the loading rate [20], we used a consistent strain rate for all simulations, with a strain increment of 1×10^{-6} per time step (for the quasi-static conditions of this study, in which the particles' mass densities are scaled, the time-rate is less relevant than the strain increment, [21]).

17.1.2 Setup and running the simulation

The GitHub repository contains all of the files for performing this simulation. The files for the simulation are in the GitHub folder `examples/CIUC_pq_unsat_loose`. These are the steps for running DEMPLA on a Linux system.

1. create the executable file from the fortran source code. The steps for doing this with the gfortran compiler are described in Section 4.1. You will need to know the name of the executable file and its path.

   For this simulation the parameter `mlistJ` should be set to 800 in the `param-dempla.X.X.XX.f` file (an `imodel=9` blunted-cone Jager cont was used in the simulation), which was sufficient for the simulation.

   Because this is a Mode 1 simulation of a single assembly, there is no advantage of

parallelized multi-thread running. The `-fopenmp` compile option is not needed (see Item 2 on page 14).

2. Locate the RunFile and StartFile for the DEMPLA simulation. These files should be located in the same directory in which DEMPLA is being run. If necessary, create links (symlinks, shortcuts, aliases) to the RunFile, StartFile, and executable file in the run directory to these target files (in Linux, "`ln -s <path to source file> <given name in current dirctory>`").

3. now run the DEMPLA simulation by entering the following commands from the main directory (one directory above the BaseName directory):

```
# Change to your path!!!
PATH_TO_EXECUTABLE="../../source/a.out"
#
$PATH_TO_EXECUTABLE << END
1
CIUC_pq_unsat_loose_Bvalue_0.42
DStartFile
END
```

where `CIUC_pq_unsat_loose_Bvalue_0.42` is a link to the RunFile, and `DStartFile` is a link to the StartFile. The simulation might take several hours to run. If you receice the following error message,

```
Insufficient space in array listJ in subroutin Jager3D
```

then change `mlistJ` to 800 in the `param-dempla` file of the Fortran source code and recompile.

17.1.3 Features of the RunFile

The RunFile is shown in Fig. 15 and Fig. 16. The RunFile for the simulation are in the GitHub folder `examples/CIUC_pq_unsat_loose`, which is present as a symbolic link to the source RunFile in the `RunFiles` folder. Most of the parameters in this RunFile are described in the Mode 2 wave propagation simulation of Section 17.4. The friction coefficient `frict` (Section 11.1.24) and the shape parameters for the blunted cone contact (`A_1` and `palpha`) were selected by conducting a series of separate simulations, in a trial-and-error manner (Section 11.1.35 and Section 11.1.50 and the reference [6]). The values of `A_1` and `palpha` were chosen so that the following results wer obtained: (1) the assembly had the same shear modulus as Toyoura Sand when the mean stress was 100 kPa, and (2) the shear modulus increased with mean stress with a similar power-law as Toyoura Sand. The friction coefficient `frict` was adjusted with a trial-and-error process, so that the strength (monotonic and cyclic) was similar to Toyoura Sand. In these simulations, a consistent loading rate was used. Also see the paper [2]. Nite that the initial water saturation `S_o` is 95%, and the initial water pressure is atmospheric (`p_o=0.`, Section 11.1.38).

```
Heading with useful information about this particular simulation
       1         : algori   | the algorithm for advancing the particle positions (1 or 2)
       6         : ivers    | add extra input lines (0)
     400         : ncownt   | fequency for updating non-periodic boundaries (0)
       0         : iout(2)  | output files with avg. def. and gradients in layers (0)
       0         : iout(3)  | output files with avg. stresses within layers (0)
       1         : istart   | type of file defining the initial configuration (1 or 3)
       0         : iend     | type of file to be created at end of the run (0, 1, or 3)
       0         : idef     | reference configuration for reporting deformations (0)
     500         : iupdtm   | max. no. of time steps between linked-list updates
       0         : icirct   | compute and regularly update the particle graph (0)
       9         : imodel   | contact model (linear, Hertz, etc.) (0 1, 5, 6, 7, or 9)
       0         : nplatn   | number of additional D-files with boundary particles (0)
       5         : nloop1   | minimum number of iteration loops when algori=2
       0         : iexact   | don't use the same mass for every particle (0 or 1)
       0         : isub     | number of submerged particles (0)
       1         : idamp    | standard or Potyondy/Cundall damping (0, 1, 2, or 3)
       0         : iheat    | temperature-dependent model (0)
       0         : icoef    | enable changing the friction coefficient during a run (0)
       3         : iporo    | poromechanic fluid model (0, 1, 3, or 4)
       0         : ifree(11)| a free input integer.  Placeholder for future versions
       0         : ifree(12)| a free input integer.  Placeholder for future versions
       0         : ifree(13)| a free input integer.  Placeholder for future versions
       0         : ifree(14)| a free input integer.  Placeholder for future versions
       0         : ifree(15)| a free input integer.  Placeholder for future versions
       0         : ifree(16)| a free input integer.  Placeholder for future versions
       0         : ifree(17)| a free input integer.  Placeholder for future versions
       0         : ifree(18)| a free input integer.  Placeholder for future versions
    29.d9        : kn       | normal contact stiffness (force/indentation)
     0.15        : kratio   | ratio of tangential/normal contact stiffnesses
     0.45        : frict    | coefficient of friction at particle contacts
     0.          : frictw   | coefficient of friction between particles and wall
    -1.          : rho      | the mass density of the particle material
     0.250       : sep      | threshold separation during the near-neighbor searches
     0.03        : pcrit(1) | viscosity coefficient for translational body damping
     0.05        : pcrit(2) | viscosity coefficient for rotational body damping
     0.          : pcrit(3) | viscosity coefficient for contact damping
```

Figure 15: An example RunFile – the upper 38 lines — for simulating triaxial undrained compression of a quasi-saturated sand specimen.

```
    0.64        : xseed    | seed for assigning random initial velocities (when motion=1)
    0.          : rmsvel   | rms initial velocity (when motion=1)
    0.10        : pdif     | parameter for reducing Jager memory demand (when imodel>=6)
    0.          : tmax     | maximum time for the simulation run
  180.0         : A_1      | shape factor for conical contact profile (A_1)
   -1.          : dt       | time increment
    0.          : gravty(1)| gravity in x_1 direction. Depracated. (0.)
    0.          : gravty(2)| gravity in x_2 direction. Depracated. (0.)
    0.          : gravty(3)| gravity in x_3 direction. Depracated. (0.)
    0.          : p_o      | initial fluid pressure of pore liquid relative to atm. pressure
   31.8d9       : K_s      | bulk modulus of grain bodies (when iporo = 1, 3, or 4)
    2.2d9       : K_f      | bulk modulus of pore fluid (when iporo = 1 or 3)
    0.95d0      : S_o      | initial fluid saturation of pore space at p_o (iporo=2,3)
  100.d3        : p_atm    | the reference atmospheric pressure (iporo=2,3)
    0.0187d0    : Hcc      | Henry's coefficient of pore liquid/gass (iporo=3)
   72.75d-3     : gamm     | surface tension of pore bubbles (iporo=3)
    0.0001d0    : D_o      | initial bubble diameter at p_o and S_o (iporo=3)
    0.          : N_o      | number/density bubbles per unit initial pore volume(iporo=3)
 2337.d0        : p_vap    | water vapor pressure relative to atm. pressure (iporo=3)
    0.          : p_wcav   | cavitation pressure relative to atm. pressure (iporo=3)
    0.          : rfree(15)| placeholder
    0.          : rfree(16)| placeholder
    0.          : rfree(17)| placeholder
    0.          : rfree(18)| placeholder
    1.5         : palpha   | alpha power in contact profile (when imodel=9)
```

		**********	Deformation-Stress Path Segments			**********										imicro										
	icontp	(100000)	(10000)	(1000)	(100)	(10)	(1)		krotat				iflexc	|						iplot				
icontr	|	rate_11	rate_22	rate_33	rate_12	rate_13	rate_23	vrate	igoal	finalv			idump	ipts	|	|	|	ibodyf	|	defdot	|	ipts2	|	|
111000	1	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	70	0	5000.		500	0	0	0	0		0.		0	0			
111000	1	-25.	-25.	-25.	0.000e+0	0.000e+0	0.000e+0	0.000e+0	11	0	-100000.		100	0	0	0	0		0.		0	0			
111000	1	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	70	0	1000.		100	0	0	0	0		0.		0	0			
111000	1	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	50.	70	0	5400.		50	0	0	0	0		0.		0	0			
111000	1	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.	70	0	500.		100	0	0	0	0		0.		0	0			
066000	0	-1.000e-6	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	70	0	400.		20	0	0	0	0		0.		0	0			
066000	0	-1.000e-6	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	70	0	1000.		50	0	0	0	0		0.		0	0			
066000	0	-1.000e-6	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	70	0	50000.		100	0	0	0	0		0.		0	0			
066000	0	-1.000e-6	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	70	0	250000.		100	0	0	0	0		0.		0	0			

Figure 16: An example RunFile – the lower lines — for simulating triaxial undrained compression of a quasi-saturated sand specimen.

Another important aspect of the RunFile are the values `rho=-1.` and `dt=-1.` (see Section 11.1.26 and Section 11.1.36). With these values, there is no need to decide upon a value of the DEM time step. The time step is set to 1 second, and the masses of the particles are adjusted, so that the simulation will be nearly quasi-static.

The deformation-stress control sequence is given in the final 9 lines of Fig. 16, and these lines are described belowa:

1. The first line is 5000 time-steps of a quiescent period (`igoal=70` specifies that that the period is limited by the number of time steps, and the value of `finalv=5000` specifies the number of steps, described in Section 11.2.5 and Section 11.2.7). This quiescent period is needed to allow the DFile assembly to become equilibrated under the conditions of the contacts' stiffnesses. The period is "quiescent" because the value of `icontr=111000` means that the three normal effective stresses are being controlled (the 1's) with rates of zero (the values of `defrat(1)`, or "rate_11" etc.). See Section 11.2.1 and Section 11.2.3 for descriptions of `icontr` and `defrat`. The shearing deformation rates are also zero. The value `icontp=1`, meaning that the pore water pressure is also being controlled The initial water pressure `p_o=0.` is zero relative to atmospheric pressure, and the `defrat(8)=0.` for the first period, so the water pressure is maintained at zero relative to atmospheric pressure (see Section 11.2.2 and Section 11.2.4 for descriptions of `icontp` and `defrat(8)`).

2. In the second period, the effective normal stresses are controlled, the shearing deformations are controlled (`icontr=111000`), and their rates are given by `defrat`. In this period, the shear rates are zero (`defrat(4)`, `defrat(5)`, and `defrat(6)`), but the effective normal stresses are increased (negative, greater compression) at 25Pa per time step — isotropic compression (`defrat(1)`, `defrat(2)`, and `defrat(3)`). The effecive stresses are to be increased until the effective stress σ_{11} reaches 100,000Pa. Note that `igoal=11`: the first "1" means that a terminal value is given in the x_1 direction, and the second "1" means that a terminal value of effective stress (not deformation or total stress) is given for the x_1 direction. During this period, the, drained conditions are maintained: the pore pressure (not water infusion) is maintained (`icontp=1`), and the pressure rate is zero (`defrat(8)=0.`, i.e. `vrate`). Note that the choice of direction of the 25kPa/step depends on whether the effective stress at the start of the second period is greater than or less than the target of -100,000Pa. In the case of this simulation, the stress at the end of the first period is -3,557Pa, so the rate should be -25Pa/step. If a positive value was given, a runaway simulation would occur.

3. The third period is similar to the first period — a quiescent period of 1000 time steps. However, the effective stress is now being maintained at 100,000Pa. The water pressure is being maintained at atmospheric pressure.

4. In the fourth period, we maintain the same effective stresses, but the pore water pressure (back pressure) is controlled (`icontp=1`, meaning drained) and is increased at a rate of 50Pa per time step. The water pressure is increased for 5400 time steps (`igoal=70` means that a specified number of time steps are run, anf `finalv=5400`). This will increase the water pressure to 270,000Pa above atmospheric pressure. At

the end of the fourth period, the saturation is 99.939%. (A pressure of 283,000 would be required to achieve full saturation.)

5. The fifth period is another quiescent period, like the first and third periods. The fifth period lasts for 500 time steps.

6. The next four periods (periods 6, 7, 8, and 9) conduct the triaxial undrained compression. The value `icontr=066000` means that deformation is controlled in the x_1 direction (the first 0), the total normal stress is controlled in the x_2 and x_3 directions (the second and third 6's), and the shearing strains are zero. The x_1 deformation rate $\dot{F}_{11}$ is -1.00×10^{-6} per time step. The rates of the x_2 and x_3 total stresses are zero (i.e., constant chamber pressure). The periods are undrained: the value of `icontp=0` means that the inflow of fluid into the specimen is being controlled; the `defrat(8)` (i.e., `vrate`) of zero means that no fluid flows into or out of the specimen (Section 11.2.2 and Section 11.2.4). Note that the volume of the specimen can change, if the air bubbles increase or decrease in size. The four periods only differ in the frequency at which the output results are written to the AFile and BFile (the values of `ipts`, Section 11.2.8).

17.2 Mode 1 simulation of cyclic undrained triaxial compression of unsaturated sand

17.2.1 Introduction

As with the previous example in Section 17.1, this example is of an isotropically consolidated undrained compression test, but the previous example was of monotonic loading. The current example is of cyclic loading. This example is described in the author's paper [3], in Sections 3 and 3.3, with results depicted in Fig. 5a of that paper. All of the files for the simulation are in the GitHub folder `examples/CIUC_Cyclic_unsat_loose`. The simulation is of a loose Toyoura Sand at a relative density of about 30%. The sand is consolidated to an isotropic stress of 100kPa. At this stress, the water saturation is 95%. The water pressure (back pressure) is increased to 270kPa while maintaining an effective stress of 100kPa. This achieves a 99.9% saturation (in separate simulations, discussed in Section 3.1 of the paper, the B-value was 0.42). The assembly is then cyclically loaded with undrained triaxial compression at an amplitude of 30kPa while maintaining constant total stress in the transverse directions. The specimen fully liquefies. In figure 5a of the paper [3], shows a line for the B-value of 0.42, which is the results of six different simulations, each with a different cyclic amplitude. This particular simulation has an amplitude of 30kPa.

Section 17.1.1 gives background information about the assembly preparation and details about the contact model. We skip repeating this information, and now describe the setup and running of the simulation and details about the RunFile. Because the same soil is simulated, the DFile for this simulation is the as that used in the previous example.

17.2.2 Setup and running the simulation

The GitHub repository contains all of the files for performing this simulation. The files for the simulation are in the GitHub folder `examples/CIUC_Cyclic_unsat_loose`. These are

the steps for running DEMPLA on a Linux system.

1. create the executable file from the fortran source code. The steps for doing this with the gfortran compiler are described in Section 4.1. You will need to know the name of the executable file and its path.

   For this simulation the parameter `mlistJ` should be set to 800 in the `param-dempla.X.X.XX.f` file (an `imodel=9` blunted-cone Jager cont was used in the simulation), which was sufficient for the simulation.

   Because this is a Mode 1 simulation of a single assembly, there is no advantage of parallelized multi-thread running. The `-fopenmp` compile option is not needed (see Item 2 on page 14).

2. Locate the RunFile and StartFile for the DEMPLA simulation. These files should be located in the same directory in which DEMPLA is being run. If necessary, create links (symlinks, shortcuts, aliases) to the RunFile, StartFile, and executable file in the run directory to these target files (in Linux, "`ln -s <path to source file> <given name in current dirctory>`").

3. now run the DEMPLA simulation by entering the following commands from the main directory (one directory above the BaseName directory):

   ```
   # Change to your path!!!
   PATH_TO_EXECUTABLE="../../source/a.out"
   #
   $PATH_TO_EXECUTABLE << END
   1
   CIUC_Cyclic_unsat_loose_Bvalue_0.42
   DStartFile
   END
   ```

   where `CIUC_pq_unsat_loose_Bvalue_0.42` is a link to the RunFile, and `DStartFile` is a link to the StartFile. The simulation might take several hours to run. If you receice the following error message,

   ```
   Insufficient space in array listJ in subroutin Jager3D
   ```

   then change `mlistJ` to 800 in the `param-dempla` file of the Fortran source code and recompile.

17.2.3 Features of the RunFile

A part of the RunFile is shown in Fig. 17. Only the middle portion is shown. The RunFile for the simulation is in the GitHub folder `examples/CIUC_Cyclic_unsat_loose`, which is present as a symbolic link to the source RunFile in the `RunFiles` folder. Most of the parameters in this RunFile are described in the Mode 2 simulation of wave propagation in Section 17.4 and in the monotonic CIUC simulation of Section 17.1.3. In this section,

```
    6.d5        : tmax     | maximum time for the simulation run
  180.0         : A_1      | shape factor for conical contact profile (A_1)
   -1.          : dt       | time increment
    0.          : gravty(1)| gravity in x_1 direction
    0.          : gravty(2)| gravity in x_2 direction
    0.          : gravty(3)| gravity in x_3 direction
    0.          : p_o      | initial fluid pressure of pore liquid
   31.8d9       : K_s      | bulk modulus of grain bodies
    2.2d9       : K_f      | bulk modulus of pore fluid
    0.95d0      : S_o      | initial fluid saturation of pore space at p_o (iporo=2,3)
  100.d3        : p_atm    | the reference atmospheric pressure (iporo=2,3)
    0.0187d0    : Hcc      | Henry's coefficient of pore liquid/gass (iporo=3)
   72.75d-3     : gamm     | surface tension of pore bubbles (iporo=3)
    0.0001d0    : D_o      | initial bubble diameter at p_o and S_o (iporo=3)
    0.          : N_o      | number/density bubbles per unit initial pore volume(iporo=3)
 2337.d0        : p_vap    | water vapor pressure relative to atm. pressure (iporo=3)
    0.          : p_wcav   | cavitation pressure relative to atm. pressure (iporo=3)
    0.          : rfree(15)| placeholder
    0.          : rfree(16)| placeholder
    0.          : rfree(17)| placeholder
    0.          : rfree(18)| placeholder
    1.5         : palpha   | alpha power in contact profile (when imodel=9)
```

															imicro							
	********** Deformation-Stress Path Segments **********							krotat			iflexc								iplot		
	icontp (100000)	(10000)	(1000)	(100)	(10)	(1)					idump									ipts2		
icontr	rate_11	rate_22	rate_33	rate_12	rate_13	rate_23	vrate	igoal	finalv	ipts						ibodyf		defdot				
111000	1 0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	70	0	5000.	500	0	0	0	0		0.		0	0		
111000	1 -25.	-25.	-25.	0.000e+0	0.000e+0	0.000e+0	0.000e+0	11	0	-100000.	100	0	0	0	0		0.		0	0		
111000	1 0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	70	0	1000.	100	0	0	0	0		0.		0	0		
111000	1 0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	50.	70	0	5400.	50	0	0	0	0		0.		0	0		
111000	1 0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.	70	0	500.	100	0	0	0	0		0.		0	0		
066000	0 -1.000e-6	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	12	0	-4.00e+05	20	0	0	0	0		0.		0	0		
066000	0 1.000e-6	-0.000e+0	-0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	12	0	-3.40e+05	20	0	0	0	0		0.		0	0		
066000	0 -1.000e-6	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	12	0	-4.00e+05	20	0	0	0	0		0.		0	0		
066000	0 1.000e-6	-0.000e+0	-0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	12	0	-3.40e+05	20	0	0	0	0		0.		0	0		
066000	0 -1.000e-6	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	0.000e+0	12	0	-4.00e+05	20	0	0	0	0		0.		0	0		

Figure 17: An example RunFile – the lower lines — for simulating cyclic triaxial undrained compression of a quasi-saturated sand specimen.

we will only describe differences between the monotonic and cyclic CIUC RunFiles. The monotonic loading RunFile is shown in Fig. 15 and Fig. 16.

An important difference between the monotonic and cyclic loading RunFiles is the value of `t_max` in the cyclic file. A value of 600,000 time steps is given for cyclic loading. The cyclic RunFile includes over 100 cycles of cyclic loading. The Fig. 17 only shows the first 5 lines of this cyclic loading in the deformation–stress control portion of the file (the last 5 lines of the figure). When running the simulation, you would not know how many cycles of loading are needed before the assembly liquefies. When liquefaction occurs and the specimen fails, DEMPLA will run indefinitely, continuing until the target terminal stress of the particular deformation–stress period is reached. The use of `t_max` will halt the DEMPLA run, so that it does not continue to run indefinitely (see Section 11.1.34 for a description of `t_max`). (In older cyclic triaxial laboratory equipment, the operator would need to stand by, ready to kill the test, so that the equipment would not destroy itself when cyclic failure had occurred.)

In Fig. 17, the first five lines of deformation–strain control are the same as those of Section 17.1.3: an initial period of quiescence to allow the particles to equilibrate; an isotropic increase in the effective stress to 100kPa; another quiescent period; an increase of the pore fluid pressure from atmospheric pressure to 270kPa above atmospheric pressure; and another quiescent period. At this stage, the total isotropic stress was −370kPa, the water pressure was 270kPa, the effective stress was −100kpa, and a water saturation was 99.939%.

The next five lines are the beginning of cyclic CIUC loading (there are at least 200 of these lines). In all of these lines, `icontp=0`, meaning that the inflow of fluid is being controlled. The value of `defrat(8)=0.` (i.e., `vrate`), meaning that the rate of inflow (as a compressive strain in the pore fluid) is zero. Undrained conditions are maintained with these values of `icontp` and `defrat(8)=0` (see Section 11.2.4). With these lines, the value of `defrat(1)` in the x_1 direction (i.e., `rate_11`) alternates between `-1.000e-6` and `1.000e-6`, meaning that these are changes (per time step) in the deformation gradient F_{11} (see Section 11.2.3). The alternation of strain rate produces the cyclic loading. When the rate is `-1.000e-6`, the value of `igoal=12`: the first digit "1" means that terminal condition is related to the x_1 direction of deformation or stress, anf the second digit "2" means that the terminal condition is a total stress (i.e., the total stress σ_{11}, see Section 11.2.5). The value of `finalv=-4.00e+05`, meaning that this deformation–stress control period is terminated when the stress becomes −400kPa (remember that the ambient total stress is −400kPa and the x_1 `defrat(1)` is negative). See Section 11.2.7 for a description of `finalv`. These lines alternate with lines having a terminal total stress of −340kPa. Thus, the cyclic amplitude is 30kPa.

17.3 Mode 2 wave propagation in dry soil column

17.3.1 Introduction

This example is described in the author's paper [2], in Sections 4 and 4.1. The simulation is intended to give a realistic simulation of wave propagation that compares well with a centrifuge test with Nevada Sand. The simulation is of a 11 m deposit of dense dry sand at $D_r = 101\%$ subjected to horizontal shaking with a seismic motion [22]. Although the paper gives the results of a set of three separate simulations (and centrifuge tests), only

one of the simulations is described here. The set of three centrifuge tests was performed at the University of California, Davis, on a homogeneous, level, and dry deposit of dense sand with $D_r = 100\%$ [22]. The tests were performed on a model of 558 mm depth. Two tests, including the simulation of this example were at a lower centrifuge speed (acceleration 20 g, DKS02_bb and DKS02_be), simulating an 11 m thick deposit; the third test (not described here), at a higher speed (acceleration 40 g, DKS02_bl), simulated a 22 m thickness. The centrifuge box was a flexible shear beam, and in the simulations, the sand's mass was increased by 24% to account for the box mass (e.g., [23]). Three different seismic motions, taken from the same flight DKS02, were selected for simulation (only one seismic motion is used with this example). The base motions were scaled versions of the 1989 Loma Prieta SCZ090 record, available from the Strong-Motion Virtual Data Center [24]. As was done in [23], the motions at model depth 483 mm (accelerometer A17) were used as input for the simulations. The data streams from all instruments are available from [25], and the full program is documented by Stevens et al. [22].

17.3.2 Setup and running the simulation

The GitHub repository contains all of the files for performing this simulation. These are the steps for running DEMPLA on a Linux system.

1. create the executable file from the fortran source code. The steps for doing this with the gfortran compiler are described in Section 4.1. You will need to know the name of the executable file and its path.

   For this simulation the parameter `mlistJ` was set to 200 in the `param-dempla.X.X.XX.f` file (an `imodel=9` blunted-cone Jager cont was used in the simulation), which was sufficient for the simulation.

2. from the main directory (folder) from which you are running the simulation, create a link to the directory "`MotionInput`" that contains the MFiles of motion histories (see Section 9 for a description of MFiles). For example, if you are using the same directory structure as in the github archive, you could run the simulation from within the `examples/` directory. Note that the `MotionInput` directory is not located inside of `examples/`, but it probably contains a symbolic link (symlink, similar to a shortcut in Windows or an Alias in MacOS) to the `MotionInput` directory. Create a link commands similar to these:

   ```
   cd <folder from which you are running the simulation>
   ln -s <path to the MotionInput directory> MotionInput
   ```

   Note: if the directory from which you are running the simulation contains the actual `MotionInput` folder, then you won't need to create the link.

   The particular MFile for this simulation is named `Motion_DKS02_bb_A17_Model`, and it is included in the github repository. Make sure that this file is included in your own `MotionInput` directory.

3. from the same directory (the main directory from which you are running the simulation), create a link to the directory `RunFiles` that contains the RunFiles (these files give the parameters for running the DEM simulations at each RVE, see Section 11).

   ```
   cd <folder from which you are running the simulation>
   ln -s <path to the RunFiles directory> RunFiles
   ```

   Note: if the directory from which you are running the simulation contains the actual `RunFiles` directory, then you won't need to create the link.

4. from the same directory (the main directory from which you are running the simulation), create a link to the directory `StartFiles` that contains the StartFiles (these files give the parameters for running the DEM simulations at each RVE, see Section 12.1).

   ```
   cd <folder from which you are running the simulation>
   ln -s <path to the StartFiles directory> StartFiles
   ```

   Note: if the directory from which you are running the simulation contains the actual `StartFiles` folder, then you won't need to create the link.

5. create the directory with the BaseName of the simulation, along with three additional directories. The BaseName of this particular simulation is `DKS02_bb_B_mu005_f040_A17`:

   ```
   mkdir DKS02_bb_B_mu005_f040_A17
   mkdir DKS02_bb_B_mu005_f040_A17/02_LayerSetup
   mkdir DKS02_bb_B_mu005_f040_A17/03_RVESetup
   mkdir DKS02_bb_B_mu005_f040_A17/04_RunOutput
   ```

6. create the GFile that gives the conditions of the soil column (see Section 7). Place this GFileinside of the BaseName folder (in this example, the folder `DKS02_bb_B_mu005_f040_A17`). For this example, the GFileis named `DKS02_bb_B_mu005_f040_A17a`, with an "a" suffix. This file is included in the github repository, inside of the `examples/DKS02_bb_B_mu005_f040_A17/` directory.

7. create the LFiles that give information about each soil layer (see Section 8). Place these LFiles inside of the BaseName folder (in this example, the folder `DKS02_bb_B_mu005_f040_A17`). For this example, the soil column has only one stratigraphic layer, and the single LFile, named

   ```
   L0001_DKS02_bb_B_mu005_f040_A17
   ```

   The file is included in the github repository, inside of the `examples/DKS02_bb_B_mu005_f040_A17/` directory.

8. now run the DEMPLA simulation by entering the following commands from the main directory (one directory above the BaseName directory):

```
# Change to your path!!!
PATH_TO_EXECUTABLE="../source/a.out"
#
time $PATH_TO_EXECUTABLE << END
2
DKS02_bb_B_mu005_f040_A17
a
Motion_DKS02_bb_A17_Model
END
```

17.4 Features of the simulation and input files

Centrifuge tests are conducted at elevated accelerations on small models, which are intended to represent larger prototype systems. The centrifuge model had a depth of 558 mm, although the base motion was recorded in the centrifuge experiment at a depth of 483 mm, which was used as the column height for the simulation (`BaseDepth`, see Section 7.1.7). A scaling factor of 20 was applied to to simulate an 11 m depth (hence a gravity of 20 g was used, a `grav` of 196.14 m/s^2, see Section 7.1.13). Scaling factors are applied to results of the model system to compute the corresponding prototype response [26]. Similar to centrifuge tests, the simulations herein were conducted at model-scale, avoiding the problem of matching the scales of multiple quantities in the prototype system (particle size, viscosity, stress, depth, surface tension, permeability, etc.). The simulation results were then converted to prototype-scale as with the centrifuge studies, so that results could be compared (acceleration response spectra, pore pressures, movements, etc.).

The sand use in the centrifuge tests was Nevada Sand, a fine uniformly graded sand (SP) used in many laboratory and centrifuge programs, including the VELACS program of the 1990s [27, 28, 29]. For the centrifuge study, the sand was prepared as a homogeneous mass within a centrifuge box, and seismic base motions were applied in a single lateral direction with no intentional vertical motion. The containers was a flexible shear-beam, creating nearly 1-D conditions in lateral directions (however, friction along side walls restricts vertical movement).

A series of DEM assemblies, each with 10,648 particles (`np`, see Section 12.1.2), was prepared to simulate Nevada Sand at a range of densities. Unlike FEM simulations in which a macro-scale constitutive model is chosen along with its macro-scale parameters, DEM simulations rely solely on micro-scale information. This information is of four types, listed in the order in which they are assigned: (1) a geometric description of the particles' sizes and shapes and of their arrangement, (2) a contact normal-force model that defines the particles' one-to-one displacement–force interactions, (3) a frictional contact model, and (4) a poromechanic relationship between pore volume and fluid pressure. Ideally, the macro-scale behavior of a DEM assembly resembles that of the intended soil, in regard to density, stiffness, and liquefaction resistance. The author relied upon laboratory tests of Nevada Sand conducted by Arulmoli [27] and Kwan [30] to check and calibrate the DEM assemblies' properties.

DEM particles were created with a distribution of sizes similar to that of Nevada Sand,

with a median size (by cumulative weight) $D_{50} = 0.165$ mm and a polydisperity $C_u = 2.1$. A non-spherical particle shape was adopted, because assemblies of spheres have less bulk strength than those of sands, and because the range of packing densities for spheres differs from that of sand particles. Although progress has been made to mimic the shapes of sand particles in DEM simulations [31], the author chose a simpler proxy shape: a non-convex cluster of seven overlapping spheres (see [6] for a rendering, `nobs=6` plus a central sphere, see Section 12.1.5), allowing multiple contacts (interlocking) between two particles, and thus inhibiting inter-particle rolling. The correlations of Cho et al. [32] guided the selection of the relative sizes and overlaps of the seven spheres, leading to a particle shape that could be compacted to a range of void ratios similar to Nevada Sand.

The various shape parameters for the particles are given in the top portion of the DFile, which is included in the Github repository, in the StartFiles directory. The name of the particular StartFile for this simulation is as follows:

```
DEquil_Template_Frict_0.05_DBumpies-10648-6-06-08-064-c_Frict_0.05_LRforce_-0.20e-4-Int10-1-
```

DEMPLA knows that this is the StartFile for the simulation, because its name given in the LFile for the single soil layer of the simulation (`StartFileLayer`, see Section 8.8). The name of the particular LFile is as follows:

```
L0001_DKS02_bb_B_mu005_f040_A17
```

and it is found in the `examples/DKS02_bb_B_mu005_f040_A17` folder. A bumpy particle shape was used for all particles (`kshape=7`, the first line in the DFile, see Section 12.1.1). The bumpies had 6 satellite spheres and a central sphere (`nobs=6` plus a central sphere, see Section 12.1.5). The values of `satrad`, `cenrad`, and `cirrad` are given in 5th line of the DFile (see Section 12.1.6).

A suite of DEM assemblies was created, with void ratios ranging from 0.511 to 0.892 (see [3] for assembly protocol), similar to the range e_{min} to e_{max} of Arulmoli [27] (0.511 to 0.887). Different density ranges are reported in the various centrifuge studies, so the D_r value of each study was converted to a void ratio, and the DEM assembly with the closest void ratio was used for the study's simulation. At the start of each simulation, the assemblies were anisotropically consolidated (with zero lateral strain) to the geostatic conditions of the RVE points.

The contacts between particles were modeled as blunted cone asperities (see [5, 6, 33] for details about this type of contact). The parameters for this contact model are given in the RunFile for the single layer of the simulation. The name of this RunFile is given in the LFile for the layer (the file `L0001_DKS02_bb_B_mu005_f040_A17` in the `examples/DKS02_bb_B_mu005_f040_A17` folder). The name is given as the input value of `RunFileLayer` in the LFile (see Section 8.7). The name of the RunFile is as follows:

```
Prep_B_frict_040_Isotropic_b_Gen1.5_Aalph_100_dry
```

and the file is included in the Github repository in the `RunFiles` directory (see the second bullet in Section 6.2).

The parameters of the blunted cone asperities are given in the RunFile. These include `imodel` (specifying a blunted cone, Section 11.1.12); the shear modulus `G` (or `kn`, see Section 11.1.22); the Poisson ration `kratio` (see Section 11.1.23); the cone shape parameter `A_1` (see Section 11.1.35); and the cone shape parameter `palpha` (see Section 11.1.50).

With this contact model, the shear modulus G of the bulk material was measured as 104 MPa at mean stress $\bar{p} = 80$ kPa for strain γ of 0.0002% at void ratio $e = 0.653$, results

Table 6: Poromechanic parameters for DEM simulations (see [3]).

Property		Input	Section	Value
K_s,	bulk modulus, grains	`K_s`	11.1.39	31.8 GPa
K_w,	bulk modulus, liquid water	`K_f`	11.1.40	2.2 GPa
ν_s,	Poisson ratio, grains	`kratio`	11.1.23	0.15
k_G,	hydraulic conductivity	`k_geot`	7.2.3	0.0034 cm/s
μ_a,	viscosity, air	`visc_a`	7.1.10	0.019 mPa·s
μ_w,	viscosity, water	`visc_w`	7.1.9	0.008 Pa·s
$p_{w,vap}$,	vapor pressure, water	`p_vap`	11.1.47	2.34 kPa
γ,	surface tension, water–air	`gamm`	11.1.44	0.073 N/m
H^{cc},	Henry's solubility, water–air	`Hcc`	11.1.43	0.0173
p_{atm},	atmospheric pressure	`p_o`	11.1.38	100 kPa
D,	bubble diameter	`D_o`	11.1.45	0.0001 m
S_o,	initial saturation	`S_o`	11.1.41	0%

similar to Arulmoli's tests on specimens with relative density $D_r = 60\%$. The contact profile produced a dependence of G on the mean stress $\bar{p}$, with proportionality $G \propto \bar{p}^{0.50}$, along with a dependence of G on e similar to Nevada Sand.

A DEM assembly's susceptibility to liquefaction is sensitive to the friction coefficient μ that is assigned to the particles' contacts. A coefficient of 0.40 was found to give results similar to the cyclic simple-shear tests of Kwan [30], noting that this type of testing is most similar to that of the centrifuge tests (i.e., vertical propagation of s-waves). The value of 0.40 is input with the RunFile as parameter `frict` (Section 11.1.24). The annular-slip Jager contact model was used to compute tangential forces that result from complex sequences of normal and tangential movements [5].

The poromechanic properties of the DEM assemblies are given in Table 6, and except for D, k_G and S_o, are accepted values for quartz, water, and air. Table 6 also gives the names of the input parameters and the corresponding sections of this guide. These input values are given in the RunFile, GFile, and LFile. The assumed bubble diameter D is roughly the size of inter-grain pores, and its value is used to compute the dissolution of air in water for quasi-saturated mixtures. Although different hydraulic conductivities k_G have been reported for Nevada Sand, the simulation values were based on the results of Arulmoli [27], approximated as $k_G = (0.3 \text{ mm/s})\, e^3/(1+e)$.

Besides the input values described above, the GFile contains other input: 17 RVEs were used in the simulation (`nRVEs=17`, see Section 7.1.3); the water table was placed below the soil depth, with at table depth of 1 m (`WaterDepth`, see Section 7.1.8); the mass of the centrifuge box rings increased the soil mass by a factor of 1.20 (`centrifuge`, see Section 7.1.14); and the ground surface and water table were horizontal (`Dip_soil=0.`, see Section 7.1.20).

Besides the above input, other input information is given in the LFile (see step 7 at the beginning of this Section 17.3.2). The file `L0001_DKS02_bb_B_mu005_f040_A17` gives the thickness of the single layer as 20 m, which is thicker than the 483 mm height of the soil column, so this one soil layer comprises the entire column (`LayerThickness`, see Section 8.4). The DEM assemblies will be anisotropically consolidated for each of the RVE levels (`Isotropic=2`, Section 8.3). The initial void ratio is said to be 0.563, even though the

DEM assemblies may have a different void ratio. The evolution of void ratio is computed from the volumetric strains and the original 0.563 value rather than from the actual void ratio of the DEM assembly. Parameter `VoidRatio-0.563`, see Section 8.5. Finally, the specific gravity of the soil solids, G_s is said to be 2.67 (`G_s`, Section 8.6). This specific gravity is use in DEMPLA to compute the density of the soil layer so that geostatic stresses can be computed and the soils mass can be found for the wave propagation analysis.

The MFile that gives the motions (i.e., the time-history of displacements) at the base of the soil column has this name,

`Motion_DKS02_bb_A17_Model`

This file should be located in the `MotionInput` folder. See step 2 near the beginning of this Section 17.3.2 and the first bullet of Section 6.2. The input base motions were obtained from published seismic acceleration records. The simulations require the base displacements, so accelerations were double-integrated and, when necessary, adjusted to end with zero velocity. See Section 9 for the format of MFiles.

Another important aspect of the RunFile is the values `rho=-1.` and `dt=-1.`. With these values, there is no need to decide upon a value of the DEM time step. The time step is set to 1 second, and the masses of the particles are adjusted, so that the simulation will be nearly quasi-static.

17.5 Other Mode 2 simulations of wave propagation in centrifuge tests

Besides the simulation described in Section 17.3, the GitHub `examples/` directory also contains the simulations of ten other centrifuge tests, all described in the paper [2]:

`DKS02_be_B_mu005_f040_A17/`
`DKS02_bl_B_mu005_f040_A17/`
`DKS04_15_B_mu005_f040_A17/`
`DKS04_41_B1_mu005_f040_A17/`
`DKS04_41_B_mu005_f040_A17/`
`DKS04_49_B_mu005_f040_A17/`
`Kramer_B1_Air_mu019_f040/`
`L45V_2_10_B_mu022_f040/`
`L65V_2_10_B_mu014_f040/`
`L75V_2_10_B_mu010_f040/`

18 Octave (Matlab) files

The GitHub repository includes some Octave function files that can be used for analyzing DEMPLA data. Octave is an open source software that is almost fully compatible with Matlab. The Octave files include the following:

- `Read_A_file6.m`: a function for reading AFiles.
- `Read_B_file7.m`: a function for reading BFiles.
- `Read_Dempla2.m`: a function to read the Mode 2 wave propagation ResultsFiles that are described in Section 14.

- `Read_D_file3.m`: a function to read the contents of a DFile.
- `Shapes3.m`: a utility function to assign numbers to particles' shapes. A dependency of other functions. See Section 12.1.1 for a list of particle shapes (parameter `kshape`).
- `Write_Dfile_Bumpies.m`: a function for writing assembly information to a DFile.
- `Write_Dfile_Spheres.m`: a function for writing a DFile of an assembly of spheres (see Section 12.2.4).
- `Write_Dfile_Ovoids.m`: a function for writing a DFile of an assembly of ovoids (see Section 12.2.5).
- `Write_Dfile_Bumpies.m`: a function for writing a DFile of an assembly of bumpy particles (a sphere cluster, see Section 12.2.7).
- `Append_RunFile_line3.m`: this function appends a line of input to the bottom of an existing RunFile. The line gives data for a single period of the deformation–stress control path (see Section 11.2).

19 Some advice on using Dempla

When Dempla(or Oval) is properly used, the program is efficient and provides repeatable results. The program can be maddening, however, when it is unknowingly being stretched beyond its limits. You may want to consider the following advice.

1. When you are seeking quasi-static conditions, slow is (usually) better. When choosing deformation or stress rates with the input values `defrat(i,j)`, the program's performance can be greatly improved by choosing appropriately slow rates (Section 11.2.1 and Section 11.2.3). What is an appropriate rate? If the rate is too slow, you will needlessly waste time (days, weeks, months) waiting for your simulation to finish. If the rate is too fast, the program will either fail to maintain quasi-static conditions (when `algori=1`, Section 11.1.2), will run excessively slow while attempting to establish quasi-static conditions (when `algori=2`), or will crash. A common error message upon crashing is the following:

   ```
   An illegitimate contact in subroutin lister.
   ```

   This error occurs when the particle velocities are excessive, causing two particles that were previously not even in the linked-list of near-neighbors to come into contact within a single time step (Section 11.1.27). Dempla will not stop running, but if this message repeatedly occurs or the results become erratic, you will probably want to reduce the deformation rate. (I am fairly tolerant of this error when I am not particularly interested in the accuracy of the results, for example when I am preparing an assembly from a sparse arrangement of particles.)

   The proper deformation rate depends primarily on (and is almost proportional to) the average overlap among particles in their initial configuration. If the overlaps are too large (relative to the particle radii), the simulation will not be very realistic. With

smaller average overlaps, slower deformation rates will be required to maintain nearly quasi-static conditions. Note that the relative overlaps are fairly small for the sample assemblies that are included in the DEMPLA package (Table 5, page 87).

The best way to determine a suitable deformation rate may be by trial and error. If you are testing an initially dense assembly, choose `algori=2` and try a few `defrat` values. The first deformation-stress control segment, however, should not involve any deformation. A beginning period of quiescence is required to allow the initial particle arrangement to come to near-equilibrium. The first segment should, therefore, have `icontr=000000` and should be long enough (`igoal=70`, with sufficient time `finalv`) to allow a enough steps for the initial particle arrangement to equilibrate (Section 11.2.5 and Section 11.2.1).

The second and subsequent stress-deformation control segments are where you can test the deformation rate. As the program runs and output appears on the screen, wait until these segments are entered (see the `istep` values on the screen output, page 86), and then monitor the values of `chi1`, `chi2`, and `xloops` (Sections 13.3.5, 13.3.9, and 13.3.15). The option `algori=2` provides a minimum of 3 equilibrating time steps per deformation step, with a maximum of 101 time steps (Section 11.1.2). If `xloops` is consistently 3.0, you can probably increase the trial deformation rate. If `xloops` is consistently much greater than 3.0, then the deformation rates are probably too high. I usually try to keep `xloops` near 3 or 4 and `chi1` and `chi2` below 0.005.

2. For diffuse, sparse assemblies, use `algori=1` (Section 11.1.2). This will be the case if you are trying to compact a gaseous assembly into a dense one. For dense assemblies, use `algori=2`, as this will enforce a self-regulating mechanism for maintain nearly quasi-static conditions.

3. I sometimes give the particles initial velocities (`rmsvel>0`) to help in densifying an initially loose particle arrangement. As has already been stated, slow is usually better. If you get the message,

   ```
   An illegitimate contact in subroutin lister.
   ```

   then you have probably assigned an `rmsvel` value that is too large.

4. When starting a simulation with a D-file or E-file, the particles will likely not be in an equilibrium configuration, since these files only specify the particle positions and provide no information about the contact forces (see `istart=3`, Section 11.1.7). This means that the initial calculation of contact forces will give zero tangential force, a condition not likely to produce equilibrium. The first deformation-stress segment, therefore, should not involve any deformation. Instead, a beginning period of quiescence is required to allow for the initial particle arrangement to come to near-equilibrium. The first segment should, therefore, have `icontr=000000` and should be long enough (`igoal=70`, with sufficient time `finalv`) to allow a few time steps for the initial particle arrangement to equilibrate (Sections 11.2.5 and 11.2.1). The second and subsequent stress-deformation control segments are where you can start the desired deformation process.

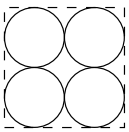
5. Binary C-file formats might not be portable between different platforms or operating systems.

6. If you are assigning initial random velocities to the particles (an input value `rmsvel`$\neq$`0`), do not assign velocities that are too large. You will probably need to reduce the value of `rmsvel` if you get the error message:

   ```
   An illegitimate contact in subroutine lister.
   ```

   See item 1 above.

7. DEMPLA (and before it, OVAL) does not work when the assembly contains too few particles, say fewer than nine 2D particles or twenty-seven 3D particles. This problem is related to the use of periodic boundaries. As an example, consider a square arrangement of four particles of equal size with this arrangement:

   Each particle touches a neighboring particle twice: once within the core assembly and once again across a periodic boundary. DEMPLA's underlying data structure allows for a single contact per particle pair. This problem is avoided with a larger number of particles.

8. Do not try to control a boundary stress when the boundary stress is zero. For example, an input value `icontr=11100` will not work with a diffuse, gaseous assembly (Section 11.2.1).

9. If you get an error message that your input files can not be properly read, you will want to carefully check the formatting of the file's input fields. Microsoft Windows users may have problems with hidden characters that can be embedded in files when using Word and Word Pad. You will probably want to install a genuine text processor and avoid using word processors.

10. If at the start of a simulation you get the unexpected message, "`Name of a platen file:`", then you have probably given `nplatn` a non-zero value in your RunFile (see Section 11.1.13).

20 Change Log

This section documents the changes that have been made between various version of OVAL and DEMPLA.

21 References

[1] P. A. Cundall and O. D. L. Strack. A discrete numerical model for granular assemblies. *Géotechnique*, 29(1):47–65, 1979.

[2] M. R. Kuhn. Multi-scale simulation of wave propagation and liquefaction in a one-dimensional soil column: hybrid DEM and finite-difference procedure. page in review.

[3] M. R. Kuhn and A. Daouadji. Simulation of undrained quasi-saturated soil with pore pressure measurements using a discrete element (DEM) algorithm. *Soils and Found.*, 60(5):1097–1111, 2020. ISSN 0038-0806. doi: https://doi.org/10.1016/j.sandf.2020.05.013.

[4] J. Jäger. *New Solutions in Contact Mechanics*. WIT Press, Southampton, UK, 2005.

[5] M. R. Kuhn. Implementation of the Jäger contact model for discrete element simulations. *Int. J. Numer. Methods Eng.*, 88(1):66–82, 2011. doi: https://doi.org/10.1002/nme.3166.

[6] M. R. Kuhn, H. Renken, A. Mixsell, and S. Kramer. Investigation of cyclic liquefaction with discrete element simulations. *J. Geotech. and Geoenv. Eng.*, 140(12):04014075, 2014. doi: https://doi.org/10.1061/(ASCE)GT.1943-5606.0001181.

[7] D. O. Potyondy and P. A. Cundall. A bonded-particle model for rock. *Int. J. Rock. Mech. and Mining Sci.*, 41(8):1329–1364, 2004.

[8] J. Jäger. Uniaxial deformation of a random packing of particles. *Arch. Appl. Mech.*, 69(3):181–203, 1999.

[9] M. Satake. A discrete-mechanical approach to granular materials. *Int. J. Eng. Sci.*, 30 (10):1525–1533, 1992.

[10] M. Satake. New formulation of graph-theoretical approach in the mechanics of granular materials. *Mech. Mater.*, 16:65–72, 1993.

[11] M. R. Kuhn. Structured deformation in granular materials. *Mech. of Mater.*, 31(6): 407–429, 1999.

[12] M. Satake. Fabric tensor in granular materials. In P. A. Vermeer and H. J. Luger, editors, *Proc. IUTAM Symp. on Deformation and Failure of Granular Materials*, pages 63–68. A.A. Balkema, Rotterdam, 1982.

[13] F[ranco] P. Preparata and M[ichael] I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

[14] C. Thornton and C. W. Randall. Applications of theoretical contact mechanics to solid particle system simulation. In M. Satake and J.T. Jenkins, editors, *Micromechanics of Granular Materials*, pages 133–142. Elsevier Science Pub. B.V., Amsterdam, The Netherlands, 1988.

[15] X. Huang, K. J. Hanley, C. O'Sullivan, and C.-Y. Kwok. Implementation of rotational resistance models: a critical appraisal. *Particuology*, 34:14–23, 2017.

[16] H. B. K. Nguyen, M. M. Rahman, and A. B. Fourie. Characteristic behavior of drained and undrained triaxial compression tests: DEM study. *J. Geotech. and Geoenv. Eng.*, 144(9):04018060, 2018.

[17] W. Pabst and E. Gregorová. Elastic properties of silica polymorphs—a review. *Ceramics-Silikaty*, 57(3):167–184, 2013.

[18] B. O. Hardin and F. E. Richart. Elastic wave velocities in granular soils. *J. of Soil Mech. and Found. Div., ASCE*, 89(1):33–66, 1963.

[19] T. Iwasaki, R. Tatsuoka, and Y. Takagi. Shear moduli of sands under cyclic torsional shear loading. *Soils and Found.*, 18(1):39–56, 1978.

[20] K. J. Hanley, X. Huang, C. O'Sullivan, and F. Kwok. Challenges of simulating undrained tests using the constant volume method in DEM. In *AIP Conference Proceedings*, volume 1542, pages 277–280. AIP, 2013.

[21] K. Suzuki and M. R. Kuhn. Uniqueness of discrete element simulations in monotonic biaxial shear tests. *Int. J. Geomech.*, 14(5):06014010, 2014. doi: https://doi.org/10.1061/(ASCE)GM.1943-5622.0000365.

[22] D. K. Stevens, B. I. Kim, D. W. Wilson, and B. L. Kutter. Comprehensive investigation of nonlinear site response–centrifuge data report for DKS02. Technical Report Data Report UCD/CGMDR-99/02, Center for Geotechnical Modeling, Univ. of Cal., Davis, 1999.

[23] A. Elgamal, Z. Yang, T. Lai, B. L. Kutter, and D. W. Wilson. Dynamic response of saturated dense sand in laminated centrifuge container. *J. Geotech. and Geoenv. Eng.*, 131(5):598–609, 2005. doi: https://doi.org/10.1061/(ASCE)1090-0241(2005)131:5(598).

[24] Center for Engineering Strong Motion Data. Strong-Motion Virtual Data Center (VDC), Jan. 2021. URL https://www.strongmotioncenter.org/vdc/scripts/default.plx.

[25] Center for Geotechnical Modeling. University of California, Davis, Jan. 2021. URL https://cgm.engr.ucdavis.edu/about/.

[26] R. Butterfield. Scale-modelling of fluid flow in geotechnical centrifuges. *Soils and Found.*, 40(6):39–45, 2000. doi: https://doi.org/10.3208/sandf.40.6_39.

[27] K. Arulmoli, K. K. Muraleetharan, M. M. Hossain, and L. S. Fruth. VELACS verification of liquefaction analyses by centrifuge studies laboratory testing program soil data report. Technical Report Project No. 90-0562, The Earth Technology Corporation, Irvine, CA, 1992. Data available through http://yees.usc.edu/velacs.

[28] K. Arulmoli. VELACS: selection, distribution and laboratory testing of soils. In K. Arulanandan and R. F. Scott, editors, *Verifications of Numerical Procedures for the Analysis fo Soil Liquefaction Problems*, volume 2, pages 1281–1292. Balkema, Rotterdam, 1994.

[29] B. L. Kutter, Y.-R. Chen, and C. K. Shen. Triaxial and torsional shear test results for sand. Technical Report CR 94.003, Office of Naval Research, Arlington, VA, 1994.

[30] W. S. Kwan. *Laboratory investigation into evaluation of sand liquefaction under transient loadings*. Dissertation, University of Texas at Austin, 2015.

[31] K.-W. Lim and J. E. Andrade. Granular element method for three-dimensional discrete element calculations. *Int. J. Numer. and Anal. Methods in Geomech.*, 38(2):167–188, 2014. doi: https://doi.org/10.1016/j.cma.2012.06.012.

[32] G.-C. Cho, J. Dodds, and J. C. Santamarina. Particle shape effects on packing density, stiffness, and strength: natural and crushed sands. *J. Geotech. and Geoenv. Eng.*, 132 (5):591–602, 2006. doi: https://doi.org/10.1061/(ASCE)1090-0241(2007)133:11(1474).

[33] M. R. Kuhn and A. Daouadji. Quasi-static incremental behavior of granular materials: Elastic–plastic coupling and micro-scale dissipation. *J. Mech. Phys. Solids*, 114:219–237, 2018. ISSN 0022-5096. doi: https://doi.org/10.1016/j.jmps.2018.02.019.