YADE - An extensible framework for the interactive simulation of multiscale, multiphase, and multiphysics particulate systems

Vasileios Angelidakis

Lecturer in Geotechnics, Queen's University Belfast, UK

v.angelidakis@qub.ac.uk

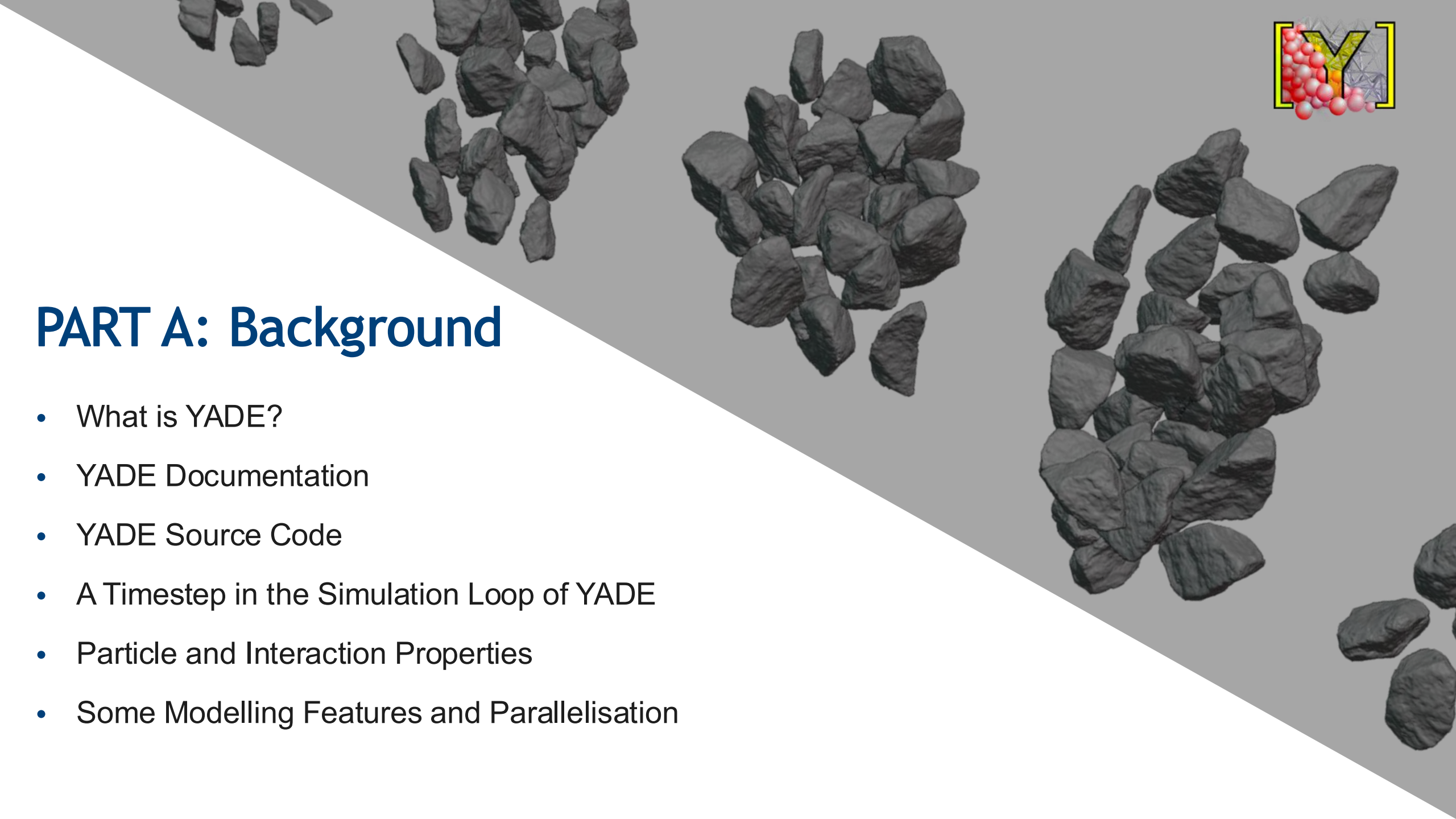
QUEEN'S UNIVERSITY BELFAST

CCC-ParaSolS Training Event

Edinburgh, May 2025

ParaSolS

Particulate Solids Simulations



PART A: Background

- What is YADE?
- YADE Documentation
- YADE Source Code
- A Timestep in the Simulation Loop of YADE
- Particle and Interaction Properties
- Some Modelling Features and Parallelisation

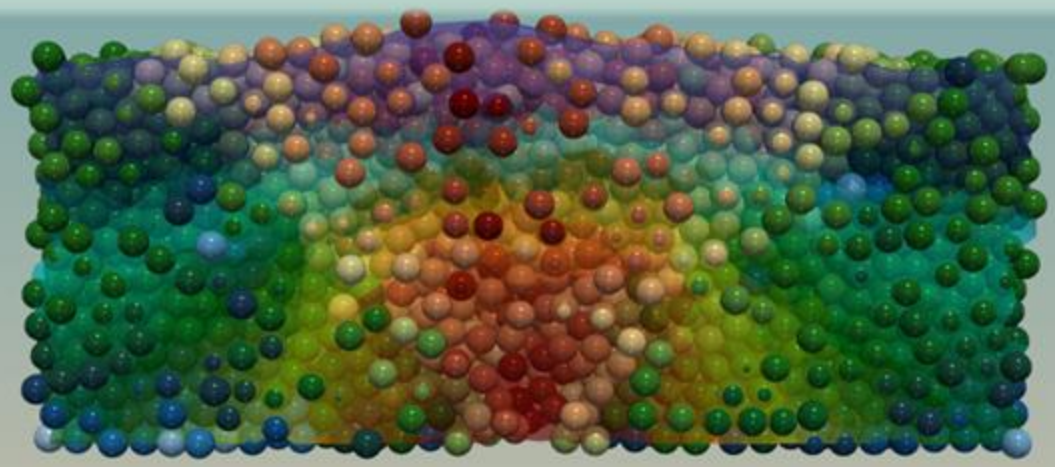
Yet Another Dynamic Engine (YADE)

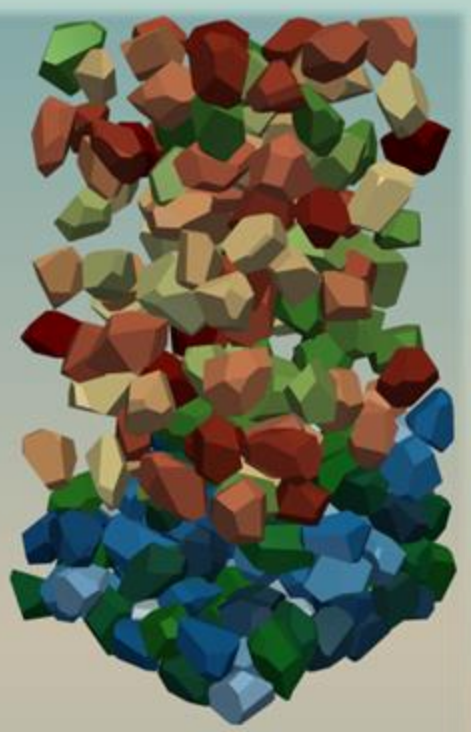
YADE is an **open-source** framework employing the Discrete Element Method (DEM), distributed under the **GPLv2.0** license.

Originating from 3SR-LAB in Université Grenoble Alpes some 20 years ago, YADE has evolved from “Yet Another Dynamic Engine” to an **international collaborative project** and to a **versatile multiscale and multiphysics solver**, counting a large, active, and growing community of users and developers.

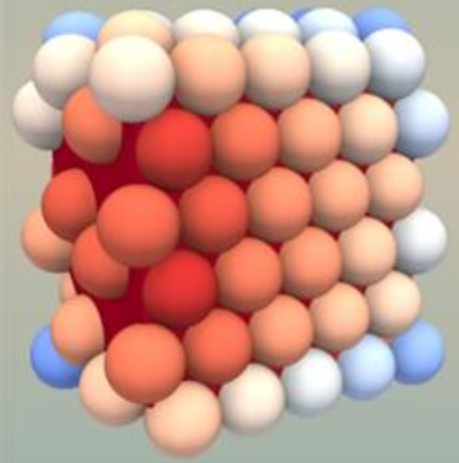
The computationally intense parts of the source code are written in **C++**, using **flexible object models** that allow for easy implementation of new features. **The source code is wrapped in Python**, equipping the software with an **interactive kernel** used for rapid and concise scene construction.

https://yade-dem.org



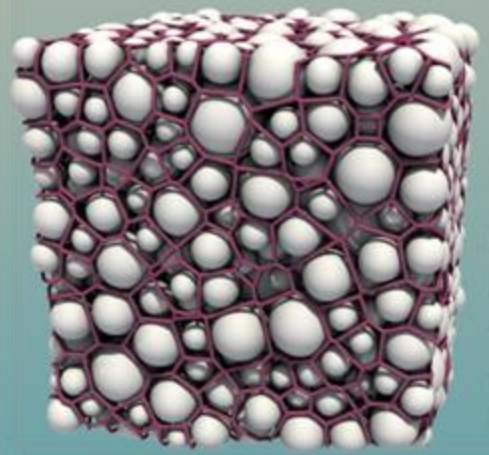


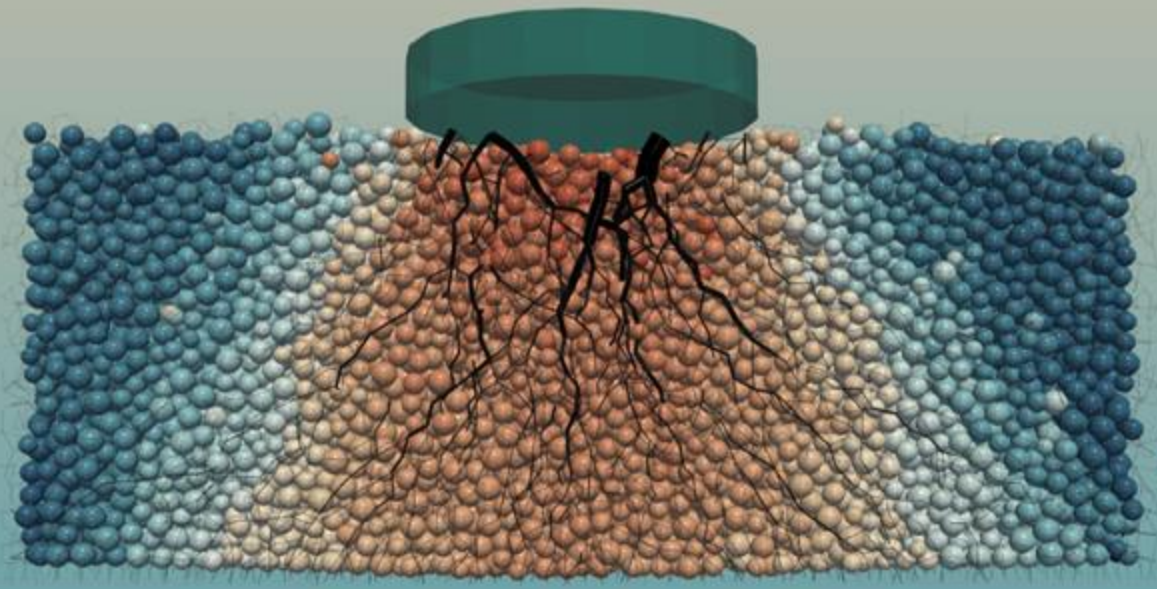
(Angelidakis et al, CPC, 2024)



YADE

yade-dem.org



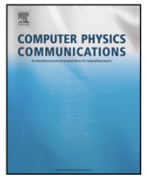


Computer Physics Communications 304 (2024) 109293

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc



Computer Programs in Physics

YADE - An extensible framework for the interactive simulation of multiscale, multiphase, and multiphysics particulate systems ☆,☆☆

Vasileios Angelidakis [a,b,*], Katia Boschi [c], Karol Brzeziński [d], Robert A. Caulk [e], Bruno Chareyre [e,*], Carlos Andrés del Valle [f], Jérôme Duriez [g], Anton Gladky [h], Dingeman L.H. van der Haven [i,j], Janek Kozicki [k,l], Gerald Pekmezi [m], Luc Scholtès [n], Klaus Thoeni [o]

[a] School of Natural and Built Environment, Queen's University Belfast, BT9 5AG Belfast, United Kingdom
[b] School of Engineering, Newcastle University, NE1 7RU, Newcastle upon Tyne, United Kingdom
[c] Department of Civil and Environmental Engineering, Politecnico di Milano, 20133 Milan, Italy
[d] Faculty of Civil Engineering, Warsaw University of Technology, 00 637 Warsaw, Poland
[e] Univ. Grenoble Alpes, CNRS, Grenoble INP, 3SR, 38000 Grenoble, France
[f] Departamento de Física, Universidad Nacional de Colombia, Carrera 45 No. 26-85, Edificio Uriel Gutiérrez, Bogotá D.C., Colombia
[g] INRAE, Aix Marseille Univ, RECOVER, Aix-en-Provence, France
[h] Independent researcher, 09618 Brand-Erbisdorf, Germany
[i] Department of Materials Science & Metallurgy, University of Cambridge, CB3 0FS Cambridge, United Kingdom
[j] Oral Drug Product Process Development, Novo Nordisk A/S, 2760 Måløv, Denmark
[k] Faculty of Applied Physics and Mathematics, Gdańsk University of Technology, 80-233 Gdańsk, Poland
[l] Advanced Materials Center, Gdańsk University of Technology, 80-233 Gdańsk, Poland
[m] Department of Mechanical Engineering, The University of Alabama at Birmingham, Birmingham, AL, USA
[n] Université Clermont Auvergne, CNRS, IRD, OPGC, Laboratoire Magmas et Volcans, Clermont-Ferrand, France
[o] Centre for Geotechnical Science and Engineering, The University of Newcastle, 2308 Callaghan, Australia

ARTICLE INFO

Dataset link: https://gitlab.com/yade-dev/trunk

Keywords:
Discrete element method (DEM)
Open-source software
Granular materials
Non-spherical particles
Coupled methods
Parallel computing

ABSTRACT

This contribution presents the key elements of YADE, an extensible open-source framework for dynamic simulations. During the past 19 years, YADE has evolved from "Yet Another Dynamic Engine" to a versatile multiscale and multiphysics solver, counting a large, active, and growing community of users and developers. The computationally intense parts of the source code are written in C++, using flexible object models that allow for easy implementation of new features. The source code is wrapped in Python, equipping the software with an interactive kernel used for rapid and concise scene construction, simulation control, post-processing, and debugging. The project, including documentation and examples, is hosted on https://yade-dem.org, while the source code is freely available on GitLab. Over the last decade, YADE has expanded in terms of capabilities thanks to the contribution of many developers from different fields of expertise, including soil and rock mechanics, chemical engineering, physics, bulk material handling, and mineral processing. The rapid growth of YADE can be attributed to (1) the careful and robust design of the framework core, (2) a continuous integration pipeline with fully embedded thorough tests which are executed upon each merge request, ensuring stable compilation for various operating systems, and (3) user-friendliness, facilitated by the Python interface, detailed documentation, and rigorous user support. In this paper, we review the main features of YADE, highlighting its versatility in terms of applications, its flexibility in terms of code development, as well as recent improvements in terms of computational efficiency.

(Angelidakis et al, CPC, 2024)

YADE: State-of-the-art

This recent paper in CPC provides:

- a single-stop point for information on Yade,
- reporting on current features,
- linkage to past work of developed features,
- analysis of the code structure, and
- reporting on performance & parallelisation.

Yade documentation (https://yade-dem.org)

Quick search

Go

Welcome to Yade - Open Source Discrete Element Method

The Yade project is a strong community based entirely on peaceful international cooperations. This diverse foundation has bound us to many brilliant scien-tists from Ukraine and Russia who are unjustly affected by the current hostilities perpetrated by Russia on Ukraine. We, the Yade developers and users, hope for the immediate cessation of agression followed by the prompt restoration of peace in Ukraine. We stand with the people of Ukraine, the citizens of Russia and the thousands of Russian scientists who vehemently protest this war. As these scientists point out, this unlawful invasion puts many innocent lives at risk in both Ukraine and Russia. Without a doubt, we will continue to keep these Ukranians and Russians in our thoughts as we await a peaceful conclusion ahead.

Yade is an extensible open-source framework for discrete numerical models, focused on the Discrete Element Method. The computation parts are written in c++ using a flexible object model and allowing independent implementation of new algorithms and interfaces. Python is used for rapid and concise scene construction, simulation con-trol, postprocessing and debugging.

Yade is located at yade-dem.org, which contains this documentation and wiki. Development is kindly hosted on launchpad and GitLab ; they are used for source code, bug tracking and source downloads and more. Building, regression tests and packages distribution are hosted on servers of the Grenoble Geomechanics group at Laboratoire 3SR, UMS Gricad and Gdańsk University of Technology.

Yade supports high precision calculations (following this publication) and Python 3 (see backward compatibility for Python 2). The development branch is on GitLab.

This documentation describes Yade version 2024-11-11.git-fcdd8ae, see changelogs. You can also download a PDF version of this documentation.

Acknowledging Yade

Please make sure you read the "Acknowledging Yade" section if you plan to use Yade for publications.

Documentation

(Šmilauer et al, Yade documentation 3rd ed, 2021)

Introduction

getting familiar with Yade

Tutorial

first steps with practical examples

DEM Formulation

the Discrete Element Method (DEM)

Class reference

simulation building blocks, c++ & python

Yade documentation (https://yade-dem.org)

Yade | Documentation » Overview

Quick search

Go

Welcome to Yade - Open Source Discrete Ele

The Yade project is a strong community based entirely on peaceful international c tists from Ukraine and Russia who are unjustly affected by the current hostilities p for the immediate cessation of agression followed by the prompt restoration of pe and the thousands of Russian scientists who vehemently protest this war. As the risk in both Ukraine and Russia. Without a doubt, we will continue to keep these U ahead.

Yade is an extensible open-source framework for discrete numerical models, focused or flexible object model and allowing independent implementation of new algorithms and int trol, postprocessing and debugging.

Yade is located at yade-dem.org, which contains this documentation and wiki. Developm bug tracking and source downloads and more. Building, regression tests and packages d Laboratoire 3SR, UMS Gricad and Gdańsk University of Technology.

Yade supports high precision calculations (following this publication) and Python 3 (see b

This documentation describes Yade version 2024-11-11.git-fcdd8ae, see changelogs. Yo

Acknowledging Yade

Please make sure you read the "Acknowledging Yade" section if you plan to use Yad

Documentation

Introduction

getting familiar with Yade

Tutorial

first steps with practical examples

DEM

the Disc

Class

simulatio

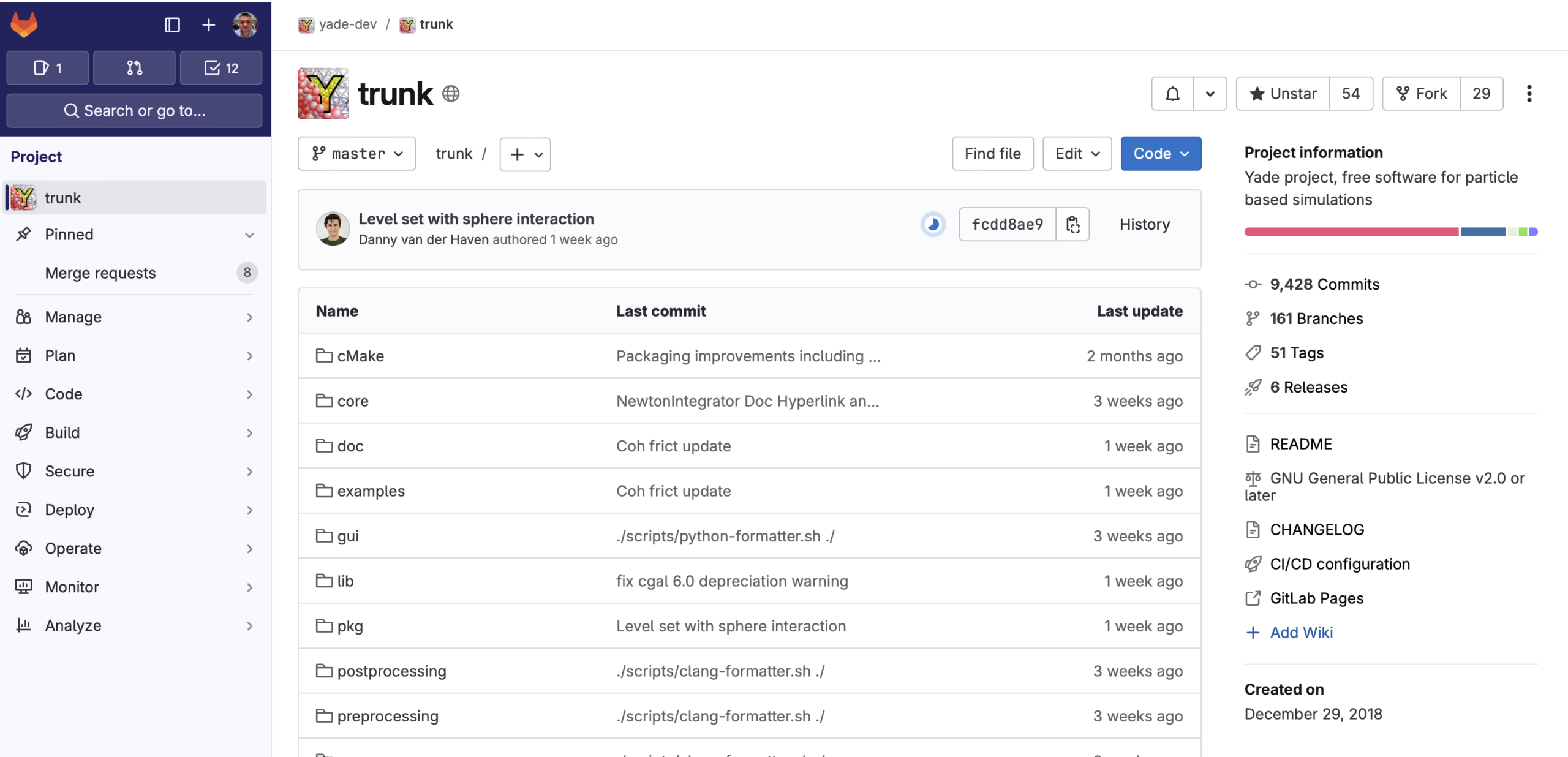
Yade Documentation (3rd ed.)

The documentation is built in html, epub, and pdf formats.

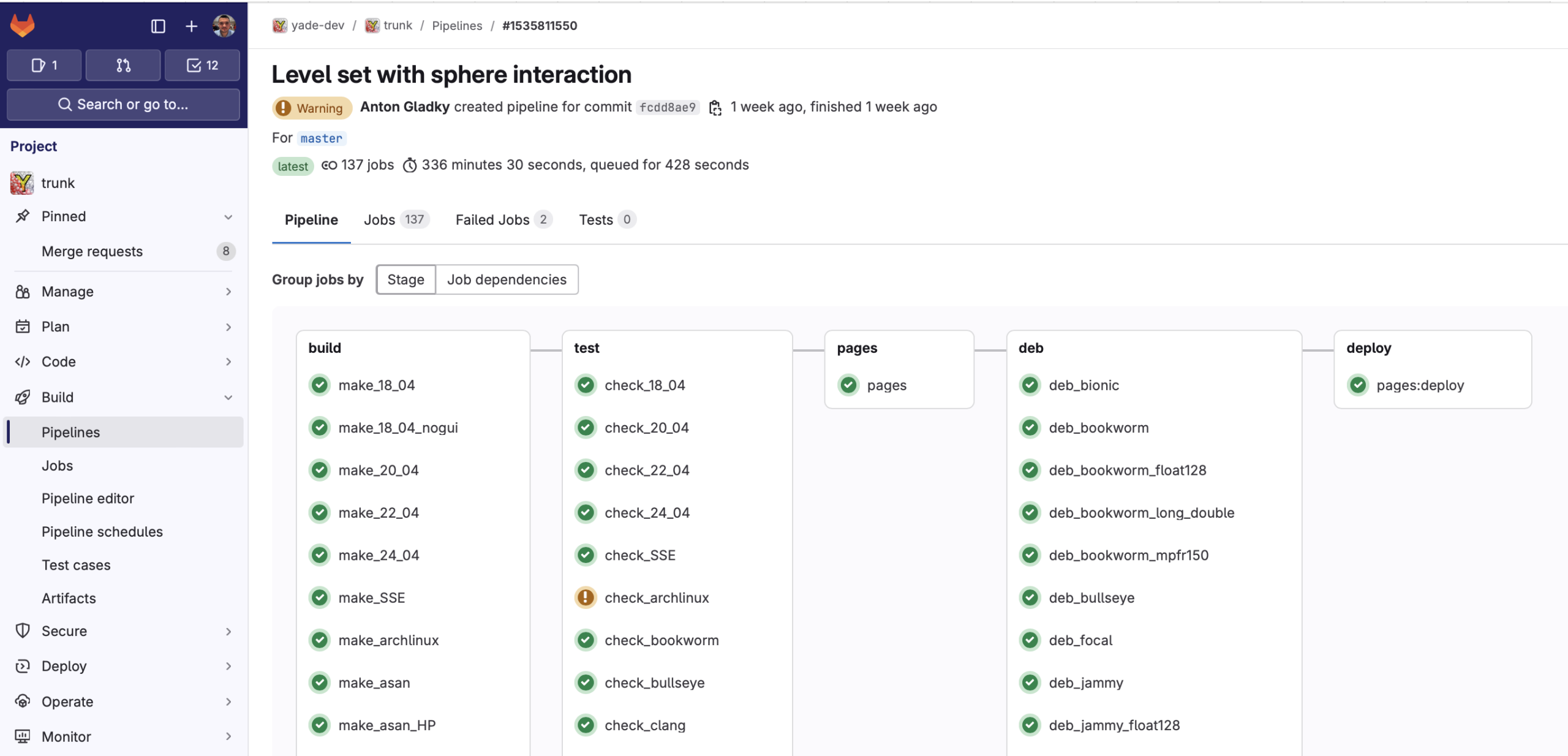
(Šmilauer et al, Yade documentation 3rd ed, 2021)

3rd Edition based on Yade version 2024-11-11.git-fcdd8ae, November 14, 2024

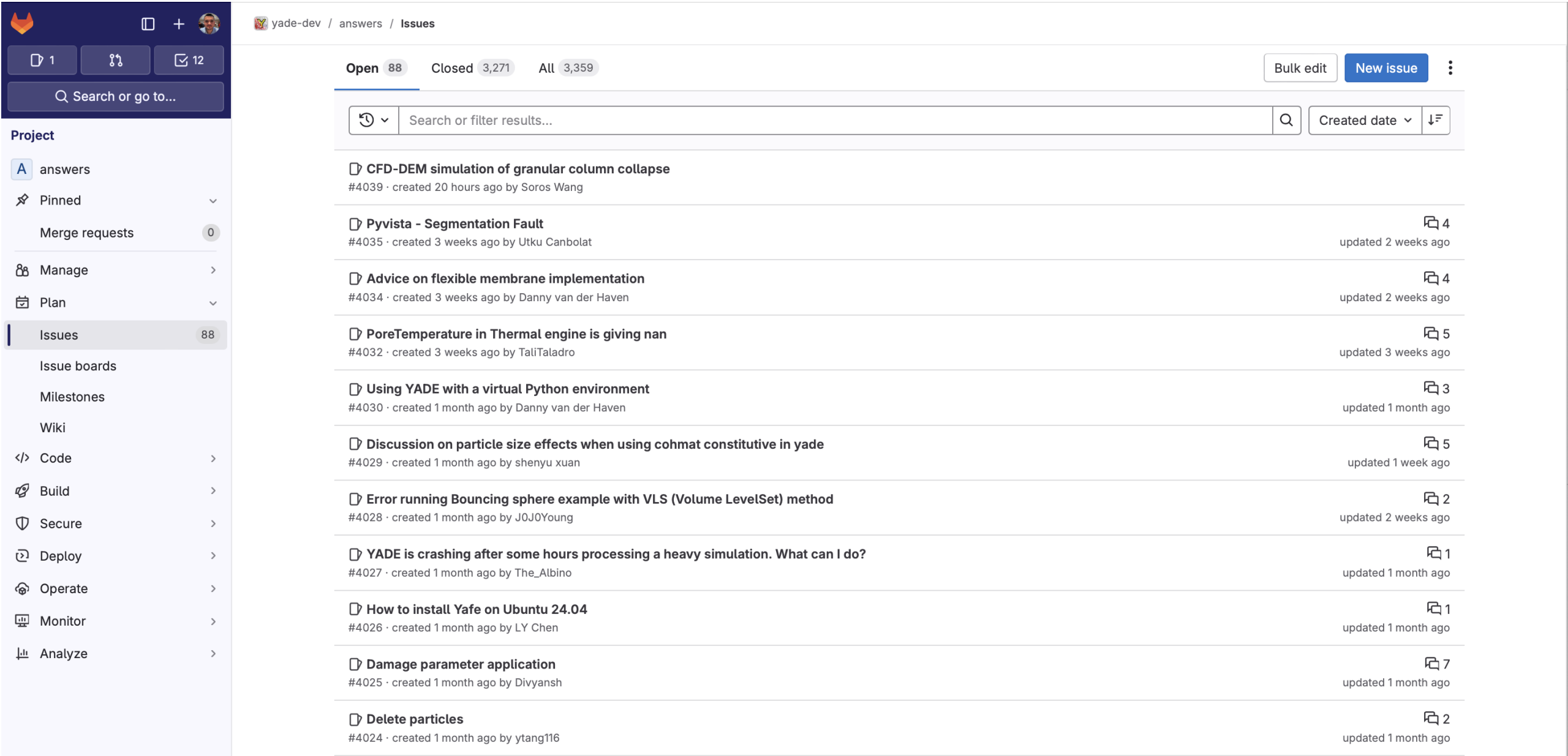
Hosting on Gitlab - Source code (gitlab.com/yade-dev)



Hosting on Gitlab → Continuous Integration (CI)



Hosting on Gitlab → Active user forum for Q&A



The Discrete Element Method (DEM)

The DEM is a numerical simulation technique to model systems made of particles.

Proposed by Cundall and Strack (1979), the DEM is the leading modelling approach for granular materials and other many-body systems, such as soils, powders, and grains.

Solving Newton's equations of motion for a system of particles, one can get:

$$\frac{dv_i}{dt} = \frac{\sum F_{ci}}{m_i} + g_i$$ **linear acceleration = force/mass**

$$\frac{d\omega_i}{dt} = \frac{\sum F_{ti} R}{I}$$ **angular acceleration = torque/inertia**

(Cundall and Strack, Géotechnique 1979)

CUNDALL, P. A. & STRACK, O. D. L. (1979). *Géotechnique* **29**, No. 1, 47–65

A discrete numerical model for granular assemblies

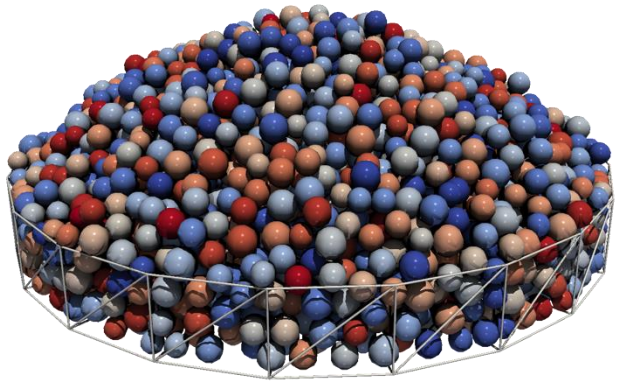
P. A. CUNDALL* and O. D. L. STRACK†

The distinct element method is a numerical model capable of describing the mechanical behaviour of assemblies of discs and spheres. The method is based on the use of an explicit numerical scheme in which the interaction of the particles is monitored contact by contact and the motion of the particles modelled particle by particle. The main features of the distinct element method are described. The method is validated by comparing force vector plots obtained from the computer program BALL with the corresponding plots obtained from a photoelastic analysis. The photoelastic analysis used for the comparison is the one applied to an assembly of discs by De Josselin de Jong and Verruijt (1969). The force vector diagrams obtained numerically closely resemble those obtained photoelastically. It is concluded from this comparison that the distinct element method and the program BALL are valid tools for research into the behaviour of granular assemblies.

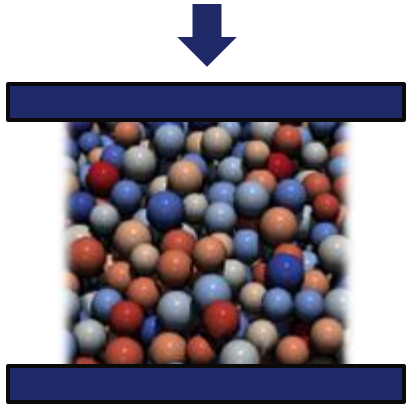
La méthode des éléments distincts est un modèle numérique capable de décrire le comportement mécanique de l'assemblage de disques et de sphères. La méthode est basée sur l'utilisation d'un système numérique explicite dans lequel l'interaction des particules est contrôlée contact par contact et le mouvement des particules simulé particule par particule. Les caracteristiques principales de la méthode des eléments distints sont décrites. La méthode est validée en comparant les tracés de vecteur de force obtenus par le programme sur ordicateur BALL avec les tracés correspondants obtanus à l'aide d'une analyse photo-élastique. L'analyse photo-élastique utilisée pour la comparaison est celle appliquée sur un assemblage de disques par De Josselin de Jong et Verruijt (1969). Les diagrammes de vecteur de force obtenus numériquement sont très voisins de ceux obtenus photo-élastiquement. Cette comparaison permet de conclure que la methode des éléments distincts et le programme BALL sont des instruments valables pour la recherche du comportement des assemblages granulaires.

DEM for material characterisation

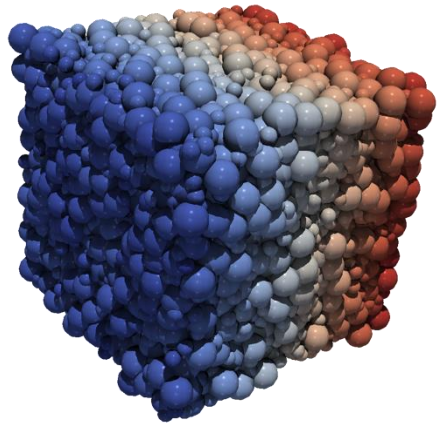
- Many of us use the DEM to characterise granular materials at the bulk scale, e.g. in terms of angle of repose, compression and shear resistance, contact fabric and contact network properties.



Angle of Repose (AOR)



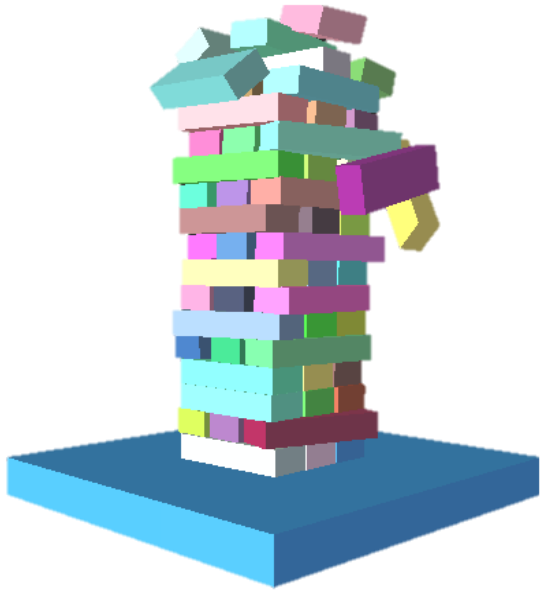
Uniaxial compression: (un-/re-)loading stiffness



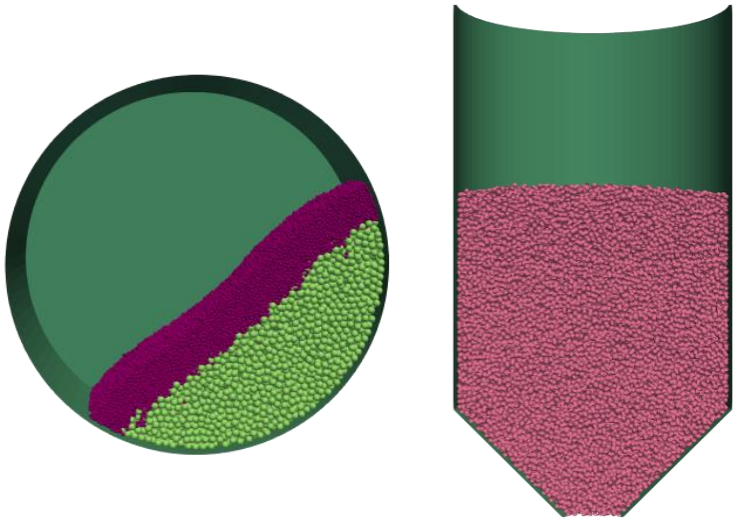
Triaxial compression: shearing resistance

DEM for Boundary Value Problems (just some..)

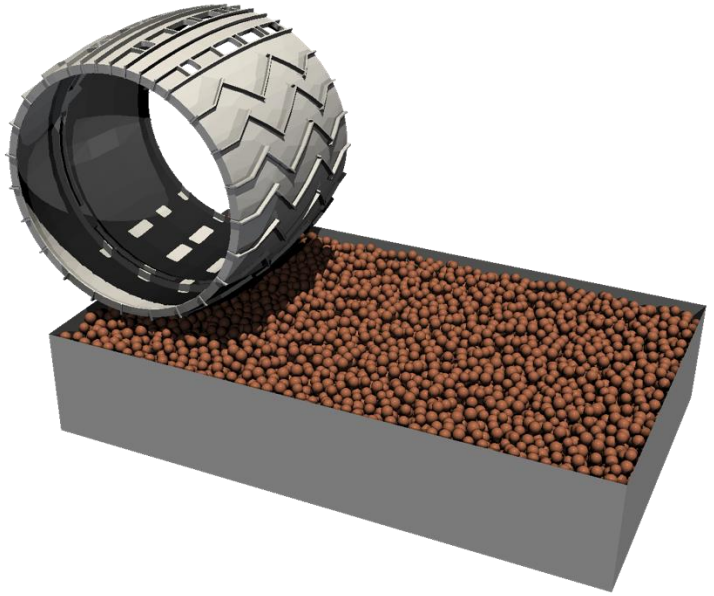
- The DEM is also used to model various processes, e.g.



Stability of tower made of polyhedra



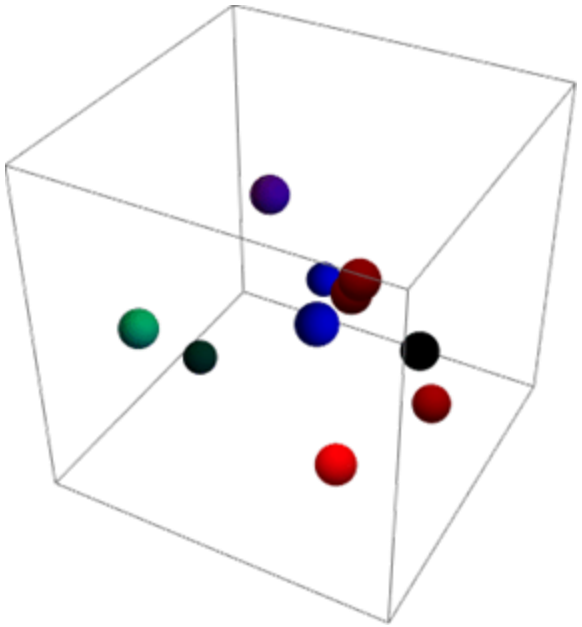
Granular flows in drums and silos

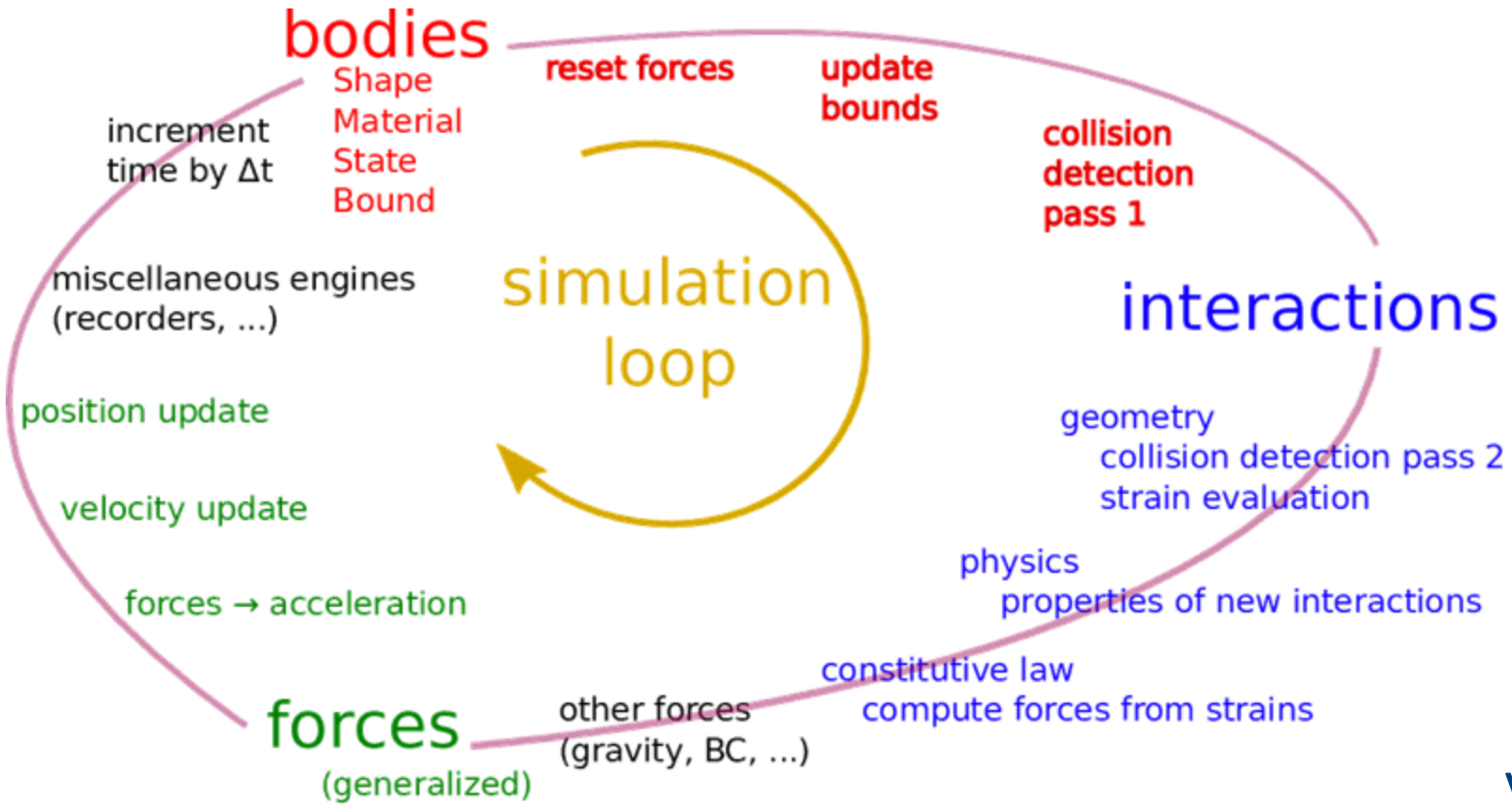


Soil-structure interaction, e.g. locomotion on soil

A Timestep in the Simulation Loop of YADE

Simulation loop: Individual particles move based on Newton's and Euler's laws of motion. When particles meet, they develop contacts which alter their trajectories. **The schematic shows what happens during one timestep of the DEM in YADE.**



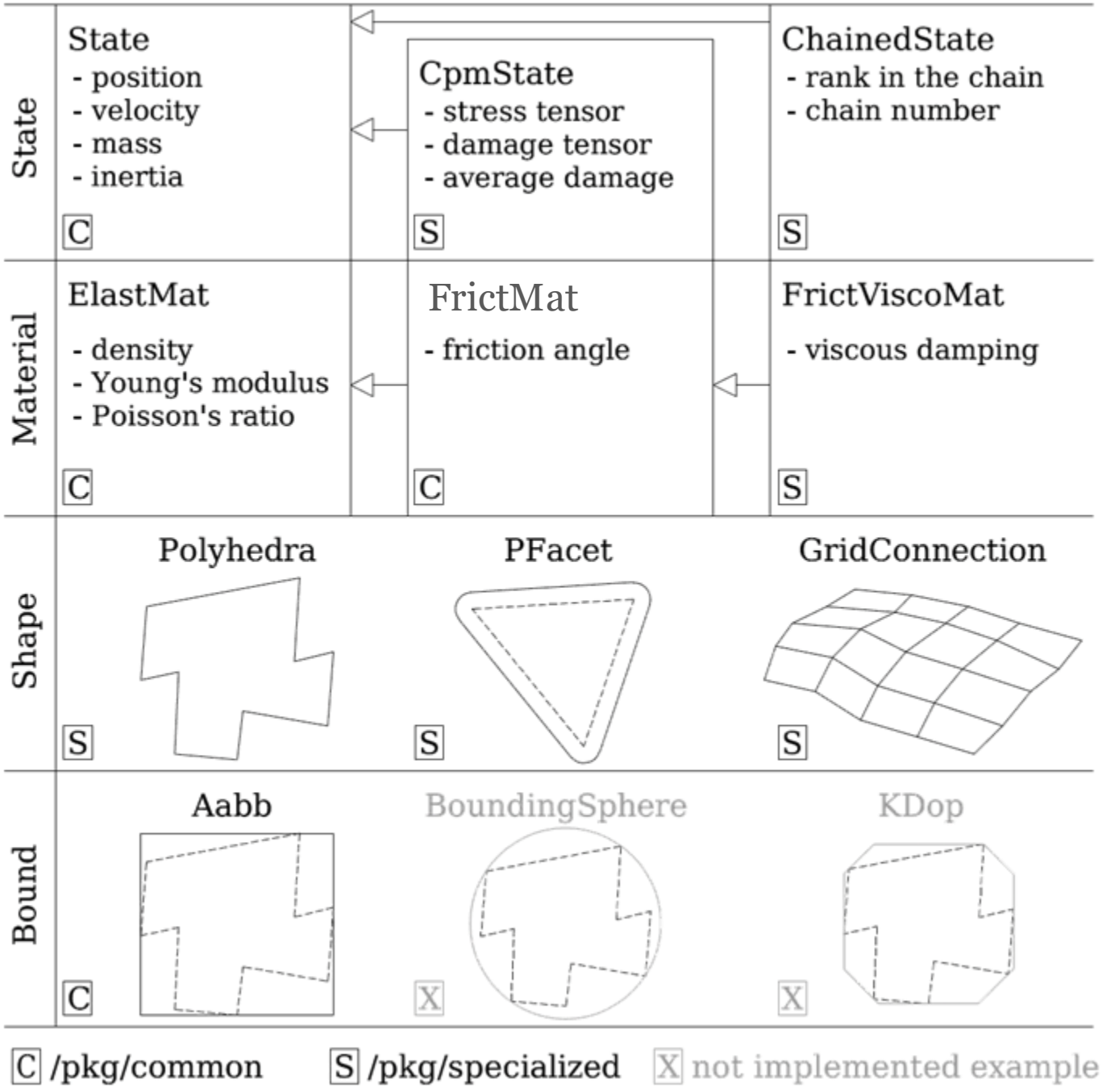


(Šmilauer et al, Yade documentation 3rd ed, 2021)

Body properties

Each particle (body) has a:

- **state**,
- **material**,
- **shape**, and
- **bound**.

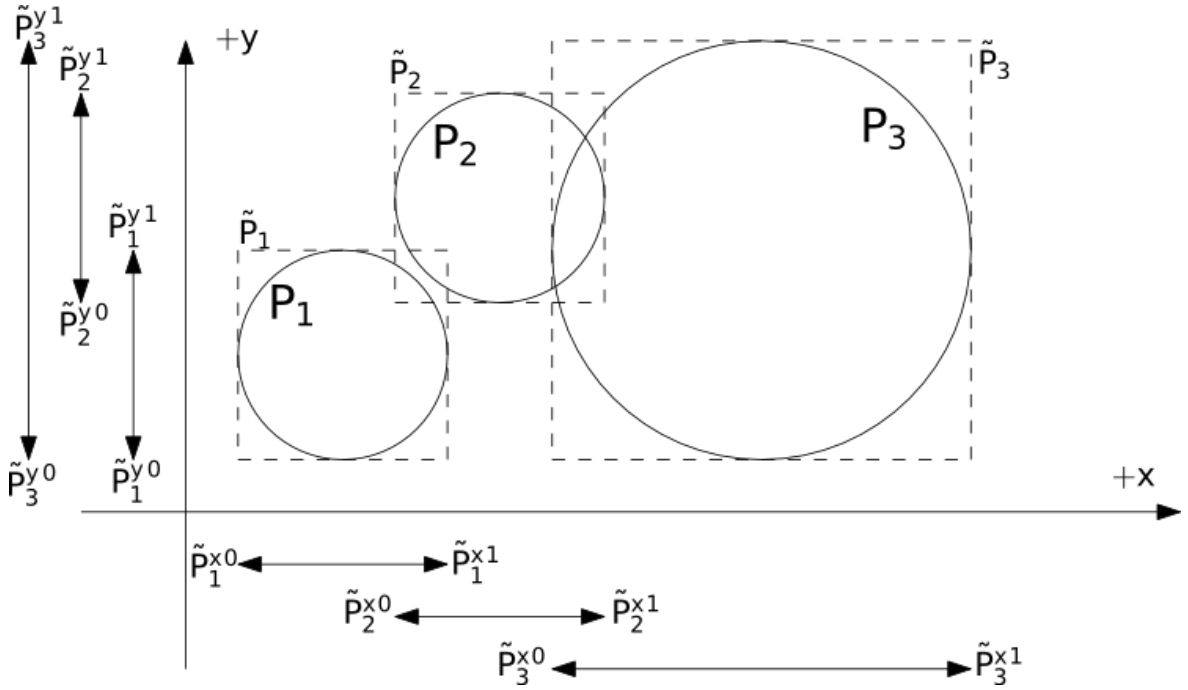


(Šmilauer et al, Yade documentation 3rd ed, 2021)

Collision detection in the DEM

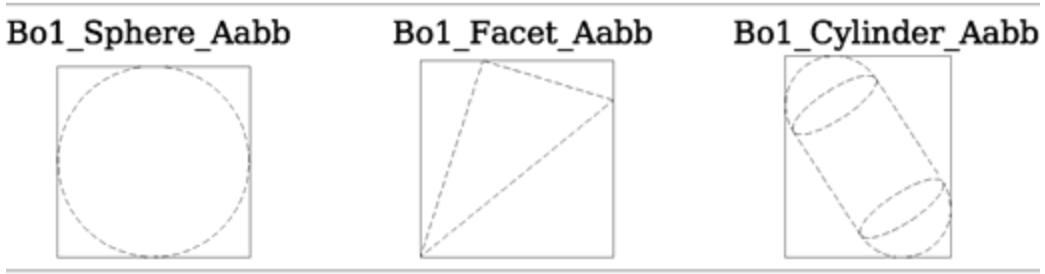
Two types of collision detection:

1. **Approximate**, using Axis-aligned Bounding Boxes (AABBs) → **Bo1 functors**
2. **Exact**, using geometric contact detection algorithms.

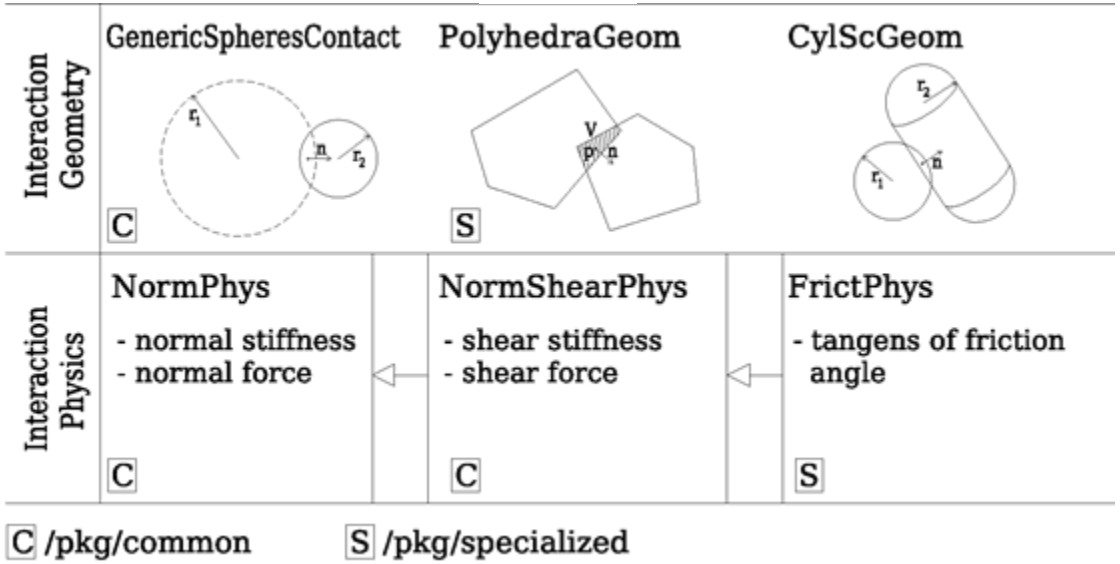


(Šmilauer et al, Yade documentation 3rd ed, 2021)

Approximate



Exact

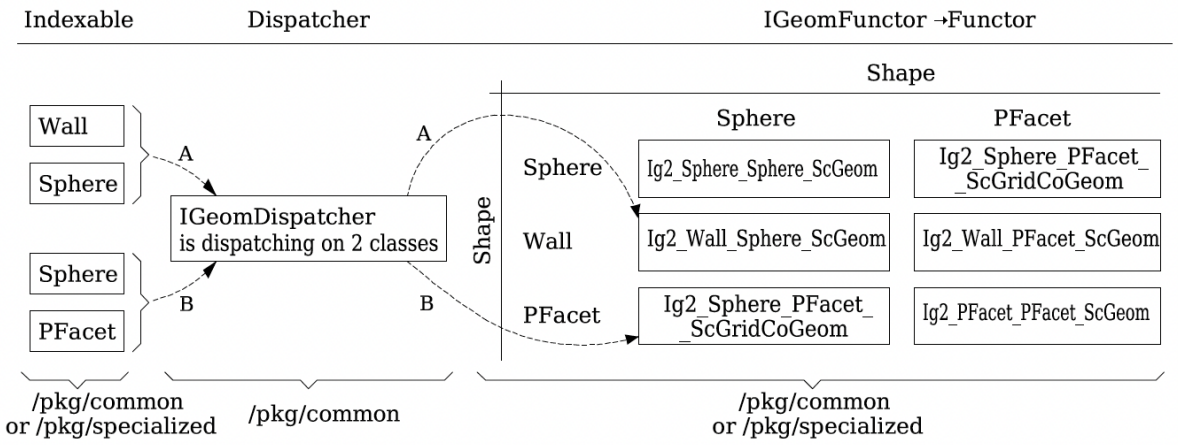


Interaction Geometry (IGeom) + Interaction Physics (IPhys)

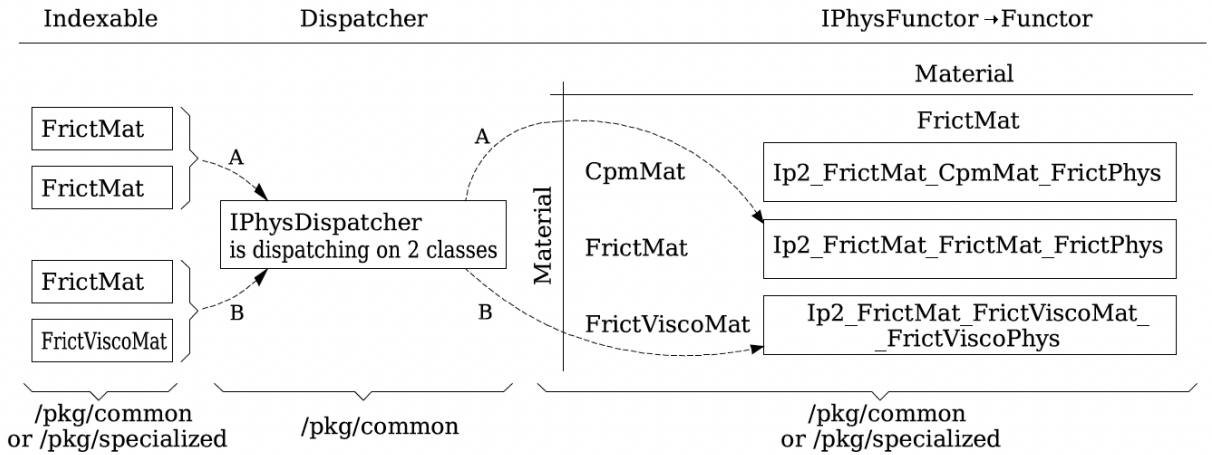
Functors to determine interaction properties:

1. **IGeom (Ig2)**, combination of two shapes to define interaction geometry.
2. **IPhys (Ip2)**, combination of two materials to define interaction physics.
3. **Law (Law2)**, combination of **IGeom** and **IPhys** to define contact law.

IGeom (Ig2 functors)



IPhys (Ip2 functors)



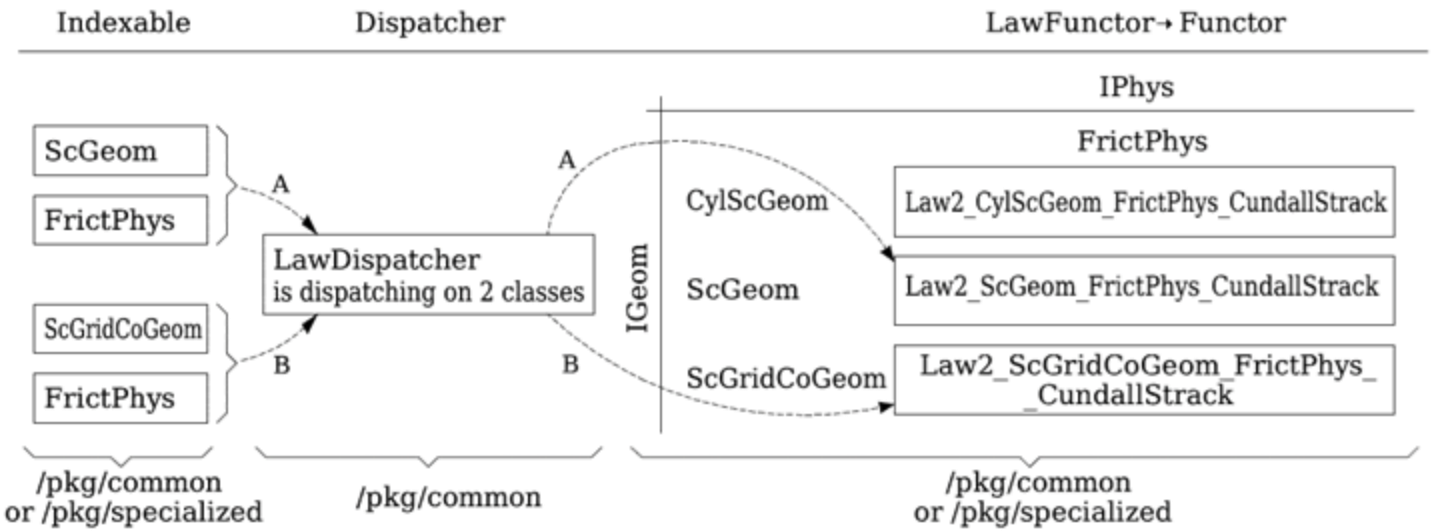
(Šmilauer et al, Yade documentation 3rd ed, 2021)

Interaction Geometry (IGeom) + Interaction Physics (IPhys)

Functors to determine interaction properties:

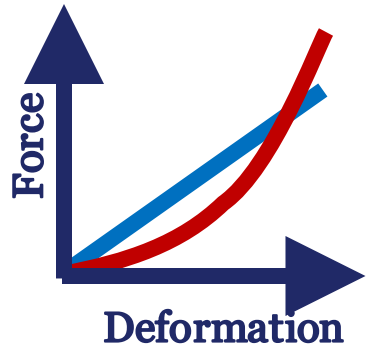
1. **IGeom (Ig2)**, combination of two shapes to define interaction geometry.
2. **IPhys (Ip2)**, combination of two materials to define interaction physics.
3. **Law (Law2)**, combination of **IGeom** and **IPhys** to define contact law.

Law (Law2 functors)

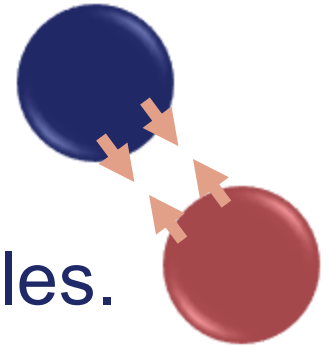


(Šmilauer et al, Yade documentation 3rd ed, 2021)

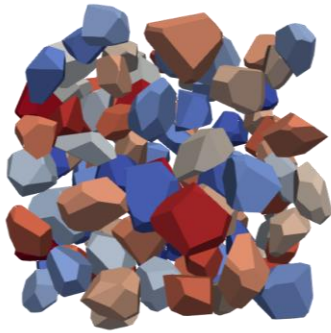
YADE: Some modelling capabilities…



- **Wide variety of interaction models**
  Linear stiffness & Hertz-Mindlin models, rolling resistance, adhesion & capillarity for pendular liquid bridges.



- **Particle shape + deformable particles**
  Clumps, polyhedra, potential particles, level-set particles.



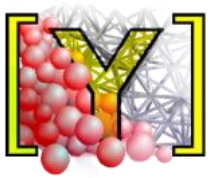
- **Crushable particles**
  Replacement particle method and bonded particle method

- **Particle-fluid systems**
  PFV-DEM and CFD-DEM schemes

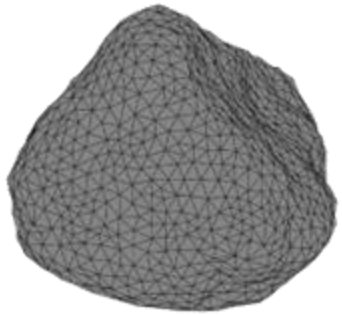
Interaction models (Law2)

YADE features a modest list of 35 contact laws to define the interaction forces between particles

(Šmilauer et al, Yade documentation 3rd ed, 2021)



Rigid particles

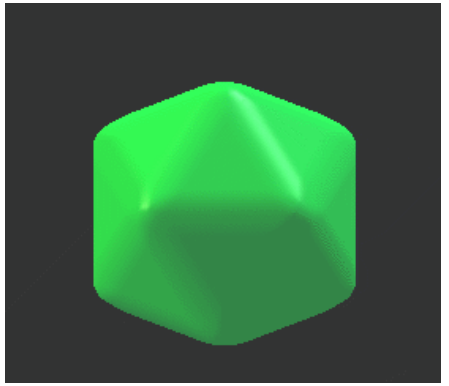


Sphere

Box
Non-dynamic cuboids

Facet
Non-dynamic triangles

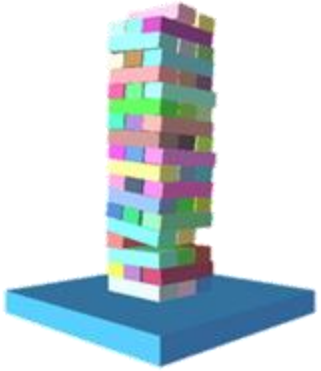
Wall
Non-dynamic infinite wall



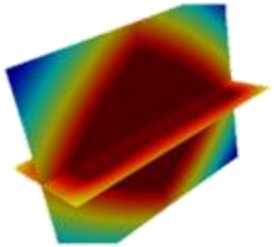
Clump
Rigid agglomeration of particles

PotentialParticle
Rounded convex particles defined by a potential function

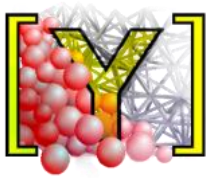
Polyhedra & PotentialBlock
Convex polyhedra with sharp edges



LevelSet
Particles defined by a distance-field function



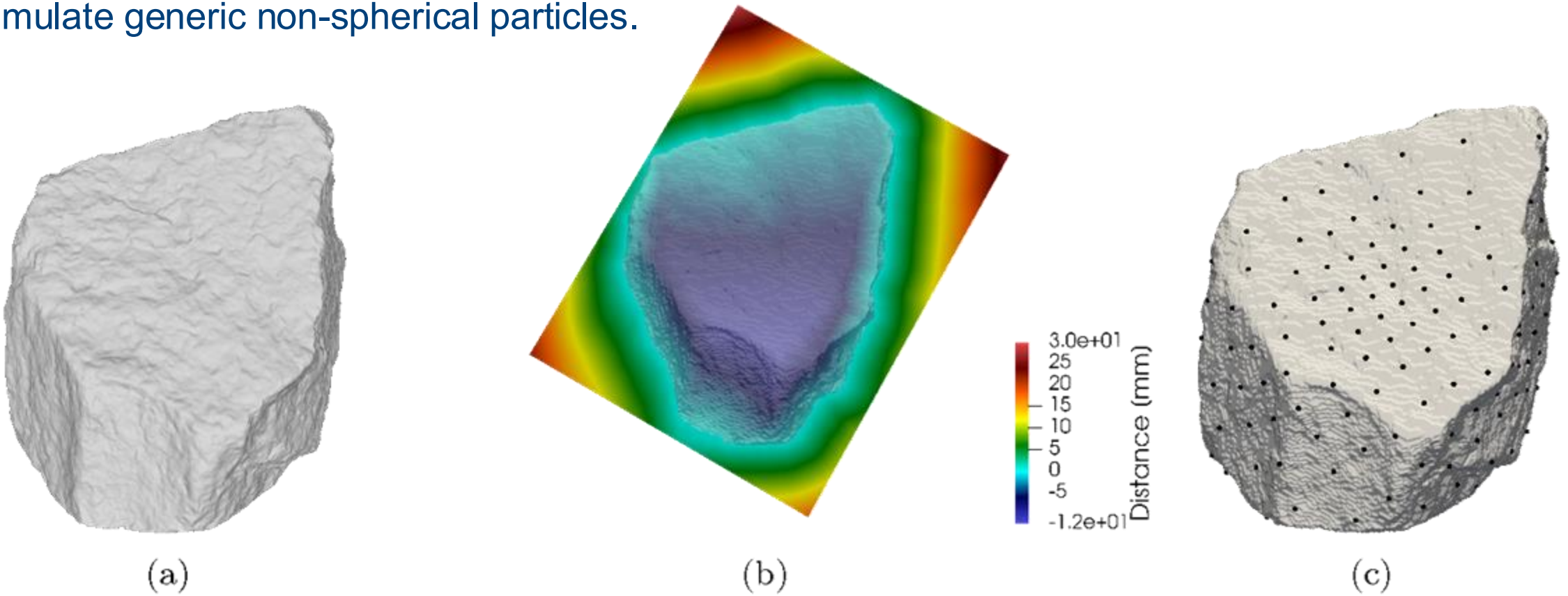
(Angelidakis et al, CPC, 2024)



Rigid particles: Level-Set DEM (LS-DEM)

The LS-DEM has been developed to simulate generic non-spherical particles.

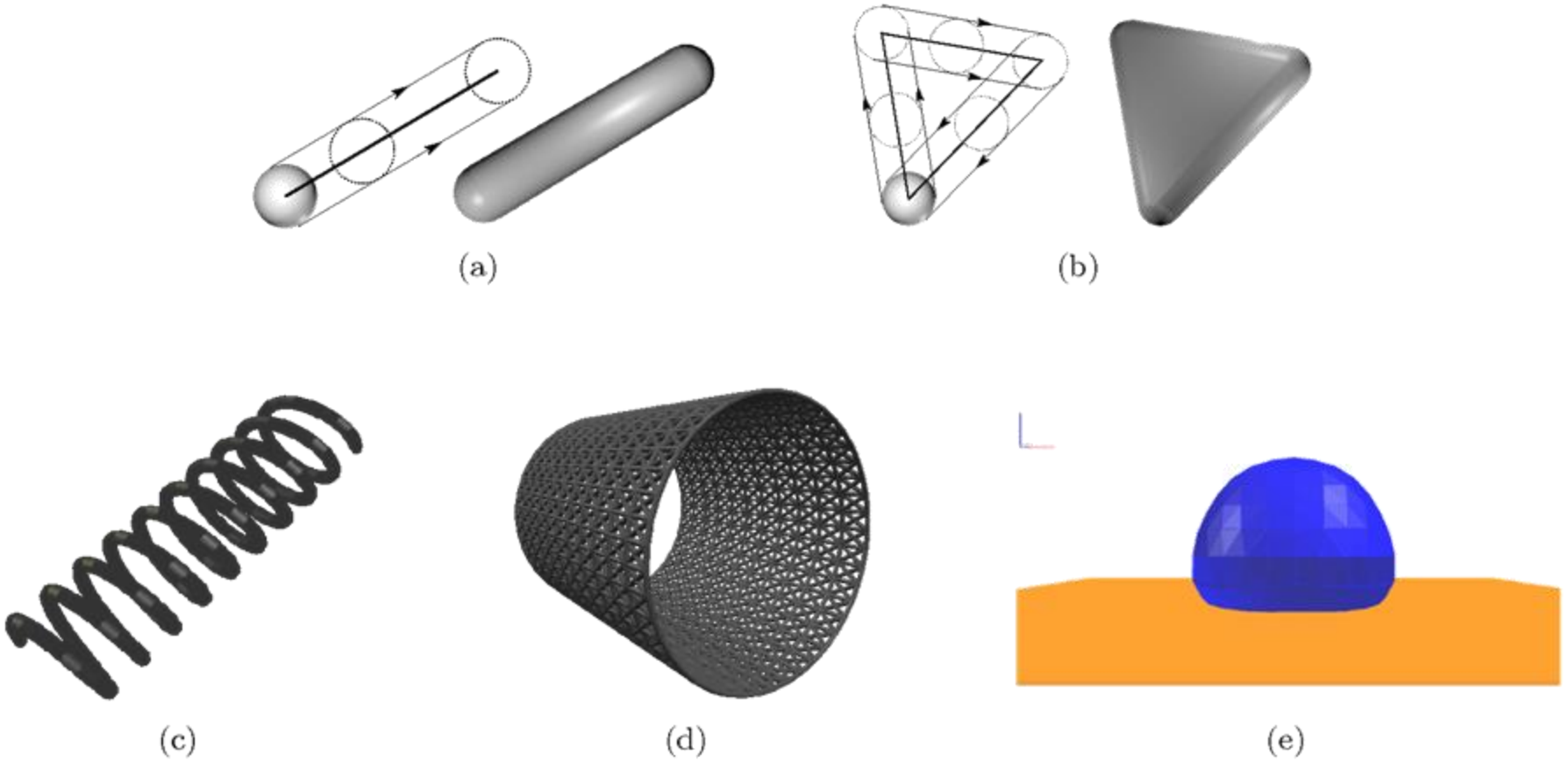
(Duriez and Galusinski, Comput. Geosci, 2021)



The Volumetric LS-DEM (VLS-DEM) is also developed to provide a different handling of contact mechanics.

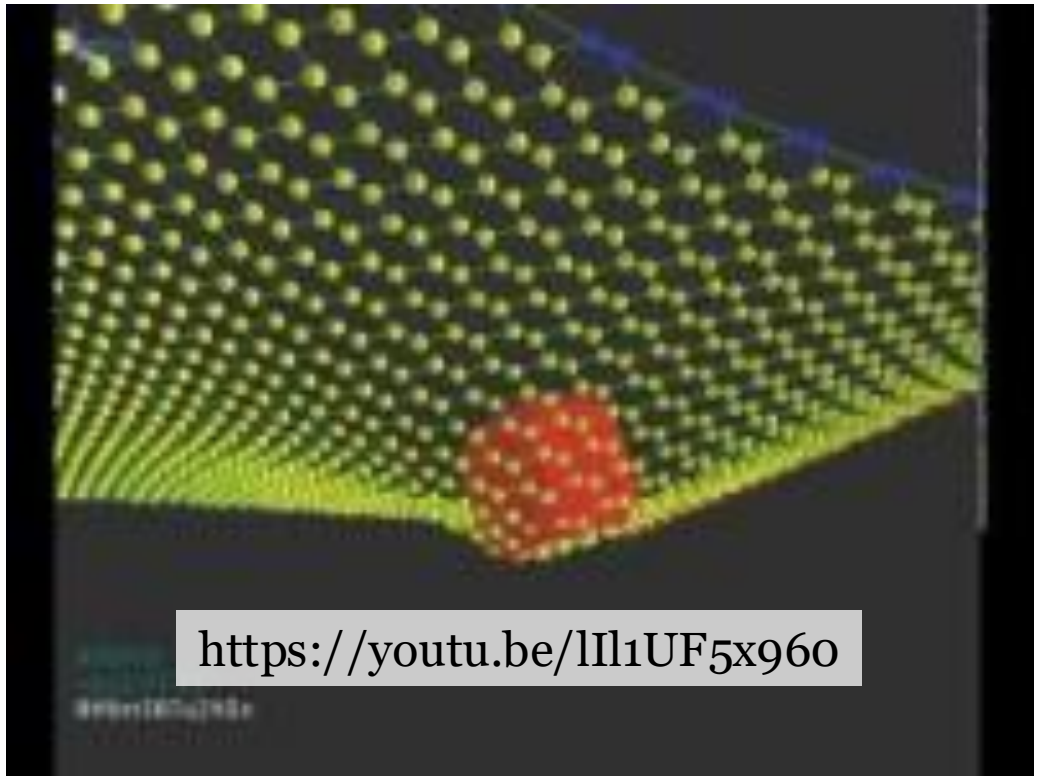
(van der Haven et al, CMAME, 2023)

Deformable particles: Spherocylinders/Spheropolyhedra



(Effeindzourou et al, Geotext. Geomembr, 2016)

Deformable particles: Wires & spherocylinders

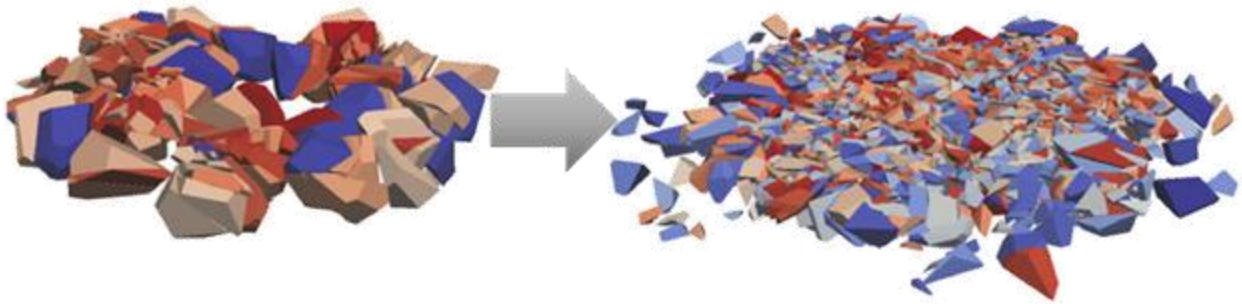


(Thoeni et al, Comput. Geotech, 2013)

(Effeindzourou et al, Geotext. Geomembr, 2016)

Crushable particles: Polyhedra and Clumps

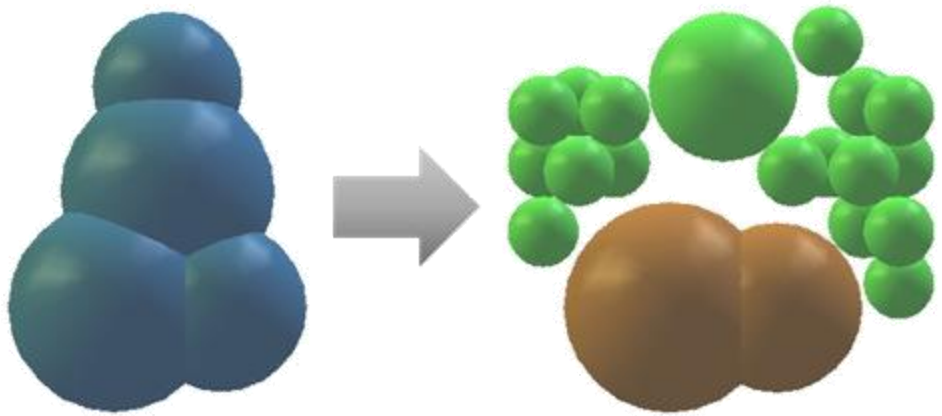
Particle replacement method for sharp polyhedra



(a)

(Gladky and Kuna, Granul. Matter, 2017)

Particle replacement method for multispheres (clumps)

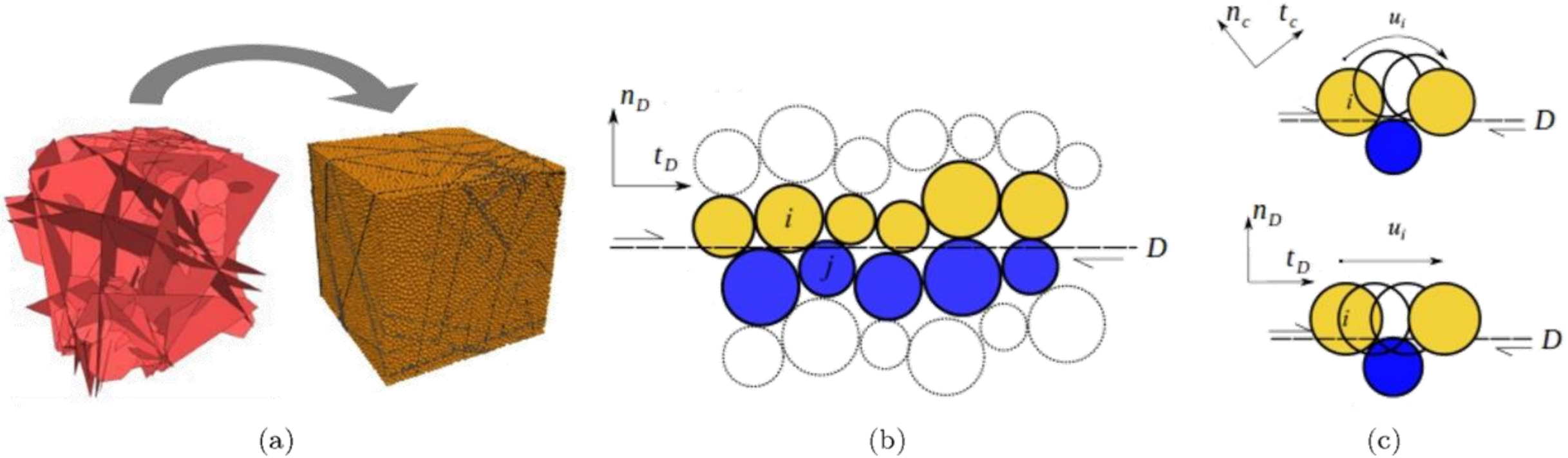


(b)

(Brzeziński and Gladky, Tribol. Int, 2022)

Jointed Cohesive Frictional Particle Model (JCFPM)

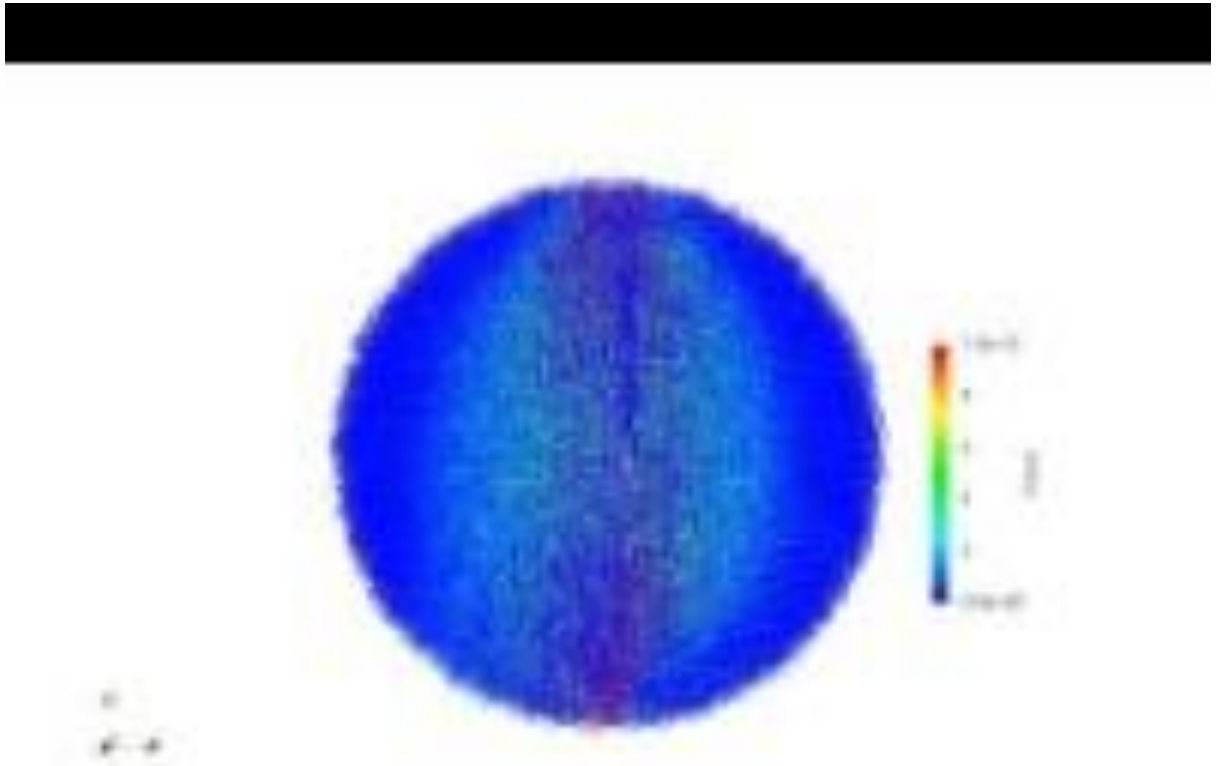
- Planar surfaces are overlain with a set of discrete elements to define discontinuities, with frictional and cohesive properties.



(a) (b) (c)

(Scholtès et al, Int. J. Rock. Mech, 2012)

Jointed Cohesive Frictional Particle Model (JCFPM)



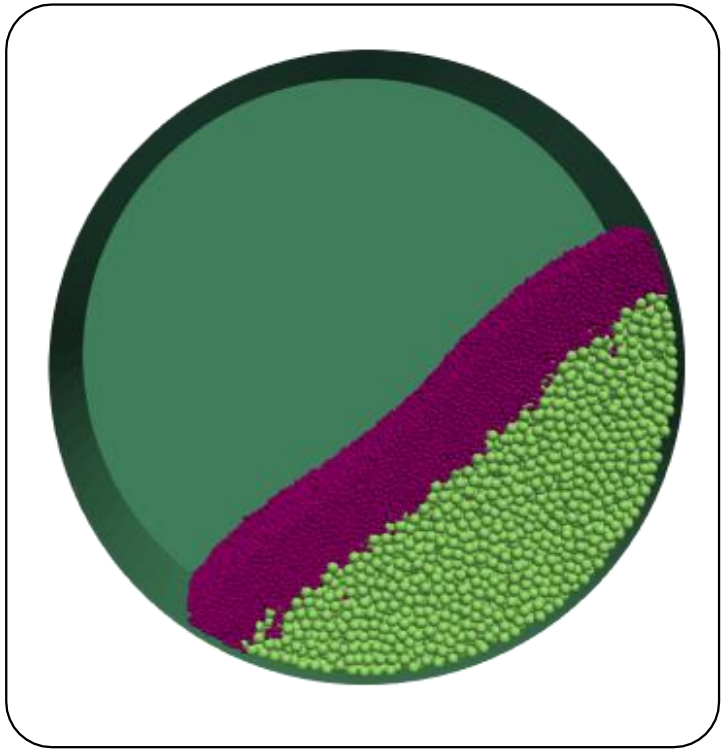
https://youtu.be/K8Z_ySLIXrQ

Brazilian disc test

(Scholtès et al, Int. J. Rock. Mech, 2012)

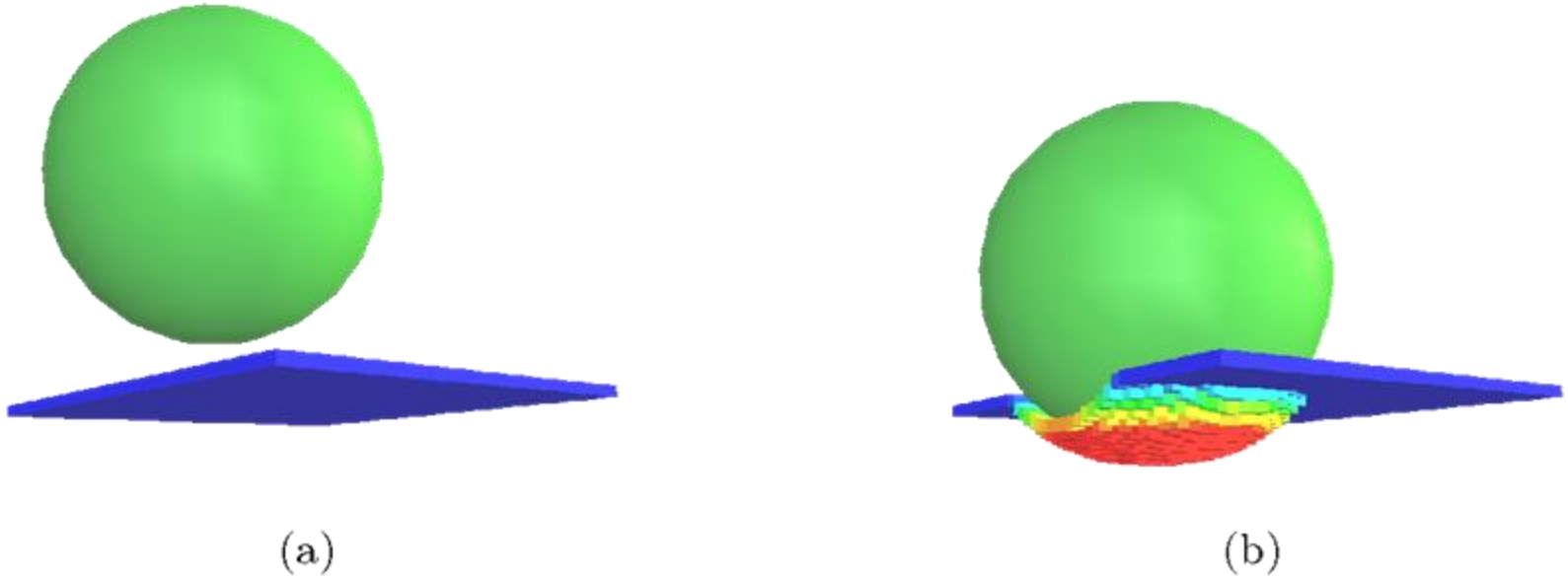
Boundary conditions: Rigid boundaries

- Any body can be used to form static or dynamic boundaries. These can be flat walls, or triangulated surfaces to form any shape, such as the drum in the graphic below.



Boundary conditions: Viscoelastic boundaries

- Burger's model can be used as a boundary condition, to allow for compliant boundaries, such as the cuboidal boundaries in the graphic below.

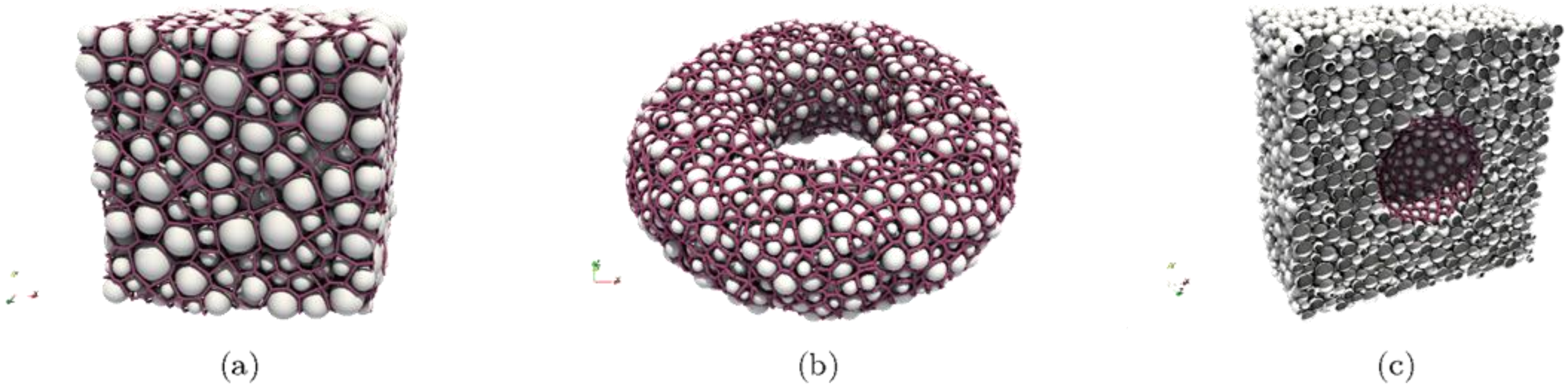


(a) (b)

(Brzeziński et al, J. Theor. Appl. Mech, 2023)

Boundary conditions: Uniform boundaries

- Uniform boundary conditions, enclosing a set of particles, allowing for uniform stress application.

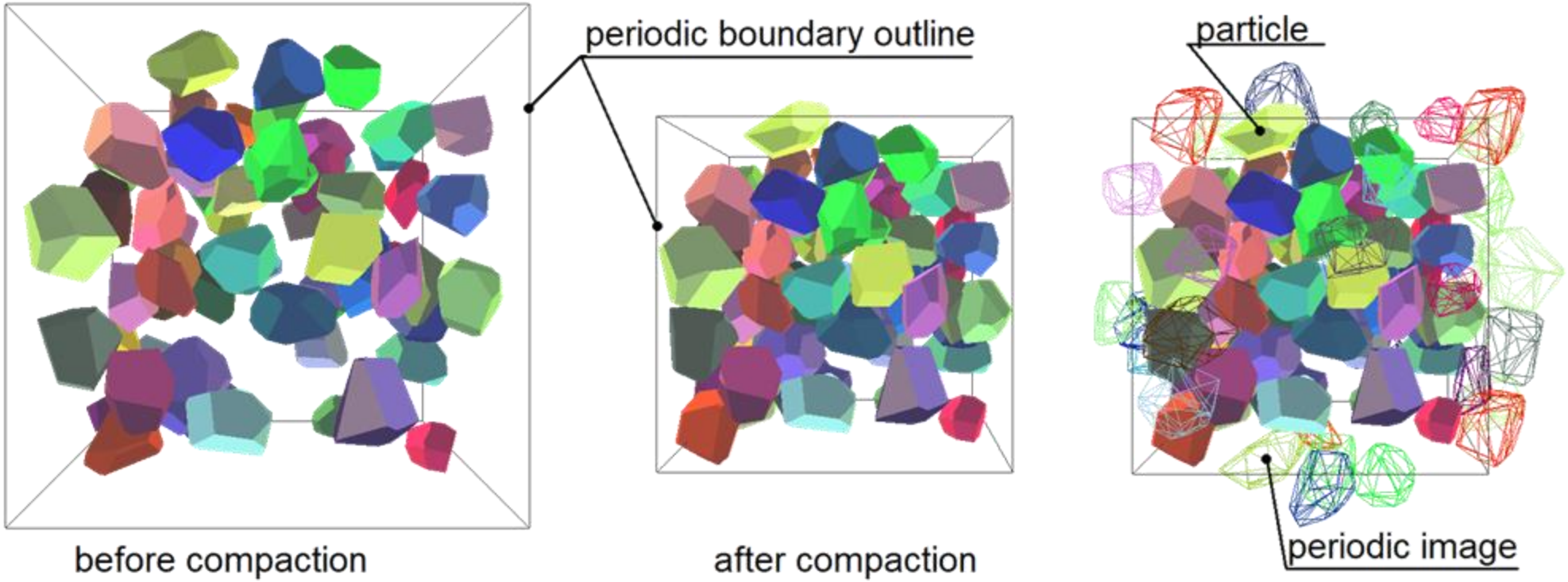


(a) (b) (c)

(Pekmezi et al, CPC, 2024)

Boundary conditions: Periodic boundaries

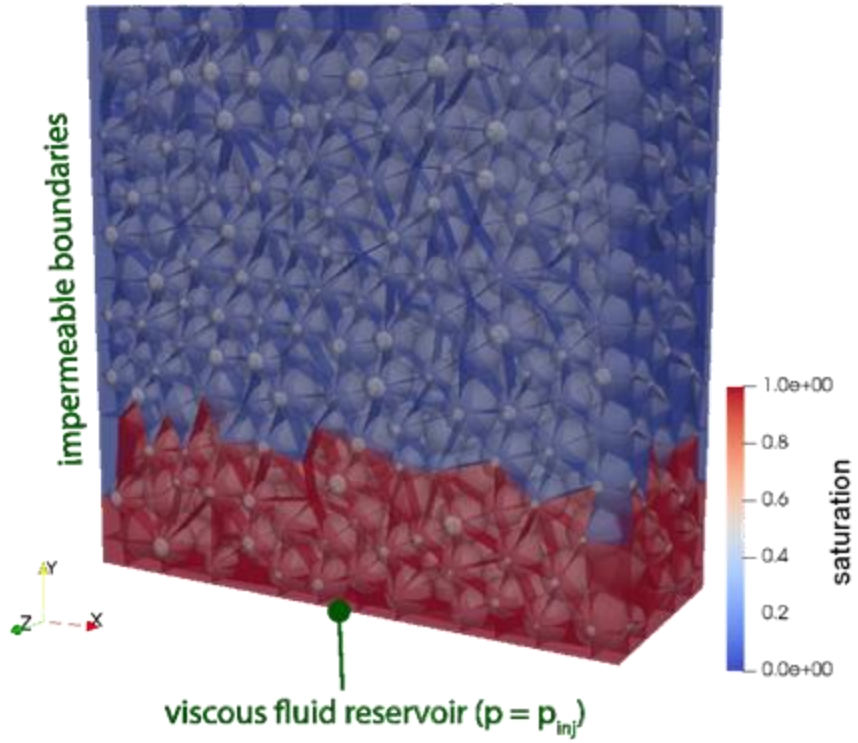
- Periodic boundaries can be used to eliminate boundary effects, creating homogeneous conditions via application of velocity gradients on a granular packing.



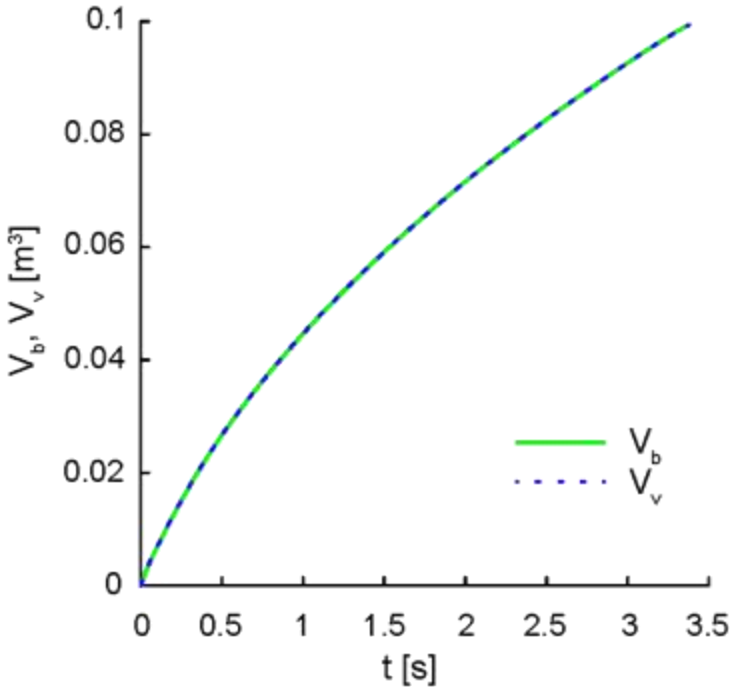
(Šmilauer et al, Yade documentation 3rd ed, 2021)

Particle-fluid interaction: PFV-DEM coupling

- Yade provides a **FlowEngine** and **TwoPhaseFlowEngine** to simulate particle-fluid systems, based on an in-house Particle-Finite Volume (PFV) framework. Here we see the injection of a viscous fluid through a granular bed, where the fluid flow follows Stoke's law. Compressible flow is also supported.



(a)

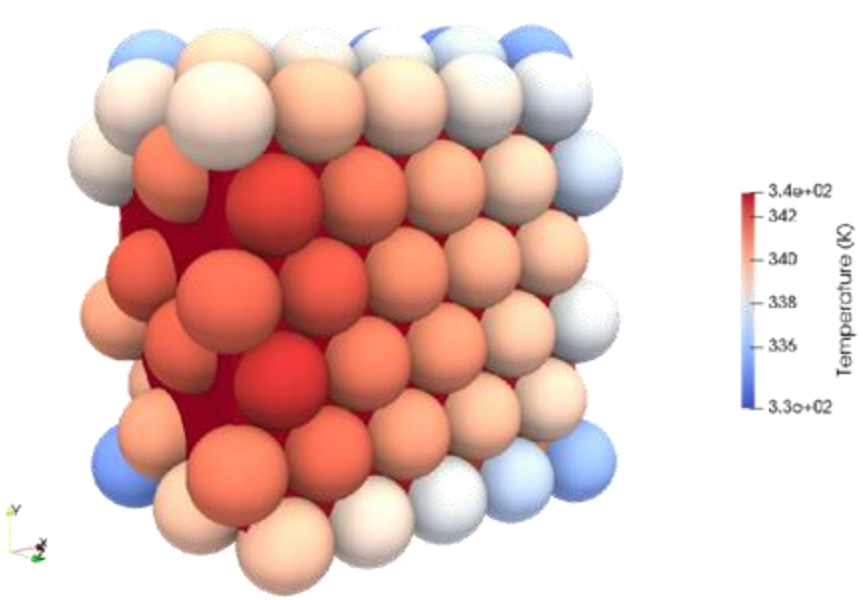


(b)

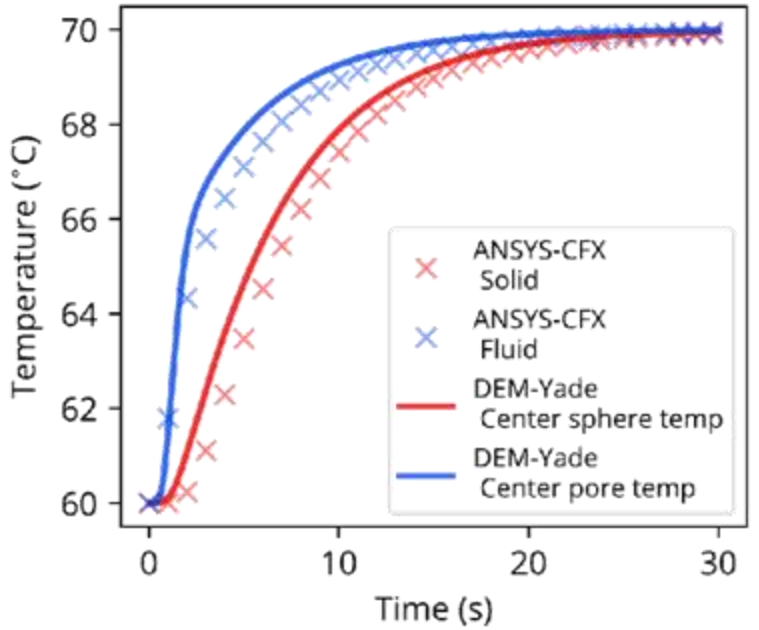
(Chareyre et al, Transp. Porous Media, 2012; Catalano et al, Int. J. Num. Anal. Methods Geomech, 2014; Scholtès et al, Contin. Mech. Thermodyn, 2015)

Particle-fluid interaction: Thermal Engine

- The **ThermalEngine** opens up a wide range of Thermal-Hydraulic-Mechanical (THM) modelling possibilities, such as non-isothermal flow through a conductive sphere packing and geomechanical thermo-poroelasticity.



(a)



(b)

(Caulk et al, CMAME, 2020)

Particle-fluid interaction: CFD-DEM

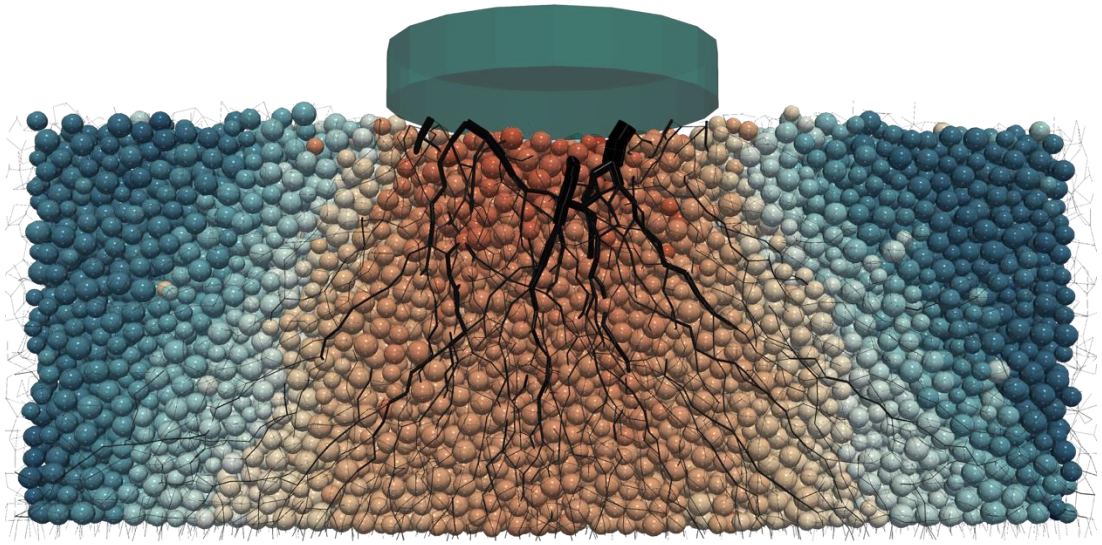
- YADE has been coupled with OpenFOAM, to simulate particle-fluid interaction.
- Original coupling with **OpenFOAM 6**, which has been expanded recently up to OpenFOAM 11, and OpenFOAM2312, supporting both the OpenFoam Foundation and ESI branches.
- The initial coupling was with **icoFOAM** and **pimpleFOAM**, for single-phase flow.
- We are currently coupling YADE with **interFOAM**, and **interIsoFOAM**, for two-phase flows.

Example of CFD-DEM simulation with OpenFOAM: 50 000 spheres in lid-driven cavity flow.

(Angelidakis et al, CPC, 2024)

From discrete to continuum results: Tessellation wrapper

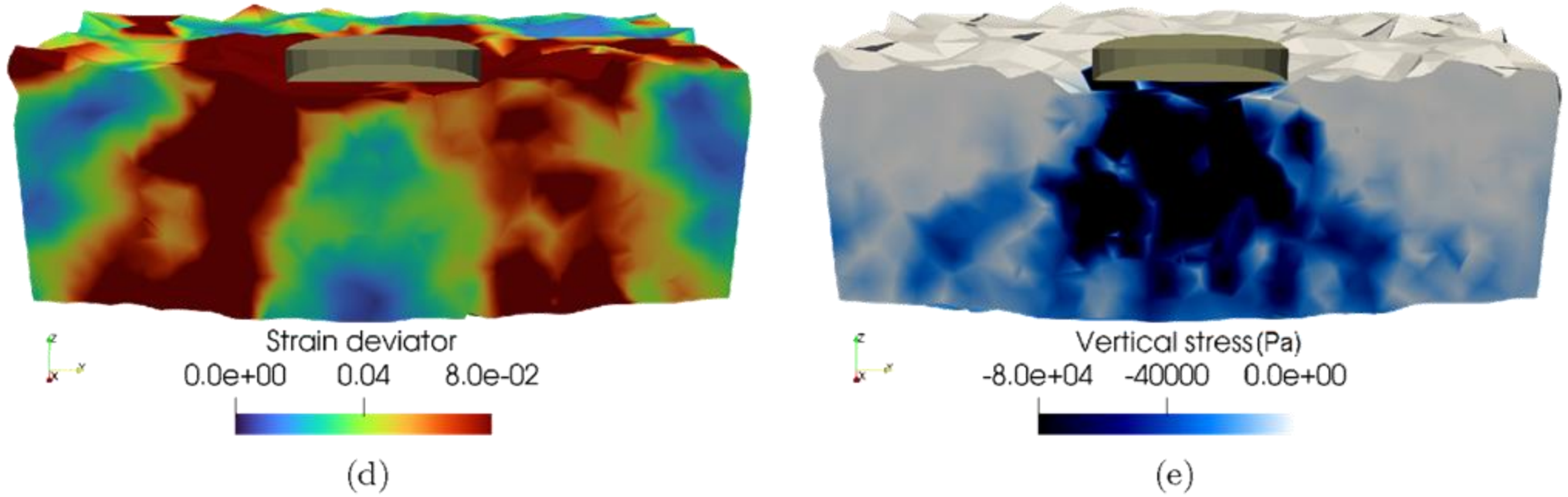
- The Tessellation wrapper is a suite of in-house YADE tools, to generate continuum fields from discrete results.



(Catalano et al, Int. J. Num. Anal. Methods Geomech, 2014)

From discrete to continuum results: Tessellation wrapper

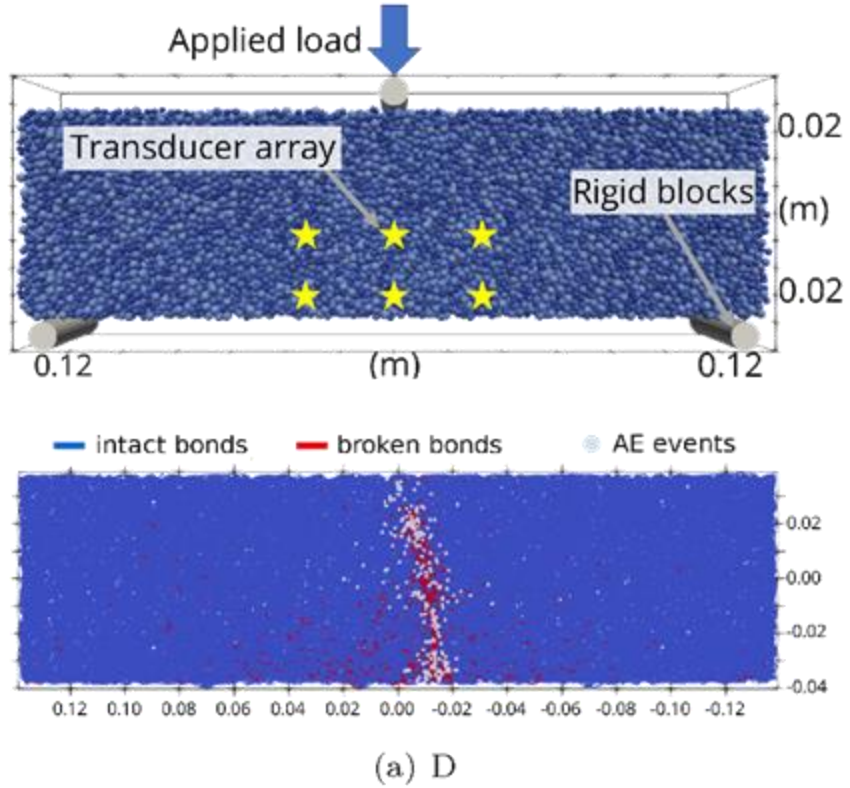
- The Tessellation wrapper is a suite of in-house YADE tools, to generate continuum fields from discrete results, e.g. micro-strain (left) and micro-stress (right) in the graphic below.

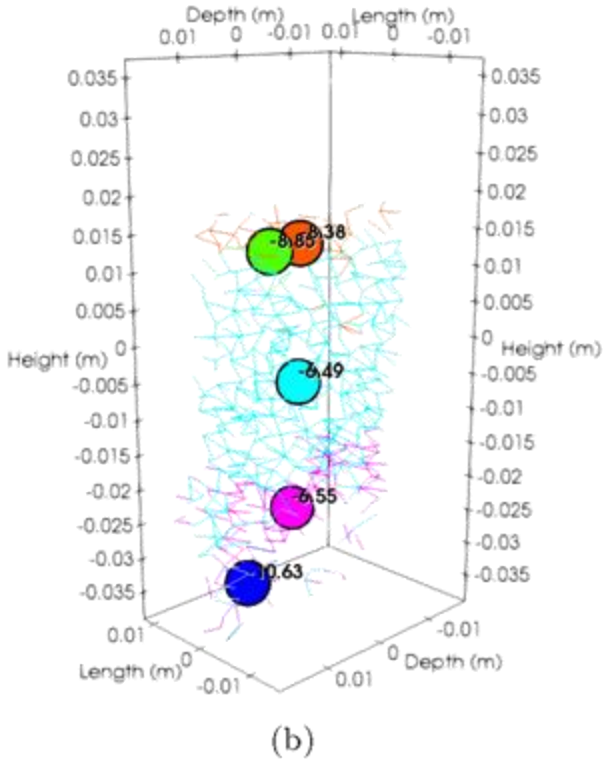


(Catalano et al, Int. J. Num. Anal. Methods Geomech, 2014)

Acoustic emissions (AE)

- YADE can simulate acoustic emission events in cohesive discrete element assemblies. The AE module, enables the clustering of bonds that break close to one another which allows for the realistic quantification of AE magnitude during material failure.



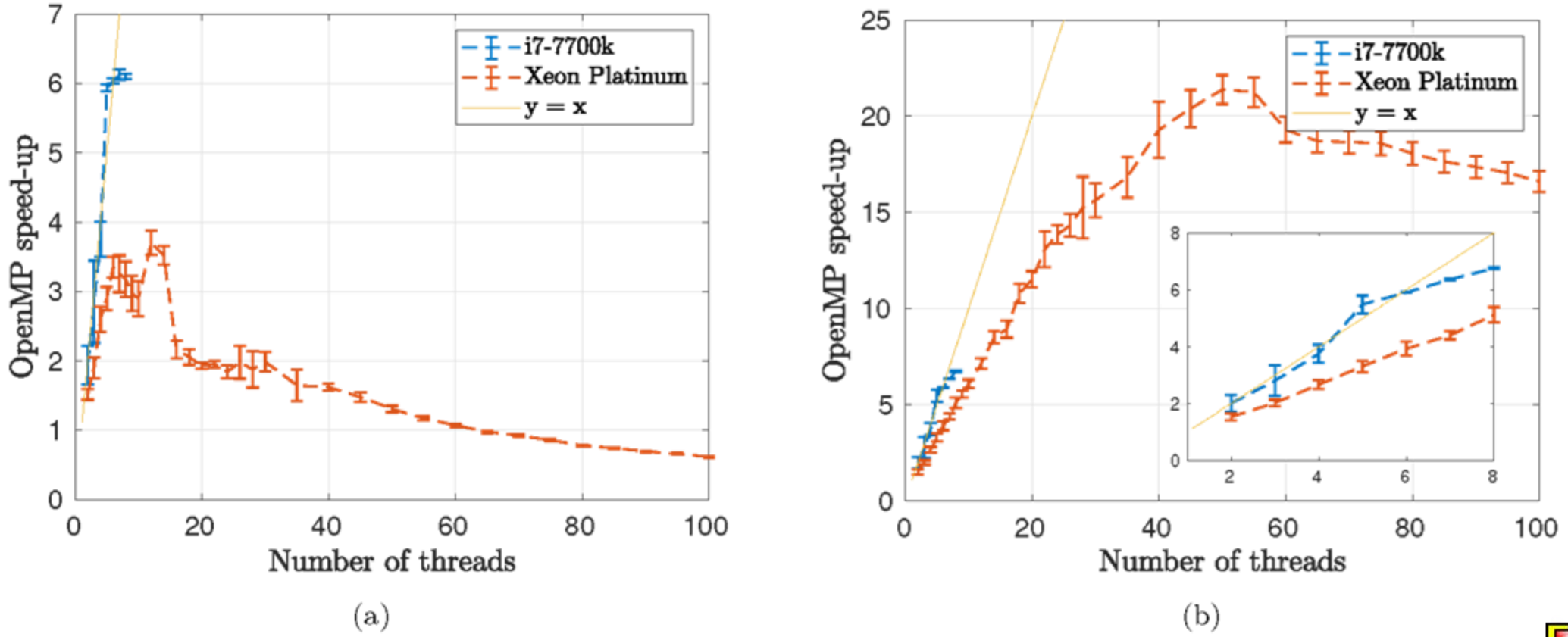


(Caulk, Open Geomech, 2020)

Parallelisation: OpenMP

- YADE supports both OpenMP and MPI paralellisation schemes, as well as hybrid schemes combining the two.

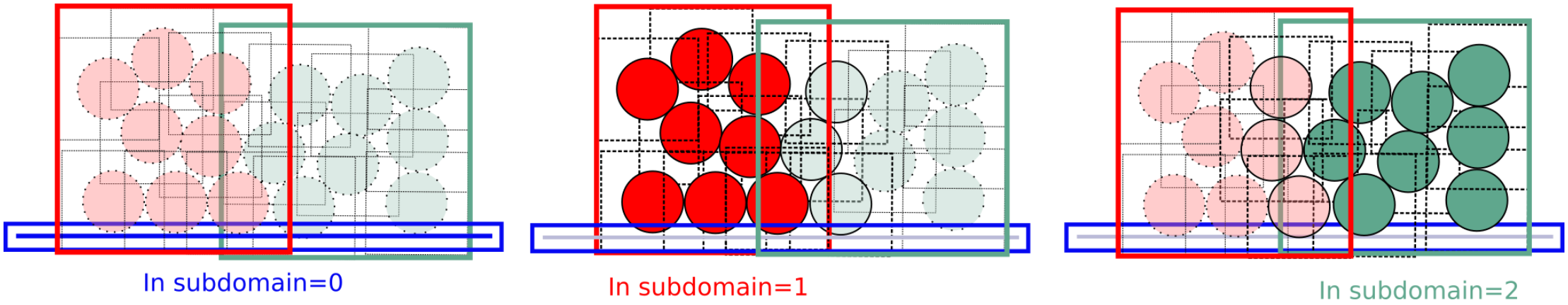
(Duriez and Bonelli, Comput. Geotech, 2021)



OpenMP speed up for a quasi-static axisymmetric compression of a packing including 8 000 grains, (a) simple spheres and (b) LevelSet shapes.

Parallelisation: MPI using the *mpy* module

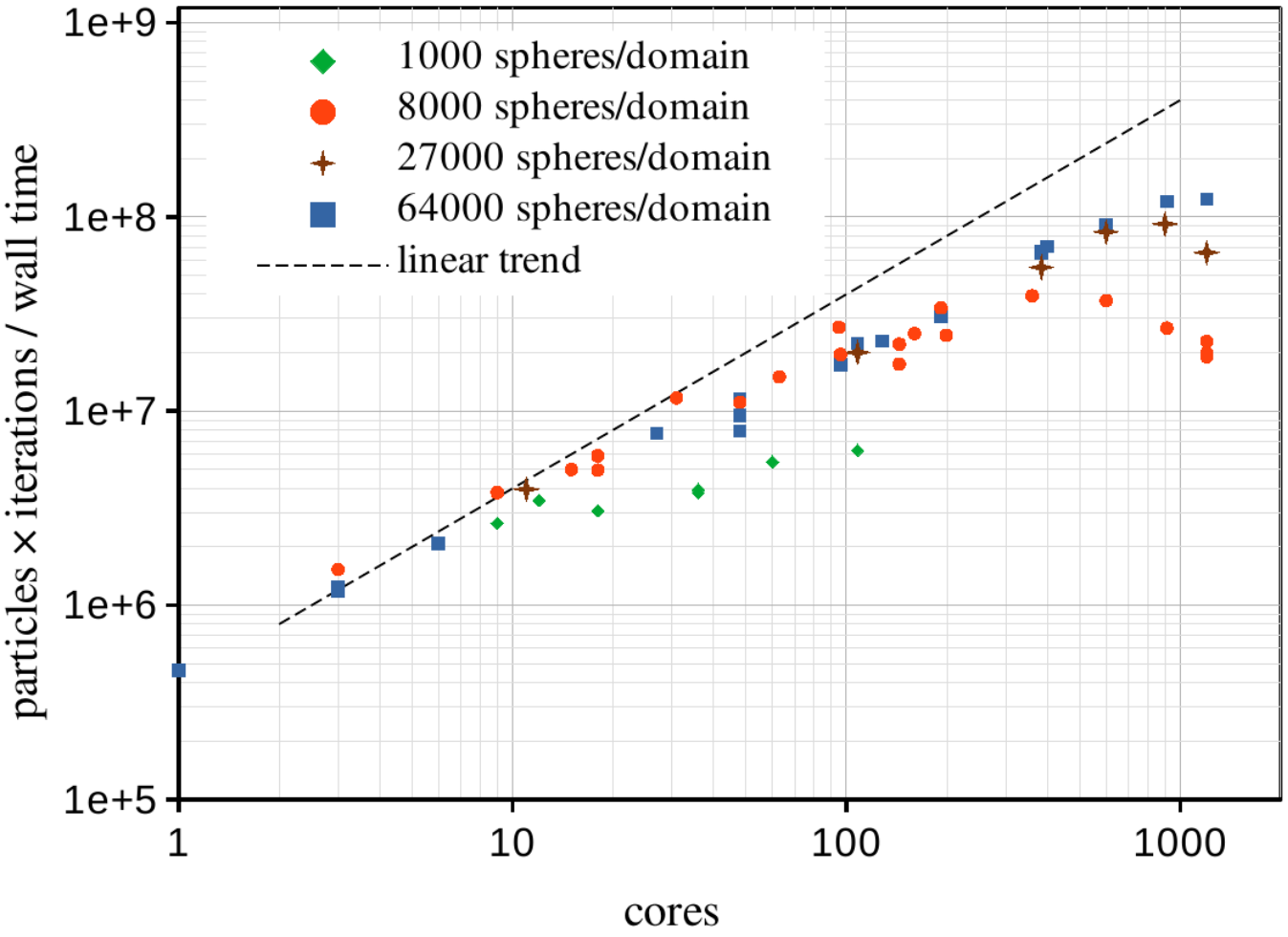
- YADE supports both OpenMP and MPI paralellisation schemes, as well as hybrid schemes.



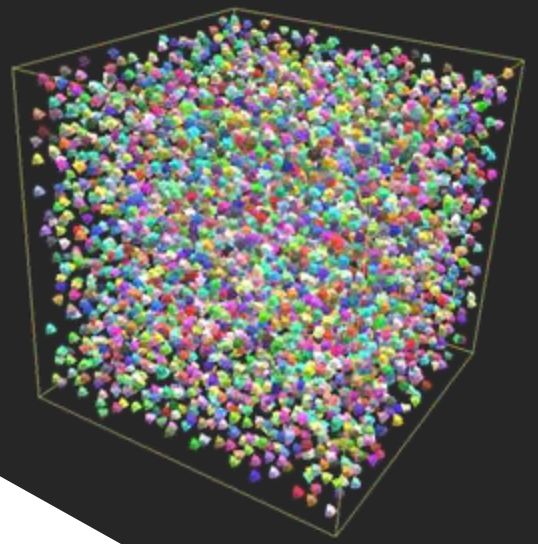
Subdomains and their AABBs as defined by domain decomposition in the mpy (MPI) module. Subdomain=0 owns the base, and it passes the bounding boxes of the other subdomains to handle collision detection between subdomains.

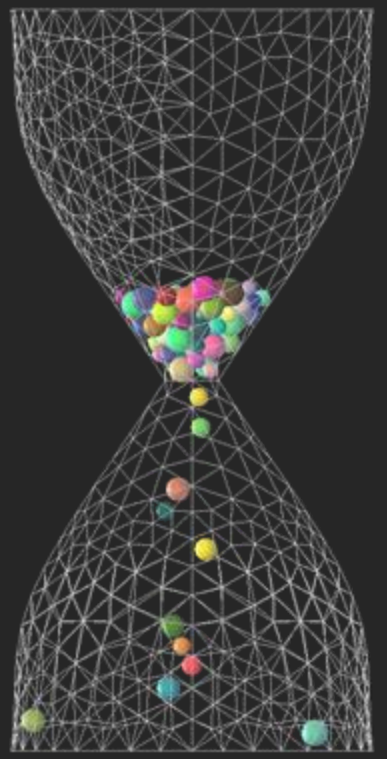
(Šmilauer et al, Yade documentation 3rd ed, 2021)

Parallelisation: MPI using the *mpy* module



YADE's throughput with mpy's domain decomposition and a constant number of spheres per core (weak scaling). The dotted line indicates a linear trend. The maximum number of spheres (1200 cores with 64 × 103 spheres) is 76.8 × 106.





PART B: Hands-on

- Tutorial 0: GUI navigation
- Tutorial 1: Bouncing sphere
- Tutorial 2: Angle of repose
- Tutorial 3: Granular flow in an hourglass
- Tutorial 4: Uniaxial compression test (oedometer)
- Tutorial 5: Triaxial compression test
- Tutorial 6: Geogrid pull-out test
- Tutorial 7: Impact of granular flow on flexible barrier

Accessing YADE

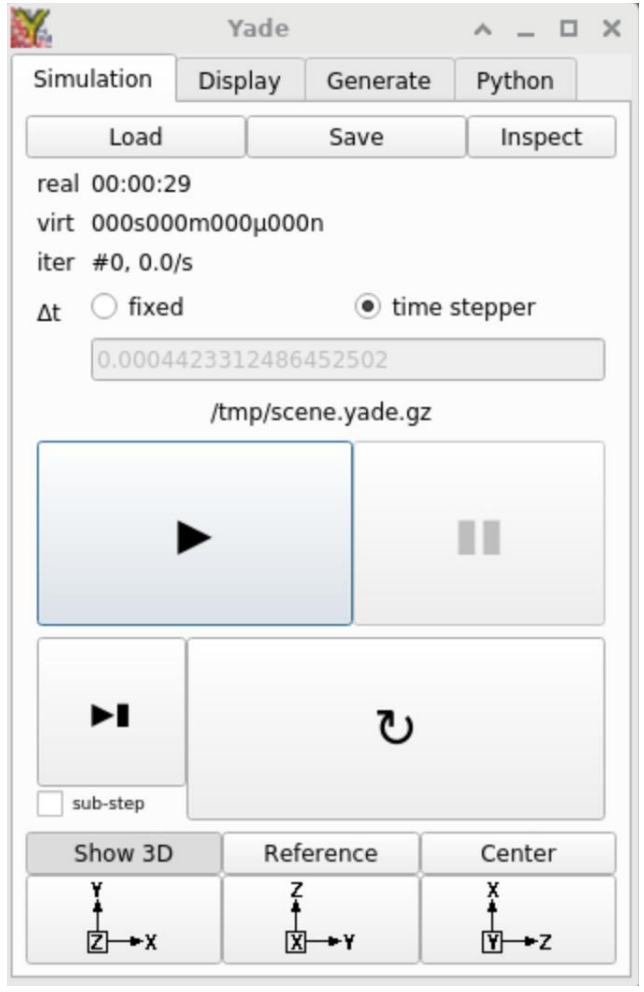
- YADE installation: https://www.yade-dem.org/doc/installation.html

- Upon installation, you can open a terminal and run the command yade

- YADE is supported by various unit tests and complex checks, e.g. run:
  - Run yade --test → Run unit tests
  - Run yade --check → Run full simulation script checks

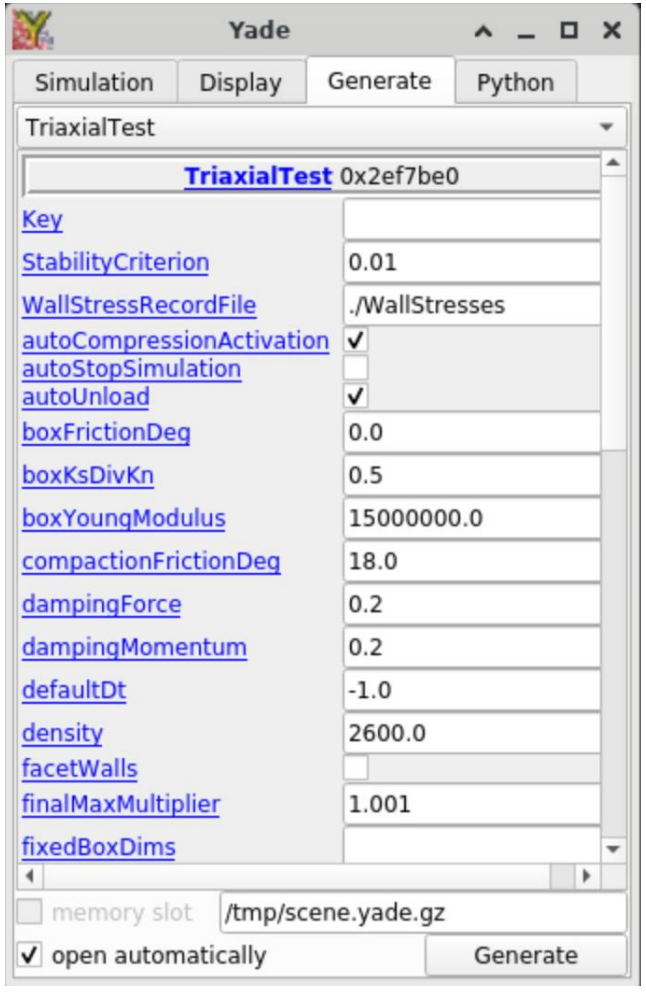
Tutorial 0 - GUI navigation

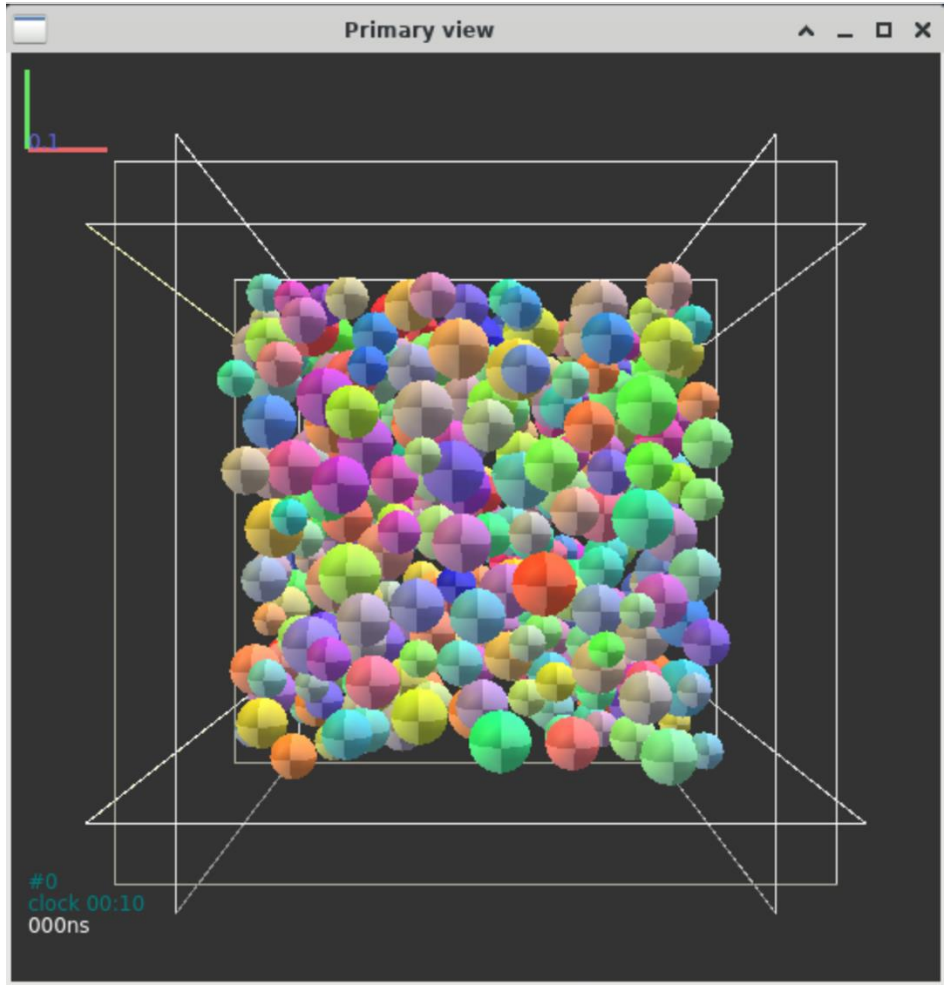
- Running YADE launches an interactive Python terminal.
- YADE has a GUI build using the qt library, which allows for fast scene creation.
- Open a terminal and type yade.
- Click on F12, and you should see the qt.Controller
  - Alternatively, type in the terminal:
    - from yade import qt
    - qt.Controller()

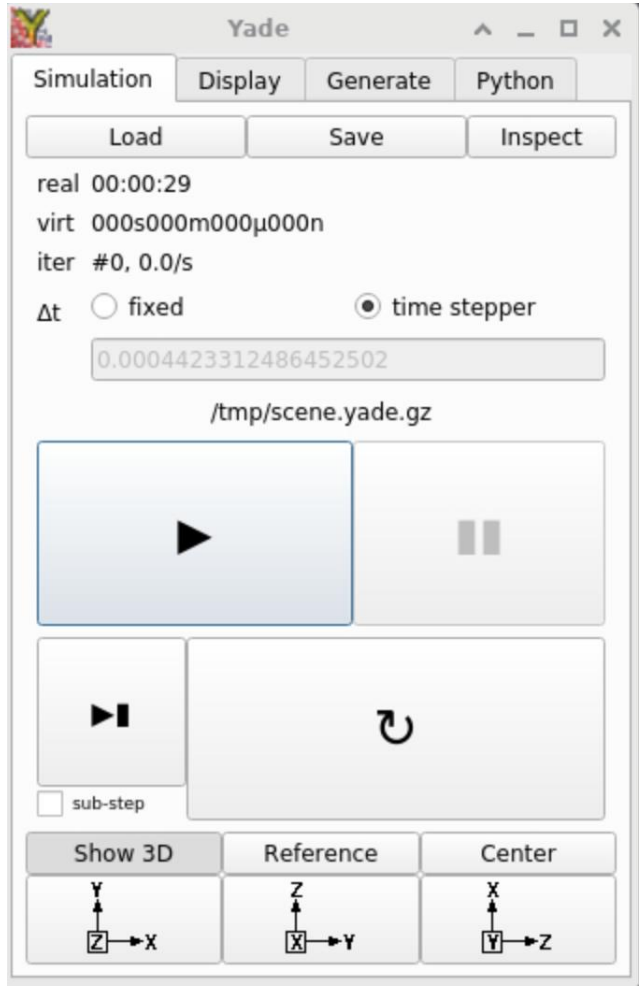


Tutorial 0 - GUI navigation

- Go to the Generate tab and click on Generate at the bottom of the controller.
- Navigate the Simulation, Display, Generate, Python tabs







Let’s dive in – Omega and its containers

The Omega class (abbreviated with “O”) holds all information of a YADE simulation:

- class yade.wrapper.Omega

Containers are hold various elements of a YADE simulation:

- O.bodies particles present in the simulation
- O.materials materials
- O.interactions interactions
- O.forces generalised forces (i.e. forces and torques)
- O.engines periodic functions of the interaction loop
- O.energy energies of the simulation (only if O.trackEnergy=True)

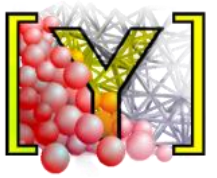
Let’s dive in – Omega and its containers

Most of the YADE source code is written in C++ and many parts have been “wrapped” in Python, to expose properties to the user and provide an interactive interface, easy scene generation and real-time processing.

As a result, everything in YADE is an object with attributes.

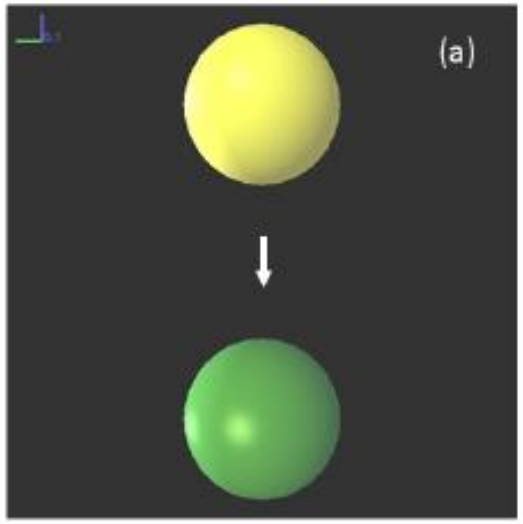
For instance, to define a material:

```
m1=FrictMat(young=50e9, poisson=0.2, frictionAngle=0.3, label=“mat1")
O.materials.append(m1)          # add material to the scene
```

Tutorial 1 – Bouncing sphere

- A free sphere bounces on top of a fixed sphere under gravity.
- Change the coefficient of restitution: $e_n = \frac{v_1}{v_0}$
- Increase the timestep and rerun




```
1  # Script to simulate a free sphere bouncing under gravity on a fixed sphere
2  # 2025 © Vasileios Angelidakis <v.angelidakis@qub.ac.uk>
3  # Developed based on https://yade-dem.org/doc/tutorial-examples.html with
   modifications to adopt the viscoelastic Hertz-Mindlin contact law
4
5  # Create materials
6  O.materials.append(FrictMat(young=5e9, poisson=.25, frictionAngle=0.0, density=2500,
   label='particles'))
7
8  # Create particles
9  O.bodies.append([
10         sphere(center=(0, 0, 0), radius=.5, material='particles', fixed=True), #
   fixed sphere
11         sphere(center=(0, 0, 2), radius=.5, material='particles') # free sphere
12 ])
13
14 # Engines - Simulation loop
15 O.engines = [
16         ForceResetter(),
17         InsertionSortCollider([Bo1_Sphere_Aabb()]),
18         InteractionLoop(
19                 [Ig2_Sphere_Sphere_ScGeom()],  # collision geometry
20                 [Ip2_FrictMat_FrictMat_MindlinPhys(en=1)],  # collision "physics"
21                 [Law2_ScGeom_MindlinPhys_Mindlin()]  # contact law -- apply forces
22         ),
23         # Apply gravity force to particles. damping: numerical dissipation of energy.
24         NewtonIntegrator(gravity=(0, 0, -9.81), damping=0.0) # No numerical damping
25 ]
26
27 # Timestep
28 O.dt = 0.0005 * RayleighWaveTimeStep() # set timestep to a fraction of the Rayleigh
   Wave timestep
29
30 # save the simulation, so that it can be reloaded later, for experimentation
31 O.saveTmp()
32
33 Gl1_Sphere.quality=5 # Make the visualisation of spheres nicer
34
```

Bodies

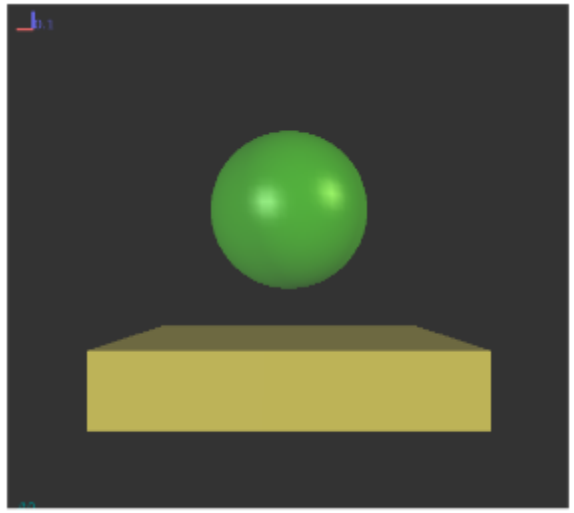
Engines

Timestep

Tutorial 1 - Let's replace the fixed sphere with a box

We need to add (1) a box particle, (2) an Aabb for the box, (3) contact geometry

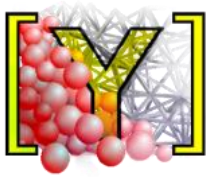
1) O.bodies.append([…,
   box(center=(0,0,0), extents=(1,1,0.25), material='particles', fixed=True),
   …])
2) Bo1_Box_Aabb()
3) Ig2_Box_Sphere_ScGeom(hertzian=True)



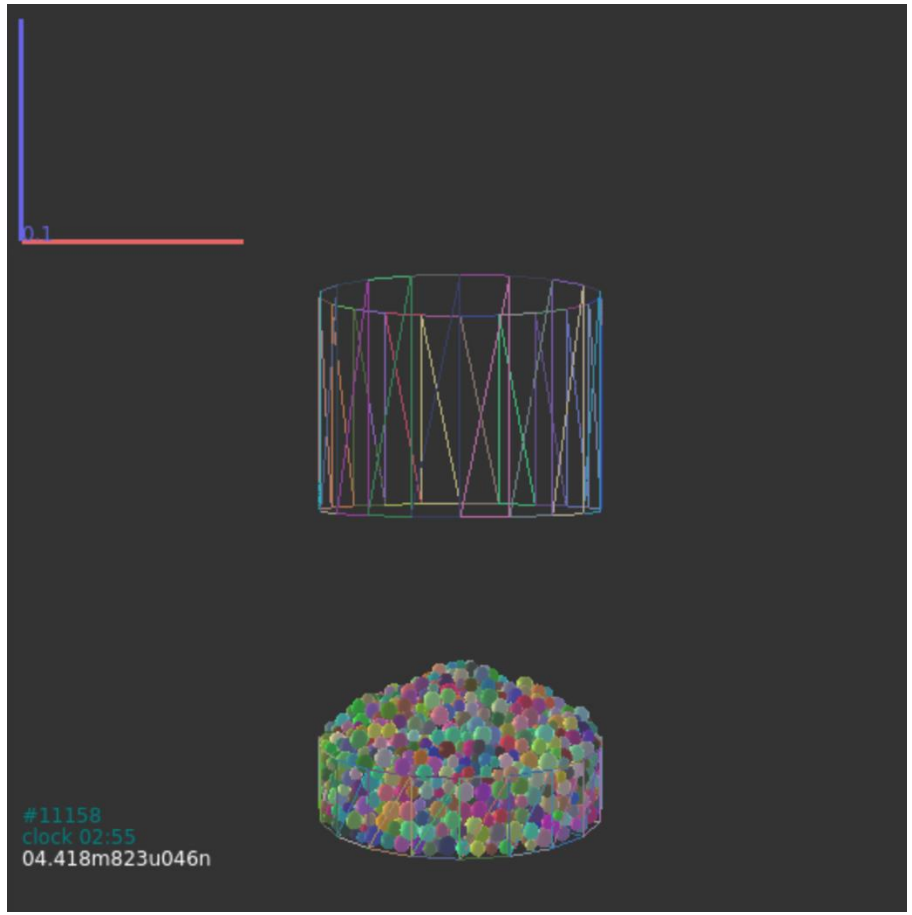
Tutorial 1 – Let's activate live plotting of results

```
37 # Live plotting of results
38 showPlots = True
39
40 if showPlots:
41         from yade import plot
42
43         # Function to plot data periodically
44         def addPlotData():
45                 if O.interactions.countReal()>0: # If the system has "real"
   interactions, then plot force vs displacement
46                         i=O.interactions[0,1]
47                         a=i.geom.penetrationDepth       # Normal deformation
48                         Fn=i.phys.normalForce[2]        # Normal force -> Z component
49                 else:
50                         a=0
51                         Fn=0
52                 plot.addData(deformation=a, force=Fn,
   position=O.bodies[1].state.pos[2], iteration=O.iter)
53 #                plot.saveDataTxt('results/bouncing-sphere.txt')
54
55         # besides unbalanced force evolution, also plot the displacement-force
   diagram
56         plot.plots = {'deformation': ('force'), 'iteration': ('position')}
57         plot.plot()
58
59         O.engines = O.engines + [PyRunner(command='addPlotData()', iterPeriod=200)]
60
```

- The plot module of YADE allows for real-time live plots.
- Here we define a function addPlotData() to call the plot module, and we call this function periodically by making it an engine using the PyRunner module of YADE.
- Uncomment the saveDataTxt command in line 53 to save the results in a txt file.



Tutorial 2 – Angle of Repose (AoR)



- Now let's define some rigid boundaries and let's make them move.
- The Angle of Repose is used in many communities to assess the flowability of granular materials.
- In this example we will generate a granular packing in a cylinder made of facets, we will let it settle under gravity, and then we will lift the top part of the cylinder, to make the material form a repose state.
- We start lifting once the unbalancedForce() ratio is low, i.e. when the system is relaxed.
- VTK output available for Paraview.

Tutorial 2 - Unbalanced force ratio

There are various metrics to decide whether a system of particles is relaxed, i.e. when it is in equilibrium and no significant movement or deformation is happening. This is essential for quasi-static simulations, where the systems needs to deform slowly, and not to develop inertial (dynamic) effects.

1. Unbalanced force ratio: An index calculating the ratio of average unbalanced forces per body (i.e. all resultant forces acting on its centroid) over the average contact forces per contact. Two calculations of this index:

(Ng, J. Eng. Mech, 2006)

Arithmetic mean value

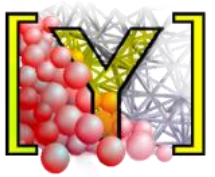
(YADE)

$$I_{uf} = \frac{\sum_{1}^{n_b} (\text{unbalanced forces})/n_b}{\sum_{1}^{n_c} (\text{contact forces})/n_c}$$

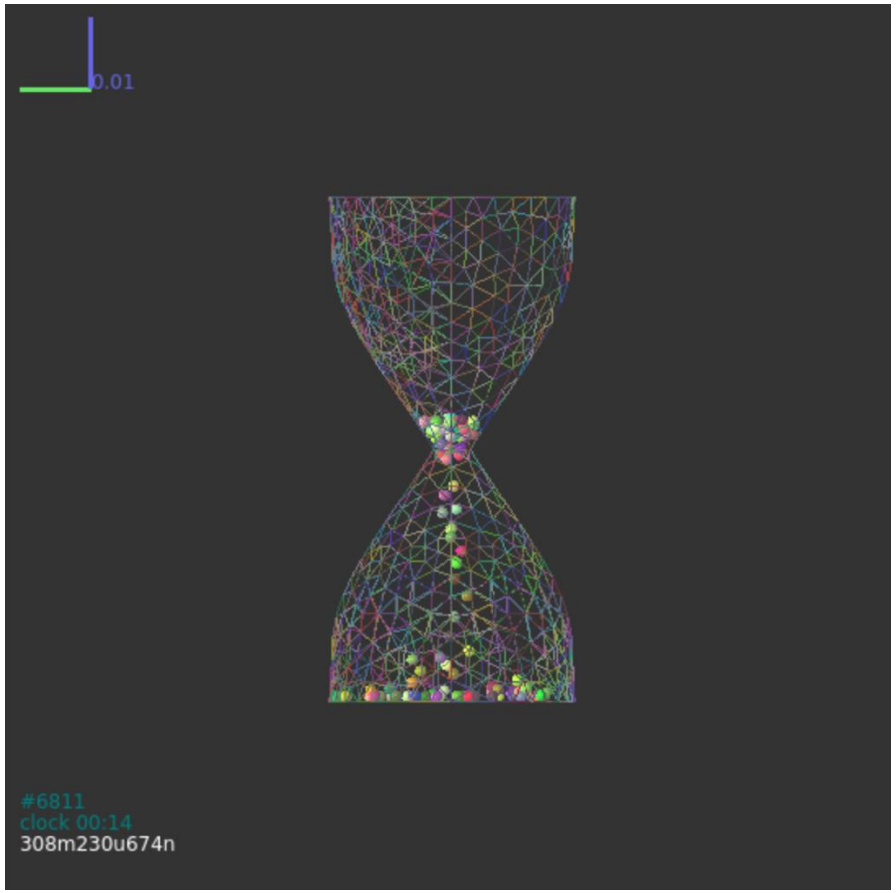
Root mean square value

(Ng, 2006)

$$I_{uf} = \sqrt{\frac{\sum_{1}^{n_b} (\text{unbalanced forces})^2/n_b}{\sum_{1}^{n_c} (\text{contact forces})^2/n_c}}$$



Tutorial 3 – Granular flow in an hourglass



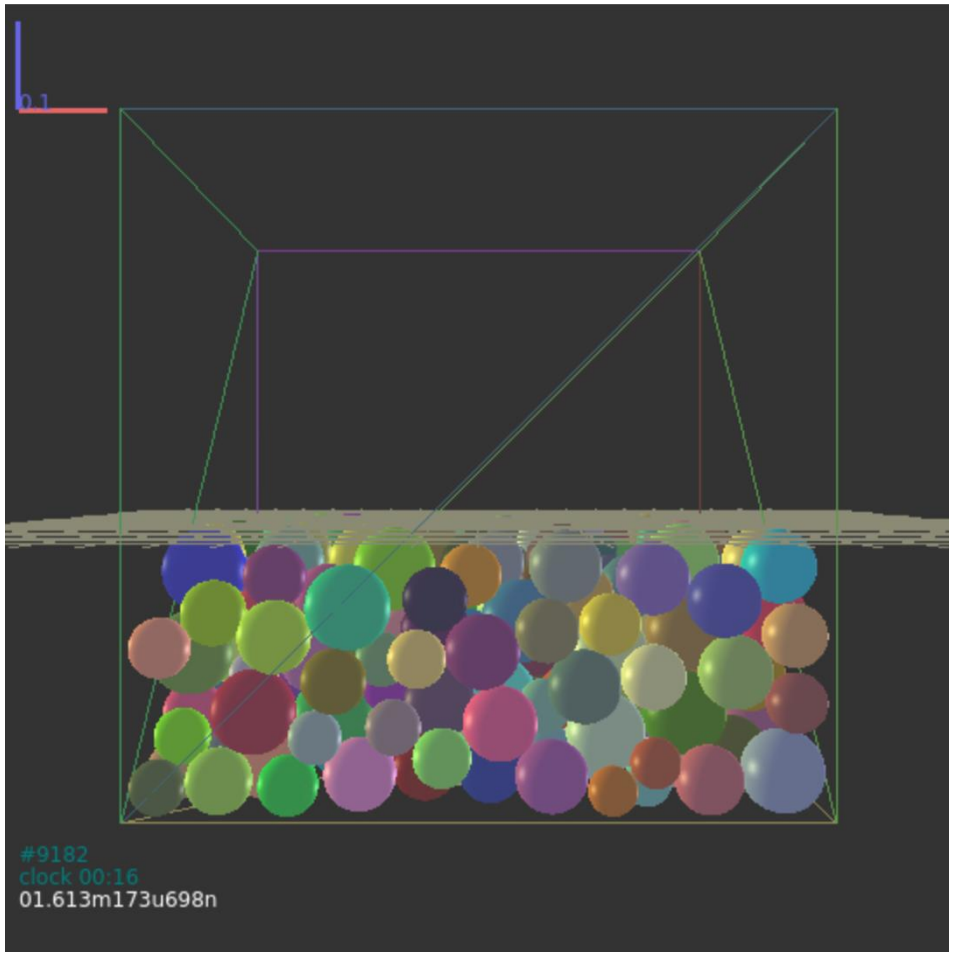
Now let's generate a granular packing inside an hourglass, and let it flow.

In this example, we will use the pack module of YADE:

- pack.ortho: simple cubic packing
- pack.hexa: hexagonal close packing
- pack.makeCloud: random loose packing

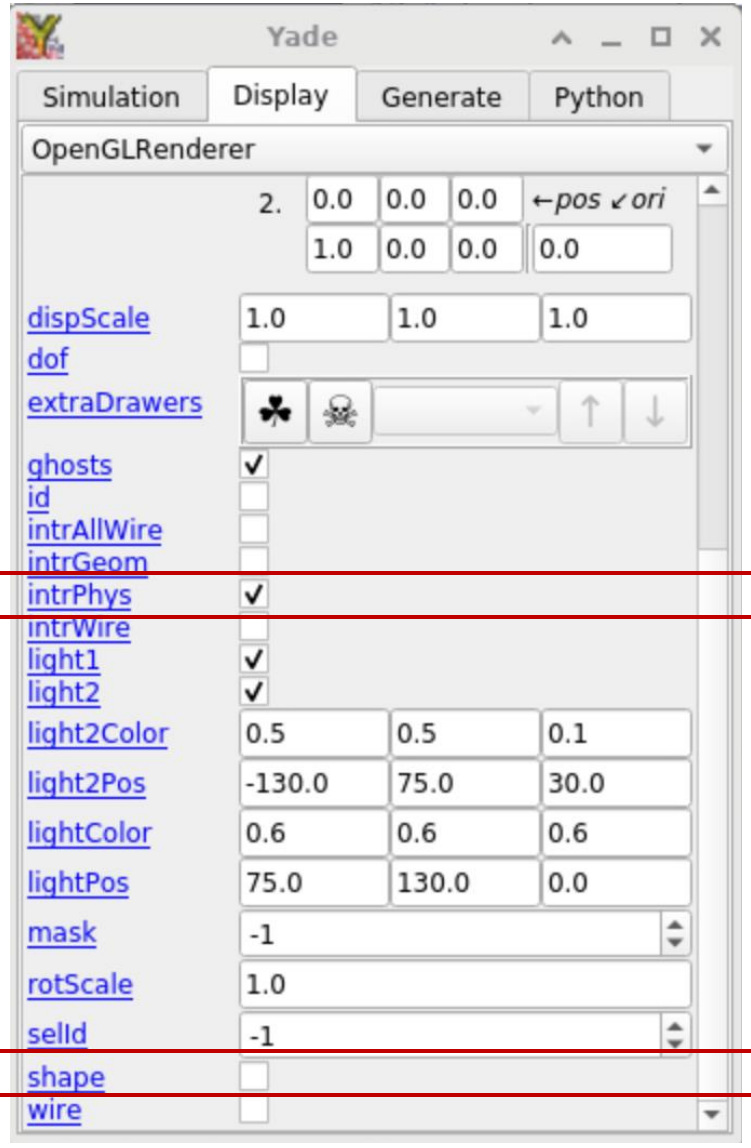
The interparticle friction is set to zero. Increase the friction value. What do you observe?

Tutorial 4 - Oedometer (Uniaxial compression) test

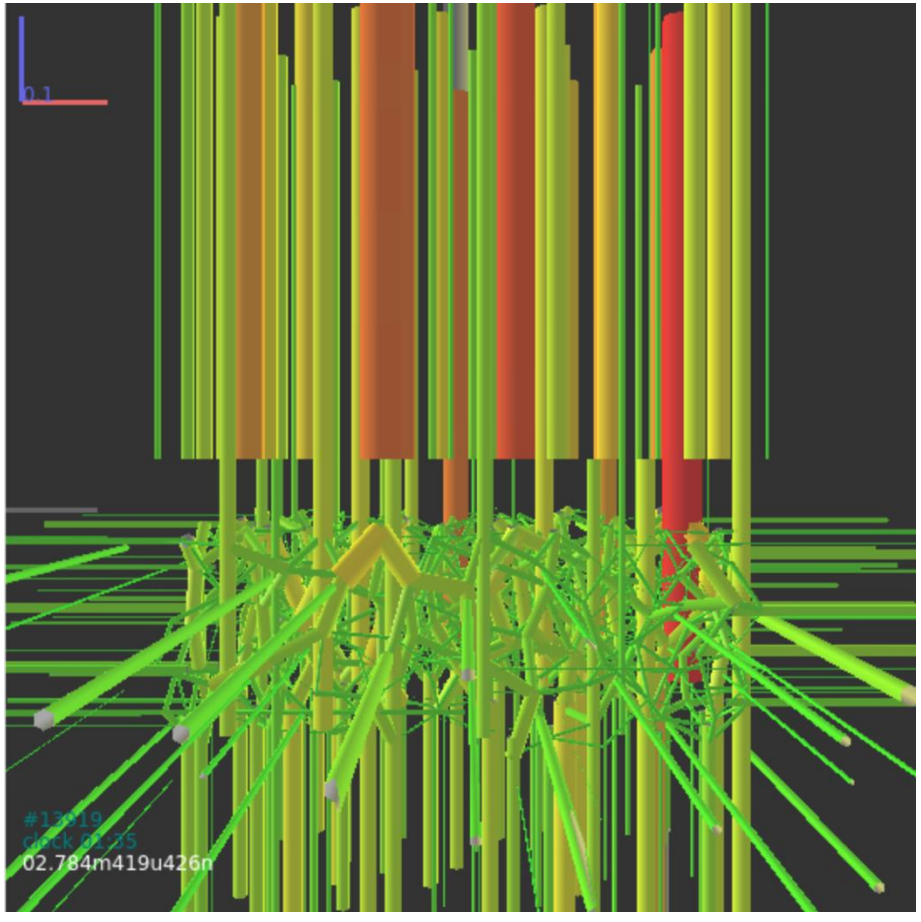


- In this tutorial we will generate a granular packing inside a facetBox(), and then we will compress it uniaxially.
- We make use of the pack module.
- We start loading once the unbalancedForce() ratio is low, i.e. when the system is relaxed.
- We can monitor the porosity() of the system in real time.
- We can monitor the fabricTensor() of the system in real time.
- Run this in parallel with yade –j4 oedometer.py using OpenMP.

Tutorial 4 – Visualise the contact forces



- Go to the Display tab of the qt.Controller() and tick intrPhys to visualise the normal contact forces. Untick shape to hide the particles.



Tutorial 4 – Oedometer (Uniaxial compression) test

- Batch simulation capacity: YADE can read input parameters from a .table file, and then run multiple simulations in batch mode → Parametric studies.
- Run:

```
yade-batch -job--threads=1 oedometer.table oedometer.py
```

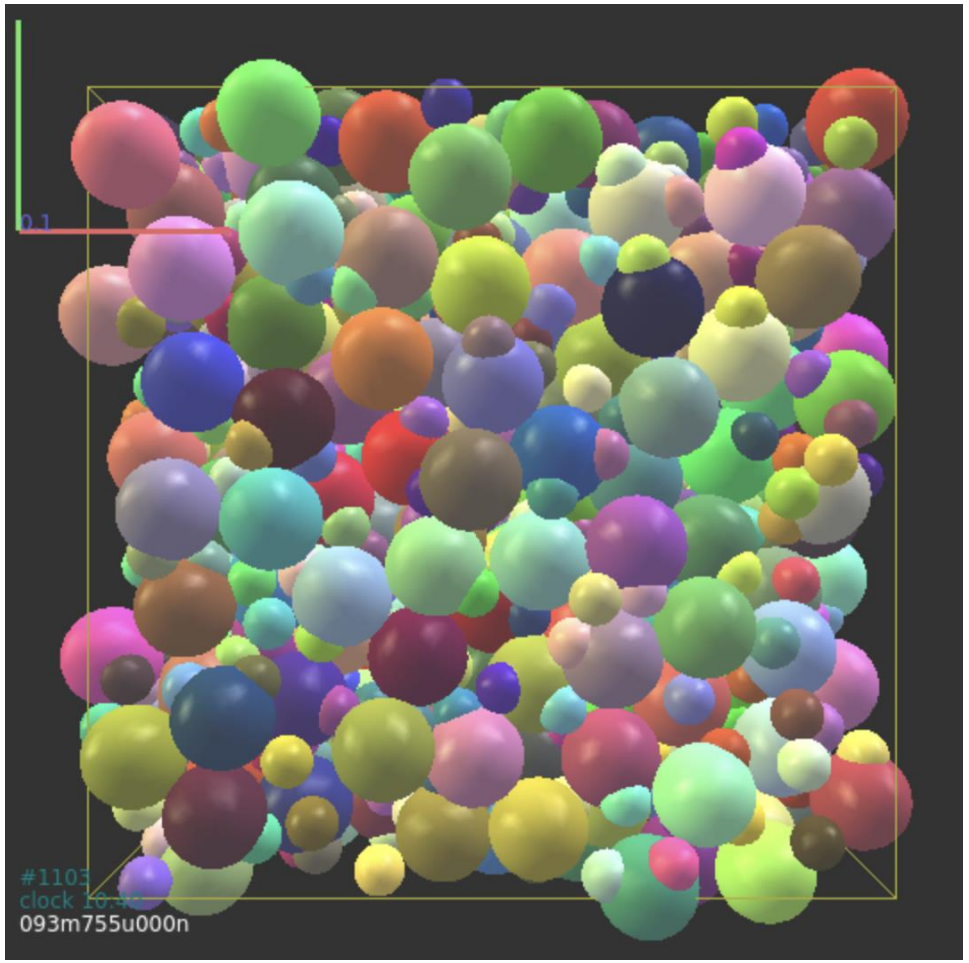
In the **oedometer.py** file:

```
15 readParamsFromTable(rMean=.05, maxLoad=1e5, minLoad=1e4)
```

In the **oedometer.table** file:

```
1 rMean maxLoad
2 .04 1e5
3 .05 1e5
4 .06 1e5
```


Tutorial 5 - Triaxial compression test



- In this tutorial we will generate a granular packing inside periodic space, we will compress it isotropically and then we will compress it deviatorically.
- We start the deviatoring loading once the unbalancedForce() ratio is low, i.e. when the system is relaxed.
- We can monitor the porosity() of the system in real time.
- We can monitor the coordination numbers of the system in real time.
- We can measure the average stressTensor of the system in real time.

Tutorial 5 - Triaxial compression test

Average stress tensor:

$$\sigma = \frac{1}{V}\sum_{i=1}^{N} F_i \otimes x_i$$

Where **V** the volume of the periodic box, $\mathbf{F_i}$ the contact forces, and $\mathbf{x_i}$ the branch vector connecting the centroid of each particle to the contact point.

getStress()

Coordination numbers: Measures of the connectivity among particles, i.e. how many contacts in average do the particles of the system have with their neighbours. More contacts per particle (e.g. 5-6) hint to a dense, more stable configuration. Fewer contacts (i.e. <4-5) hint to a loose, less stable configuration. Two coordination numbers exist, the average coordination number and the mechanical coordination number:

Average coordination number:

$$Z = 2C/N$$

Mechanical coordination number:

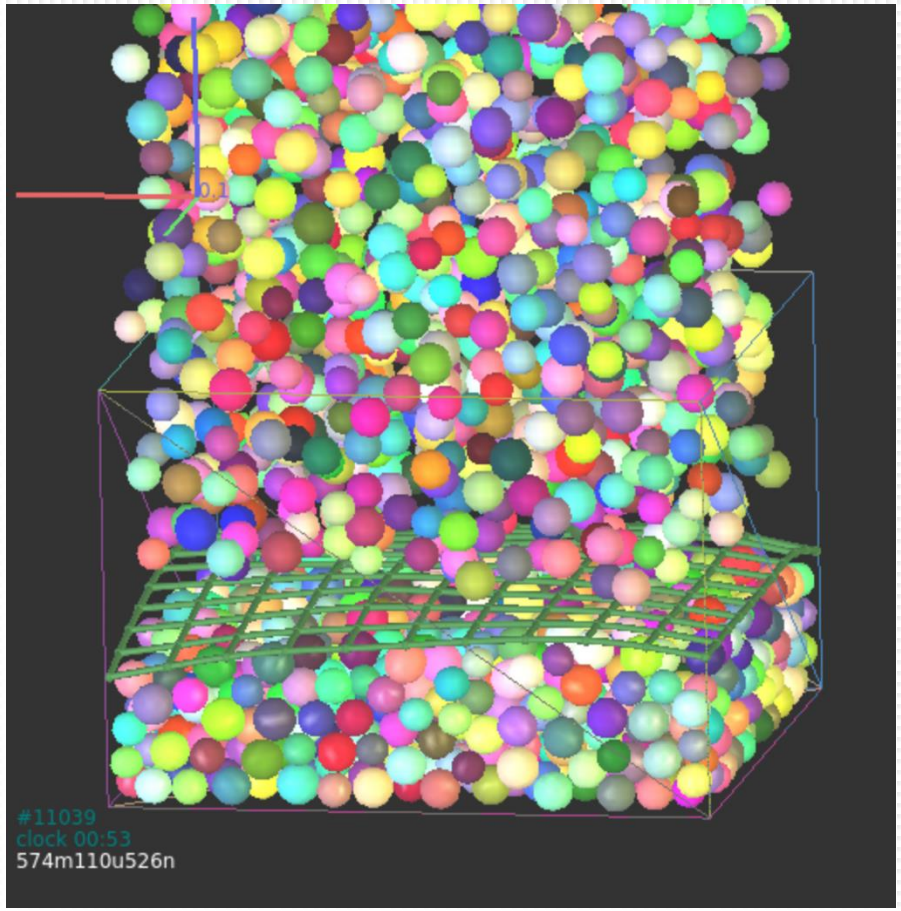
$$Z_m = \frac{(2C - N_1)}{(N - N_0 - N_1)}$$

avgNumInteractions(considerClumps=True)

(Thornton, 2015)

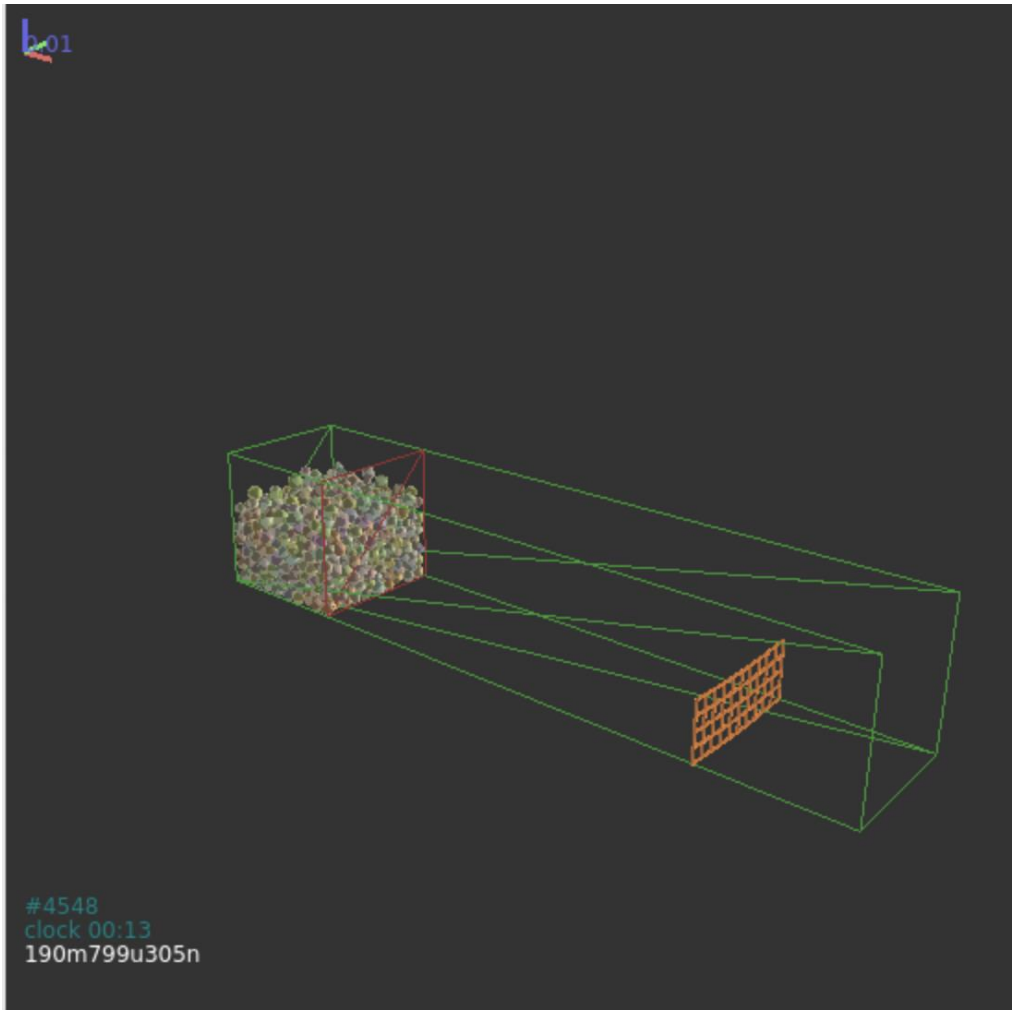
where C the number of contacts and N the number of particles, N_1 and N_0 are the number of particles with only one or no contacts respectively.

Tutorial 6 - Pull out of geogrid from granular sample



- This tutorial shows a pull out test on a geogrid embedded in a granular material.
- Flexible structures are modelled using spherocylinders and deformable joints, using the GridConnection and GridNode shapes in YADE, respectively.
- Planar, membrane-like elements also exist, using the PFacet class.

Tutorial 7 - Flow on flexible rockfall protection nets



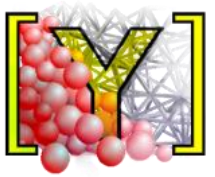
- This tutorial shows the impact of a granular flow of debris on a flexible net.
- Flexible structures are modelled using spherocylinders and deformable joints, using the GridConnection and GridNode shapes in YADE, respectively.
- Planar, membrane-like elements also exist, using the PFacet class.

In case you have time to spare..

- Hundreds of more examples are available in the examples folder:

  https://gitlab.com/yade-dev/trunk/-/tree/master/examples

- Try running the following examples:
  - YADE/examples/bulldozer/bulldozer.py
  - YADE/examples/clumps/clump-hopper-viscoelastic.py
  - YADE/examples/pfacet/pFacets_grids_spheres_interacting.py
  - YADE/examples/PotentialBlocks/Jenga.py



Reflection – What did I learn today?

- Today I (hopefully) learnt:
  - That Python is fun. ☺
  - How to define materials, particles, interaction properties in YADE.
  - How to import boundaries from stl files and in-house functions.
  - How to run simulations with Periodic Boundary Conditions in YADE.
  - How to access particle properties (e.g. position, velocity) in real time.
  - How to access interaction properties (e.g. normal forces) in real time.
  - How to live-plot any parameter in a YADE simulation.
  - How to calculate the average stress tensor.
  - How to calculate the porosity, coordination number, and fabric tensor.
  - How to run a parallelised YADE simulation using OpenMP.
  - How to visualise YADE results in Paraview.

https://yade-dem.org

Thanks for listening.

Vasileios Angelidakis