LMGC90

Frédéric Dubois, Michel Jean, Mathieu Renouf, Rémy Mozul, Alexandre Martin, Marine Bagnéris

CSMA 2011
10e Colloque National en Calcul des Structures
9-13 Mai 2011, Presqu'île de Giens (Var)

LMGC90

F. Dubois [1], M. Jean [2], M. Renouf [3], R. Mozul [1], A. Martin [1], M. Bagneris [1]

[1] *LMGC, Université Montpellier 2, France, {frederic.dubois,remy.mozul,alexandre.martin, marine.bagneris}@univ-montp2.fr*
[2] *Planète Marseille,mjean.recherche@wanadoo.fr*
[3] *LaMCoS, INSA Lyon, France, Mathieu.Renouf@insa-lyon.fr*

Résumé — This paper gives a short overview of LMGC90 platform in terms of architecture and main features. Some examples illustrate its possibilities.
Mots clés — contact, dynamics, multiple physics, DEM, FEM, OpenSource, Fortran, Python.

1 Introduction

LMGC90 is an open platform dedicated to the modeling of large collections of interacting objects (2D/3D). It aims at modeling objects of any shape with various mechanical behavior and to take into account interaction laws as complex as necessary. Furthermore multi-physics couplings (thermal effects, fluids, *etc*) are progressively taken into account.
LMGC90 is designed as a research software which offers to developers the possibility to add new physical models (behavior law, interaction law, *etc*), numerical models (finite element, natural element, *etc*), technical features (contact detection, visualization, parallelism, *etc*) and numerical strategies (time integration, numerical solver, *etc*). Furthermore end-users are able, through the supervisor, to interact with the internal database and to use/modify these informations during the simulation.
As all research software, versatility of LMGC90 software has been obtained designing a dedicated architecture, however it needs to evolve more or less deeply with the new scientific requirements. In the present paper we refer to the architecture of the second version.

2 Architecture

As summarized on FIGURE 1, LMGC90 is divided into independent parts :
– **Core** which is dedicated to the simulation itself. The core of LMGC90 software is written in Fortran9x, which language is widely known as computationally efficient. **Core** provides a Fortran9x public interface allowing to drive a simulation, access to the database, *etc*.
– a supervising part, which rely on the **Core** public API. The common usage is possible through the **ChiPy** Python module which is build on a C API provided by a Fortran200x wrapping of the **Core** public API. An advanced-user can directly write a standalone program calling the Core public API or can build new functionality which can be added to the standard **ChiPy** module. A **Sandbox** mechanism is proposed to manage these developments (compiling, versioning, *etc*).
– **Bindings** which embed full external libraries. A general API has been defined for sets of key features on the **Core** side and a wrapping to some existing libraries to fulfill the desired functionality has been done on the **Bindings** side. Existing bindings concern linear algebra (use of Lapack as ExternalLinearAlgebra), bulk behavior (Matlib as ExternalModels) and contact detection algorithms (ExternalDetection). Various other bindings are also available (FE code, *etc*).
– **User module** which permits to define, through Fortran9x functions, some very specific possibilities as the orthotropic frame, the value of some fields at Gauss points, *etc*.

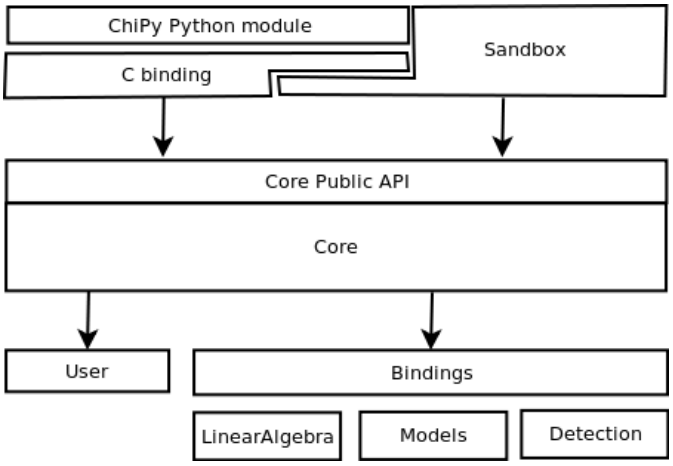

FIGURE 1 – LMGC90 modules

2.1 Core

Core relies on various Fortran9x modules. A module gathers type definitions and methods working on data of these types. Moreover modules also hold data of these defined types in order to limit side effects. Usually the data and types are private, and a set of public functions are available to use the module. Thus a structure close to an object (in the Oriented Object Programming sense) is obtained but without polymorphism or inheritance.

The general architecture of the **Core** part is depicted FIGURE 2. **Core** provides functionalities to describe the model and to run a simulation with it. The model is built using the preprocessor part of LMGC90 [7].

The input data coming through **MeliMelo** from the preprocessor are heterogeneous (objects may be different) but self content (the consistency checks are made within the **Pre**) and need to be split into a physical model (**Modelization**) and locus of interaction (**Contactor**). A numerical model is a discretization in space of a physical model. It will be used to compute the physical evolution of a body. The data pushed by **MeliMelo** depend on the kind of model :
– for a rigid body : position of the center of inertia, inertia matrix, boundary conditions, material parameters.
– for a deformable meshed body : position of the nodes, connectivity of each element and for each element its model and material and boundary conditions
A contactor contains a geometrical model of a potential interaction locus which can be described using simple convex primitives (disk, sphere, polygon, polyhedron, *etc*), a cluster of these primitives or a general triangulated surface. An object may have several contactors.

During a simulation the **Interaction Handler** module manages the creation of interactions parsing the contactors list and using smart contact detection algorithms. An interaction contains some geometrical data (position, local frame, gap, maps, *etc*) and modeling data (reaction, internal variables, *etc*). An other module, **Model Handler**, manages the creation of systems of equations (**SoE**) integrating in time the models contained in **Modelization**.
These two handler modules are driven by the simulation which provides integration rules depending on the chosen time integrator. The current state of a bulk model or of an interaction is stored in a generic database : the **State** module. Numerical methods are provided by the **Solver** module to solve all the systems of equations while fulfilling the constraints given by the interactions.

Even if this structure sounds very simple, it is not so easy to implement since the modelization, being rigid or deformable (small or large deformation) are treated rather differently before obtaining a generic system of equations. Thus the **Modelization** module is in fact a kind of abstraction module on any numerical representation of a physical model. The **Contactor** module may be seen in the same manner with regards to the various geometric shapes of contactors.

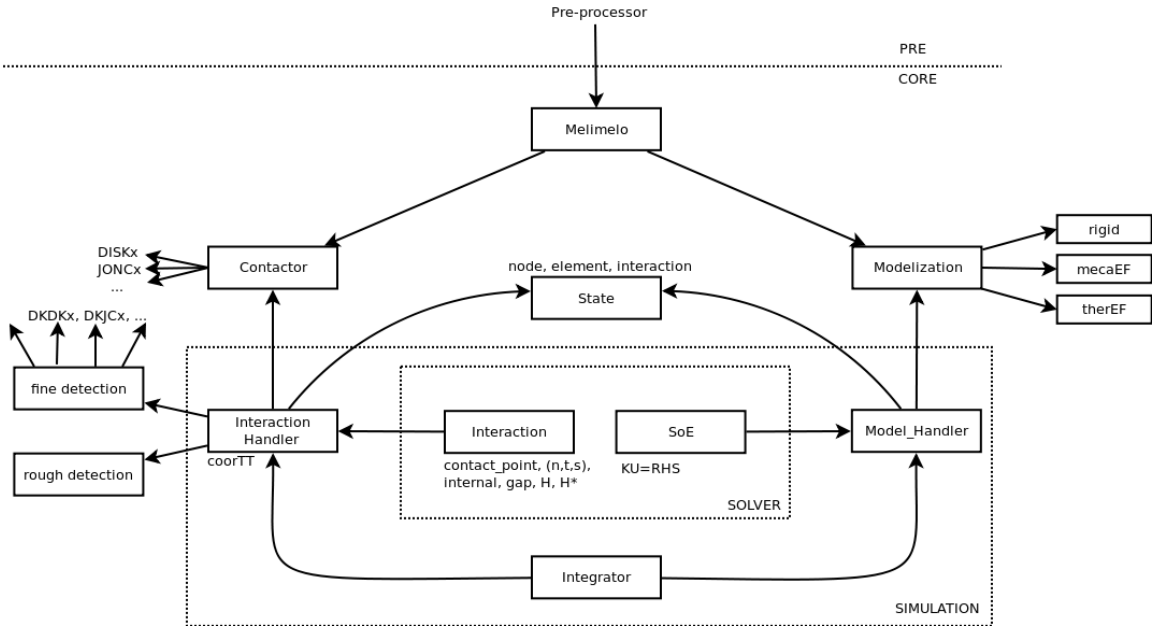

FIGURE 2 – Architecture of lmgc90

Either **Modelization** or **Contactor** provides a generic interface to different modules which implement the specific functionality for a given data type. The same logic is used for every set of module also one can imagine easily adding its new solver, time integrator, contactor geometry and the corresponding detection features.

Since the inheritance of objects is now supported in the Fortran200x norm, one could expect we use such a mechanism. But unfortunately Fortran compilers have still partially integrated the new norms and from one compiler to another the integrated features are not the same. Thus it seems irrelevant to rely on too new features proposed by the Fortran norm.

2.2 Some advanced programming features

As a foreword, we would like to say that developing a large Fortran software while trying to make it readable for new developers and friendly for users is quite difficult (almost painful). One reason comes from the fact that Fortran lacks of several really convenient features or mechanisms available in other languages (templates, polymorphism, inheritance,*etc*). As we noticed before some of them are present in the Fortran200x norm but not yet fully implemented in the available compilers. Moreover the performance of such advanced programming features is questionable. Thus our policy is to stay as close as possible to the Fortran9x norm in the **Core**'s CPU consuming parts and progressively integrate some well supported features of the new norm, like the C language binding in the interface, to give us some flexibility in programming.

However some necessary programming features are missing and here is described the way we mimic them using Fortran :

– template :
  One of the most envied features of the C++ language, is the Standard Template Library (STL), which provides vector, list, queue, *etc*, for any data type.
  When building dynamic data structures a basic convenient structure is the linked list. With standard Fortran9x such structures can be created through user defined types and pointers. The issue raises from the fact that the mechanism is type dependent and can not be deferred to a module. One understand that copying the same subroutines in each place where linked list are needed is error prone and tedious to maintain. So, to mimic a STL like behavior in Fortran one needs to trickily mix various possibilities. Key features are the overloading of the name of a type and the possibility to include the content of a

file in a module. This idea is fully used in Flibs [6]. However inclusion of file is considered as dirty programing since modules are available, but in this particular case it seems to be the only way.
We emphasize that linked list once filled are transferred into an allocatable array which is more relevant from a computational point of view.

– polymorphism :
Fortran9x allows some static polymorphism mechanism through the use of generic function interfaces. The main drawback of this mechanism is that the choice of the real function is done regarding to the type of the arguments. Therefore it is irrelevant if the prototype (*e.g.* arguments) are similar. With Fortran200x norm comes the possibility to have pointer on procedures and any procedure with the same prototype can be pointed at by a field of a data structure. Then at run-time the pointer can be set imitating a kind of dynamic polymorphism. This is particularly interesting to write loop on objects without checks on type. Contact detection is a typical example, a first generic rough detection is made (using bounding box for example) and then a fine detection depending on the type of the two contactors needs to be performed. A pointer can be set to select the fine detection to use instead of having to go through a tedious selection.

2.3 Generating a Python interface

Plugging Fortran code into Python is usually seen as an awkward problem, despite f2py which aims to automatically generate a python interface to Fortran source code. Unfortunately f2py needs to compile some Fortran code and indeed to determine which compiler and version of the compiler is available ; which makes it system dependent and may need to modify some python script located in the system. Furthermore it is impossible to add docstrings (Python documentation) to the generated module. Therefore we have decided to use a new feature of the Fortran 200x norm which allows to automatically bind a Fortran routine in C through the use of the ISO_C_BINDING module. Adding a header file, which describes the prototypes of the routines, allows to use SWIG [3], an automatic code conversion tool. Combining this approach with the use of doxygen permits to generate the docstrings.

3 Main features

3.1 Modeling

LMGC90 offers a framework to describe complex 2D/3D multi-body systems :

– **Shape of objects** : described by simple convex primitive (disk, polygon, sphere, polyhedron, *etc*), cluster of primitives or general triangulated surface.
Contact detection is performed for any combination of primitives.

– **Bulk behavior** of objects may be rigid or deformable (linear : elastic or non linear : hyper-elastic, viscous, plastic, *etc*) by means of the finite element method. Behavior parameters may depend on external fields.
Thermal effects and other physics (*e.g.* fluid dynamics) can also be modeled.
An external natural element method will be soon available.

– **Interaction laws** : a large set of laws is available, textite.g. frictional contact, cohesion (capillarity, damage, brittle, *etc*), wire, rod.
Thermal effects at contact may be taken into account.

Thanks to multiple discretizations of the same object, physic couplings can be performed at different scales through interpolation/projection methods.

Preprocessing tools are available to automatically generate granular samples, masonry structures, *etc*. Importing meshed geometry is also possible. See [7]

Once models are loaded in LMGC90 various possibilities are offered to check the validity of the model and eventually to correct it (*e.g.* log, visualization).

3.2 Analysis

LMGC90 multi-contact modeling strategy is mainly based on the **Non Smooth Contact Dynamics** method : non smooth dynamics framework, implicit time integration, implicit contact solvers (NLGS, PCG, *etc*). See [11] for details on this method. Python supervisor is very useful to finely drive the analysis.

Thanks to the modular architecture other strategies are or may also be implemented (molecular dynamics strategy, explicit time integration, quasi-static time evolution, *etc*).

Applications with a large number of interacting bodies are reachable through parallel computing (Shared or distributed memory).

A non stationary thermal solver is available.

During computation it is possible to check the relevance of the analysis through some global indicators (convergence norm, quality of interaction laws computation, *etc*).

3.3 Post-processing

LMGC90 generates two categories of post-processing files of its simulation results :
– xml-vtk visualization files. These files describe the state of objects with various physical nodal fields, contact networks, Gauss points with their values, *etc*. Visualization may be performed with any tool supporting xml-vtk format as Paraview or Mayavi.
– multi column text files. These files contain the result of analyses performed on all or part of the studied samples as contact orientation distribution, total dissipated energy, *etc*. They are suitable for any kind of plotting tool (xmgrace, MatPlotLib, gnuplot, *etc*).

Through the Python interface its also possible to access to various data which can be analyzed using built-in Python functions.

3.4 User Interface

LMGC90 is not integrated into a graphical environment and is usually driven by a Python script. Moreover it provides access to LMGC90 database, which offers the possibility to manage finely a simulation or couple with an external code.

A Python preprocessor aims at defining geometries, material properties, numerical modeling options, boundary conditions, *etc*.

3.5 An open platform

Even if Python allows to customized simulation, advanced developments benefit from an in **Core** Fortran90 implementation, which needs to have access to the sources which are open under CECILL license (textiti.e. GPL). This kind of license is particularly relevant to convince researchers to capitalize their scientific developments.

Such an open platform enables coupling capabilities with external software (Finite Element code : Code_Aster, Pelicans) and libraries (Physical Models : MatLib, Linear Algebra : Lapack, Contact detection : Rapid, *etc*.). Furthermore the LMGC90 developer team is part of the Saladyn project (funded by ANR) and the Degrip project (funded by Oseo/FEDER).
LMGC90 relies on collaborative programming tool (subversion & trac) :
`https://subver.lmgc.univ-montp2.fr/trac_LMGC90v2`

4 Examples of Application

4.1 Granular Material from rheology to structure

A typical use of LMGC90 concerns the study of the rheology of granular materials. As an example FIGURE 3 shows two types of results concerning the deformation of a sample made of 40,000 polyhedra (8 faces). On the left a detail of the packing is given ; the grey scale corresponds to the coordination number. On the right the contact network is represented ; line thickness is proportional to the force. The strong network is in grey and the weak in red.

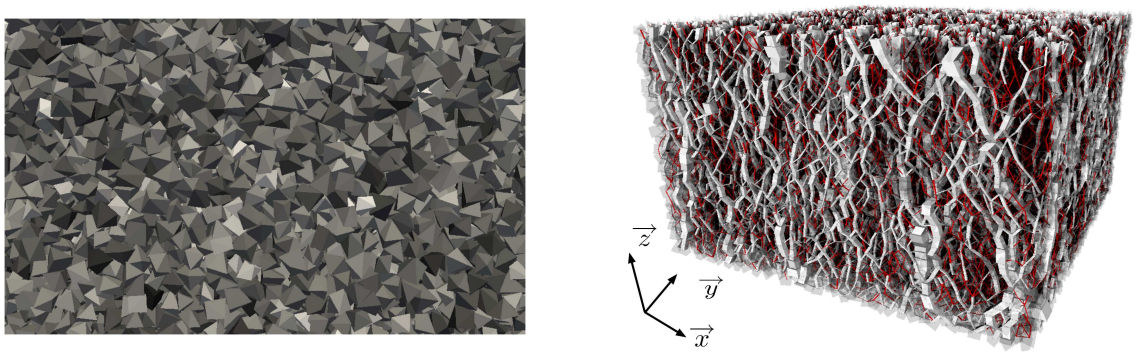

FIGURE 3 – Tri-axial compression of a polyhedral sample [13]

Basic features help with studying influence of shapes, interaction laws and bulk physics on the behavior of a collection of objects.

4.2 Masonry structures

A typical example is the stability study of ancient monument as Nîmes arena. On FIGURE 4 block geometry details of the and computed pressure into the joints due to dead load are given. This example was built using the Roman's conception rules. This kind of modeling allows assessing the stability and the

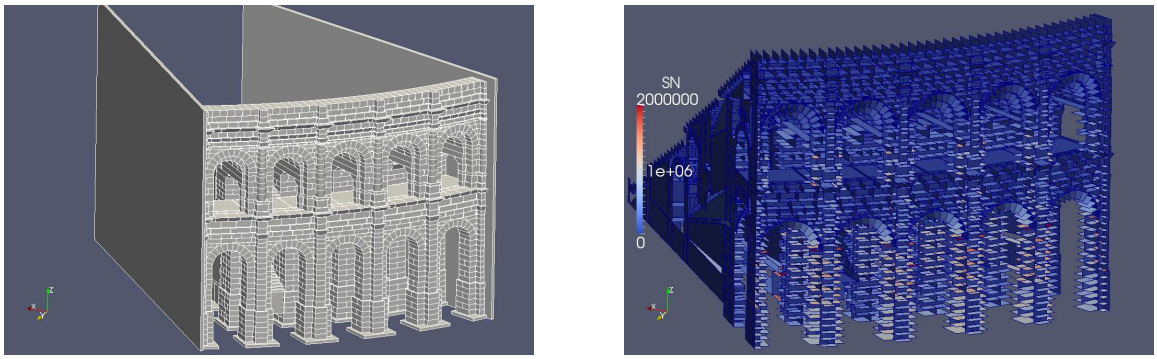

FIGURE 4 – Modeling of Nîmes arena. Left the geometry. Right the pressure between blocks once at rest

safety of masonry structures under static loads or dynamical natural risk (seism, landslide, *etc*) taking into account the influence of the design pattern and the joint behavior.

4.3 Modeling Fracture

LMGC90 allows to model fracture in various media. FIGURE 5 shows (left) fracture of an heterogenous media under horizontal traction load [8] and (right) the study of stability of a rock mass under self weight load [12]. Using a Frictional Cohesive Zone Model, fracture can be modeled, at microscopic or meso-

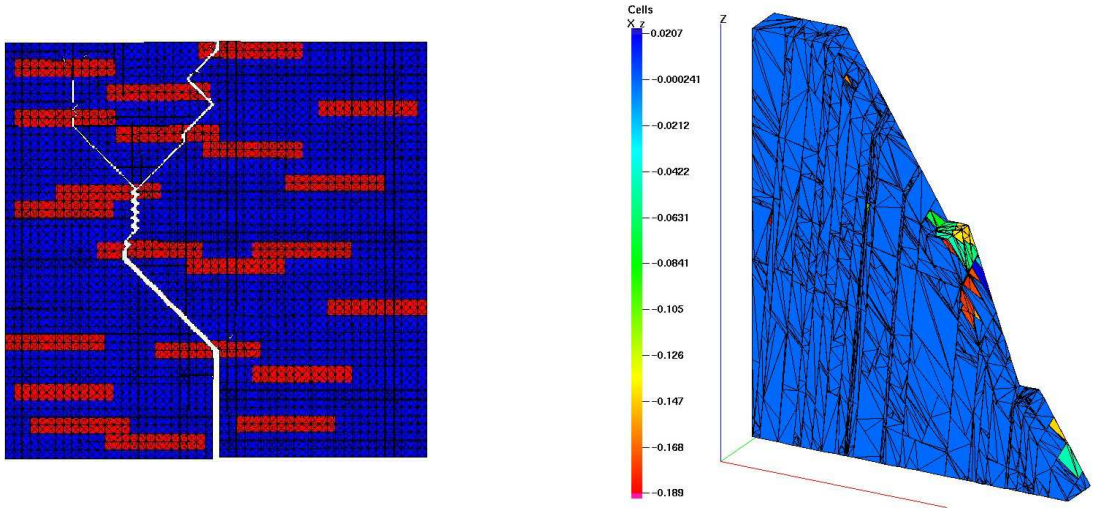

FIGURE 5 – fracture modeling examples

scopic scale, from initiation to post-failure.
Considering rock mass as a fractured media, avalanches on natural or mining slopes, stability tunnels, *etc* can be studied.

4.4 Multiple physics couplings

LMGC90 allows to consider various physics at different scales such as thermal coupling [11], fluid particle interaction [9, 10], electrical conductivity [5]. FIGURE 5 shows (left) Sedimentation of thin particles in gaz and (right) an immersed avalanche of a loose granular assembly in a small closed box.

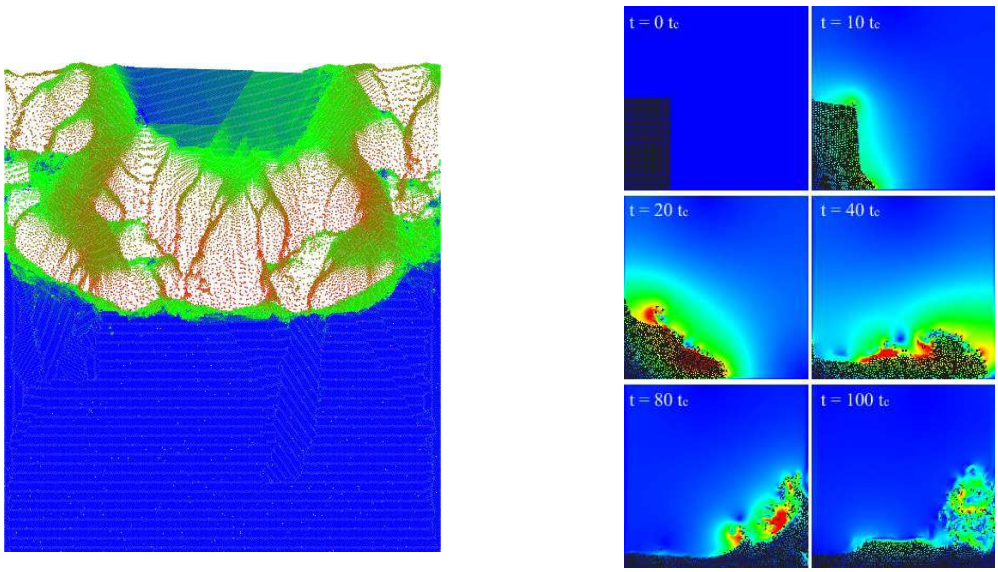

FIGURE 6 – coupling modeling examples

5 Conclusion

Started more than 10 years ago, the LMGC90 project is becoming more and more active from a research or application point of view.
The nowadays challenges are multi-physics and multi-representation either to perform the switch of an object models (rigid - deformable) or to reduce-refine a part of a structure made of several objects.

The price to pay is a constant evolution of the overall software architecture and an improvement of the core technology.
Fortunately the Open-Source world provides a bunch of impressing software libraries that can be used.
One aim of the Saladyn project is to couple LMGC90, Siconos and Code_Aster software through the Salome platform in order to take advantages of all these tools.

Acknowledgements : the authors want to thanks all the LMGC90's contributing people in particular : G. Saussine, B. Chetouane, F. Perales, R. Perales, A. Rafiee, M. Vinches, M. Schryve, F. Kuss, H. Baudriller, C Silvani, J.J. Moreau, F. Radjai.
The project is currently funded by ANR (Saladyn Project) and OSEO-FEDER (Degrip Project).

Références

[1] M. Jean, J.J. Moreau. *Unilaterality and dry friction in the dynamic of rigid body collections*, Proc. Contact Mechanics Int. Symp., Edt A. Curnier, 31-48, 1992.

[2] M. Jean. *The non-smooth contact dynamic method*, Computer methods in applied mechanics and engineering, 177, No 3-4, 235-257, 1999.

[3] D. M. Beazley. *Automated scientific software scripting with SWIG.* Future Generation Computer Systems, 19(5) :599–609, 2003.

[4] F. Dubois, M. Jean. *LMGC90 : une plateforme de développement dédiée à la modélisation des problèmes d'interaction*, In M. Pottier-Ferry, M. Bonnet et A. Bignonnet, éditeurs : 6ème Colloque National en Calcul des Structures, Giens, volume 1, 111-118, 2003.

[5] M. Renouf, N. Fillot. *Coupling electrical and mechanical effects in discrete element simulations*, IJNME, 74(2), 238-254, 2008

[6] A. Markus, M. Baudin, *FLIBS - A collection of Fortran modules*, http ://flibs.sourceforge.net/, 2010

[7] A. Martin, M. Bagnéris, F. Dubois, R. Mozul. *Conception d'un outil adapté à la mise en données des systèmes discrets*, 10éme Colloque National en Calcul des Structures, Giens, 2011.

[8] F. Perales, F. Dubois, Y. Monerie, B. Piar, L. Stainier.*A NonSmooth Contact Dynamics-based multi-domain solver. Code coupling (Xper) and application to fracture*. European Journal of Computational Mechanics 19, 389-417, 2010.

[9] A. Martin. *Ecoulement confiné d'un matériau granulaire en interaction avec un gaz, application à la relocalisation du combustible nucléaire*, Thèse de doctorat de l'Université de Montpellier 2, 2010.

[10] V. Topin, F. Dubois, Y. Monerie, F. Perales, A. Wachs. *Micro-rheology of dense particulate flows : Application to immersed avalanches*.Journal of Non-Newtonian Fluid Mechanics 166 (1-2), 63-72, 2011.

[11] F. Radjai, F. Dubois *Discrete Numerical Modeling of Granular Materials*, ISTE Ltd and John Wiley & Sons Inc, 2011.

[12] A. Rafiee, M. Vinches, F.Dubois. *The Non-Smooth Contact Dynamics method applied to the mechanical simulation of a jointed rock mass*, ENOC Congress, 2011

[13] E. Azema, F. Dubois, F. Radjai. *On the quasi-static behavior of granular material composed with irregular polyhedral particles : effects of grains angularity*, to be published.