18th International Conference
ENGINEERING MECHANICS 2012
Svratka, Czech Republic, May 14 – 17, 2012

pp. 1237–1251
Paper #18

# OPEN SOURCE FEM-DEM COUPLING

**J. Stránský [*], M. Jirásek [**]**

**Abstract:** *Finite element method (FEM) and discrete element method (DEM) are leading strategies for numerical solution of engineering problems of solid phase. Both are applicable in different situations and sometimes can be beneficially coupled. Coupling of two free open source programs (finite element code OOFEM and discrete element code YADE, both with C++ core and Python user interface) is presented. Some of the basic coupling strategies (surface coupling, volume coupling, multi-scale approach and contact analysis) are explained on patch tests and simple simulations.*

**Keywords:** ***FEM, DEM, coupling, Python scripting, open source***

## 1. Introduction

Numerical simulations are an indispensible part of the current engineering and science development. For different engineering areas there are different numerical methods used. In solid phase mechanics, the leading methods are the finite element method (FEM) and the discrete (distinct) element method (DEM). FEM is rigorously derived from the continuum theory and is being used for the description of deformable continuous bodies, while DEM describes particulate materials, usually modeled by perfectly rigid particles and their interactions determined from fictitious overlaps of these rigid particles.

Often an engineering problem can be modeled using only one of the aforementioned methods. A steel beam would be simulated by FEM, a small assembly of gravel particles by DEM. But what if we wanted to simulate an impact of the steel bar on the gravel? One possible approach would be to split the problem into two domains (the steel part modeled by FEM and the gravel part modeled by DEM) and appropriately *couple* them.

There are countless software programs for both FEM and DEM. Some of them are commercial (usually) without possibility to change the code and adjust the behavior to our requirements (combination with another software for instance). However, there exist programs with open source code, which the user can modify, possibly for coupling with another programs. In the present article, coupling of FEM code OOFEM (Patzák & Bittnar, 2001) and DEM code YADE (Šmilauer et al., 2010) is presented. Both programs have the core written in C++ (providing efficient execution of time consuming routines), user interface written in Python (modern dynamic object oriented scripting language, providing easy to use scripting while preserving the C++ efficiency) and extensible object oriented architecture allowing independent implementation of new features - new material model or new particle shapes for instance.

Basic principles of different coupling strategies (surface, volume, multiscale and contact coupling) are explained in section 2, implementation issues are covered in section 3 and specific examples for each coupling strategy are presented in section 4. Python scripts controlling these examples are for illustration placed in Appendix A.

## 2. Theory

In this section, firstly the two numerical methods are quickly reviewed for the simplest case of small strain/displacement linear elasticity (to establish consistent notation and to help readers familiar with

[*]Ing. Jan Stránský: Czech Technical University in Prague, Faculty of Civil Engineering, Department of Mechanics, Thákurova 7; 166 29, Prague; CZ, email: jan.stransky@fsv.cvut.cz
[**]Prof. Ing. Milan Jirásek, DrSc.: Czech Technical University in Prague, Faculty of Civil Engineering, Department of Mechanics, Thákurova 7; 166 29, Prague; CZ, email: milan.jirasek@fsv.cvut.cz

one method to understand the other one). Then the basic principles of the chosen coupling strategies are explained.

### 2.1. Finite element method

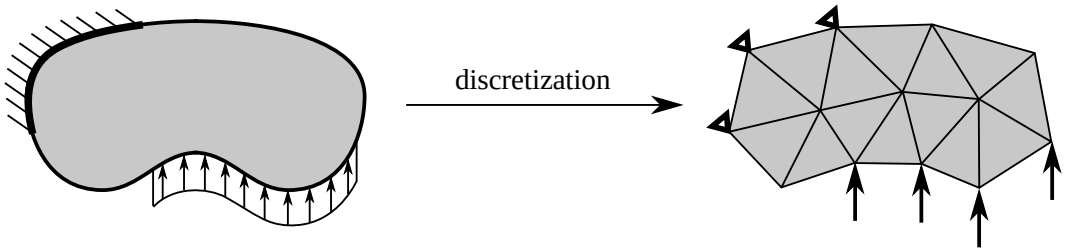



*Fig. 1: Simplified illustration of FEM discretization*

The continuous (static) linear elasticity solves the boundary value problem

$$\boldsymbol{\varepsilon} = \boldsymbol{\nabla}^{\mathrm{sym}}\mathbf{u} \qquad \mathbf{x} \in \Omega \tag{1}$$

$$\boldsymbol{\sigma} = {}^4\mathbf{D} : \boldsymbol{\varepsilon} \qquad \mathbf{x} \in \Omega \tag{2}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \mathbf{b} = \mathbf{0} \qquad \mathbf{x} \in \Omega \tag{3}$$

$$\mathbf{u} = \overline{\mathbf{u}} \qquad \mathbf{x} \in \Gamma_u \tag{4}$$

$$\mathbf{n} \cdot \boldsymbol{\sigma} = \overline{\mathbf{t}} \qquad \mathbf{x} \in \Gamma_t \tag{5}$$

where $\mathbf{x}$ = vector of Cartesian coordinates, $\mathbf{u} = \mathbf{u}(\mathbf{x})$ = displacement vector, $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}(\mathbf{x})$ = strain tensor, $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{x})$ = stress tensor, ${}^4\mathbf{D} = {}^4\mathbf{D}(\mathbf{x})$ = fourth-order stiffness tensor, $\mathbf{b} = \mathbf{b}(\mathbf{x})$ = body forces, $\mathbf{t} = \mathbf{t}(\mathbf{x})$ = surface tractions, $\Omega$ = domain of the problem, $\Gamma_u$ = boundary of the domain with prescribed displacements, $\Gamma_t$ = boundary of the domain with prescribed tractions and $\mathbf{n} = \mathbf{n}(\mathbf{x})$ = outer unit normal vector of the boundary. $\boldsymbol{\nabla}$ is the vector of spatial partial derivatives, $\cdot$ is contraction and : is double contraction (tensor operations). Overbars indicate prescribed values.

Combining geometric equations (1), constitutive equations (2) and equilibrium (Cauchy) equations (3) yields Lamé equations

$$\boldsymbol{\nabla} \cdot \left[ {}^4\mathbf{D} : (\boldsymbol{\nabla}\mathbf{u}) \right] + \mathbf{b} = \mathbf{0}, \tag{6}$$

which describe the elasticity problem in the strong sense.

The basic idea of FEM is to discretize the domain $\Omega$ into finite elements. On each element $e$, the displacement field $\mathbf{u}^e(\mathbf{x})$ is approximated as a combination of nodal displacement values $\{\mathbf{d}\}^e$. From the element geometry and material parameters, the element stiffness matrix $[\mathbf{K}]^e$ can be constructed, providing the relationship between the nodal displacements $\{\mathbf{d}\}^e$ and nodal internal forces $\{\mathbf{f}\}^e$ (7). Approximating the load by nodal forces, the global system of equations

$$[\mathbf{K}]^e\{\mathbf{d}\}^e = \{\mathbf{f}\}^e \quad \rightarrow \quad [\mathbf{K}]^g\{\mathbf{d}\}^g = \{\mathbf{f}\}^g \tag{7}$$

is assembled and unknown nodal displacements are solved for defined boundary conditions. Equation (7) can be derived from the virtual work principle and describes the elasticity problem in the weak sense.

### 2.2. Discrete element method

Consider an assembly of rigid particles. The contact (bond, interaction etc.) $c$ between particles is created either initially in the beginning of the simulation or when the particles overlap. The contact (see equations (8-9) and figure 2) can be described by a branch vector $\mathbf{l}^c$ (connecting the centers of linked particles) with normal $\mathbf{n}^c = \mathbf{l}^c / \|\mathbf{l}^c\|$. The (linearized) relative contact displacement $\mathbf{u}^c$ can be expressed in terms of particle displacement $\mathbf{x}^p$ and rotation $\boldsymbol{\varphi}^p$ and decomposed into the normal component $u_N^c$ and the shear component $\mathbf{u}_T^c$. Constitutive (linear elastic) contact law is also expressed (independently) in the normal and shear directions in terms of the normal and shear forces $f_N^c$ and $\mathbf{f}_T^c$ and normal and shear contact stiffnesses $k_N^c$ and $k_T^c$. The total contact force $\mathbf{f}^c$ is composed from normal and shear

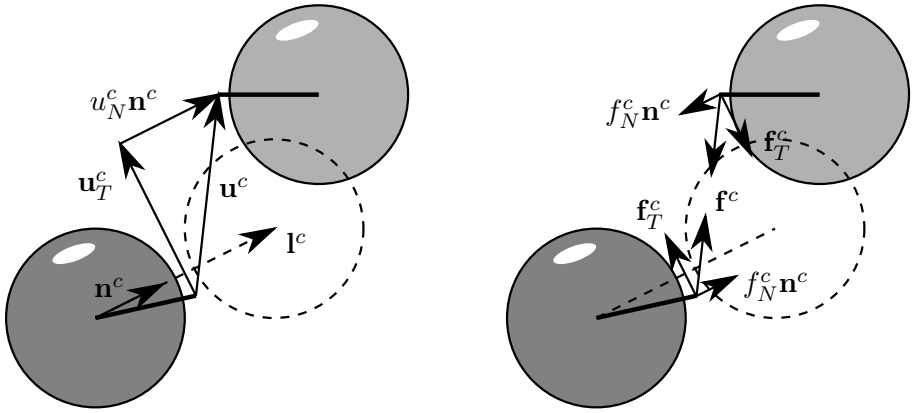



*Fig. 2: Simplified illustration of the contact displacement and contact forces and their decompositions*

components. The contact force is considered to act at the contact point (in the middle between particle centers for example), and so its shear component also causes a moment on both particles.

$$u_N^c = \mathbf{u}^c \cdot \mathbf{n}^c, \qquad \mathbf{u}^c = u_N^c \mathbf{n}^c + \mathbf{u}_T^c \tag{8}$$

$$f_N^c = k_N^c u_N^c, \qquad \mathbf{f}_T^c = k_T^c \mathbf{u}_T^c, \qquad \mathbf{f}^c = f_N^c \mathbf{n}^c + \mathbf{f}_T^c \tag{9}$$

For each particle $p$, the forces are then summed from all particles contacts and the Newton's equations of motion

$$\mathbf{f}^p = \sum_{c \in p} \mathbf{f}^c, \quad \ddot{\mathbf{u}}^p = \frac{\mathbf{f}^p}{m^p} \tag{10}$$

are solved using Verlet explicit time integration scheme. $m^p$ is the mass of particle $p$ and $\ddot{\mathbf{u}}^p$ its acceleration. The summation and integration of the equation of motion is also defined for moments and angular accelerations. For more details about time integration as well as DEM in general see (Šmilauer et al., 2010) or (Kuhl et. al, 2001).

### 2.3. Surface coupling

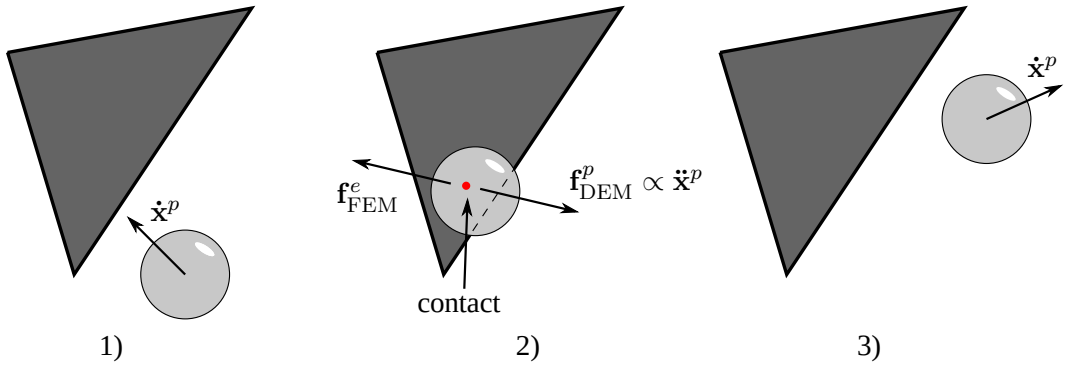



*Fig. 3: Illustration of FEM/DEM surface coupling*

The so-called surface coupling (Fakhimi, 2009; Nakashima & Oida, 2004; Oñate & Rojek, 2004; Villard et al., 2009) is probably the most straightforward coupling method. The principle is to split the whole problem into two non-overlapping domains, one modeled by FEM and the other by DEM (as already illustrated on the steel beam – gravel example in the introduction). As long as there is no overlap between the two domains, nothing special happens - both methods are applied independently.

If a contact between a finite element and a DEM particle is detected, the new force becomes acting on the DEM particle (causing its acceleration). The same force (with the same magnitude and the opposite direction) of course acts also on the FEM element and is processed as a load boundary condition, see figure 3.

In the current OOFEM/YADE implementation, the FEM boundary (surface) elements are copied into DEM part as special particles. This approach allows us to exploit efficient YADE contact detection algorithms. The resulting DEM force and moment acting on surface element are in FEM part transfered into nodal forces and further assumed as external load.

### 2.4. Volume coupling

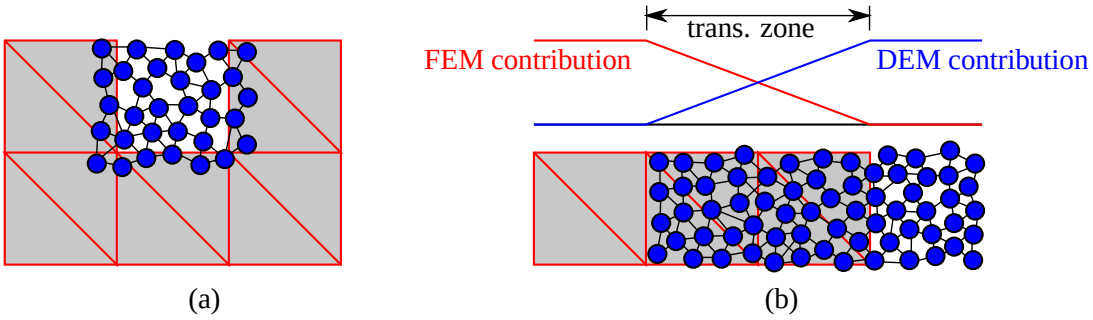



*Fig. 4: Illustration of "master/slave" (a) and "Arlequin" (b) FEM/DEM volume coupling*

Volume coupling is similar to the surface coupling. The difference is that the two subdomains overlap each other. The possible usage of this approach could be a model of concrete beam subjected to an impact load (blast for example). The whole beam would be modeled by FEM and only a small volume of the concrete (the volume to be fragmented and crushed) would be modeled by DEM.

There are two basic strategies how to model transition between FEM and DEM domains. The first one, "direct" or "master/slave" method, (Azvedo & Lemos, 2004) considers DEM particles overlapping with FEM as direct slaves of the FEM mesh (using standard "master/slave" or "hanging nodes" approach). The second one, the "Arlequin" method (Rousseau et al., 2009; Wellmann & Wriggers, 2012), considers a transition bridging zone, where the total response is superposed from contributions of the two models and is interpolated between both domains.

### 2.5. Multiscale coupling

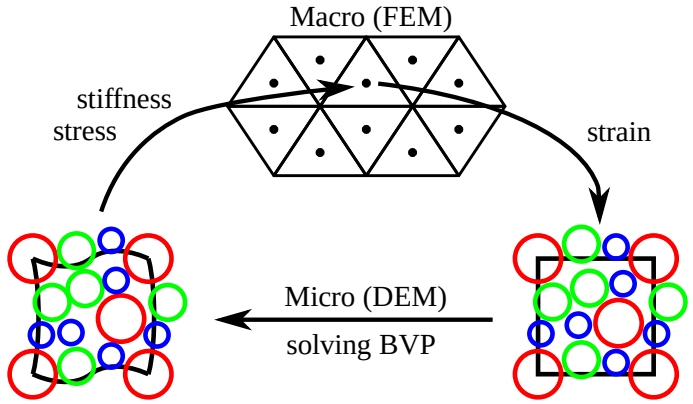



*Fig. 5: Illustration of multiscale coupling according to (Geers et. al, 2010) applied to FEM/DEM*

The idea of multiscale simulations is to model the problem on the large (macro) scale using information from a lower (micro) scale. In the current context, the (first order) homogenization (Geers et. al, 2010) is presented. Geometric information (strain) from macro scale (Gauss points of FEM mesh) is transfered to the micro scale (representative volume element - RVE - modeled by DEM), see figure 5. On the micro scale, the boundary value problem (BVP) governed by the transfered prescribed strain is solved using periodic boundary conditions (Stránský & Jirásek, 2010). The output of the micro-scale problem are the stress tensor and the constitutive characteristics (stiffness tensor), which are transfered back to the macro-scale problem.

As an example of this approach, consider a model of a large sample of sand. On macro scale, sand is usually considered as a continuous material. DEM modeling of a large sand sample with individual grains modeled by individual particles would not be feasible. According to these two facts, the macro scale problem is modeled by FEM. However, the particular nature of sand is preserved in the microscale RVE simulations, which provides the FEM part with stress and stiffness. Thus we do not need any explicit expression of the material law on the FEM scale (it is determined from the actual micro RVE response).

The stress and stiffness can be evaluated either analytically or numerically. The analytical formulas (Kuhl et. al, 2001) are inspired by the microplane theory and are derived from Voigt's hypothesis. It assumes that the strain $\boldsymbol{\varepsilon}$ is distributed uniformly in the RVE and, consequently, the relative displacement $\mathbf{u}^c$ of each contact $c$ can be expressed and decomposed in the form (see (Kuhl et. al, 2001) and equations (8-9) for more details):

$$\mathbf{u}^c = \boldsymbol{\varepsilon} \cdot \mathbf{l}^c = \|\mathbf{l}^c\| \, \boldsymbol{\varepsilon} \cdot \mathbf{n}^c = u_N^c \mathbf{n}^c + \mathbf{u}_T^c \tag{11}$$

$$u_N^c = \mathbf{u}^c \cdot \mathbf{n}^c = (\|\mathbf{l}^c\| \, \boldsymbol{\varepsilon} \cdot \mathbf{n}^c) \cdot \mathbf{n}^c = \|\mathbf{l}^c\| \, (\mathbf{n}^c \otimes \mathbf{n}^c) : \boldsymbol{\varepsilon} = \|\mathbf{l}^c\| \, \mathbf{N}^c : \boldsymbol{\varepsilon} \tag{12}$$

$$\mathbf{u}_T^c = \mathbf{u}^c - u_N^c \mathbf{n}^c = \|\mathbf{l}^c\| \, \boldsymbol{\varepsilon} \cdot \mathbf{n}^c - \|\mathbf{l}^c\| \, \boldsymbol{\varepsilon} : (\mathbf{n}^c \otimes \mathbf{n}^c)\mathbf{n}^c = \|\mathbf{l}^c\| \, \boldsymbol{\varepsilon} : (\mathbf{I}^{\mathrm{sym}} \cdot \mathbf{n} - \mathbf{n} \otimes \mathbf{n} \otimes \mathbf{n}) = \|\mathbf{l}^c\| \, \mathbf{T}^c : \boldsymbol{\varepsilon} \tag{13}$$

$\mathbf{N}^c = \mathbf{n}^c \otimes \mathbf{n}^c$ and $\mathbf{T}^c = \mathbf{I}^{\mathrm{sym}} \cdot \mathbf{n}^c - \mathbf{n}^c \otimes \mathbf{n}^c \otimes \mathbf{n}^c$ are auxiliary projection tensors and $\mathbf{I}^{\mathrm{sym}} = \frac{1}{2}[\delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}]$ is the fourth order symmetric identity tensor. The equivalence of macroscopic ($M$) and microscopic ($m$) virtual work

$$\boldsymbol{\sigma} : \delta\boldsymbol{\varepsilon} = \delta W^M \quad = \quad \delta W^m = \frac{1}{V} \sum_{c \in V} \mathbf{f}^c \cdot \delta \mathbf{u}^c = \frac{1}{V} \sum_{c \in V} \mathbf{f}^c \cdot \delta\boldsymbol{\varepsilon} \cdot \mathbf{l}^c = \frac{1}{V} \sum_{c \in V} [\mathbf{f}^c \otimes \mathbf{l}^c] : \delta\boldsymbol{\varepsilon} \tag{14}$$

yields the expression for the stress tensor (with substitution of equations (8) and (9))

$$\boldsymbol{\sigma} = \frac{1}{V} \sum_{c \in V} [\mathbf{f}^c \otimes \mathbf{l}^c]^{\mathrm{sym}} = \frac{1}{V} \sum_{c \in V} \|\mathbf{l}^c\| \, (\mathbf{N}^c f_N^c + [\mathbf{f}_T^c \otimes \mathbf{n}^c]^{\mathrm{sym}}). \tag{15}$$

Substituting equations (9), (12) and (13) into (15)

$$\begin{aligned} \boldsymbol{\sigma} &= \frac{1}{V} \sum_{c \in V} \|\mathbf{l}^c\| \, (\mathbf{N}^c k_N^c u_N^c + [\mathbf{n}^c \otimes k_T^c \mathbf{u}_T^c]^{\mathrm{sym}}) = \\ &= \frac{1}{V} \sum_{c \in V} \|\mathbf{l}^c\| \, (\mathbf{N}^c k_N^c \|\mathbf{l}^c\| \, \mathbf{N}^c : \boldsymbol{\varepsilon} + [\mathbf{n}^c \otimes k_T^c \|\mathbf{l}^c\| \, \mathbf{T}^c : \boldsymbol{\varepsilon}]^{\mathrm{sym}}) = \\ &= \frac{1}{V} \sum_{c \in V} \|\mathbf{l}^c\|^2 (k_N^c \mathbf{N}^c \otimes \mathbf{N}^c + k_T^c [\mathbf{n}^c \otimes \mathbf{T}^c]^{\mathrm{sym}}) : \boldsymbol{\varepsilon} \end{aligned} \tag{16}$$

and comparing (16) with (2) yields the expression for the stiffness tensor

$$\begin{aligned} {}^4\mathbf{D} &= \frac{1}{V} \sum_{c \in V} \|\mathbf{l}^c\|^2 (k_N^c \mathbf{N}^c \otimes \mathbf{N}^c + k_T^c [\mathbf{n}^c \otimes \mathbf{T}^c]^{\mathrm{sym}}) \\ D_{ijkl} &= \frac{1}{V} \sum_{c \in V} \|\mathbf{l}^c\|^2 \left[ (k_N^c - k_T^c) n_i^c n_j^c n_k^c n_l^c + k_T^c \frac{1}{4} \left( n_i^c n_l^c \delta_{jk} + n_i^c n_k^c \delta_{jl} + n_j^c n_l^c \delta_{ik} + n_j^c n_k^c \delta_{il} \right) \right]. \end{aligned} \tag{17}$$

This estimation of the stiffness tensor is derived from the kinematic constraint, thus it is an upper bound of the real one. If the real strain state differs too much from the assumed uniform state (strain localization for instance), the analytical stiffness estimation is no more valid and the stiffness has to be computed numerically. The aspect of strain localization can be in same cases captured by the implemented periodic boundary conditions, see (Šmilauer et al., 2010) and (Stránský & Jirásek, 2010), but it will be among other aspects (second order homogenization for instance) subjected to further analysis and development.

### 2.6. Contact analysis

The idea of contact analysis (Frenning, 2009) is very simple and opposite to the multiscale approach. The material on the large scale is considered to be of a particulate nature and is modeled by particles using DEM. Each such particle is further modeled by FEM.

There is no strict border between the cases when the solution can be considered as contact FEM analysis and when it is already DEM. For only a few particles we would probably use the former one, but when the number of particles is large, the DEM modeling (with its efficient contact detection algorithms) would be more convenient. This strategy can be actually considered as full FEM, only the contact detection is ”borrowed” from the DEM program.

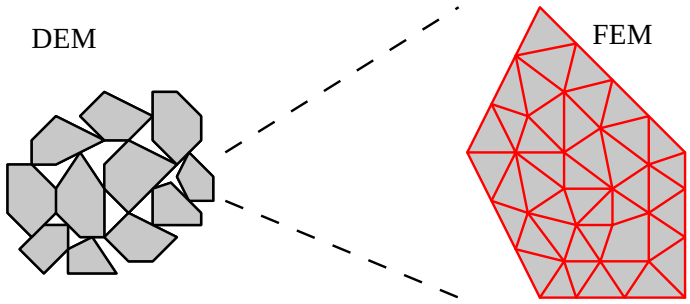



*Fig. 6: Illustration of multimethod FEM/DEM contact analysis*

## 3. Implementation

The current implementation serves only for testing of the methods and functionality, therefore not all features are in public versions of OOFEM and YADE yet. Since this paper is about open source coupling, the changes will be (after further testing and bug-fixing) of course merged to public versions and will be available for any potential user/developer.

From the point of view of Python, all important functionality is concentrated into the **`fakemupif`** module, which provides several classes derived from the base **`FemDemCoupler`** class. Each such class has its **`step`** method, which is the only needed additional command in comparison with non-coupled simulations (see the example scripts in the following sections). When the testing is finished, the functionality (probably with some syntax and internal code modifications, but with same high-level simplicity) will be implemented into the **`MuPIF`** project (Patzák, 2011) to enable coupling with other programs or other physical models (heat transfer for instance).

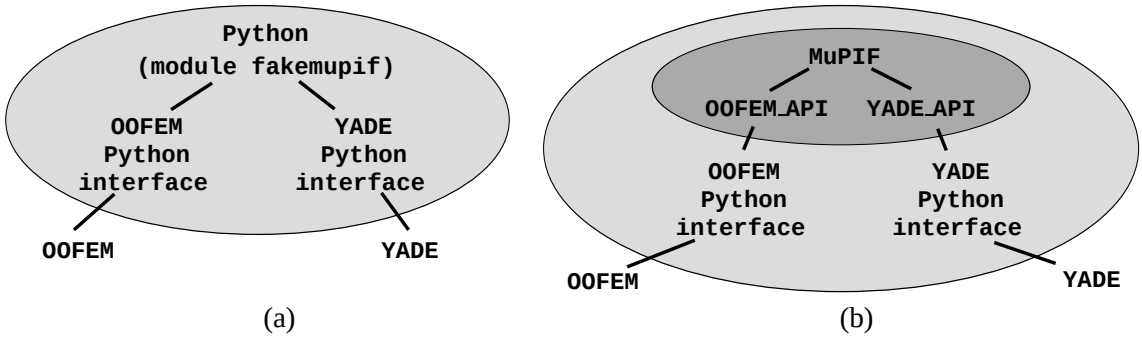
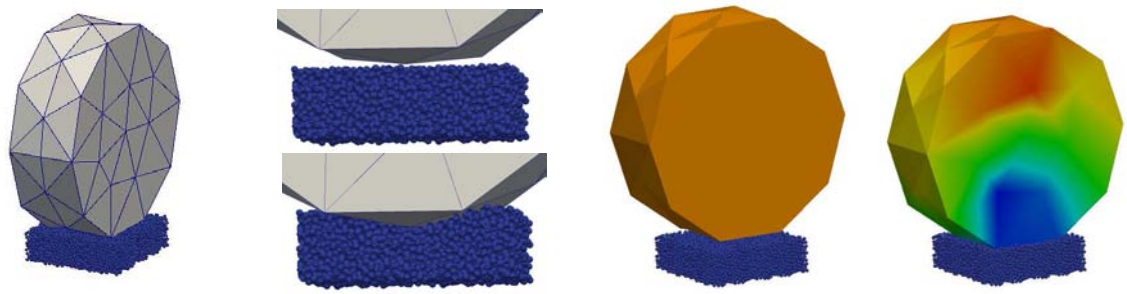



*Fig. 7: Current (testing) (a) and planned (b) implementation*

## 4. Examples

In this section, one specific simple example for each discussed coupling strategy is presented.



(a) (b) (c)

*Fig. 8: Results of surface coupling - soil-tire contact:*
*simulation setup (a), detail of contact (b) and resulting vertical normal stress in FEM domain (c)*

### 4.1. Surface coupling: soil-tire contact

Simulation of the soil-tire contact, inspired by (Nakashima & Oida, 2004), is presented here. The tire is modeled by FEM as a linear elastic material, the soil is modeled by DEM as spherical particles. Of course, in a real simulation we could use a more complicated soil grain geometry, a more advanced (than just linear elastic) material model for the tire and so on, but for an illustrating and testing example the present assumptions are sufficient. See figure 8 and codes 1 and 5.

### 4.2. Volume coupling: three point bending

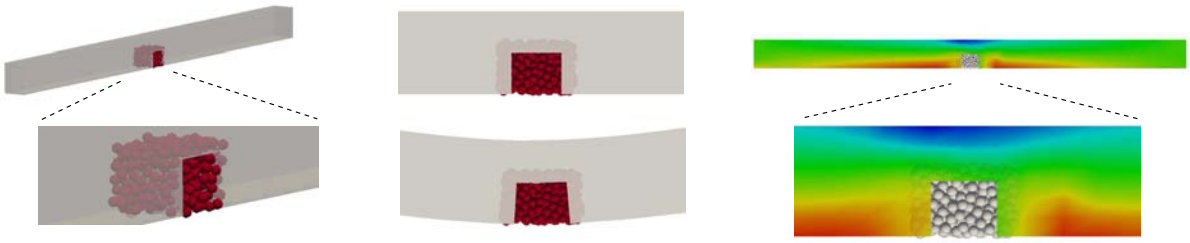


*Fig. 9: Results of three point bending:*
*initial state (a), undeformed and deformed final state (b) and normal stress (c)*

In this example, a simply supported beam was simulated. The whole beam body was simulated by FEM, only the part with maximal tensile normal stress was simulated by DEM. The vertical displacement of the middle cross section was prescribed.

Again, the elastic solution is nothing special here, but instead of linear elastic particles we could use a kind of material model for fracture description (Azvedo & Lemos, 2004), thus the expected crack initiation and propagation in the middle cross section would be modeled with the help of discrete models. See figure 9 and codes 2 and 6.

### 4.3. Multiscale coupling: uniaxial strain

Uniaxial strain (oedometric test) of a sample consisting of two different (linear elastic) materials (with stiffness ratio $1/2$) is simulated in this example. The macro-scale problem is modeled by two brick elements. Each FEM element has eight integration points. For each integration point, a DEM micro-scale RVE simulation is performed, in which the FEM prescribed strain is imposed. The resulting stress and stiffness are transfered back to the macro-scale FEM simulation.

In figure 10, magnified results are plotted. In each material, one micro RVE result is displayed (all RVEs in the same material, due to the simulation setup, correspond to each other). The results of linear elastic behavior are not extremely spectacular indeed, but using nonlinear behavior of RVEs (resulting

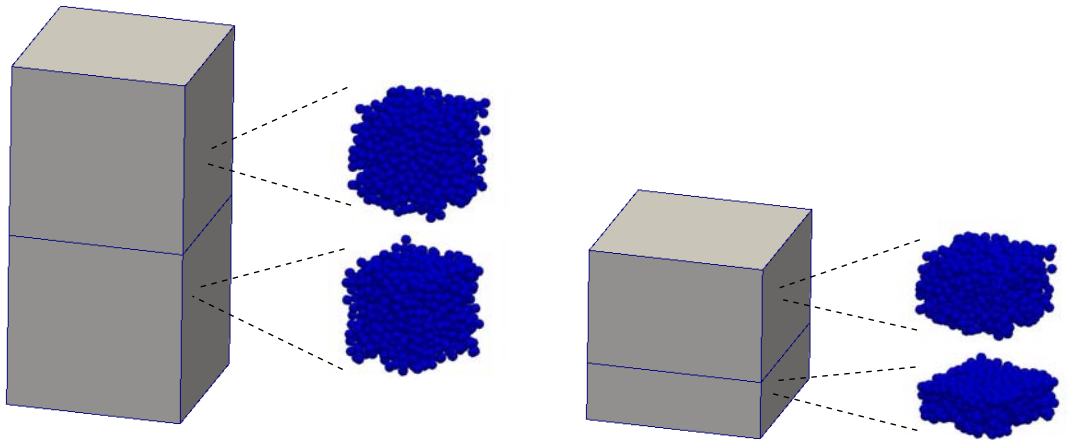


*Fig. 10: Results of multiscale uniaxial strain test*

in higher stiffness when more inter-particle contact occur for instance) could be very useful for certain applications.

See figure 10 and codes 3 and 7.

### 4.4. Contact analysis: cantilever shock analysis

In this example, the simulation of a cantilever shock is presented. The shock is caused by the fall of an impactor – another beam in our example – on the end of the cantilever. Both cantilever and the falling beam are modeled by FEM, only a detection algorithm is borrowed from DEM. This contact detection between particles of FEM elements shapes (tetrahedra or bricks) is a typical example of the code, which is not public yet and needs more testing to be committed to public version. See figure 11 and codes 4 and 8.

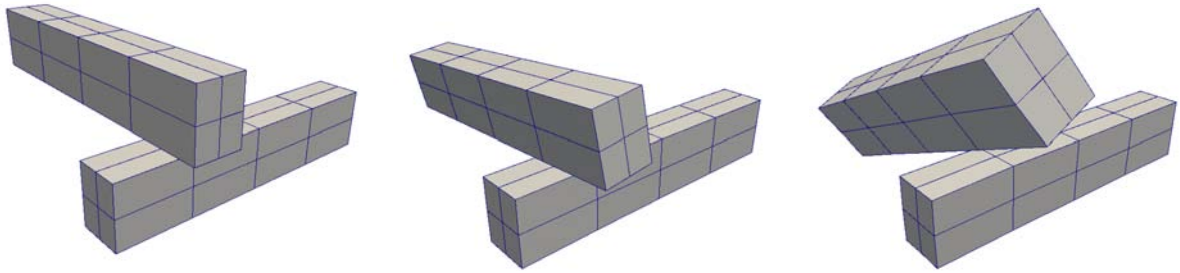


*Fig. 11: Results of cantilever shock analysis – different stages of impact*

## 5. Conclusions

The basic principles of the most popular FEM/DEM coupling strategies (surface, volume, multiscale and contact coupling) were presented, together with specific examples and corresponding Python scripts. All methods can be arbitrarily combined with each other or with different methods/programs (which uses Python user interface).

As an example, consider a dynamic soil compaction. The compacted soil would be definitely modeled by DEM, the compactor by FEM (here we have surface coupling) and the rest of the soil domain by FEM. The soil DEM / soil FEM interface would probably be of a volume coupling kind. Of course, the FEM soil could be modeled using the multiscale approach, and we could go in coupling further and further.

This example was just to show the variety of possible coupling combinations and that there are a lot of real world problems, where such coupled methods could be useful. Together with the simplicity of creating, modifying and running such simulations and extensibility of the used programs (due to the open source character of the code) it makes this approach attractive for a variety of engineering problems.

Future work on this topic will address (among others) second order DEM homogenization, adjustment of DEM periodic boundary conditions for arbitrary localization analysis, implementation and testing of Arlequin volume coupling method, implementation of contact detection algorithms of FEM element shaped particles and implementation of testing **`fakemupif`** interface into **`MuPIF`** framework.

## Acknowledgments

Financial support of the Czech Technical University in Prague under project SGS12/027/OHK1/1T/11 is gratefully acknowledged.

## References

Azvedo, N. M. & Lemos, L. V. (2006) Hybrid discrete element/finite element method for fracture analysis. *Computer Methods in Applied Mechanics and Engineering*, 195, pp. 4579–4593.

Fakhimi, A. (2009) A hybrid discretefinite element model for numerical simulation of geomaterials. *Computers and Geotechnics*, 36, pp. 386–395.

Frenning, D. (2008) An efficient finite/discrete element procedure for simulating compression of 3D particle assemblies. *Computer Methods in Applied Mechanics and Engineering*, 197, pp. 4266–4272.

Geers, M. G. D., Kouznetsova, Brekelmans, W. A. M. (2010) Multi-scale computational homogenization: Trends and challenges. *Journal of Computational and Applied Mathematics*, 234, pp. 2175–2182.

Kuhl, E., DAddetta, G. A., Leukart, M. and Ramm, E. (2001) Microplane modelling and particle modelling of cohesive-frictional materials. In: *Continuous and Discontinuous Modelling of Cohesive-Frictional Materials* (P. A. Veemer et al. eds). Springer, Berlin, pp. 31–46

Nakashima, H. & Oida, A. (2004) Algorithm and implementation of soiltire contact analysis code based on dynamic FEDE method. *Journal of Terramechanics*, 41, pp. 127–137.

Oñate, E. & Rojek, J. (2004) Combination of discrete element and finite element methods for dynamic analysis of geomechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 193, pp. 3087–3128.

Patzák, B. & Bittnar, Z. (2001) Design of object oriented finite element code. *Advances in Engineering Software*, 32, pp. 759–767.

Patzák, B. (2011) MuPIF: A Distributed Multi-Physics Integration Tool. In: *Proceedings of the Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering* (P. Ivnyi & B. H. V. Topping eds.). Stirlingshire, United Kingdom, paper 15.

Rousseau, J., Frangin, E., Marin, P. & Daudeville, L. (2009) Multidomain finite and discrete elements method for impact analysis of a concrete structure. *Engineering Structures*, 31, pp. 2735–2743.

Stránský, J. & Jirásek, M. (2010) Calibration of particle-based models using cells with periodic boundary conditions. In: *Proc. II International Conference on Particle-based Methods - Fundamentals and Applications* (E. Oñate and D.R.J. Owen eds.), CIMNE, Barcelona, pp. 274–285.

Šmilauer, V., Catalano, E., Chareyre, B., Dorofeenko, S., Duriez, J., Gladky, A., Kozicki, J., Modenese, C., Scholtès, L., Sibille, L., Stránský, J. & Thoeni, K. (2010) *Yade Documentation* (V. Šmilauer ed.), The Yade Project, 1st ed. http://yade-dem.org/doc/.

Villard, P., Chevalier, B., Le Hello, B. & Combe, G. (2009) Coupling between finite and discrete element methods for the modelling of earth structures reinforced by geosynthetic. *Computers and Geotechnics*, 36, pp. 709–717.

Wellmann, C. & Wriggers, P. (2012) A two-scale model of granular materials. *Computer Methods in Applied Mechanics and Engineering*, 205–208, pp. 46–58.

Xu, M., Gracie, R. & Belytschko, T. (2002) Multiscale Modeling with Extended Bridging Domain Method. In: *Bridging the Scales in Science and Engineering* (J. Fish ed.). Oxford University Press.

## Appendix A. Example scripts and input files

Python scripts controlling examples in section 4 are presented in this section.

Currently, OOFEM simulations use input text files, while YADE constructs the simulation directly in a Python script. Therefore the OOFEM input files (or a significant parts of them) are followed by the actual Python scripts.

Since the **`fakemupif`** module will be changed and adjusted to **`MuPIF`** requirements, the actual code **`fakemupif`** is not presented. However, the simplicity of **`coupler.step()`** line (from the user point of view representing the entire coupling process) should be preserved.

```
test_surface.oofem.out
surface coupling test - tire-soil contact
LinearStatic nsteps 1000 nmodules 1
vtk 1 tstep_step 20 domain_all vars 2 1 4 primvars 1 1
domain 3d
OutputManager tstep_all dofman_all element_all
ndofman 64 nelem 192 ncrosssect 1 nmat 1 nbc 2 nic 0 nltf 2
node 1 coords 3 -3.000000e-01 0.000000e+00 -1.000000e-01 bc 3 0 0 0
...
node 34 coords 3 0.000000e+00 0.000000e+00 -1.000000e-01 bc 3 1 2 1
...
node 64 coords 3 -1.307587e-01 1.822355e-01 -4.467372e-02 bc 3 0 0 0
ltrspace 1 nodes 4 32 40 42 45 crosssect 1 mat 1
...
ltrspace 192 nodes 4 38 32 55 46 crosssect 1 mat 1
SimpleCS 1 thick 1.0 width 1.0
IsoLE 1 d 0. E 100e9 n 0.4 talpha 0.0
BoundaryCondition 1 loadTimeFunction 1 prescribedvalue 0.0
BoundaryCondition 2 loadTimeFunction 2 prescribedvalue 1.0
ConstantFunction 1 f(t) 1.0
PiecewiseLinFunction 2 npoints 3 t 3 0 500 1000 f(t) 3 0 0.01 0
```

*Code 1: test_surface.oofem.in*

```
test_volume.oofem.out
volume coupling test - hanging nodes - three point bending
NonLinearStatic nsteps 50 nmodules 1
vtk 1 tstep_step 1 domain_all vars 2 1 4 primvars 1 1
domain 3d
OutputManager tstep_all dofman_all element_all
ndofman 2478 nelem 1764 ncrosssect 1 nmat 1 nbc 2 nic 0 nltf 1
node 1 coords 3 0.000000e+00 0.000000e+00 0.000000e+00 bc 3 0 1 0
...
node 2478 coords 3 6.000000e+00 3.000000e-01 4.000000e-01 bc 3 0 1 1
lspace 1 nodes 8 2 9 58 51 1 8 57 50 crosssect 1 mat 1
...
lspace 1800 nodes 8 2422 ... 2470 crosssect 1 mat 1
SimpleCS 1 thick 1.0 width 1.0
IsoLE 1 d 0. E 40e9 n 0.2 talpha 0.0
BoundaryCondition 1 loadTimeFunction 1 prescribedvalue 0.0
BoundaryCondition 2 loadTimeFunction 1 prescribedvalue 1e-2
PiecewiseLinFunction 1 npoints 2 t 2 0 50 f(t) 2 0. 1.
```

*Code 2: test_volume.oofem.in*

```
test_multi.oofem.out
multiscale coupling test
NonLinearStatic nsteps 50 nmodules 1
vtk 1 tstep_step 1 domain_all vars 2 1 4 primvars 1 1
domain 3d
OutputManager tstep_all dofman_all element_all
ndofman 12 nelem 2 ncrosssect 1 nmat 2 nbc 2 nic 0 nltf 2
node 1  coords 3 0.0 0.0 0.1 bc 3 1 1 1
...
lspace 1 nodes 8 1 2 3 4  5 6 7  8  crosssect 1 mat 1
lspace 2 nodes 8 4 3 9 10 8 7 11 12 crosssect 1 mat 2
SimpleCS 1 thick 1.0 width 1.0
MultiScaleSMMat 1 d 0. E 40e6 n 0.2 talpha 0.0
MultiScaleSMMat 2 d 0. E 80e6 n 0.2 talpha 0.0
BoundaryCondition 1 loadTimeFunction 2 prescribedvalue 0.0
BoundaryCondition 2 loadTimeFunction 2 prescribedvalue -1e-2
PiecewiseLinFunction 1 npoints 2 t 2 0 50 f(t) 2 0. 1.
ConstantFunction 2 f(t) 1.
```

*Code 3: test_multi.oofem.in*

```
test_contact.oofem.out
contact coupling test - cantilever inpact
DEIDynamic nsteps 12000 nmodules 1 dumpcoef 0 deltaT 1e-4
vtk 1 tstep_step 30 domain_all vars 2 1 4 primvars 1 1
domain 3d
OutputManager tstep_all dofman_all element_all
ndofman 90 nelem 32 ncrosssect 1 nmat 1 nbc 2 nic 0 nltf 1
node 1 coords 3 0.000000e+00 0.000000e+00 0.000000e+00  bc 3 1 1 1
...
node 90 coords 3 1.723500e+00 1.744000e+00 -1.000000e-01
lspace 1 nodes 8 2 5 14 11 1 4 13 10 crosssect 1 mat 1 bodyloads 1 2
...
lspace 32 nodes 8 72 75 90 87 71 74 89 86 crosssect 1 mat 1 bodyloads 1 2
SimpleCS 1 thick 1.0 width 1
IsoLE 1 d 1000. E 1e8 n 0.2 talpha 0.0
BoundaryCondition 1 loadTimeFunction 1 prescribedvalue 0.0
deadweight 2 loadTimeFunction 1 components 3 0 0 10
ConstantFunction 1 f(t) 1.
```

*Code 4: test_contact.oofem.in*

```
# example script for surface coupling - soil-tire contact
# first import required modules
import fakemupif
from fakemupif import oofem,yade

nSteps = 1000
output = 50
# then instanciate FEM ...
oofemFile = 'test_surface.oofem.in'
fem = fakemupif.instanciateOofemProblem(oofemFile)
dem = yade.Omega()

# ... as well as DEM problem
dem.materials.append(fakemupif.defaultMat)
c1,c2 = (0,-.015,-.008), (.03,-.004,.017)
c1,c2 = (-.15,-.4,-.15), (.15,-.3,.15)
rect = yade.pack.inAlignedBox(c1,c2)
rad = .005
sphs = yade.pack.randomDensePack(rect,rad,spheresInCell=1000)
dem.bodies.append([sph for sph in sphs])

# DEM BCs
bcw = 3*rad
for b in dem.bodies:
   p = b.state.pos
   if not (p[0]<c1[0]+bcw or p[0]>c2[0]-bcw): continue
   if not (p[1]<c1[1]+bcw or p[2]<c1[2]+bcw or p[2]>c2[2]-bcw): continue
   b.state.blockedDOFs = 'xyzXYZ'

#
coupler = fakemupif.SurfaceFemDemCoupler(fem,dem,oofemFile)
dem.bodies.append([facet for facet in coupler.facets])
#

dem.engines=[
   yade.ForceResetter(),
   yade.InsertionSortCollider([
      yade.Bo1_Sphere_Aabb(),
      yade.Bo1_Facet_Aabb()]),
   yade.InteractionLoop(
      [yade.Ig2_Sphere_Sphere_Dem3DofGeom(),
         yade.Ig2_Facet_Sphere_Dem3DofGeom()],
      [yade.Ip2_CpmMat_CpmMat_CpmPhys()],
      [yade.Law2_Dem3DofGeom_CpmPhys_Cpm()]
   ),
   yade.NewtonIntegrator(),
   yade.PyRunner(command='vtk.exportSpheres(); vtk.exportFacets()',\
      iterPeriod=max(1,nSteps/output)),
]
dem.dt = yade.utils.PWaveTimeStep()/2.

# Yade vtk export
import yade.export
vtk = yade.export.VTKExporter('test_surface.yade',startSnap=1)

# run
for i in xrange(nSteps):
   coupler.step()

# exit
print 'Finished!'
fem.terminateAnalysis()
dem.exitNoBacktrace()
```

*Code 5: controlling script for soil-tire contact simulation*

```
# example script for volume coupling - three point ending
# first import required modules
import fakemupif
from fakemupif import oofem,yade

# then instanciate FEM ...
oofemFile = 'test_volume.oofem.in'
fem = fakemupif.instanciateOofemProblem(oofemFile)

# ... as well as DEM problem
dem = yade.Omega()
dem.materials.append(fakemupif.defaultMat)
rect = yade.pack.inAlignedBox((2.8,0.0,0.12),(3.2,0.3,0.4))
sphs = yade.pack.randomDensePack(rect,0.02,spheresInCell=1000)
dem.bodies.append(sphs)

#
coupler = fakemupif.VolumeFemDemCoupler(fem,dem,oofemFile)
#

dem.dt = .5*yade.utils.SpherePWaveTimeStep(.02,1000,25e9)
dem.engines=[
   yade.ForceResetter(),
   yade.InsertionSortCollider([
      yade.Bo1_Sphere_Aabb(aabbEnlargeFactor=1.5,label='is2aabb')
      ]),
   yade.InteractionLoop(
      [yade.Ig2_Sphere_Sphere_Dem3DofGeom(distFactor=1.5,label='ss2d3dg')],
      [yade.Ip2_CpmMat_CpmMat_CpmPhys()],
      [yade.Law2_Dem3DofGeom_CpmPhys_Cpm()]),
   yade.NewtonIntegrator(damping=.3),
]
dem.step()
is2aabb.aabbEnlargeFactor = ss2d3dg.distFactor = -1.

# Yade vtk export
import yade.export
vtk = yade.export.VTKExporter('test_volume.yade',startSnap=1)

# run 50 steps of simulation and save results
for i in xrange(50):
   coupler.step()
   vtk.exportSpheres(what=[('dspl','b.state.displ()')])

# exit
print 'Finished!'
fem.terminateAnalysis()
dem.exitNoBacktrace()
```

*Code 6: controlling script for three point bending volume coupling example*

```
# example script for multiscale coupling - uniaxial strain
# first import required modules
import fakemupif
from fakemupif import oofem,yade

# then instanciate FEM ...
oofemFile = 'test_multi.oofem.in'
fem = fakemupif.instanciateOofemProblem(oofemFile)

# ... as well as DEM problem
dem = yade.Omega()
demMat1 = dem.materials.append(fakemupif.defaultMat)
demMat2 = dem.materials.append(fakemupif.defaultMat2)

#
coupler = fakemupif.MultiScaleFemDemCoupler(fem,dem,oofemFile)
#

# Yade vtk export
import yade.export
vtks = {}
for gp in coupler.gps:
   newScene = dem.addScene()
   coupler.scenes.append(newScene)
   dem.switchToScene(newScene)
   dem.dt = .5*yade.utils.SpherePWaveTimeStep(.001,1000,25e9)
   dem.bodies.append(yade.pack.randomPeriPack(.001,.02))
   dem.engines=[
      yade.ForceResetter(),
      yade.InsertionSortCollider([
         yade.Bo1_Sphere_Aabb(aabbEnlargeFactor=1.5,label='is2aabb')
         ]),
      yade.InteractionLoop(
         [yade.Ig2_Sphere_Sphere_Dem3DofGeom(distFactor=1.5,label='ss2d3dg')],
         [yade.Ip2_CpmMat_CpmMat_CpmPhys()],
         [yade.Law2_Dem3DofGeom_CpmPhys_Cpm()]),
      yade.NewtonIntegrator(damping=.3),
   ]
   dem.step()
   is2aabb.aabbEnlargeFactor = ss2d3dg.distFactor = -1.
   vtks[newScene] = yade.export.VTKExporter('test_multi.yade%d'%newScene)

# run 50 steps of simulation and save results
for i in xrange(50):
   coupler.step()
   for scene in coupler.scenes:
      vtks[scene].exportSpheres(what=[('dspl','b.state.displ()')])

# exit
print 'Finished!'
fem.terminateAnalysis()
dem.exitNoBacktrace()
```

*Code 7: controlling script for multiscale uniaxial strain simulation*

```
# example script for contact coupling - cantilever impact
# first import required modules
import fakemupif
from fakemupif import oofem,yade

nSteps = 12000
# then instanciate FEM ...
oofemFile = 'test_contact.oofem.in'
fem = fakemupif.instanciateOofemProblem(oofemFile)

# ... as well as DEM problem
dem = yade.Omega()
dem.materials.append(fakemupif.defaultMat)

#
coupler = fakemupif.ContactFemDemCoupler(fem,dem,oofemFile)
#

dem.bodies.append(coupler.demImages)
dem.engines=[
   yade.ForceResetter(),
   yade.InsertionSortCollider([
      yade.Bo1_Facet_Aabb()
      ]),
   yade.InteractionLoop(
      [yade.Ig2_Facet_Facet_Dem3DofGeom()],
      [yade.Ip2_CpmMat_CpmMat_CpmPhys()],
      [yade.Law2_Dem3DofGeom_CpmPhys_Cpm()]),
]

# run
for i in xrange(nSteps):
   coupler.step()

# exit
print 'Finished!'
fem.terminateAnalysis()
dem.exitNoBacktrace()
```

*Code 8: controlling script for cantilever shock analysis*