Pkdgrav/Soft-Sphere DEM Tutorial

Last Update 7/31/18 by Derek C. Richardson

This tutorial assumes familiarity with a Linux-based operating system and the command line.[1]

1. Software Installation

a. You will be provided with a recent version of the pkdgrav source code, something like `pkdgrav_current-v1.0.0.zip`. It is important to note the version you have (in this example, `1.0.0`), to help with any debugging. The master code is maintained on github in a private repository. Contact DCR if you need access.

b. Place the source code somewhere (recommended: `~/src`), and unzip it: `unzip pkdgrav_current-v1.0.0.zip` — this will create a subdirectory that for ease of this tutorial should be renamed: `mv pkdgrav_current-v1.0.0 pkdsrc` (but it is recommended to keep the full name, or at least rename it back later, if you expect to install multiple versions down the line).

c. Build the package for basic operation:
   i. `cd pkdsrc`
   ii. Edit the file `Makefile.in` to check only `USE_COLLISIONS=true` under "pkdgrav functionality options" is uncommented (no leading hashtag).[2]
   iii. `make` [Macs need the XCode command line tools installed!]

d. Make sure everything works by performing the simple self-test:
   i. `cd ssrun`
   ii. `./sstest`
   iii. Type "null".
   iv. Keep pressing "ENTER" to run successive tests. CTRL-C to quit.

e. In order to perform all the steps in this tutorial, you also need several 3rd-party packages (you may be able to use an installation utility, e.g., MacPorts on Mac, or Ubuntu's `apt-get` command, if you have superuser privilege).
   i. `ffmpeg` — get it from http://www.ffmpeg.org/download.html (source code and static binaries are available). Place the executable in, say, `~/bin` (create the directory if it does not exist) if it doesn't install automatically. For Mac with MacPorts installed, `sudo port install ffmpeg` should work. For Ubuntu, try `sudo apt-get install ffmpeg`.
   ii. `povray` — get it from http://www.povray.org/download/ (source or binary).
      - *Note on installation: one time I had to "./install -no-arch-check", choose option "U", specify "/PATH/lib/povray" (where PATH is the absolute path to my home directory; this created the directory*

[1] For extra practice, try https://www.codecademy.com/en/courses/learn-the-command-line

[2] If you are not used to editors in Linux, such as `vi` or `emacs`, try `nano` or `pico`.

"lib/povray" there—you can specify whatever you want, but note lots of files will be installed), and then "ln -s ~/lib/povray/bin/povray ~/bin/povray" (this puts the binary in your search path; be sure to refresh either by typing "rehash" (csh) or starting a new shell); to use, copy pkdsrc/etc/povray.inc to your run directory; see below.

- Again, with MacPorts or Ubuntu, etc., try `install povray` first!

iii. `netpbm` (http://netpbm.sourceforge.net/) if image file conversion tools like `tgatoppm` are not available on your system.

f. Finally, add the executable locations to your search path. E.g., add the following to your `~/.cshrc` file, after any other paths are set and before the setup exits if not interactive (remember to source the file afterward, or start a new shell):

```
set path = ($path ~/bin ~/src/pkdsrc/bin/Linux-x86_64 ~/src/pkdsrc/bin/scripts)
```

The "`Linux-x86_64`" part above depends on your operating system and machine architecture.[3] E.g., on a Mac, this would be "`Darwin-x86_64`". Note if you're using `bash` shell, the equivalent path-setting line in `~/.bashrc` would be:

```
export PATH=$PATH:~/bin:~/src/pkdsrc/bin/Linux-x86_64:~/src/pkdsrc/bin/scripts
```

Note, `.cshrc` (or equivalent) may not exist in your home directory, in which case just create it. For the changes to take effect, either `source` the dot file or logout (exit the shell) and log back in again.

2. Setting Up a New Run

a. For this tutorial we will generate 2 (soft-sphere) rubble piles and smash them into each other. First, we need to recompile pkdgrav with SSDEM support turned on:

i. `cd ~/src/pkdsrc`

ii. Edit `Makefile.in` and uncomment the line "#USE_DEM=true" (remove the leading hashtag). Also uncomment "#USE_DEM_FIXED_BALL=true" and "#USE_DEM_ROTATION_DASHPOT=true".

iii. `make`

b. Next, create a run directory and copy some template files (as well as the new `pkdgrav` executable) into it:

i. `mkdir ~/Run; cd ~/Run`

ii. `cp ~/src/pkdsrc/etc/rpg.par .`

iii. `cp ~/src/pkdsrc/etc/ss.par .`

iv. `cp ~/src/pkdsrc/etc/ssdraw.par .`

v. `cp ~/src/pkdsrc/bin/Linux-x86_64/pkdgrav .`

Again, replace "`Linux-x86_64`" with "`Darwin-x86_64`" on a Mac.

c. Now let's make the rubble piles. First, run `rpg` — this makes the file `rpg.ss`, containing a simple rubble pile. Then, run `rpx` and enter the following sequence

[3] You can check for yourself what this label will be by typing `uname` and `uname -m` at a shell prompt.

of commands at the prompts (this will result in two 1-km radius rubble piles separated by 0.5 km approaching one another at 10 m/s in the *x* direction):

i. [CR] (this means press "enter" to choose the default option)
ii. rpg.ss
iii. 0
iv. rpg.ss
v. 4
vi. [CR]
vii. 2.5 0 0
viii. 5
ix. [CR]
x. -10 0 0
xi. 8
xii. 5
xiii. 0
xiv. [CR]
xv. [CR]
xvi. initcond.ss

d. Let's check to make sure everything looks right. Edit the file `ssdraw.par` and change "Starting view size" from 0 to –6000, run `ssdraw initcond.ss`, then `xv initcond.ras` (assuming `xv` or an equivalent image viewer such as `display` that can handle `.ras` files is installed; otherwise it may be necessary to convert the image to something else, e.g., `rastoppm initcond.ras | ppmtogif > initcond.gif`, then use a generic image viewer or copy the file somewhere you can view it).[4]

e. Before we run `pkdgrav`, we need to do a few preliminary calculations. The soft-sphere discrete element method (SSDEM) resolves collisions by allowing particles to interpenetrate and experience repulsion and friction forces (see Schwartz et al. 2012[5]). This introduces a lot of material parameters that must be specified carefully in conjunction with the integration timestep. The `demparams` utility was created to help with this step, but we need the following information first (the units are chosen to be compatible with `demparams`):

i. Particle mass (3.35103e13 g; get this from the screen output of `rpg`, or by running `ssinfo initcond.ss`).
ii. Particle radius (1.27441e4 cm; ditto).
iii. Rubble pile surface gravity ($g = GM/R^2$, where R = 1 km (or 1e5 cm) and M = 7.4728e+12 kg from `ssinfo`—remember to divide by 2, giving $g \sim 0.0005$ m/s^2).

[4] If you're remotely viewing a Linux session on your Mac, you may need to install X11 first (it's not automatic anymore)—visit http://xquartz.macosforge.org/ for the self-install package, and use `ssh -Y` when logging in to enable X11 forwarding.

[5] http://www.astro.umd.edu/~dcr/reprints/schwartz_gm14,363.pdf

Now run `demparams` (choose the default at the first prompt, to NOT use system units), then change the particle mass and radius (options 1 & 2), surface gravity (6), and bulk radius (7, set it to *R* in cm). Also set the speed to 1000 cm/s (option 4), which is how fast we set the rubble piles colliding. You can now read off that the recommended value for the SSDEM normal spring constant is 2.616767e-05 in `pkdgrav` units and the recommended timestep is 1.878896e-09. (Note: `pkdgrav` takes the gravitational constant to be unity, so for solar system problems, masses are expressed in solar masses, lengths in au, and times in yr/2π (so speeds are in 2π au/yr, or about 30 km/s). This is a very small timestep (about 9 ms)! Enter "0" to exit out of `demparams` (a copy of the final screenshot will appear in `demparams.log`).

f. Now it's time to edit the `pkdgrav` run parameters in `ss.par`. Change the following values as indicated:
   i. nDigits = 6
   ii. dDelta = 1.878896e-09
   iii. nSteps = 100000
   iv. iOutInterval = 10000
   v. iRedOutInterval = 1000
   vi. iLogInterval = 100000
   vii. iCheckInterval = 0
   viii. dEpsN = 0.5
   ix. iForceOverrideOption = 2
   x. dKn = 2.616767e-05 [uncomment the lines in this section]
   xi. iDEMStatsInterval = 1000

g. Start the run: "`./pkdgrav ss.par >& output &`" (in `bash`, the equivalent command is "`./pkdgrav ss.par > output 2>&1 &`"). You can monitor the progress by typing `ls`, `tail -f output`, or running "`pkd_status .`". To make later analysis easier, also do "`ln -s initcond.ss ss.000000`".

h. To generate a movie at any time, run `mkmov.py` (this will make `movie.mp4` — view using `ffplay` or *quicktime* or whatever). To generate the DEM quality-control movies, the plotting package `sm` (not free) must be installed; if you have it, copy `pkdsrc/etc/demstats.sm` to the run directory, type `demstatsanim`, and view the resulting `.mp4` files.

i. Feel free to repeat this exercise varying the rubble pile and/or encounter parameters. Be sure to run `demparams` again if you make drastic changes.

3. SSDEM With Walls

Granular dynamics simulations generally need boundary conditions (walls) in order to be contained. You can specify wall parameters by supplying a "walls file" in `ss.par` (achWallsFile). You need to uncomment the line "#USE_WALLS=true" in `Makefile.in` and recompile. The syntax for specifying walls is fully described in `pkdsrc/docs/walls.pdf`. For visualization, include the walls file in `ssdraw.par`, set the particle shape to 2 (*POV-Ray*), and be sure to place the camera and light source

relatively close to the scene. You will also need to copy `pkdsrc/etc/povray.inc` to your run directory. Then run `mkmov.py` to make a ray-traced movie (or use the `pov` script on individual `.pov.gz` frames generated by `ssdraw`).

4. Running in Parallel

The supplied version of `pkdgrav` supports MPI and pthread parallelization (select via the macro "PKDGRAV_TYPE" in `Makefile.in`). For MPI, whatever system version (e.g., MPICH, LAM MPI, OpenMPI) specified by `mpicc` is used. (Note: you can alter the compilation behavior by editing the macros at the top of `Makefile.in` — this is also how you can easily change the non-MPI compiler from `gcc` to `icc`, for example.) To run, use `mpirun` for MPI (with whatever flags your particular flavor requires) or the `pkdgrav` command-line option `-sz [# threads]` for pthreads.

5. Generating Your Own Data

The utilities `ss2bt` and `bt2ss` convert between binary `ss` format and human-readable text format. This allows you to write your own generators and filters for particle data. The format of a `bt` file is one particle per line, with the following data on each line:

```
i1 i2 m R x y z vx vy vz wx wy wz c
```

where `i1` and `i2` are integer indices (which normally just monotonically increase from zero), `m` is the particle mass, `R` is the radius, `x y z` is the position vector, `vx vy vz` is the velocity vector, `wx wy wz` is the spin vector, and `c` is a color index between 0 and 255 (see the end of `ssdraw.par`, for example, to see the color codes). The units are solar masses, au, and yr/2π, as usual. For very large files, it's better to manipulate the `ss` data directly. The `ssio` library in `pkdsrc/src/ss` can be used for this purpose (see the source files `ss2bt.c` and `bt2ss.c` for pertinent usage examples). Recently a utility `ssio.py` was added to `pkdsrc/bin/scripts` that provides I/O functionality with `ss` files for *Python* scripts. See `pkdsrc/docs/ssio_py.pdf` for usage.

IMPORTANT NOTE: By default when using SSDEM, `pkdgrav` assumes particles are ordered by *radius* from smallest to largest. The `rpx` utility has a `-s` option to sort particles in ascending or descending order by radius (use `rpx -h` for a list of options). The initial conditions generator `ssgen` also has reordering options. The prefered ordering is set by the search ball strategy — see `pkdgrav/docs/SSDEM.pdf` for more information. (That document also contains information on restarts and how to analyze the DEM statistics — check it out!)

6. Visualizing Your Data

The `pkdgrav` package includes support for *POV-Ray* and *ParaView* visualization of simulation data (in addition to more basic supported formats).

For *POV-Ray*, the utility `ssdraw` generates `.pov` files based on the given binary `ss` and walls data (edit `ssdraw.par` to set up the scene); then use the utility `pov` to convert the generated `.pov` file to a `.png` file (view using `display` or whatever). You can also add other objects to the scene by editing the `.pov` file by hand (visit http://www.povray.org/ for ideas). For example, the text `Hello world!` can be presented by adding "`text { ttf "timrom.ttf" "Hello world!" 0.1, 0 pigment { Turquoise } scale 0.01 rotate <90,0,0> translate <0.028,0,0.048> }`" in the `.pov` file, where `timrom.ttf` specifies the font (included with *POV-Ray*), the next two floats are the thickness and offset values, color can be specified in `pigment`, `scale` controls the word size, and `rotate` and `translate` are used to adjust the orientation and position of the text in the scene.

For *ParaView*, the utilities `ss2vtk` and `walls2vtp` convert binary `ss` format and walls data to `.vtk` and `.vtp` formats, which can be opened directly in *ParaView* for visualization (see `pkdsrc/docs/Paraview_tutorial.pdf` for details).

7. Please remember the code is not public.

You should not distribute the code to anyone else without checking with one of the authors first. Any publications using the code should have at least one of the code authors as a co-author, and that person must be given a chance to comment on the work thoroughly before publication. This is to maintain the integrity of studies done with `pkdgrav`, since we do not have enough people-power to support a public release at this time.