The Saladyn toolbox

Vincent Acary, Maurice Brémond, Hong Phong Cao, Frédéric Dubois, Sylvain Mazet, Rémy Mozul

CSMA 2013
11e Colloque National en Calcul des Structures
13-17 Mai 2013

The Saladyn toolbox

V. Acary [1], M. Brémond [1], H.P. Cao [2] , F. Dubois [3], S. Mazet [4], R. Mozul [3]

[1] INRIA Rhône-Alpes, BIPOP, vincent.acary@inria.fr, maurice.bremond@inria.fr
[2] LAMSID, caohongphong@gmail.com
[3] LMGC, frederic.dubois@univ-montp2.fr, remy.mozul@univ-montp2.fr
[4] EDF R&D, AMA, sylvain.mazet@gmail.com

Abstract. The Saladyn software toolbox is the main outcome of the ANR project Saladyn (COSINUS ANR-08-COSI-014-01), which aims at designing and implementing a new software platform by coupling several kinds of mechanical models and modeling approaches in a single framework : a) solid bodies (deformable), mainly through their finite element representation, b) rigid multi-body systems and c) multi-contact systems. The goal is to obtain a close coupling of these models for the modeling and the simulation in a nonsmooth dynamical framework, able to deal rigorously with the unilateral contact and Coulomb's friction. This platform is composed of the integration of the following components : SALOMÉ, an open–source platform for the pre and post-processing and the coupling of numerical software codes, CODE_ASTER an open–source Finite Element Application, which has already been integrated in Salomé, under the name of Salomé-méca, LMGC90 an open–source software for the modeling and the simulation of multi-contact systems and SICONOS an open–source software for the modeling, the simulation and the control of nonsmooth dynamical systems. In this article, the common formalism and numerical methods are delineated. Several general coupling schemes are described and the interest of the approach is illustrated on several industrial realistic examples.
Mots clés — contact, friction, impact, nonsmooth dynamics, computational contact Mechanics, multiple physics, DEM, FEM, Open Source software, coupling.

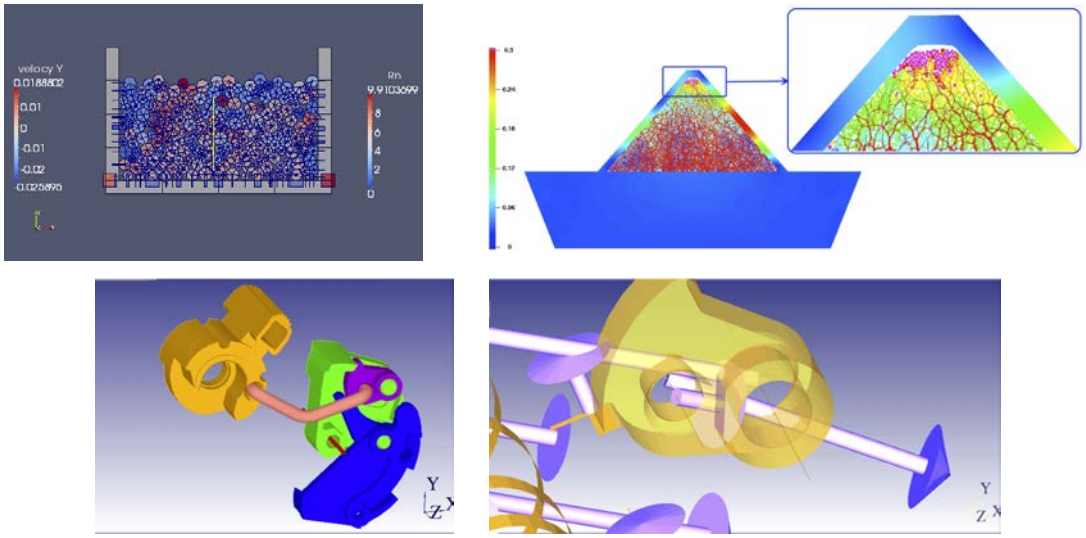


Fig. 1 – Applications of the Saladyn Toolbox

1 Introduction

The Saladyn software toolbox is the main outcome of the ANR project Saladyn (COSINUS ANR-08-COSI-014-01), which aims at designing and implementing a new software platform by coupling several kinds of mechanical models and modeling approaches in a single framework : a) solid bodies (deformable), mainly through their finite element representation, b) rigid multi-body systems and c) multi-contact systems. The goal is to obtain a close coupling of these models for the modeling and the simulation in a nonsmooth dynamical framework, able to deal rigorously with the unilateral contact and Coulomb's friction. This platform is composed of the integration of the following components : SALOMÉ, an open–source platform for the pre and post-processing and the coupling of numerical software

codes, CODE_ASTER an open–source Finite Element Application, which has already been integrated in Salomé, under the name of Salomé-méca, LMGC90 an open–source software for the modeling and the simulation of multi-contact systems and SICONOS an open–source software for the modeling, the simulation and the control of nonsmooth dynamical systems.

The Saladyn project was an experimental development project coordinated by INRIA Grenoble-Rhône Alpes. EDF and Schneider Electric were associated as industrial partners together with two other laboratories LMGC Montpellier and LAMSID Clamart.

In order to provide mechanical engineering with robust and efficient tools, the aim of the first step of the project is to accumulate, gather and couple numerous functions of the three codes (CODE_ASTER, LMGC90, SICONOS) with different ranges of applications in solid mechanics (continuum solid Mechanics, multi-body systems and multi-contact systems) into a single software environment (Salomé). In the second step, methods have been proposed to dynamically adapt the modeling and simulation strategy. Once the main issues of coupling several software codes (I/O data and methods exchanges) are fixed, the designed environment is meant to be sufficiently open such that it allows to developers to integrate other software. This design attempts to limit the most important bottlenecks : data transfer and memory consumption. These developments have enabled the simulation of circuit breakers for Schneider Electric and rock-fill dams for EdF with low development costs (in terms of human and financial resources) compared to the re-writing ab nihilo new software.

2 Goals, Objectives and Means

Objectives In the framework of the Computational Contact Mechanics, a lot of academic and commercial simulation codes can nowadays model and simulate the quasi-statics and the smooth dynamics of mechanical systems with unilateral contact, Coulomb's friction and possibly enriched interfaces laws such as cohesive zone models. However few of them are able to take into account the non smooth events occurring in dynamics, when abrupt failures, collapses, frictional paradoxes or impact are encountered. These physical phenomena can be efficiently modeled and simulated in the framework of the nonsmooth dynamics mainly initiated and developed in the mechanical community by the pioneering work of J.J. Moreau and M. Jean [1–5]. Within this framework, the Contact Dynamics (CD) or the NonSmooth Contact Dynamics (NSCD) has been mainly implemented in the two software codes : LMGC90 and SICONOS. One of the goal in this project is to bridge the gap between these two codes and the standard available software in the Contact Mechanics, where high-level FEM implementation can be found with enhanced and sophisticated bulk mechanical behavior laws.

More specifically, the core of this work is to try to share some specific features of several specialized pieces of software which are :

- LMGC90 designed for multi-body/multi-contact simulation [1] [6, 7]
- SICONOS designed for the modeling, the simulation and the control of non smooth dynamical systems, with a special focus on mechanical with contact, impact and friction [2] [8, 9]
- CODE_ASTER designed for finite element simulation in Solid Mechanics.

Although CODE_ASTER is an experienced and specialized code for the simulation of solid bodies (linear or non-linear mechanical constitutive laws), the simulation of the nonsmooth motion of large collection of bodies cannot be efficiently performed. Firstly, this is mainly due to the choice in the architecture of the standard code. The standard architecture of the most FEM simulation software does not allow to take into account large collections of bodies with possibly floating and rigid body motions. Secondly, the occurrence of nonsmooth events prevents the use of standard time–stepping schemes. Since most of FEM codes are based on Newmark–like integration schemes (Newmark, HHT, α-generalized) which takes into account the unilateral contact at the position level, it is not possible to consistently integrate in time nonsmooth evolutions.

On the other hand, LMGC90 offers efficient contact detection features relevant for large collection of objects and a robust implementation of the NSCD method, which treats the nonsmooth laws mainly at the velocity level with a consistent time integration scheme based on a modified θ-method : The Moreau–Jean scheme. SICONOS is also dedicated to the nonsmooth dynamics simulation. Besides the

1. https://subver.lmgc.univ-montp2.fr/trac_LMGC90v2
2. http://siconos.gforge.inria.fr

Moreau–Jean scheme, it additionally offers the possibility to handle a lot of various dynamical systems formulations (Lagrangian, Newton/Euler, ...) and provides several time–stepping schemes alternative to the NSCD method (nonsmooth Newmark, Schatzman–Paoli scheme) and a collection of solvers for discrete frictional contact problems.

A summary of the available and missing features is described in Fig. 2.

	Modelling				Simulation					
	Linear		Non-linear		Quasi-static		Smooth dynamic		Non-mooth dynamic	
	Rigid	Deformable	Rigid	Deformable	Linear	Non -linear	Linear	Non-linear	Linear	Non-linear
Code_Aster	✗	✓✓	✗	✓✓	✓	✓	✓	✓	✗	✗
Siconos	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
LMGC90	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓

Fig. 2 – Software features

Goals As we said earlier, the aim is to accumulate, gather and couple numerous functionality of the three codes (CODE_ASTER, LMGC90, SICONOS) with different ranges of applications in solid mechanics (continuum solid Mechanics, multi-body systems and multi-contact systems) into a single software environment (Salomé). The main goals are :

- to drive CODE_ASTER with simulation features of SICONOS and to include rigid bodies provided by LMGC90 in the model,
- to take benefit from SICONOS contact solvers (SICONOS/NUMERICS) and time integration schemes (SICONOS/KERNEL) into LMGC90 and CODE_ASTER
- to take benefit from the very complete and efficient FE library available within CODE_ASTER for SICONOS. SICONOS doesn't provide modeling tool for solid Mechanics, and in this case it would benefit of all modeling, contact detection and pre/post processing tools available in LMGC90 and CODE_ASTER.
- to take benefit from the SICONOS/MECHANICS component which provides with a modeling tool for rigid multibody systems (mechanisms) described by CAD geometries using OpenCascade and PythonOCC.

Means : strong and weak coupling A common spine must be defined for the communication between the various engineering software. Python is the obvious solution as the best language/environment for this goal. Besides its intrinsic features, the existence of automatic wrappers gives us the opportunity to plug the various existing software programs with the pre/post-processing tools of Salomé-Méca. The so–called **SALADYN TOOLBOX** is composed of a set of Python classes defining a common application programming interface (API) for the several codes involved. The object-oriented design allows further evolution in terms of integration of new codes as well as the development of new functionality such as the dynamic adaptability of mechanical models during computation. Another strong coupling that has also been realized concerns the use of CAD models based on B-Rep representations for the mechanical simulation of unilateral contact and friction. In most of the standard simulation codes, spatial discretizations (mesh) are usually used. However, for the mechanical engineers, the contact simulation, especially with large sliding motions needs to directly deal with CAD geometries. The Saladyn/Multibody toolbox (written in C++/Python and integrated in SICONOS/MECHANICS) allows us to perform this task for the contact detection and the solving in order to give more realistic simulations.

These three software codes can be driven by python scripting. The chosen coupling method is based on a common interface implemented through python classes, the **SALADYN TOOLBOX**. Each of the three codes are then plugged to the toolbox

In the developed toolbox, as in SICONOS, a graph structure is used to describe the full model, where each node of the graph is a dynamical system and each edge an interaction between two systems. Such

structure allows the simulation tool to drive models easily and offers genericity since any new software could be plugged-in in the toolbox. Each software needs to split its functionality in components as described on Fig. 3.

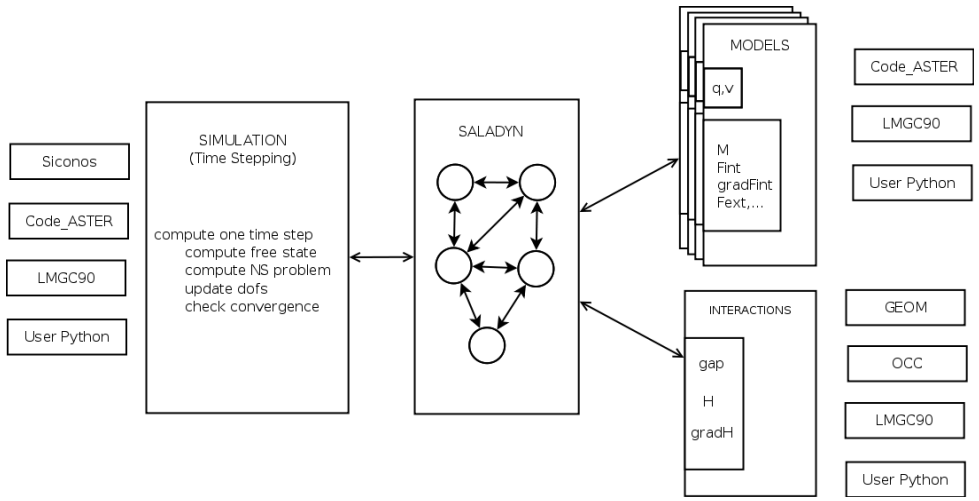


Fig. 3 – Saladyn Toolbox

The main bottleneck of our approach comes from data structures and memory management which may imply duplication in modeling tools and some copy of data between them (which is exactly the same problem with parallel computing). However, due to the power of python, some mechanisms using references on memory were tested to completely remove copies to ensure efficiency, such approach is still in progress since it has side effects on the architecture of the software.

Outline The article is organized as follows. In Section 3 and 4, the main theoretical ingredients are briefly recalled, *i.e* the common formalism for the equation of the nonsmooth dynamics and the nonsmooth constitutive and the numerical solution procedure. In Section 5, the main features of the available software are recalled. The SALADYN TOOLBOX and then coupling strategy is detailed in 6. In Section 7, the two main applications of the SALADYN TOOLBOX are illustrated : the simulation of clearances in electrical circuit breakers for Schneider Electric in Section 7.1 and the simulation of rock-fill dams for EdF in Section 7.2. Section 8 concludes the article.

3 A general mechanical formulation of nonsmooth dynamical systems

In this section, we propose a common formulation for various modeling approaches that we want to couple in Saladyn. We also briefly review the main ingredients of the nonsmooth dynamical systems theory. For more details, we refer to [10–13]

As we said above, we want to take into account at least three mechanical modeling approaches in a single framework : a) solid bodies (deformable), mainly through their finite element representation, b) rigid multi-body systems (mechanisms with joints) and c) multi-contact systems (granular matter and divided materials). The goal is to obtain a close coupling of these models for the simulation in a nonsmooth dynamical framework, able to deal rigorously with the unilateral contact and Coulomb's friction and possibly with cohesive zone models. Let us just recall that we only deal with discrete systems. When applications with solid bodies are addressed, we assume that a preliminary space–discretization has been performed, for instance, by FEM.

In order to reach this goal, a common formulation of the equations of motion and the constitutive laws is proposed, which embraces the different modeling approaches :

- continuous solid bodies and Lagrangian systems. In this case, the configuration parameters $q(t) \in \mathbb{R}^{n_q}$ are usually displacements, or positions, or more generally generalized coordinates. The velocity $v(t) \in \mathbb{R}^{n_v}, n_v = n_q$ is given by the time–derivative of these coordinates, *i.e.* $v(t) = \dot{q}(t)$.
- rigid bodies and multi–body systems. In this latter case, a lot of choices are possible for the configuration parameters. In the standard Newton/Euler setting, the configuration parameters $q(t) \in \mathbb{R}^{n_q}$

are usually the position of the center of mass x_G and a set of parameters Θ that encodes the rotation of the body (rotation matrix, Euler angles, quaternion, ...). The velocity $v(t) \in \mathbb{R}^{n_v}$ is then given by the time derivative of the position of the center of mass $v_G = \dot{x}_G$ and the angular velocity Ω, described for instance in the inertial frame. Then, the velocity is not the time derivative of the configuration parameters.

In order to take into account the two previous case, we formally write

$$\dot{q} = T(t,q)v, \tag{1}$$

where the operator $T \in \mathbb{R}^{n_q \times n_v}$ links the time derivatives of the coordinates to the velocities. The generic formulation of the motion equations of a mechanical system can be written as

$$\begin{cases} \dot{q} = T(t,q)v, & \text{(2a)} \\ M(q)\dot{v} = F(t,q,v,u), & \text{(2b)} \\ \dot{u} = d(t,q,v,u), & \text{(2c)} \end{cases}$$

where $q(t) \in \mathbb{R}^{n_q}$ is a set of coordinates that gives in a unique way the configuration of the system. The vector $v(t) \in \mathbb{R}^{n_v}, n_v \leqslant n_q$ describes the velocity. The matrix $M \in \mathbb{R}^{n_v \times n_v}$ is a symmetric, positive and (semi-)definite matrix representing the inertia. The vector of forces $F \in \mathbb{R}^{n_v}$ is applied to the system includes the gyroscopic effects, the internal and external forces and the effect of the vector $u \in \mathbb{R}^{n_u}$ which might be considered as a control input vector. The dynamics of this control can be described by another first order dynamical system (2c). These system of ordinary differential equations (ODE) is completed with initial conditions $q(t_0) = q_0$, $v(t_0) = v_0$ and $u(t_0) = u_0$.

3.1 Lagrangian formulation with unilateral constraints

For the sake of simplicity we start with the Lagrangian formulation with unilateral constraints such as

$$\begin{cases} \dot{q} = v, & \text{(3a)} \\ M(q)\dot{v} + F(t,q,v,u) = G^\top(t,q)\lambda, & \text{(3b)} \\ \dot{u} = d(t,q,v,u,\lambda), & \text{(3c)} \\ g_k(t,q) = 0, \quad k \in \mathcal{E} & \text{(3d)} \\ g_k(t,q) \geqslant 0, \quad \lambda_k \geqslant 0, \quad \lambda_k g_k(t,q) = 0 \quad k \in I, & \text{(3e)} \end{cases}$$

where $\mathcal{E} \subset \mathbb{N}$ and $I \subset \mathbb{N}$ are the index sets of bilateral constraints ($\mathcal{E}$ stand for equalities) and of *unilateral constraints* (I stand for inequalities), respectively. The matrix $G(t,q) = \nabla_q^\top g(t,q)$ is the Jacobian of the constraints. The condition (3e) is called a *complementarity condition*, or equivalently the *Signorini condition*. It can be equivalently written as

$$0 \leqslant g_k(t,q) \perp \lambda_k \geqslant 0 \quad \Longleftrightarrow \quad -\lambda_k \in N_{\mathbb{R}^+}(g_k(t,q)), \quad \text{for all } k \in I. \tag{4}$$

The mathematical symbol N_K is the normal cone to a convex set K (see [14, 15]). If we define the admissible configuration set $\Phi(t)$ as

$$\Phi(t) = \{q \mid g_k(t,q) = 0, k \in \mathcal{E}, \quad g_k(t,q) \geqslant 0, k \in I\}, \tag{5}$$

the whole set of constraints (3d-e) can be written as

$$-G^\top(t,q)\lambda \in N_{\Phi(t)}(q). \tag{6}$$

The smooth Lagrangian dynamics (3b-d-e) amounts to solving the following *Differential Inclusion* (DI)

$$-[M(q)\dot{v} + F(t,q,v,u)] \in N_{\Phi(t)}(q). \tag{7}$$

3.2 Nonsmooth Lagrangian Formulation

In order to simplify, we focus our presentation to the case of holonomic constraints and we omit the control input part. The Lagrangian dynamics in (7) is usually not so smooth. If the normal relative velocity at contact $\dot{g}_k(q)$ is positive when the bodies hit at time $t^\star$, in other words, when an impact occurs, a velocity jump must occur to satisfy the constraints after $t^\star$. From the mathematical point of view, the velocity $v(t)$ is considered as a function of Bounded Variations (BV) and the acceleration becomes a differential measure denoted by $\mathrm{d}v$, associated to v [16]. The Lagrange multiplier are also measures including some Dirac atoms when the jumps occur ; therefore we consider an impulse measure $\mathrm{d}i$ rather than $\lambda \mathrm{d}t$ to express the reaction due to the constraints.

The dynamics is written as a *Measure Differential Inclusion* (MDI) [17–19] :

$$-\left[M(q)\mathrm{d}v + F(t,q,v^+)\mathrm{d}t\right] = -G^\top(q)\,\mathrm{d}i, \tag{8}$$

where v^+ (respectively v^-) is the right limit (left limit) of the BV function v and $\mathrm{d}t$ is the Lebesgue measure. The unilateral constraints is written as a measure inclusion as

$$-G^\top(q)\,\mathrm{d}i \in N_\Phi(q). \tag{9}$$

When a jump occur in a discrete mechanical system, an impact law must be added to uniquely define the post impact velocity $v^+(t)$. Let us simply choose the Newton impact law

$$v^+(t) = -ev^-(t), \text{ for all } t \text{ such that } g(q) = 0. \tag{10}$$

where e is the coefficient of restitution. A compact form of the inclusion (9) together with the Newton impact law (10) was designed by J.J. Moreau in [1]

$$-G^\top(q)\,\mathrm{d}i \in N_{T_\Phi(q)}(v^+ + ev^-), \tag{11}$$

where T_K is the tangent cone to K. It yields the second order Moreau sweeping process :

$$-\left[M(q)\mathrm{d}v + F(t,q,v^+)\mathrm{d}t\right] \in N_{T_\Phi(q)}(v^+ + ev^-). \tag{12}$$

This MDI (12) can be recast into a complementarity form rather than inclusions into cone as

$$\begin{cases} -\left[M(q)\mathrm{d}v + F(t,q,v^+)\mathrm{d}t\right] = -G^\top(q)\,\mathrm{d}i, & \text{(13a)} \\ U = G(q)v, & \text{(13b)} \\ \text{if } g(q) \leqslant 0, \text{ then } 0 \leqslant U^+ + eU^- \perp \mathrm{d}i \geqslant 0. & \text{(13c)} \end{cases}$$

Remarque 1

– *The positivity constraint on the measure* $\mathrm{d}i$ *in (13c) can be roughly interpreted as follows. Let us consider that the measure can be decomposed as* $\mathrm{d}i = \lambda(t)\mathrm{d}t + p\delta$*, where* λ *is the standard Lagrange multiplier assumed to be a smooth function of time and* p *is the magnitude of the Dirac atom. The inequality* $\mathrm{d}i \geqslant 0$ *amounts to constraining* $\lambda(t) \geqslant 0$ *almost everywhere and* $p \geqslant 0$ *at the impact time*

– *The writing of the inclusion (11) can be interpreted as the formulation of the unilateral constraints at the velocity level. It appears to be similar to the index reduction procedure for Differential Algebraic Equations (DAE) (see [20]).*

3.3 Time–decomposition of the nonsmooth dynamics

If we neglect the singular measures, the differential measure $\mathrm{d}v$ and the impulse measure $\mathrm{d}i$ can be decomposed as follows

$$\begin{aligned} \mathrm{d}v &= \gamma \mathrm{d}t + (v^+ - v^-)\mathrm{d}\nu, \\ \mathrm{d}i &= \lambda\,\mathrm{d}t + p\mathrm{d}\nu, \end{aligned} \tag{14}$$

where $\gamma(t)$ id the standard acceleration given by $\ddot{q}(t)$ almost everywhere and $\mathrm{d}\nu = \sum_i \delta_{t_i}$ is the sum of Dirac measures supported at the instants of impact t_i. In the sequel, we write $p\mathrm{d}\nu = \sum_i p_i \delta_{t_i}$. By substituting the decomposition (14) into the dynamics (13), we obtain

– *the impact equations* at the instants of impact t_i

$$\left\{\begin{array}{l} M(q(t_i))(v^+(t_i)-v^-(t_i)) = G^\top(q(t_i))p_i, \\ U^+(t_i) = G(q(t_i))v^+(t_i), \quad U^-(t_i) = G(q(t_i))v^-(t_i), \\ 0 \leqslant U^+(t_i) + eU^-(t_i) \perp p_i \geqslant 0, \end{array}\right. \tag{15}$$

– and *the non–impulsive equations of motion* for $t \in]t_i, t_{i+1}[$

$$\left\{\begin{array}{l} M(q)\gamma + F(t,q,v) = G^\top(q)\lambda, \\ U = G(q)v, \\ \text{if } g(q) \leqslant 0, \text{ then } 0 \leqslant U \perp \lambda \geqslant 0. \end{array}\right. \tag{16}$$

3.4 Other settings

With great cares with the formulation of unilateral constraints on parameters for more general than standard Lagrange generalized coordinates, the unilateral contact can be included for various formulations of the equations of motion. In [11], the Newton/Euler case is treated with details. In a general manner, the formulation (7) with unilateral constraints yields a similar smooth DI as

$$\left\{\begin{array}{ll} \dot{q} = T(t,q)v, & \text{(17a)} \\ -[M(q)\dot{v} + F(t,q,v,u)] \in T^\top(t,q)N_{\Phi(t)}(q), & \text{(17b)} \\ \dot{u} = d(t,q,v,u,\lambda). & \text{(17c)} \end{array}\right.$$

The formulation of the second–order Moreau sweeping process equivalent to (12) can be extended as well.

3.5 Coulomb's friction

The introduction of more sophisticated interface force laws than the simplest unilateral constraints is not straightforward in a pure Lagrangian setting. In order to give a mechanical meaning to the description of the contact, a local frame at contact denoted by (C,n,s,t) is generally introduced. In the same way, the Lagrange multipliers are replaced by reaction forces, or contact forces which are decomposed in the local frame as $R = R_{\mathrm{N}}n + R_{\mathrm{T}}$ with $R_{\mathrm{T}} = R_{\mathrm{S}}s + R_{\mathrm{T}}t$. The local relative velocity at contact is also decomposed as $U = U_{\mathrm{N}}n + U_{\mathrm{T}}$. Usual kinematic relations between the local variables at contact and the generalized variables give

$$U = G(t,q)\dot{q} + j(t) = G(t,q)T(t,q)v + j(t), \tag{18}$$

and, by duality, the generalized forces due to contact forces

$$r = T^\top(t,q)G^\top(t,q)R. \tag{19}$$

The dynamical equations can be then written as

$$M(q)\dot{v} + F(t,q,v,u) = T^\top(t,q)r = T^\top(t,q)G^\top(t,q)R. \tag{20}$$

By formulating the unilateral contact in terms of local variables at contact we get

$$0 \leqslant g_{\mathrm{N}} \stackrel{\Delta}{=} g(q) \perp R_{\mathrm{N}} \geqslant 0, \tag{21}$$

and with the impact law at the velocity level

$$\text{if } g_{\mathrm{N}} \leqslant 0 \text{ then } 0 \leqslant U_{\mathrm{N}}^+ + eU_{\mathrm{N}}^- \perp \mathrm{d}I_{\mathrm{N}} \geqslant 0, \tag{22}$$

where $\mathrm{d}I_{\mathrm{N}}$ is the normal impulse measure in the local frame.

Coulomb's friction can be expressed in a disjunctive form as

$$\text{if } g_{\text{N}} \leqslant 0, \begin{cases} \text{if } U_{\text{T}} = 0 \quad \text{then } R \in \mathbf{C} \\ \text{if } U_{\text{T}} \neq 0 \quad \text{then } R \in \partial\mathbf{C} \text{ and it exists } a \geqslant 0 \text{ such that } R_{\text{T}} = -aU_{\text{T}} \end{cases} \tag{23}$$

where $\mathbf{C}$ is the Coulomb friction cone, $\mathbf{C} = \{R, | \, \|R_{\text{T}}\| \leqslant \mu R_{\text{N}}\}$ with μ, the coefficient of friction. Coulomb's friction can also be formulated compactly as a complementarity condition on second–order cones as [11, 21–23]

$$-\hat{U} \stackrel{\Delta}{=} -\begin{bmatrix} U_{\text{N}} + \mu\|U_{\text{T}}\| \\ U_{\text{T}} \end{bmatrix} \in N_C(R) \Longleftrightarrow \mathbf{C}^* \ni \hat{U} \perp R \in \mathbf{C} \Longleftrightarrow -R \in N_{C^*}(\hat{U}) \tag{24}$$

where $\mathbf{C}^*$ is the dual cone of $\mathbf{C}$ [15].

For one contact, the smooth contact dynamics is given by

$$\begin{cases} M(q)\dot{v} + F(t,q,v) = T^{\top}(t,q)G^{\top}(t,q)R \\ U = G(t,q)T(t,q)v + j(t) \\ \text{if } g(q) \leqslant 0 \text{ then } -R \in N_{C^*}(\hat{U}) \text{ otherwise } R = 0 \end{cases} \tag{25}$$

By following the work previously carried out for the unilateral constraints with the newton impact law, the MDI is extended for the Coulomb friction as

$$\begin{cases} M(q)\mathrm{d}v + F(t,q,v^+)\,\mathrm{d}t = T^{\top}(t,q)G^{\top}(t,q)\mathrm{d}I \\ U^+ = G(t,q)T(t,q)v^+ + j(t), \quad U^- = G(t,q)T(t,q)v^- + j(t) \\ \text{Si } g(q) \leqslant 0 \text{ alors } -\mathrm{d}I \in N_{C^*}(\hat{U}^+ + eU_{\text{N}}^- n) \text{ sinon } \mathrm{d}I = 0 \end{cases} \tag{26}$$

Remarque 2 *A more generic formulation with multiple contact points can be found in [11] as well as a discussion on the friction models that we can find the mechanical and control literature. Modeling of cohesive zone interfaces in theis framework can also be found in [24] and [11, Sect 3.9.4].*

4 Numerical methods

In this section, we give some insights on the numerical solution procedures for nonsmooth dynamical systems. We start with the introduction of the two main families of time-stepping schemes in section 4.1, and then we focus on the Moreau–Jean event–capturing scheme in section 4.2. We end the section with a brief overview of numerical methods for discrete frictional contact problems. For more details, we refer to [11, 12].

4.1 Two main families of time–stepping schemes

Two main families of time–stepping schemes can be used for solving initial value problems of nonsmooth dynamical systems : a) the event–detecting schemes (also termed event-driven schemes) and b) the event–capturing schemes (or shortly time–stepping schemes). By events, one must understand the time-instant when the solution lose its smoothness ; jumps in control law or in external applied forces, impacts, transitions between sticking and sliding are the main instances of nonsmooth events.

Event-detecting schemes These schemes are based on the hypothesis that the events are well–separated in time (at least at the machine precision) and that we can detect them with accuracy. The principle of integration is simple. The non–impulsive dynamics (16) is solved with any efficient solvers for DAE between two events. Over such a smooth period of time, the index sets of active constraints is assumed to be constant. The constraints are therefore treated as bilateral constraints. At an event, we solve the impact equations (15) and we reinitialize the DAE solver with new Cauchy conditions.

The advantages (denoted by $\oplus$) and the weaknesses (denoted by $\ominus$) of the event–detecting schemes are summarized below :

- ⊕ higher order integration over smooth period of evolution
- ⊖ No general proof of convergence
- ⊖ Extreme sensitivity to the numerical thresholds used for the event detection and the computation of the contact status (sticking, sliding, imminent take–off, ...)
- ⊖ Index reduction of the constraints sometimes difficult especially for the 3D contact with Coulomb's friction.
- ⊖ Not able to deal with accumulation of impacts in finite time.

In summary, these schemes are well–suited for systems where the density of events is low and with well-separated events. Although the principle is straightforward but the efficiency and the robustness of codes are very difficult to ensure in practice. In [10,25] and [11, Chapter 8], some examples of implementation of these schemes are detailed mainly on the 2D case.

Event-capturing schemes On the contrary, these schemes are based on the direct integration of the MDI (12) or equivalent formulations. The time-step length is not chosen with respect to the events, but depends on the local error of accuracy as in standard integration of smooth systems. The advantages and the weaknesses of the event–capturing schemes are summarized as follows :

- ⊕ several convergence proofs [26–29].
- ⊕ robustness and stability on large–scale problems
- ⊕ integration of problems with finite accumulation of impacts
- ⊖ low order of global accuracy ($\leqslant 1$)

These schemes are really efficient for systems where the density of events is large. Although the order is low, their robustness and their stability enable complex applications like granular matter [30], kinematic chains with clearances [31] and flexible multi–body systems that mix solid and rigid bodies. The two main instances of such schemes are the Moreau–Jean time–stepping scheme [1,2,4,5] and the Schatzman–Paoli scheme [27,28].

4.2 Moreau–Jean event–capturing scheme

In order to specify the main ideas on the implementation of the Moreau–Jean event–capturing scheme, we provide the reader with its formulation with unilateral contact in the Lagrangian setting. Let us consider a time–discretization $\{t_k\} \in [t_o,T], k \in \{1 \dots N\}$, The scheme can be written for one time–step of length $h = t_{k+1} - t_k$ and a set of contact points $I = \{1 \dots m\}$ as

$$\begin{cases} M(q_{k+\theta})(v_{k+1} - v_k) - h\tilde{F}_{k+\theta} = G^{\top}(q_{k+\theta})P_{k+1}, & \text{(27a)} \\ q_{k+1} = q_k + hv_{k+\theta}, & \text{(27b)} \\ U_{k+1} = G(q_{k+\theta})\, v_{k+1} & \text{(27c)} \\ -P_{k+1} \in N_{T_{\mathbb{R}^m_+}(\tilde{g}_{\mathrm{N},k+\gamma})}(U_{k+1} + eU_k), & \text{(27d)} \\ \tilde{g}_{\mathrm{N},k+\gamma} = g(q_k) + h\gamma U_k, \quad \gamma \in [0,1]. & \text{(27e)} \end{cases}$$

where $\theta \in [0,1], \gamma \geqslant 0$ and $x_{k+\theta} = (1-\theta)x_{k+1} + \theta x_k$. The value $\tilde{g}_{\mathrm{N},k+\gamma}$ is a forecast of the possible active constraints in the time–step. The term $\tilde{F}_{k+\theta}$ represent a consistent discretization of the applied forces for which details can be found in [11, Chap. 10]. The basic ingredients which makes the efficiency of the scheme, are i) the fully implicit treatment of the constraints at the velocity level and ii) the formulation with impulses and velocities as primary unknowns. Indeed, the variable P_{k+1} is not homogeneous to a force but to an impulse over the time–step that is to say :

$$P_{k+1} \approx \int_{]t_k,t_{k+1}]} \mathrm{d}I. \tag{28}$$

This feature makes the scheme consistent and stable even when (a large number of) impacts occur. This scheme has proved his efficiency on numerous applications. To cite of few of them, we refer to [2,24, 32–41].

4.3 Solving the discrete frictional contact problem

In order to further simplify, let us consider the linearized problem associated to (27) by assuming the dynamics is linear (or linearized) as follows

$$\left\{\begin{array}{l} M\mathrm{d}v + (Cv^+ + Kq)\mathrm{d}t = F_{ext}(t) + G^\top \mathrm{d}I, \\ U = Gv, \\ \text{if } g(q) \leqslant 0 \text{ then } -\mathrm{d}I \in N_{C^*}(\hat{U}^+ + eU_{\mathrm{N}}^- n) \text{ otherwise } \mathrm{d}I = 0. \end{array}\right. \tag{29}$$

The Moreau–Jean scheme (27) for a set of contacts $\alpha \in I$ reads

$$\left\{\begin{array}{l} \widehat{M}(v_{k+1} - v_{\text{free}}) = p_{k+1} = \sum_\alpha p^\alpha_{k+1} \\ U^\alpha_{k+1} = G^\alpha v_{k+1}; \qquad p^\alpha_{k+1} = G^{\alpha,\top} P^\alpha_{k+1} \\ \widehat{U}^\alpha_{k+1} = \left[U^\alpha_{\mathrm{N},k+1} + eU^\alpha_{\mathrm{N},k} + \mu^\alpha \, ||U^\alpha_{\mathrm{T},k+1}||, U^\alpha_{\mathrm{T},k+1}\right]^T \\ \text{for all } \alpha \in I_{a,k+1} \\ \mathbf{C}^{\alpha,*} \ni \widehat{U}^\alpha_{k+1} \perp P^\alpha_{k+1} \in \mathbf{C}^\alpha \\ \text{for all } \alpha \notin I_{a,k+1} \\ P^\alpha_{k+1} = 0 \end{array}\right. \tag{30}$$

where $I_{a,k+1} = \{\alpha \mid y_{k+\gamma} \leqslant 0\} \subset I$ denotes the set of forecast possible active constraints and $\widehat{M} = \left[M + h\theta C + h^2\theta^2 K\right]$ denotes the iteration matrix and v_{free} the free–flight velocity given by

$$v_{\text{free}} = v_k + \widehat{M}^{-1}\left[-hCv_k - hKq_k - h^2\theta K v_k + h\left[\theta(F_{\text{ext}})_{k+1}) + (1-\theta)(F_{\text{ext}})_k\right]\right]. \tag{31}$$

The problem appears as a Second-Order Cone Complementarity Problem (SOCCP) which can be generically defined as

$$\left\{\begin{array}{l} w = Wz + q \\ \mathbf{K}^* \ni w \perp z \in \mathbf{K} \end{array}\right. \qquad \text{(SOCCP).} \tag{32}$$

Without friction ($\mu = 0$), the Coulomb cone is reduced to $\mathbb{R}_+$ and the SOCCP reduced to a Linear Complementarity Problem (LCP), defined as

$$\left\{\begin{array}{l} w = Wz + q \\ 0 \leqslant w \perp z \geqslant 0 \end{array}\right. \qquad \text{(LCP).} \tag{33}$$

If additionally, the matrix W is semi–definite positive and symmetric, it amounts to solving the following standard convex Quadratic Programming (QP)

$$\left\{\begin{array}{ll} \min_z & \frac{1}{2}z^\top W z + z^\top q \\ \quad \text{such that} & z \geqslant 0 \end{array}\right. \qquad \text{(QP).} \tag{34}$$

The latter problems (32), (33) and (34) are very well–studied problems in numerical optimization. Most of the numerical methods for the unilateral contact and friction are extensions of standard algorithms of optimization. The case of friction is nevertheless not straightforward, since it implies nonsmoothness and non convexity, which remains a major challenge for the numerical algorithms. Furthermore, in most of the applications, the constraints are linearly independent which introduces indeterminacy and redundancy in the multipliers. Indeterminacy renders the numerical solving difficult and destroys most of the time the conditioning of the problem.

To conclude with, we give a list of the most–well know algorithms to solve discrete frictional contact problems that can be found in SICONOS/NUMERICS.

- Generalized Newton methods [42, 43]
- Projection/splitting methods for variational inequalities [44–47]
- Fixed point methods on the friction threshold (Tresca's friction approach) [48–50]
- Iterative convex minimization problems [21–23]

Reviews of these methods can be found in [11, 51, 52] and [12].

5 Software codes

A given piece of software is usually dedicated to a single type of computation. The computation step is to be clearly distinguished from the preparation of the data, and from the exploitation of the results, even if these three steps are closely related. Indeed, the pre-processing formats the input data for the software, and the post-processing parses the output data of the software to provide the desired results to the user.

Furthermore, for a better understanding of what a simulation software does, the computation phase is also divided into several components :

- modelization, i.e. the kind of modelizations the software can handle (linear/non-linear, rigid/deformable)
- simulation, i.e. the kind of simulation (linear/non-linear, quasi-static/(non-)smooth dynamic) the software can handle, and the time integrators available for each of them.
- contact features :
  - contact detection, i.e. the way interactions between the modelizations are generated on the fly
  - contact laws, i.e. the list of behaviors that are available (friction, newton impact, cohesive, etc.)
  - contact solvers, i.e. the resolution algorithms that are available (Gauss-Seidel, Jacobi, Alart-Curnier, etc)

Finally, an important feature of research software (and especially in the coupling matter) is the way the user can interact with the software during simulation and how he can access its database of objects, models, parameters, etc.... Thus in the following the possibilities of each software codes is detailed.

5.1 Code_Aster

The Code_Aster software is a Finite Element Software dedicated to the simulation of mechanical or thermal bodies. It is developed by EDF R&D as a standalone program, and also as an integrated module in the Salomé suite of software.

The Salomé platform provides an unified graphical framework for the pre-processing, execution and post-processing of heavy duty Code_Aster computations. The pre-processing tools of Salomé can be used for the definition of geometries and meshes, and for the association the desired materials and models to (groups of) elements. The preferred output of the software is usually a file in the MED open source format. This file can be read within the graphical interface of Salomé. Code_Aster is acknowledged as a powerful and complete tool for FEM simulation. About 360 finite elements are availabe and over 95 constitutive material laws are defined.

The numerical methods available are static, quasi-static or smooth dynamic, each one being either linear or non-linear. The set of possible time integrators is : HHT, θ-scheme in displacement or velocity and Krenk's scheme.

There is a possibility of contact resolution within Code_Aster : groups of elements may be tagged so that they cannot cross each other and contact with friction can be taken into account during computation using active stresses, Lagrange multipliers or augmented Lagrangian method.

Code_Aster is written in Fortran77 but this 'Core' part is completely hidden from the user. The user has access to a set of commands which are run in a python interpreter. This set of commands provides access to high level functionality of the software. Accessing low level functionality or accessing the database directly from Python is not straightforward, but has been proven to dramatically reduce overhead, acknowledging the fact that the Aster Python setter/getter on the database lack efficiency due to memory copy.

5.2 Siconos

SICONOS is an Open Source scientific software primarily targeted at modeling and simulating nons-mooth dynamical systems in the most generic sense :

- Mechanical systems (Rigid body or solid) with Unilateral contact and Coulomb friction as we find in Non-smooth mechanics, Contact dynamics or Granular material.
- Switched Electrical Circuit such as electrical circuits with ideal and piecewise linear components Power converter, Rectifier, Phase-locked loop (PLL) or Analog-to-digital converter
- Sliding mode control systems

– Other applications are found in Systems and Control (hybrid systems, differential inclusions, optimal control with state constraints), Optimization (Complementarity systems and Variational inequalities), Biology (Gene regulatory network), Fluid Mechanics and Computer graphics...

The software is based on 4 main components :

– SICONOS/NUMERICS (C API). Collection of low-level algorithms for solving basic Algebra and optimization problem arising in the simulation of nonsmooth dynamical systems : Linear complementarity problem (LCP) Mixed linear complementarity problem (MLCP) Nonlinear complementarity problem (NCP) Quadratic programming problems (QP) Friction-contact problems (2D or 3D) (Second-order cone programming (SOCP)) Primal or Dual Relay problems
– SICONOS/KERNEL. C++ API that allows one to model and simulate the NonSmooth dynamical systems. it contains : Dynamical systems classes (first order one, Lagrangian systems, Newton-Euler systems) and Nonsmooth laws (complementarity, Relay, FrictionContact, impact)
– SICONOS/MECHANICS This component is a modeling toolbox for multi-body systems based on external collision and geometry libraries.
– SICONOS/FRONT-END generated python bindings for Numerics, Kernel and Mechanics, with a special support for Siconos data structures.

The development team focused on the development of the simulation and modeling tools which are as most as possible reusable for different scientific context.

Any specific pre/post-processing tools has not been developed. Dynamical systems and interactions (unilateral constraints) classes provided by Siconos/Kernel may be configured with the help of plugins or by class derivation. For high–level applications SICONOS relies on external software and provide with an efficient computational engine. Concerning the mechanical modeling, it is possible to simulate linear or non-linear rigid or deformable bodies.

The numerical simulation strategies available in Siconos/Kernel are quasi-statics, smooth dynamics, non-smooth dynamics, each one being either linear or non-linear and simulated with event–capturing or event–detecting schemes. Available time integration schemes are Moreau–Jean scheme, Schatzman–Paoli scheme, Nonsmooth Newmark scheme [53, 54], Odepack suite and DAE solvers developed by E. Hairer [55].

Inside SICONOS/KERNEL, the description of the user model relies internally on a primary graph with dynamical systems as nodes and interactions as edges. During time evolution, the determination of active constraints corresponds to the building of sub-graphs of the primary graph. In the case of problems formulated in local coordinates an adjoin graph transformation is dynamically applied to sub graphs in order to drive the computation of the components of the optimization problem given to Numerics solver.

The primary graph structure may be entirely specified once by the user, or it may also be rebuilt during the simulation, as, for example, the result of a contact detection broad-phase. Links with pre/post processing software for collision detection (Bullet contact detection [56]) and CAD libraries (OpenCascade [57] and PythonOCC [58]) has been set–up. The latter link enables the contact detection between bodies defined by B-Rep (splines, nurbs, ...) and then to compute gaps and local frame at contact. Such a geometrical representation of the data allows to use realistic modeling of surfaces with large sliding. The possible impact laws for mechanics are complementary condition, impact with or without friction. Finally, the available contact solvers are those listed in Section 4.3.

SICONOS is written in C++ and has an object oriented architecture, there is no graphical user interface (GUI). A simulation is run through a C++ or Python script in which are gathered the objects creations and the method calls on these object that make up the computation. Furthermore, unlike CODE_ASTER, not only the coarse level of command driving is available, but the time loop can be exploded in every step to allow for an interactive simulation. In the same way the whole database is accessible through copy or reference, allowing for a efficient access and aimed at designing any post-processing functions.

More details can be found at `http://siconos.gforge.inria.fr`.

5.3 LMGC90

The LMGC90 software is dedicated to the simulation of multi-contact systems. It first aimed at the simulation of granular materials and thus focused its main features on the simulation with a dynamic contact network. This lead to other domains of application like masonry structure simulation, rock failure

simulation. It also provides features for multi-physics coupling (fluid-grains, or thermo-mechanics simulations). The software provides a Python pre-processor allowing to efficiently generate data set dedicated to the different application fields : granular, masonry, meshes. Concerning post-processing, paraview files are generated. The software has no graphical interface and works as a list of Python commands. This allows to finely tune the simulation since the time loop or the iteration loops of the solver can be exploded within the Python script.

In term of functionality LMGC90 can :

- model linear rigid bodies (2/3D), linear or non-linear deformable bodies using a Finite Element discretization.
- use time stepping schemes for both linear or non-linear quasi-static, smooth dynamic, non-smooth dynamic simulation.
- use θ-scheme in velocity, gear and verlet time integrators.
- use Non-Linear Gauss Seidel, Jacobi and conjugated projected gradient contact solvers.

A large effort in the development of the software was put in contact detection features. A wide range of primitive geometries is available and can be combined to define complex boundaries. It is important to note that these geometries are always discretized in space. The software also provide contact detection for most pair of these primitive contact geometries and generates the corresponding interactions in form independent of the involved shapes. Furthermore the main contact solver used is the Non-Linear Gauss-Seidel, which is very robust. This allows to have a large set of contact laws available, including dynamic friction, cohesive laws, and various laws with internal parameters.

As already stated above, LMGC90 software is designed as a Python module providing a set of functions allowing fine tuning of the simulations steps. There are also a whole set of accessors to the database using copy in most case and references on some occasion for efficiency's sake. This Python layer hides the core of the software which is written in Fortran 90/95. The structure of the software uses extensively the concept of module allowing to have an architecture very close to an object oriented program. Concerning the resolution of linear systems, a mechanism of bindings is available to allow the use of efficient solver libraries (like MUMPs or LAPACK).

6 Software coupling

The previous section which presented different software applications gave some insights on the strong points of each :

- CODE_ASTER : efficient FEM computation with extended set of behavior laws integrated in a whole pre/post-processing suite.
- SICONOS : large modeling of dynamical systems and lots of integrator and contact solvers.
- LMGC90 : wide set of contact detection algorithms and many contact laws and a pre-processing tool dedicated to domains of applications.

Again the point of this work is to unify these complementary features in a single environment. To do so, two possible levels of coupling are considered : point to point (or strong) couplings, i.e. using the commands of each software in a common environment (see figure 4) ; or a higher level (or weak) coupling using a designed API (see figure 5). The first solution is the easiest one to develop but imposes to write a simulation case for each combination of software applications and sometimes some code dedicated to the specific coupling. Thus, the addition of a new software in the coupling environment would incur an exponential growth of the code to be written, since the combinatorial increases with each new software. The second level of coupling is more generic, and consists in defining an API that is generic enough, but can still be accommodated by every software codes. Our hope is that the three pieces of software presented here be representative enough of the field, so that any new software just has to plug into the API to be coupled with any of the other software application.

Eventually, each software application is used through a script gathering a list of commands. An important design decision is the environment in which the coupling will be written. The Code_Aster software is basically a Python interpreter enriched with Code_Aster commands, and there is little choice but to run this interpreter and write Python functions for the coupling. The SICONOS main API is in C++, but there is a possibility to generate a Python interface. The LMGC90 main API is in Python, but there is a possibility to generate a C++ interface... So in case of coupling between SICONOS and LMGC90

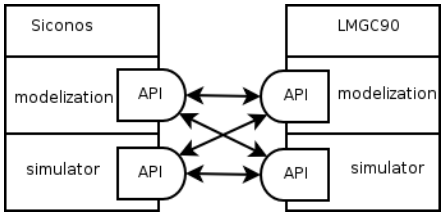


Fig. 4 – Point to point coupling

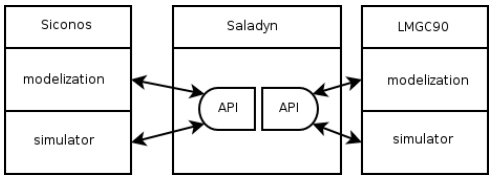


Fig. 5 – High level coupling

using either C++ or Python scripts is possible.

6.1 Strong coupling

The point-to-point coupling, in spite of their lack of genericity, as stated above, have been tried out as a prototype to help with the design of the generic API of the higher level coupling.

When coupling with SICONOS, the main issue was to initialize the database from the other piece of software, and regularly update the state of the corresponding data in the software. But this was simple enough thanks to its flexible architecture which allows to initialize the database within the running script. On the contrary, CODE_ASTER and LMGC90 must read some input files to initialize their database, each using a different format. In this case, the main issue for the pre-processing phase is the consistency of geometries and meshes between the two software applications. Then, during a simulation, consistency must also be enforced at each field transfer (ordering of nodes, cells, layouts of degrees of freedom).

After that, only a choice of accessors were to be smartly called (or implemented if they did not exist) in order to allow the transfer of data between the two software applications. In this case copy was always used, the first reason being that Code_Aster has no choice but to copy to (or from) its database. SICONOS extensively uses reference and pointers. Since some data may be referenced through pointers in LMGC90, it could be desired to make the software share memory. Although this may be possible on some vectors, this solution is not feasible when matrices are involved : first of all, since Fortran and C++ do not order their matrices in the same way by default (Fortran in column major whereas C/C++ is row major), the exchange implies a transposition when passing from one to the other ; furthermore, they are always some small differences in the formalisms used to described the problem to be solved. In the end a slight transformation on the data is often needed anyway, forbiding the direct use of references and pointers.

Finally SICONOS/LMGC90 and LMGC90/CODE_ASTER couplings have been implemented using this strong coupling method.

6.2 Weak coupling

This high level coupling is done through the definition of a common API. Python naturally arose as the language of choice for writing this intermediate layer, since all three software applications have a python interface. This layer will be referred to as the SALADYN toolbox from now on. Python allows object oriented programming, which is used within the toolbox to define the different components of a computation in a anonymous form, where the specificity of each underlying software application is hidden using the inheritance mechanism. This method introduces more genericity since a coupled computation uses only SALADYN classes/methods/functions and not a heavy mix of commands coming from different software applications assorted with blocks of code to reshape the data. Any new software application just has to inherit the SALADYN classes and fulfill the API requirement to be coupled with other software.

To be able to couple the different software applications the first step in the design of the SALADYN toolbox was to split the different components of the software. Each software code is split in term of *modeling* and *simulation*. While *modeling* defines how to compute the physics (mass, rigidity, external forces, etc) and the interactions (contact laws, contact points, etc), *simulation* will gather all time integration and solving features. SALADYN must access these two different parts of each software applications in an independent manner. The general organization of the toolbox is described on figures 3 and 6.

There is always only one *simulation* object. At the best in case of CODE_ASTER/*LMGC90* coupling an *InteractionSolver* object is aggregated to the *Simulation* one, but the features they rely on are different.

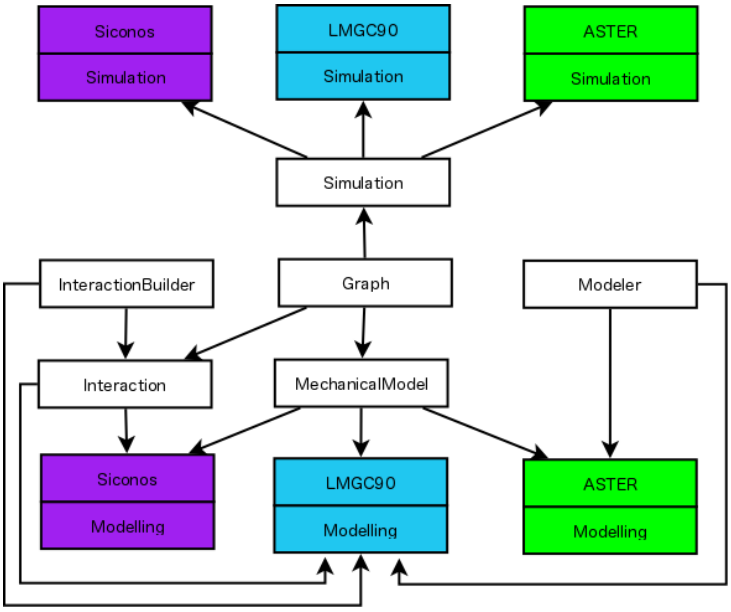


Fig. 6 – Saladyn toolbox

The *CollisionHandler* object allows to generate the list of interactions within the SALADYN's graph. The detail of all relevant classes on the design of the toolbox are given figure 7.

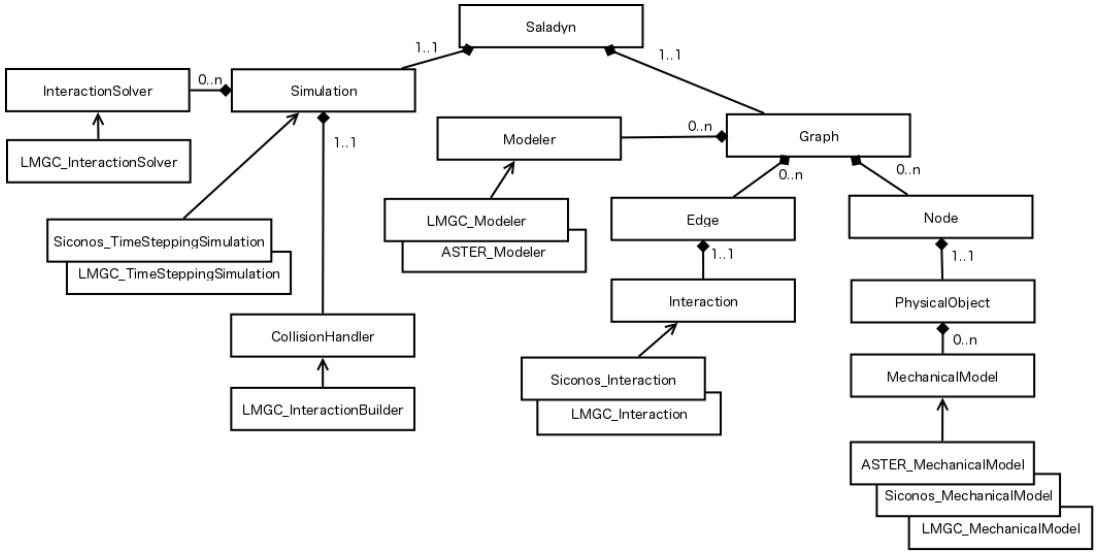


Fig. 7 – Saladyn classes

Using the SALADYN toolbox it is possible to couple the three software applications in a single computation where CODE_ASTER defines the deformable bodies, LMGC90 defines rigid bodies and performs the contact detection and SICONOS is in charge of the time integration and the contact solving.

The main drawback of this coupling method is that, to ensure the data transfers in the software, some specific functions must be designed. A possible way would be to explicitly state the source and destination software for each data transfer. This seems simple but the set of functions doing the transfer would be completely dedicated to one pair of software code, thus the main drawback of combinatorial mentioned in the point to point coupling section would still be present.

The method chosen, which ensures genericity, is to use an intermediate data structure as a buffer, and having functions to retrieve data from a given software code and load it into SALADYN, and functions that push data from SALADYN to the different software applications. This method is more in agreement with the philosophy of a weak coupling scheme, even if it adds an intermediate data structure and increases the number of copies at each computation step. This is particularly true when CODE_ASTER is involved. It has already been stated that references and pointers are usable with SICONOS and LMGC90, but cannot

be plugged directly from one to another, because of differences in the data structures and formulations. To reduce this loss of efficiency due to the large number of copies, the memory could be shared between Saladyn components and one of the software. This could be used as a mean to reduce the number of copies of one level and use the data structure of SALADYN to do the shapeshift of data. The differences in architecture brings another difficulty, usually the different functions of LMGC90 and CODE_ASTER apply to the whole set of bodies initialized in the software. But SICONOS preferably uses methods of its objects, thus work body by body. That is why a class of *Modeler* has been designed and help with unifying this difference.

Furthermore, the dynamic modeling adaptability has been studied within the ANR project. Some dedicated classes were designed within the coupling environment to help with this new feature without modifying the underlying software applications. These classes are the *PhysicalObject* and *Mechanical-Model*. Where the first one must represent the real object without taking into account the modeling used, the later does represent the object for a modeling type (rigid, deformable, etc). The *PhysicalObject* class would provide a set of method allowing to perform the modeling switch.

6.3 Coupling through binding

A third coupling method, at a complete different scale, has also been tested. It is, as a matter of fact, a binding. As stated in section 5.2 SICONOS is divided in four parts, the NUMERICS one which gathers low level algorithms is independent of the modeling of the problem to solve and is designed as a stand-alone library. It would be a great benefit to LMGC90 if it could call the SICONOS/NUMERICS library. Using some recent features of Fortran, it is now easy to bind C code in Fortran. Thus the only work was to add the possibility to call external contact solver within LMGC90.

The main point of this binding is that only little data transfer is needed and it can be designed to be efficient. In this way LMGC90 can benefit from the development of contact solvers made by the INRIA. This also allows to have a real comparison of the different layers of the software applications.

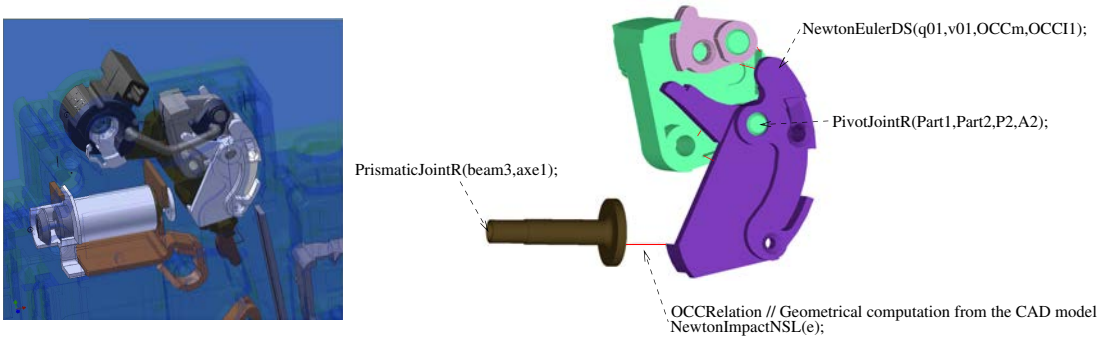
7 Examples

7.1 Simulation of clearances in electrical circuit breakers

In this final section, we give an insight of the usefulness of the proposed approaches for the virtual prototyping of electrical circuit breakers designed by the company Schneider Electric. The model that we considered in a C60 circuit breaker, which is a domestic low voltage circuit breaker depicted on Figure 8. The mechanism is only composed of 7 moving bodies, but 12 contacts come into play when the breaker switches off. Furthermore, when the mechanism is in closed position, the equilibrium is guaranteed by means of Coulomb's friction in the contact of the two bodies described in Figure 8(c). When the breaker opens the circuit, a lot of events (impacts, stick-slip transitions, ...) are observed in experimental setups. A rather complete description of its behavior and its nonsmooth modeling in 2D can be found in [25]. The study in 3D with clearances that is performed with the projected event-capturing scheme [31] allows us to accurately study the effect of the clearances in joints on the out-of-plane motion of the breaker. Furthermore, it helps to state on the stability and the robustness of the fundamental properties of the circuit breaker with respect to the size of the clearances. Such studies are not possible with standard event-driven schemes which have a lot of difficulties to deal with 3D frictional contacts and a bunch of events. Indeed, the presence of clearances in joints generates a lot of finite accumulation of impacts and numerous stick-slip transitions. Such studies are also difficult with standard event-capturing schemes for which the violation of constraints come into play with the characteristic lengths of the clearances. In THE SALADYN TOOLBOX, a extension of an event–capturing with projection onto the constraints [31] has been implemented to efficiently this problem.

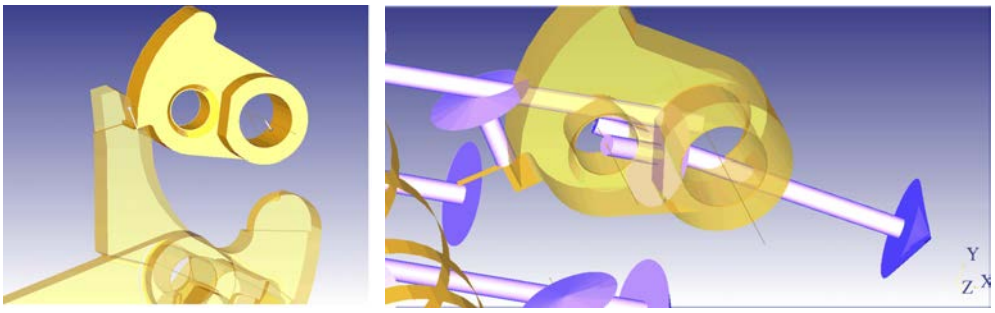
7.2 Simulation of rockfill dams

The toolbox has been successfully used to simulate rock dam where each deformable body is modeled by Code_Aster, and rigid ones by LMGC90. Contact detection and simulation are performed by LMGC90.



(a) CAD view of the mechanism

(b) Modeling as a multibody system



(c) Contact and joints with clearance

(d) An illustration of contact forces.

Fig. 8 – C60 electrical circuit breaker mechanism. Courtesy of Schneider Electric

We briefly describe the characteristics and results of the computation. For an extensive description and results, the reader is refered to [59] and [60].

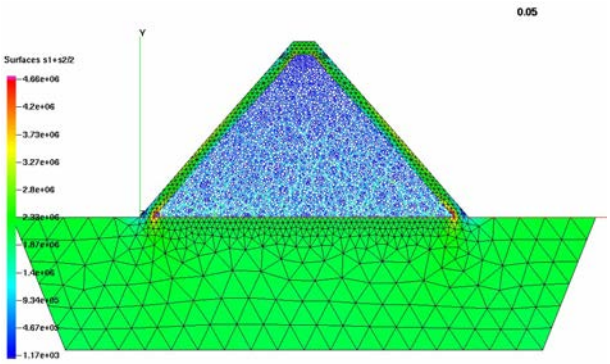


Fig. 9 – Rock dam simulation

The overall geometry is as in Fig. 9. Continuous media are modeled using linear elastic behaviors, while the bulk of rigid bodies is modeled by a pile of 10000 disks with randomized diameters between 0.1 and 0.8 m. The interaction laws are unilateral contact-friction laws, with friction coefficient 0.5.

Three phases have been modeled :

- layer by layer construction of the dam : in order to simulate the real construction process, ensure the correct compaction of the rock pile, and avoid dynamics effects in these computations, we computed the compaction of ten layers of bulk elements, one after the other ;
- flooding of the dam, in which the time scale of the flooding is much greater than the dynamic reactions of the dam ;
- and finally the effect of the sismic load has been computed, the seism being imposed on the boundary of the domain (see references [59], [60] for details).

For the two first phases, quasi-static computations have been used, while for the third phase a fully dynamic simulation has been carried out, imposing a 11 sec. sismic signal (lower parts in Fig. 10) on the boundaries.

As a summary of the results obtained, we present the temporal evolution of the total number of contacts, and of the total kinetic energy in the rock pile in Fig. 10 during the seism, and a close up on the top of the dam, after seism, in Fig. 11.

The results presented here, although the materials follow a linear elastic law, show that the non-linearity introduced by the contact-friction laws play an important role in the overall behavior of the rock pile. Previous purely FEM simulations of the rock pile (see [61]) could not represent these non-linearities : irreversible displacements appear, as well as a settlement of the pile due to the special behavior of discrete elements (Fig. 11).

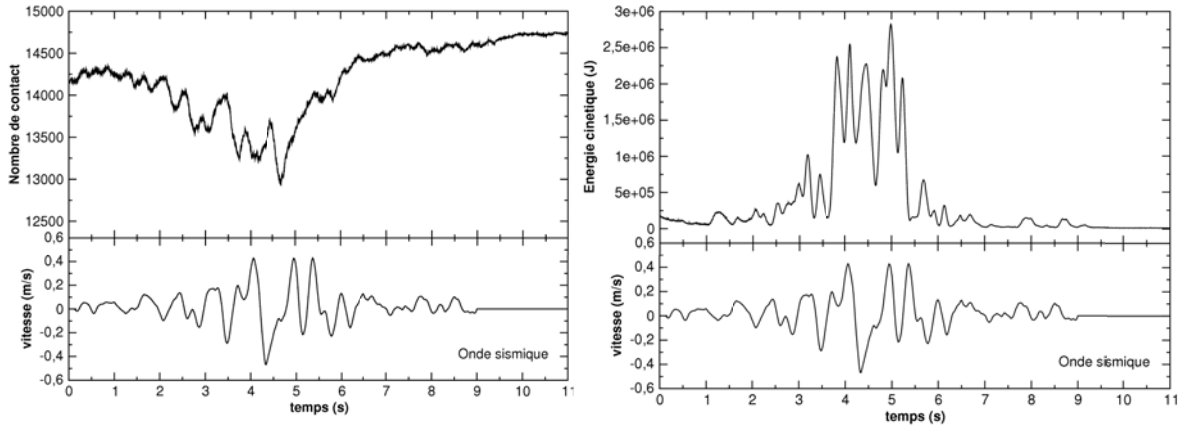


Fig. 10 – Total number of contacts in the rock pile during seism

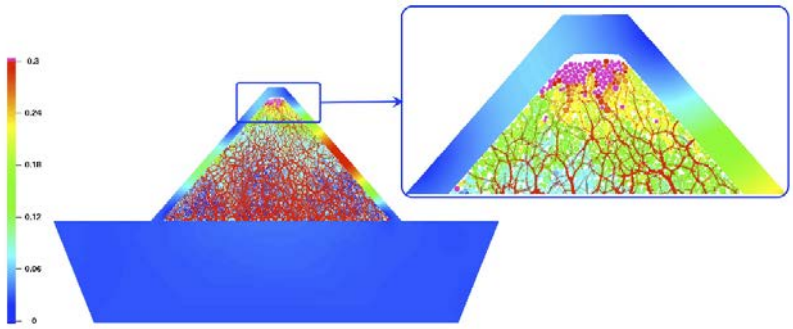


Fig. 11 – Rearrangement at top of dam after seism

8 Conclusion

Main results The main results are given by the industrial applications : efficient simulation tools and realistic simulations. On the EDF side, the simulations of rock-fill dams have been carried out with a better modeling of the core of the dam where the contact between blocks is solved with an implicit time-discretization method. On the Schneider Electric side, 3D simulations of circuit breakers have been performed using the CAD model. These simulations make possible the study of the influence of fabrication clearances on the behavior of complex mechanisms. This results yield a new PhD Thesis granted by Schneider Electric on this subject and a CIFRE PhD with the ANSYS Corporation.

Additional informations on : `http://saladyn.gforge.inria.fr/`

Thanks

This work is supported by SALADYN ANR project `http://saladyn.gforge.inria.fr` of the program COSINUS ANR-08-COSI-014-01.
The authors are grateful to F. Voldoire, A. Assire, E. Frangin, O. Bonnefon and B. Brogliato for their valuable support to this work.

Références

[1] J.J. Moreau. Unilateral contact and dry friction in finite freedom dynamics. In J.J. Moreau and Panagiotopoulos P.D., editors, *Nonsmooth Mechanics and Applications*, number 302 in CISM, Courses and lectures, pages 1–82. CISM 302, Spinger Verlag, Wien- New York, 1988.

[2] J.J. Moreau. Some numerical methods in multibody dynamics : Application to granular materials. *European Journal of Mechanics - A/Solids*, supp.(4) :93–114, 1994.

[3] M. Jean and J.J. Moreau. Unilaterality and dry friction in the dynamics of rigid bodies collections. In A. Curnier, editor, *Proc. of Contact Mech. Int. Symp.*, volume 1, pages 31–48. Presses Polytechniques et Universitaires Romandes, 1992.

[4] J.J. Moreau. Numerical aspects of the sweeping process. *Computer Methods in Applied Mechanics and Engineering*, 177 :329–349, 1999. Special issue on computational modeling of contact and friction, J.A.C. Martins and A. Klarbring, editors.

[5] M. Jean. The non smooth contact dynamics method. *Computer Methods in Applied Mechanics and Engineering*, 177 :235–257, 1999. Special issue on computational modeling of contact and friction, J.A.C. Martins and A. Klarbring, editors.

[6] Frédéric Dubois, Michel Jean, Mathieu Renouf, Rémy Mozul, Alexandre Martin, and Marine Bagneris. Lmgc90. In *Colloque CSMA 2011*, pages 1–8, 2011.

[7] Frédéric Dubois and Rémy Mozul. Lmgc90. In *Colloque CSMA 2013*, pages 1–8, 2013.

[8] V. Acary and F. Pérignon. An introduction to Siconos. Technical Report TR-0340, INRIA, http ://hal.inria.fr/inria-00162911/en/, 2007.

[9] V. Acary and B. Brogliato. *Numerical Methods for Nonsmooth Dynamical Systems : Applications in Mechanics and Electronics*, volume 35 of *Lecture Notes in Applied and Computational Mechanics*. Springer Verlag, 2008.

[10] F. Pfeiffer and C. Glocker. *Multibody Dynamics with Unilateral Contacts*. Non-linear Dynamics. John Wiley & Sons, 1996.

[11] Vincent Acary and Bernard Brogliato. *Numerical methods for nonsmooth dynamical systems. Applications in mechanics and electronics.* Lecture Notes in Applied and Computational Mechanics 35. Berlin : Springer. xxi, 525 p. , 2008.

[12] C. Studer. *Numerics of Unilateral Contacts and Friction. – Modeling and Numerical Time Integration in Non-Smooth Dynamics*, volume 47 of *Lecture Notes in Applied and Computational Mechanics*. Springer Verlag, 2009.

[13] B. Brogliato. *Nonsmooth Mechanics : Models, Dynamics and Control*. Springer-Verlag, London, 2nd edition, 1999.

[14] J.J. Moreau. *Fonctionnelles Convexes*. Séminaire sur les équations aux dérivées partielles, subventionné par le CNRS, Collège de France, Paris., 1967.

[15] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[16] J.J. Moreau. Bounded variation in time. In J.J Moreau, P.D. Panagiotopoulos, and G. Strang, editors, *Topics in Nonsmooth Mechanics*, pages 1–74, Basel, 1988. Birkhäuser.

[17] M. Schatzman. Sur une classe de problèmes hyperboliques non linéaires. *Comptes Rendus de l'Académie des Sciences Série A*, 1973.

[18] M. Schatzman. A class of nonlinear differential equations of second order in time. *Nonlinear Analysis, T.M.A*, 2(3) :355–373, 1978.

[19] J.J. Moreau. Evolution problem associated with a moving convex set in a Hilbert space. *Journal of Differential Equations*, 26 :347–374, 1977.

[20] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer, 1996.

[21] F. Cadoux. *Analyse convexe et optimisation pour la dynamique non–reguliére*. PhD thesis, Université Joseph Fourier, Grenoble I, 2009.

[22] V. Acary, F. Cadoux, C. Lemaréchal, and J. Malick. A formulation of the linear discrete Coulomb friction problem via convex optimization. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 91(2) :155–175, 2011.

[23] V. Acary and F. Cadoux. *Recent Advances in Contact Mechanics, Stavroulakis, Georgios E. (Ed.)*, volume 56 of *Lecture Notes in Applied and Computational Mechanics*, chapter Applications of an existence result for the Coulomb friction problem. Springer Verlag, 2013.

[24] V. Acary and Y. Monerie. Nonsmooth fracture dynamics using a cohesive zone approach. Research Report RR-6032, INRIA, 2006.

[25] M. Abadie. Dynamic simulation of rigid bodies : Modelling of frictional contact. In B. Brogliato, editor, *Impacts in Mechanical Systems : Analysis and Modelling*, volume 551 of *Lecture Notes in Physics (LNP)*, pages 61–144. Springer, 2000.

[26] M.D.P. Monteiro Marques. *Differential Inclusions in Nonsmooth Mechanical Problems. Shocks and Dry Friction*. Progress in Nonlinear Differential Equations and their Applications, vol.9. Birkhauser, Basel, 1993.

[27] L. Paoli and M. Schatzman. A numerical scheme for impact problems I : The one-dimensional case. *SIAM Journal of Numerical Analysis*, 40(2) :702–733, 2002.

[28] L. Paoli and M. Schatzman. A numerical scheme for impact problems II : The multi-dimensional case. *SIAM Journal of Numerical Analysis*, 40(2) :734–768, 2002.

[29] L. Paoli. An existence result for non-smooth vibro-impact problems. *Journal of Differential Equations*, 211 :247–281, 2005.

[30] F. Radjaï and F. Dubois, editors. *Discrete–element modeling of granular materials*. Iste. John Wiley & Sons, 2011.

[31] Vincent Acary. Projected event-capturing time-stepping schemes for nonsmooth mechanical systems with unilateral contact and coulomb's friction. *Computer Methods in Applied Mechanics and Engineering*, 256 :224 – 250, 2013.

[32] F. Radjai and D.E. Wolf. Features of static pressure in dense granular media. *Granular Matter*, 1(1) :3–8, 1998.

[33] I. Bratberg, F. Radjai, and A. Hansen. Dynamic rearrangements and packing regimes in randomly deposited two-dimensional granular beds. *Physical Review E, Stat. Nonlin. Soft Matter Phys.*, 66(3) :(3 Pt 1) :031303, 2002.

[34] F. Radjai and S. Roux. Turbulentlike fluctuations in a quasistatic flow of granular media. *Physical Review Letters*, 89 :064302, 2002.

[35] A. Zervos, I. Vardoulakis, M. Jean, and P. Lerat. Numerical investigation of granular interfaces kinematics. *Mechanics of Cohesive-Frictional Materials*, 5(4) :305–324, 2000.

[36] M. Renouf, F. Dubois, and P. Alart. A parallel version of the Non Smooth Contact Dynamics algorithm applied to the simulation of granular media. *J. Comput. Appl. Math.*, 168 :375–38, 2004.

[37] M. Renouf and P. Alart. Conjugate gradient type algorithms for frictional multicontact problems : applications to granular materials. *Comp. Meth. Appl. Mech. Engrg.*, 194(18-20) :2019–2041, 2004.

[38] G. Saussine, F. Dubois, C. Bohatier, C. Cholet, P.E. Gautier, and J.J. Moreau. Modelling ballast behaviour under dynamic loading, part 1 : a 2D polygonal discrete element method approach. *Computer Methods in Applied Mechanics and Engineering*, 195(19–22) :2841–2859, 2006.

[39] V. Acary and M. Jean. Numerical simulation of monuments by the contacts dynamics method. In DGEMN-LNEC-JRC, editor, *Monument-98, Workshop on seismic perfomance of monuments*, pages 69–78. Laboratório Nacional de engenharia Civil (LNEC), November 12-14 1998.

[40] V. Acary and M. Jean. Numerical modeling of three dimensional divided structures by the non smooth contact dynamics method : Application to masonry structure. In B.H.V. Topping, editor, *The Fifth international Conference on Computational Structures Technology 2000*, pages 211–222. Civil-Comp Press, 6-8 September 2000.

[41] M. Jean, V. Acary, and Y. Monerie. Non-smooth contact dynamics approach of cohesive materials. *Philosopihical Transactions of the Royal Society, Mathematical, Physical and Engineering Sciences*, 359(1789) :2497–2518, 15 December 2001. Non-smooth Mechanics, A Theme Issue compiled and edited by F.G. Pfeiffer.

[42] P. Alart and A. Curnier. A mixed formulation for frictional contact problems prone to Newton like solution method. *Computer Methods in Applied Mechanics and Engineering*, 92(3) :353–375, 1991.

[43] P.W. Christensen and J.S. Pang. Frictional contact algorithms based on semismooth newton methods. In M. Fukushima & L. Qi, editor, *Reformulation - Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, pages 81–116, Dordrecht, 1998. Kluwer Academic Publishers.

[44] F. Facchinei and J. S. Pang. *Finite-dimensional Variational Inequalities and Complementarity Problems*, volume I & II of *Springer Series in Operations Research*. Springer Verlag NY. Inc., 2003.

[45] G. De Saxcé and Z.-Q. Feng. New inequality and functional for contact with friction : The implicit standard material approach. *Mech. Struct. & Mach.*, 19 :301–325, 1991.

[46] G. De Saxcé and Z.-Q. Feng. The bipotential method : A constructive approach to design the complete contact law with friction and improved numerical algorithms. *Mathemetical and Computer Modelling*, 28(4) :225–245, 1998.

[47] F. Jourdan, P. Alart, and M. Jean. A Gauss-Seidel like algorithm to solve frictional contact problems. *Computer Methods in Applied Mechanics and Engineering*, 155 :31–47, 1998.

[48] M. Jean and G. Touzot. Implementation of unilateral contact and dry friction in computer codes dealing with large deformations problems. *J. Méc. Théor. Appl.*, 7(1) :145–160, 1988.

[49] J. Haslinger. Least square method for solving contact problems with friction obeying coulomb's law. *Applications of mathematics*, 29(3) :212–224, 1984.

[50] J. Haslinger, I. Hlaváček, and J. Nečas. Numerical methods for unilateral problems in solid mechanics. In P.G. Ciarlet and J.L. Lions, editors, *Handbook of Numerical Analysis*, volume IV, Part 2, pages 313–485, Amsterdam, 1996, 1996. North-Holland.

[51] P. Wriggers. *Computational Contact Mechanics*. Springer Verlag, second edition, 2006. originally published by John Wiley & Sons Ltd., 2002.

[52] Laursen. T.A. *Computational Contact and Impact Mechanics – Fundamentals of Modeling Interfacial Phenomena in Nonlinear Finite Element Analysis*. Springer Verlag, 2003. 1st ed. 2002. Corr. 2nd printing,.

[53] Q. Z. Chen, V. Acary, G. Virlez, and O. Brüls. A Newmark-Type Integrator for Flexible Systems Considering Nonsmooth Unilateral Constraints. In Peter Eberhard, editor, *The Second Joint International Conference on Multibody System Dynamics - IMSD 2012*, Stuttgart, Germany, March 2012.

[54] Q.-Z. Chen, V. Acary, G. Virlez, and O. Brüls. A nonsmooth generalized-α scheme for flexible multibody systems with unilateral constraints. *International Journal for Numerical Methods in Engineering*, 2012. submitted.

[55] E. Hairer, S.P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer, 1993.

[56] Bullet Physics Library. `http://bulletphysics.org`.

[57] Open CASCADE Technology. `http://www.opencascade.org`.

[58] PythonOCC. 3D CAD/CAE/PLM development framework for the Python programming language. `http://www.pythonocc.org`.

[59] Hong-Phong Cao and Francois Voldoire. Modélisation numérique d'un barrage en enrochement dans Saladyn. Code_ASTER documentation, ANR Saladyn ANR-08-COSI-014-01, 2012. Available at `http://http://saladyn.gforge.inria.fr/documents/Livrable3b.3-4.pdf`.

[60] Hong-Phong Cao and Voldoire Francois. Simulation of dynamic interaction of continuus-divided media and application for dams and building. In *Proceedings of IMSD 2012*, pages 230–238, 2012. Stuttgart-Germany.

[61] Marc Kham. Projet CODHYBAR2 : Méthodologies de modélisation numérique de la construction par couche et de la mise en eau d'un barrage en terre avec la loi élastoplastique de Hujeux . Rapport HT-62-2009-03211-FR, EDF R&D, 2009.