TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
The University of Resources. Since 1765.

imfd

A. Gladkyy, S. Roy, T. Weinhart, S. Luding, R. Schwarze
TU Bergakademie Freiberg
Lampadiusstrasse 4
09599 Freiberg
Germany

DEM simulations of weakly wetted granular materials:

implementation of capillary bridge models

September 28-30, 2015, Barcelona, Spain

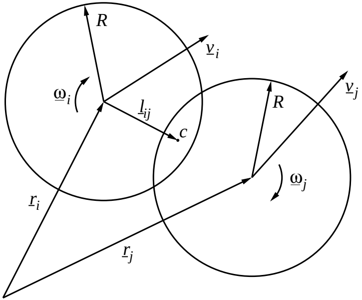
IV Conference on Particle-Based Methods, Particles 2015

TU Bergakademie Freiberg

- Situated in the city of Freiberg, Saxony, Germany
- 6 faculties, 5600 students, 86 professors
- 2 collaborative research centers
- Institute of Mechanics and Fluid Dynamics:
  - Applied Mechanics: Solid Mechanics, Dynamics
  - Fluid Mechanics, Turbomachinery
  - **Numerical modelling of granular flow processes: CFD, DEM, SPH**

Source: `https://commons.wikimedia.org/wiki/File:Weisbachbau_Freiberg_TU_Bergakademie.jpg`

Discrete element model with capillary bridges



$$m_P \frac{d\vec{v}}{dt} = \sum_c \vec{f}_c + m_P \vec{g}$$

$$I_P \frac{d\vec{\omega}}{dt} = \sum_c \left(\vec{l}_c \times \vec{f}_c\right)$$

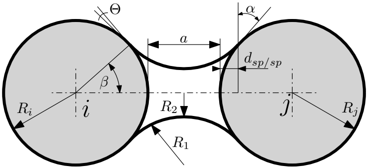
$$\vec{f}_n = \left(-k_n \delta_n - \gamma_n \frac{d\delta_n}{dt} + f_{cap}\right) \hat{n}$$

$$\vec{f}_t = \min\left[-k_t \delta_t - \gamma_t \frac{d\delta_t}{dt},\, \mu f_n\right] \hat{t}$$

$$S^+ = \frac{a}{2\sqrt{V/R}}$$

$$f_{cap} = \frac{2\,\pi R \gamma \cos\theta}{1 + 2.1(S^+) + 10(S^+)^2}$$

Willett formulation
(see Gladkyy, Schwarze 2014 for details)



Split-bottom shear cell configuration

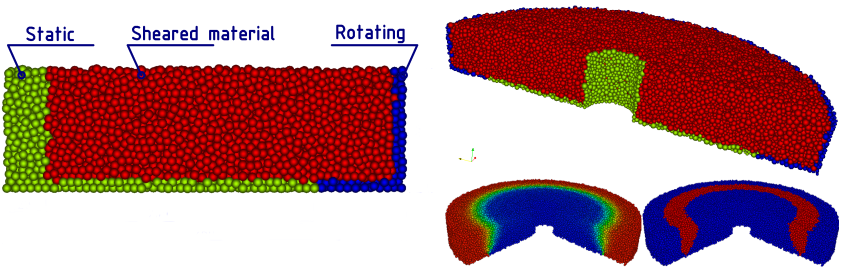


Fig. 1: Setup of split-bottom configuration (see Gladkyy, Schwarze 2014 for details)

DEM parameters

- $\overline{D}_p = 2.2\,\mathrm{mm}$; $\rho = 2000\,\mathrm{kg\,m^{-3}}$; $k_n = 110\,\mathrm{N\,m^{-1}}$; $k_t = 12\,\mathrm{N\,m^{-1}}$;
- $\gamma_n = 0.002\,\mathrm{kg\,s^{-1}}$; $\gamma_t = 0.0005\,\mathrm{kg\,s^{-1}}$; $\theta = 20°$; $\gamma = 20.6\,\mathrm{mN\,m^{-1}}$;
- Particle number ≈ 145000; Rotation 0.6 RPM; $V_b = 0$ nl to 200 nl;

DEM open-source software

MercuryDPM, YADE, LIGGGHTS

- Computational parts are written in C++, running on Linux/Unix
- Parallelization (MPI, OpenMP)
- Free GPL (v. 2 and v. 3) licenses
- Data export into different formats, including VTK
- Reach functions for work with complex geometries
- Particle import/export and generation opportunities
- Available in Debian/Ubuntu repositories (YADE, LIGGGHTS)
- CMake build system

MERCURYDPM

LIGGGHTS

DEM open-source software

MercuryDPM: http://mercurydpm.org

- C++ for the simulation setup and scenario
- Contributors: University of Twente (The Netherlands), community

Yet Another Dynamic Engine (YADE): http://yade-dem.org

- Python bindings for simulation setup, control and postprocessing
- OpenMP; GUI for simulation setup and control
- Contributors: community (Lab 3SR - Grenoble, CTU Prague, TU Freiberg ...)

LIGGGHTS (LAMMPS improved for general granular and granular heat transfer simulations): http://liggghts.com

- Own scripting language for simulation setup
- MPI; coupling with CFD-code, SPH-model, heat-transfer
- Contributors: DCS Computing, community (TU Graz, JKU Linz ...)

Implementation of capillary bridge models

MercuryDPM

```
void LiquidBridgeWilletInteraction::computeAdhesionForce()
{
    const LiquidBridgeWilletSpecies* species = getSpecies();
    if (getOverlap()>=0)
    {
        wasInContact_=true;
    Mdouble effectiveRadius = getEffectiveRadius();
    Mdouble fdotn = -2.0*constants::pi*effectiveRadius*species->getSurfaceTension()*std::cos(species->getContactAngle());
        addForce(getNormal() * fdotn);
    }
    else if (wasInContact_)
    {
    Mdouble effectiveRadius = getEffectiveRadius();
    Mdouble s_c = -getOverlap()*std::sqrt(effectiveRadius/species->getLiquidBridgeVolume());
    Mdouble fdotn = -2.0*constants::pi*effectiveRadius*species->getSurfaceTension()
     *std::cos(species->getContactAngle())/(1+(1.05+2.5*s_c)*s_c);
        addForce(getNormal() * fdotn);
    }
}
```

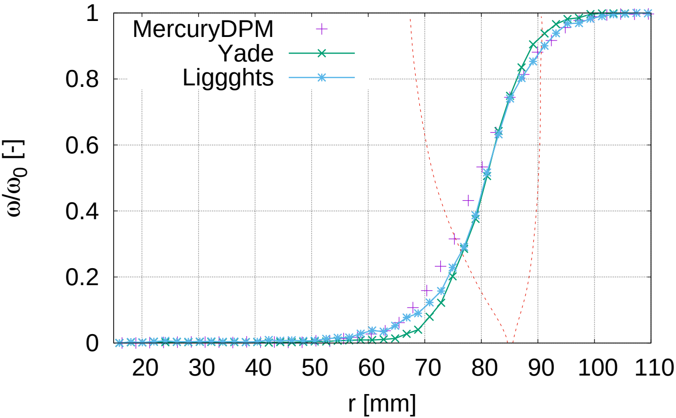
YADE

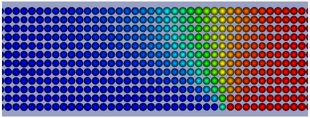
```
Real Law2_ScGeom_ViscElCapPhys_Basic::Willett_analytic_f(const ScGeom& geom, ViscElCapPhys& phys) {
  /*
   * Capillar model from Willet [Willett2000] (analytical solution)
   */

  const Real R = phys.R;
  const Real Gamma = phys.gamma;
  const Real s = -geom.penetrationDepth;
  const Real Vb = phys.Vb;

  const Real sPl = (s/2.0)/sqrt(Vb/R);                                // [Willett2000],
  const Real f_star = cos(phys.theta)/(1 + 2.1*sPl + 10.0 * pow(sPl, 2.0));   // [Willett2000],
  const Real fC = f_star * (2*M_PI*R*Gamma);                          // [Willett2

  return fC;
}
```

LIGGGHTS

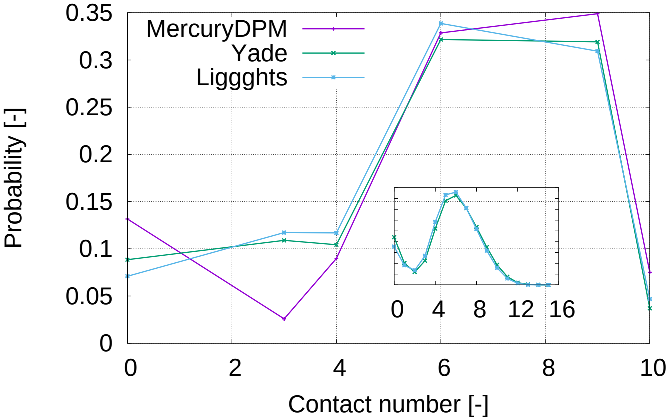
```
  static double Willett_analytic_f(const double & R, const double & Vb, const double & Gamma, const double & Theta, const double & s) {
    /*
    * Capillar model from Willet [Willett2000] (analytical solution), but
    * used also in the work of Herminghaus [Herminghaus2005]
    */

    const double sPl = (s/2.0)/sqrt(Vb/R); // [Willett2000], equation (sentence after (11)), s - half-separation, so s*2.0
    const double f_star = cos(Theta)/(1 + 2.1*sPl + 10.0 * pow(sPl, 2.0)); // [Willett2000], equation (12)
    const double fC = f_star * (2*M_PI*R*Gamma); // [Willett2000], equation (13), against F
    return fC;
  }
```

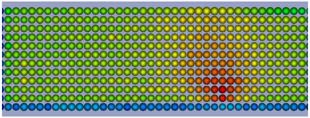
Velocity profile compare



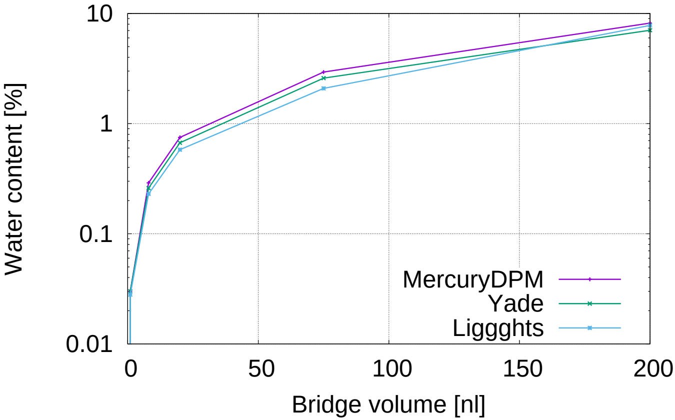


Contact number probability





Water content



Conclusions

Good practice in writing open-source programs

- Before starting a new program, try to join existing project.
- Do not invent your own license (DFSG).
- Only source files in the tarball, build everything from source.
- Avoid inclusion of 3rd-party codes.
- Document dependencies.
- Use proper versioning and prepare releases.
- Have an automatic test suit.
- Use build system: autoconf and automake, cmake.

https://wiki.debian.org/UpstreamGuide

Thank you for your attention!