**How to enable the Potential Block and Potential Particle Feature**

To enable the Potential Block and Potential Particle feature, include the bold lines during installation:

sudo apt-get install cmake git freeglut3-dev libloki-dev \

libboost-all-dev fakeroot dpkg-dev build-essential g++ \

python-dev ipython python-matplotlib libsqlite3-dev python-numpy python-tk gnuplot \

libgts-dev python-pygraphviz libvtk6-dev python-numpy libeigen3-dev \

python-xlib python-pyqt5 pyqt5-dev-tools python-pyqt5.qtwebkit gtk2-engines-pixbuf python-argparse \

libqglviewer-dev python-imaging libjs-jquery python-sphinx python-git python-bibtex \

libxmu-dev libxi-dev libcgal-dev help2man libbz2-dev zlib1g-dev python-minieigen **coinor-clp coinor-libclp-dev coinor-libclp1 coinor-libosi1v5**

And during configuration:

cmake -DCHUNKSIZE=8 **-DENABLE\_POTENTIAL\_BLOCKS=ON**

**-DENABLE\_POTENTIAL\_PARTICLES=ON** -DCMAKE\_INSTALL\_PREFIX=../install ../trunk

**Potential Particles**

The concept of “Potential Particles” was introduced and developed by Houlsby (2009) for 2-D convex non-circular particles. The problem of contact detection between a pair of Potential Particles was cast as a constrained optimization problem, and the equations are solved using the Newton-Raphson method. In Boon et al. (2013), it was extended to 3-D, and more robust solutions making use of convex optimization techniques were proposed. Many of the solvers for non-spherical particles generally cannot cope with discontinuities, ill-conditioned gradients (Jacobians) or curvatures (Hessians), and these obstacles have been overcome in Boon et al. (2013) by re-formulating the problem and solving the equations using conic optimization solvers. Users can either make use of MOSEK (using its academic licence), or using the simplified code written by Boon (2013).

A potential particle is defined as:

|  |  |  |
| --- | --- | --- |
|  | $f=\left(1−k\right)\left(\sum\_{i=1}^{N}\left⟨a\_{i}x+b\_{i}y+c\_{i}z−d\_{i}\right⟩^{2}−r^{2}\right)+k\left(x^{2}+y^{2}+z^{2}−R^{2}\right)$  | (1) |

where (*ai*, *bi, ci*) is the normal vector of the *i*th plane defined with respect to the particle local coordinate system, and *di* is the distance of the plane to the local origin. 〈 〉 are Macaulay brackets, i.e., 〈*x*〉 = *x* for *x* > 0; 〈*x*〉 = 0 for *x* ≤ 0. The planes are assembled such that their normal vectors point outwards. They are summed quadratically and expanded by a distance *r*, which is also related to the radius of the curvature at the corners. Further, a “shadow” spherical particle is added; *R* is the radius of the sphere, with0 *< k* ≤ 1 denoting the fraction of sphericity of the particle.

**Potential Blocks**

“Potential Blocks” was developed in the D.Phil. thesis of Boon (2013) and Boon et al. (2012). It was developed originally for rock engineering applications to model polygonal and polyhedral blocks. The blocks are defined with linear inequalities only.



Fig. 1 A 2-D polygon defined using a set of five inequalities

For a convex particle defined by *N* planes, the space that it occupies can be defined using the following inequalities (see Fig. 1):

|  |  |  |
| --- | --- | --- |
|  | $a\_{i}x+b\_{i}y+c\_{i}z\leq d\_{i},$ $i=1,...,N$ | (2) |

where (*ai*, *bi*, *ci*) is the unit normal vector of the *i*th plane defined with respect to the particle local coordinate system, and *di* is the distance of the plane to the local origin.

 The problem of establishing intersection between a pair of blocks is cast as a standard linear programming problem of finding a feasible region satisfying all the linear inequalities defining both blocks. The contact point is calculated as the analytic centre, which is a well-known concept in interior-point methods in convex optimisation calculations. The contact normal is obtained from the gradient of a smooth “potential particle” defined inside the block. The overlap distance is calculated through bi-section searching along the contact normal within the overlap region.

 The same data structure is used for block generation in Boon et al. (2015). The sequential subdivision concept is used together with a linear programming framework. Non-persistent joints are modelled by introducing more constraints.

 The linear programming solver for Potential Blocks was originally CPLEX, but has been updated to COIN\_OR’s CLP, because the libraries can be downloaded from Ubuntu or Debian’s distributions.

**Examples**

Some examples are being included, and the display is being output as vtk files, which can be opened through Paraview.

**Disclaimer**

The code was developed for academic purposes. Some variables are no longer in use, as the PhD thesis spanned over many years with numerous trials and errors. As this piece of code has many dependencies within the YADE ecosystem, user discretion is advised.

**References**

To acknowledge our scientific contribution, please cite the following:

Boon CW (2013) Distinct Element Modelling of Jointed Rock Masses: Algorithms and Their Verification. D.Phil. Thesis, University of Oxford

Boon CW, Houlsby GT, Utili S (2012) A new algorithm for contact detection between convex polygonal and polyhedral particles in the discrete element method. Computers and Geotechnics, 44: 73-82

Boon CW, Houlsby GT, Utili S (2013) A new contact detection algorithm for three dimensional non-spherical particles. Powder Technology, S.I. on DEM, 248: 94-102

Boon CW, Houlsby GT, Utili S (2015) A new rock slicing method based on linear programming. Computers and Geotechnics, 65:12-29