Interactive Dynamic Simulator for Multibody Systems

Jean-Rémy Chardonnet

▶ **To cite this version:**

Jean-Rémy Chardonnet. Interactive Dynamic Simulator for Multibody Systems. International Journal of Humanoid Robotics, 2012, pp.1250021-1, 1250021-24. 10.1142/S0219843612500211 . hal-00771985

HAL Id: hal-00771985
https://hal.science/hal-00771985v1

Submitted on 17 Jan 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INTERACTIVE DYNAMIC SIMULATOR FOR MULTIBODY SYSTEMS

JEAN-RÉMY CHARDONNET
Arts et Métiers ParisTech, CNRS, Le2i, Institut Image, Chalon-sur-Saône, 71100, France
jean-remy.chardonnet@ensam.eu

We propose an interactive dynamic simulator for humanoid robots using constraint-based methods for computing interaction forces with friction. This simulator is a part of a general framework for prototyping called AMELIF and is a successful integration of physical models. We focus on optimizing the computation of the dynamics to obtain real-time simulations allowing multimodal interactivity. Our simulator has been validated in two ways: first by comparing real sensors' measures and simulated values, then through different scenarios of complex manipulation tasks on the HRP-2 humanoid robot, bringing new insights to interactive robotics.

Keywords: Dynamic simulation; contact with friction; humanoid robot; haptic interaction; manipulation; framework.

1. Introduction

Many humanoid robots have been presented in the past years, mostly for entertainment or for research purposes. These robots are aimed actually at evolving in all kinds of environments and at performing tasks in collaboration with humans. This implies physical interactions with the environment and humans, more specifically nonsmooth phenomena such as contact with friction, impacts, to be considered. These goals will be achieved as long as new theoretical models will be developed. However, achieving such tasks on real robots requires to first assess these models in simulation to avoid serious damages, as humanoid robots are generally very expensive and hardly mass-produced.

A wide range of simulators has been proposed in the literature and have been mostly designed for animation or motion generation of digital actors.[1–4] Interactivity with force feedback seems to be of minor interest despite the availability on the market of haptic devices. We can note the SOFA framework, originally designed for medical purposes, which includes interactivity[5] and real-time models for computing contact with friction based on GPU use,[6,7] or the simulator of Altomonte *et al.*[8]

Although they show attractive results in terms of time computation, they propose only models leading to visually realistic simulations and do not consider necessarily physical criteria that are of primary concern for the robotics community.

In the robotics community, simulators have been proposed for control or motion planning such as SAI[9,10] which integrates haptic feedback but no friction for contacts, OpenHRP[11,12] which is not interactive, GraspIt![13] specifically designed for grasping, the framework of Son *et al.*[14] which includes interactivity but does not explain how contact with friction is handled, or those of Hale *et al.*[15] and Nagasaka *et al.*[16] Most of the simulators we can find in the literature are however not designed for general prototyping purposes nor to simulate complex systems such as humanoid robots, including accurate and fast contact models while sensing force feedback. Moreover they are not freely available to the research community.

Simulators integrate physical models, especially dynamics models that can be decomposed into free dynamics and constrained dynamics. Algorithms for free dynamics, i.e., dynamics without considering any unknown external phenomena, have been widely explored for the last decades and many authors presented now well known easily implementable, parallelizable and fast algorithms.[17–25] We will not go into details in this paper as the literature provides enough references.

The most challenging part which still receives great attention is constrained dynamics, typically nonsmooth mechanics, contact, friction, impact. A complete overview of this part can be found in Brogliato *et al.*[26] and Acary and Brogliato.[27] Despite the abundance of references, many works dealing with contact problems illustrate examples with very simple scenarios and thus do not prove their validity by experiments. We think that such implementations hide actual problems of complexity, robustness and stability that may recall into question hypotheses and proposed algorithms. In the ideal case, considering additional aspects of the problem may be required for actual purpose implementations.

Here we propose a complete interactive realistic dynamic simulator, i.e., that integrates physically consistent contact with friction while sensing force feedback, that can be applied to many kinds of multibody systems, including humanoid and android robots, virtual avatars and systems with deformable skins. The main contribution of this simulator lies on a successful integration of different theoretical models in a robust framework to handle complex simulation cases. Especially, compared to OpenHRP, that was formerly used, we wanted to prove that (i) constraint-based methods were liable and more adapted to contact simulation than penalty-based methods, (ii) it is possible to get computation time that allows to interact online with virtual environments through a haptic interface and that it is completely possible to unify haptic sensing and dynamic simulation, and (iii) we are able to use our simulator in various situations, even highly complex and extreme scenarios, and so enhance the level of development of new applications in humanoid robotics.

We will present in the next section the contact modeling, then the computation of constrained-based dynamics with an overview of our simulator architecture, we will show in Part 4 the integration of a haptic interface for sensing force feedback.

We prove in Part 5 the effectiveness of our simulator through several validation examples, both simple and complex applications, before concluding.

2. Contact Modeling and Nonsmooth Mechanics

The general dynamics equation for a multibody system can be written as:

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{f} + \mathbf{M}^{-1}(\boldsymbol{\Gamma} - \mathbf{b} - \mathbf{g}), \tag{1}$$

where $\mathbf{q}$ are the generalized coordinates of the multibody joints, $\mathbf{M}$ is the mass matrix of the system, $\boldsymbol{\Gamma}$ are the torques applied to the joints, $\mathbf{b}$ represents the Coriolis effects, $\mathbf{g}$ is the gravity and $\mathbf{f}$ are the external forces applied to the system that can be known, such as perturbations given by a user, or unknown, such as contact forces. This equation can be written in the operational space as[9,28]:

$$\mathbf{a} = \boldsymbol{\Lambda}\mathbf{f} + \mathbf{a}_{\text{free}}, \tag{2}$$

where $\boldsymbol{\Lambda}$ is the so-called Delassus operator which represents the projection of the inertia matrix in the contact space, introduced by Khatib[28] and equal to $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}$ where $\mathbf{J}$ is the Jacobian matrix, $\mathbf{a}_{\text{free}}$ is the free acceleration of the system in the contact space, that is the acceleration the system would have if there were no contact forces. $\mathbf{a}$ represents the acceleration of the system in the operational space and its dimension is $3m$, where m is the number of contact points (in the case we consider the three directions in the space).

2.1. *Contact modeling*

To model contact forces, two main approaches are generally considered: penalty-based methods and constraint-based methods. Penalty-based methods are widely used in simulators,[11,29–32] as they are easy and fast to implement. Forces are modeled as virtual spring-dampers acting when bodies are penetrating each other: they are a function of the penetration distance and its successive derivatives. As they are computed locally, they can be integrated easily into the dynamics equation (1) as known external forces. The main drawbacks of these methods are: (i) parameters of the spring-damper must be tuned for each simulation scenario, they are generally determined by experiments as there is no formula available in the literature and they can lead to high numerical instabilities, especially during numerical integration, implying to choose very small time steps which is not suitable for interactive simulations; (ii) their physical meaning is hard to justify as they are based on penetrations.

In constraint-based methods, nonpenetration constraints are explicitly integrated into the dynamics equation.[1] Generally formulated as a Linear Complementary Problem (LCP),[33] this is an elegant formulation of the contact problem as we can express the dynamics in a linear form:

$$\mathbf{a} = \boldsymbol{\Lambda}\mathbf{f} + \mathbf{a}_{\text{free}} \quad \text{and} \quad 0 \leq \mathbf{a} \perp \mathbf{f} \geq 0. \tag{3}$$

Convergence to a solution is proven to be ensured by several direct resolution methods such as Lemke's algorithm.[34,35] All contact constraints are grouped into the Delassus operator $\mathbf{\Lambda}$, with the dependencies between each constraint, allowing a global resolution of the problem. Compared to penalty-based methods, these methods are more accurate and stable, and are more and more used, especially in the video game community.[36] However, they are time-consuming, especially due to the computation of the $\mathbf{\Lambda}$ matrix, and uniqueness of the solution is not proven in the general case, for instance when the system is statically indeterminate.

Adding nonsmooth phenomena, such as dry friction, complexifies the resolution of the contact problem. Here we will consider Coulomb's friction, which is the most known and the most common friction model, but other friction models can be considered and handled in the same way.[37] Coulomb's friction is expressed as follows:

$$\|\mathbf{f}_t\| \leq \mu |f_n|, \tag{4}$$

where $\mathbf{f}_t$ is the tangential force, f_n the normal force, μ the friction coefficient. We can then consider two cases: (i) contact points are sticking ($\|\mathbf{f}_t\| < \mu|f_n|$), (ii) contact points are slipping ($\|\mathbf{f}_t\| = -\mu|f_n|\frac{\mathbf{v}_t}{\|\mathbf{v}_t\|}$). The law introduces a nonlinearity in the dynamics. Moreover, we see in the slipping case that the direction of the tangential force is determined by the tangential component of the velocity $\mathbf{v}_t$. Equation (1) is expressed at an acceleration level and shows the difficulty of the problem: acceleration is given once the forces are known and in this case, to know the direction of the frictional force, we need to integrate acceleration at least once. It is possible to express the friction problem at an acceleration level as done by Baraff,[1] but there might be cases where there is no solution for the frictional forces. Baraff applies then impulses to allow discontinuities of velocity.

Using penalty-based methods to solve friction leads to the same problems mentioned earlier: the choice of the parameters and the quantification of the tangential penetration, with numerical instabilities. Using LCP formulation of Eq. (3) requires the friction law to be discretized because of its nonlinearity.[38–43] The geometrical interpretation is to discretize the friction cone into facets. This implies additional constraints for each contact point (as many as facets) which increase drastically the size of the system but also produce a loss of accuracy, especially if the cones are not enough discretized. Other problems such as robustness or computation time needed for the computation of the Delassus operator make this method unsuitable for interactive simulations. Moreover, the Delassus operator is not symmetric anymore and can lead to conditioning illness that can be a problem for LCP solvers such as Lemke's solver.[44] Nonclassical methods of LCP resolution, such as optimization techniques, can be used, but require more complex implementation.[45,46] To keep a nondiscretized friction law, iterative methods have been introduced.[47,48] They guarantee convergence to a solution with the ability to get a compromise between accuracy and computation time, which is of interest for interactive simulations.

A comparison between different resolution methods of contact using constraint-based methods can be found in Renouf *et al.*[44] The results can be verified using the SICONOS framework, which integrates a high number of numerical solvers.[49]

More recently, new contact resolution methods based on volumic contact force models have been proposed.[7] These methods compute the penetration volume between bodies, then apply volume constraints on the bodies to get a single constraint in the dynamics equation and compute a volumic contact force. The dynamics equations get much simpler than those using the methods presented above as there is just one contact force to compute, leading to the computation of a 3×3 Delassus operator (to be compared with a $m \times m$ matrix, where m is the number of contact points). Contact volume is computed using GPUs. Dedicated to the computer graphics community, these methods are well suited to real-time visualization for entertainment, without any experimental validation. However, for the robotics community where physics is of primary concern, these methods may lead to nonphysical results.

2.2. *Whole dynamics integration*

The accuracy of the dynamics model of a nonsmooth mechanical system is not only linked to contact modeling but also lies on the numerical integration of the whole dynamics. This can also be a source of errors as simulation is performed in discrete time, meaning that nonsmooth phenomena, such as impacts, occurring between two simulation steps, cannot be precisely taken into account and solved. Two main integration schemes are widely used: the so-called event-driven and time-stepping schemes.

The event-driven scheme[50] proposes to decompose the dynamics into smooth modes, e.g., free dynamics, and nonsmooth ones, e.g., when impacts occur. Each contact instant is detected, dynamics is updated at these instants considering contact forces. Thus this scheme ensures high accuracy but is not adapted to complex environments involving a high number of contact points and so to real-time simulation we aim at in this work.

The time-stepping scheme, initiated for mechanical systems by Moreau,[47] proposes to integrate the whole dynamics over a time step, chosen as small as possible, then to discretize the result. The main feature of this scheme is not to determine nonsmooth events, as proposed by the event-driven scheme, and thus is to allow small penetrations. This method may therefore not be accurate unless small time steps or high integration orders are chosen, which is not suited to real-time needs. It is however robust and easy to implement. Since the whole dynamics is integrated, the formulation of nonsmooth constraints is modified: they can be written either in position or velocity, which may imply errors for contact simulation. Indeed, ill-conditioning of the matrices, nonlinearity of the constraints, drifts during continuous contacts can appear. However, considering velocity-based constraints can be of interest to unify into a single formulation all the constraints, both geometrical and

kinematical, and to take into account both elastic and inelastic impacts (generally written in velocity). Many work using this scheme were presented,[38,39,41,43] but considering only very simple study cases, such as a ball falling on a floor. A complete overview of the time-stepping scheme can be found in Studer's work.[51]

3. Constrained Based Dynamic Computation

In the previous part, we introduced the different models needed to build our dynamic simulator for multibody systems. We integrated them in a framework devoted to general virtual prototyping and called AMELIF.[52] To compute the free dynamics, we used Featherstone's algorithm,[17] and for constrained dynamics we extended and improved Ruspini and Khatib's framework that uses constraint-based methods,[9] to include friction.

3.1. *Architecture of our simulator*

We present the basic architecture of AMELIF, more details can be found in Evrard *et al.*[52] The goal of AMELIF is to perpetuate the developments of basic tools for the simulation and the control of multibody systems and more specifically humanoid robots. This implies the framework to be: (i) modular, so that a user can implement an algorithm efficiently and easily, which guarantees robustness and portability, (ii) able to simulate any kind of multibody systems, (iii) able to load any virtual scene or simulation context quickly, (iv) available to the community.

The global architecture is represented in Fig. 1. It consists of:

- a kernel, managing the data of the objects loaded in the virtual scene (geometrical and physical data), and running the simulation loop;
- different modules using the kernel, that can communicate between each other, and built as independent libraries. We specifically developed the following ones:
  - — dynamics (free and constrained dynamics, detailed in the following subsections);
  - — collision detection, to manage all collisions in the virtual scene. We use the PQP library[a];
  - — control, to compute command laws (joint torques);
  - — interaction, to interface the simulation environment with external events (human-machine interfaces), presented in Part 4.
- a main program, to initialize the virtual scene and the simulation loop.

3.2. *Computation of the Delassus operator*

Several methods have been presented in the literature to compute the operational space matrix. The simplest way to compute this matrix is to make a direct

[a] http://www.cs.unc.edu/~geom/SSV/.

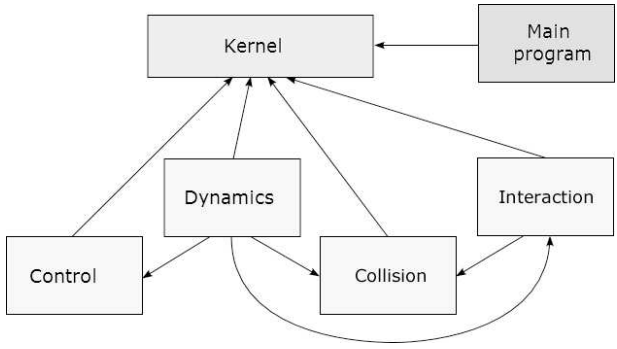


Fig. 1. General architecture of AMELIF.

computation, meaning compute $\mathbf{M}$ then its inverse, $\mathbf{J}$ and its transpose, then multiply them each other to obtain $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}$. This is obviously the most time consuming method and thus cannot be used to achieve interactive simulations. In Chardonnet *et al.*,[53] we proposed an easy implementation for computing $\mathbf{\Lambda}$ using Featherstone's algorithm, by considering no joint torques, no joint velocities, no gravity, and applying a unit force at each contact point. The main drawback of this method is that each body of the system, even those that are not contacting, must be visited during the computation, thus giving an overall complexity in $\mathcal{O}(nm + m^2)$, where n is the number of bodies and m the number of contact points. Chang and Khatib proposed to introduce an intermediate matrix $\mathbf{\Omega}$ that links the acceleration of a body i to the forces applied on a body j, then to project this matrix $\mathbf{\Omega}$ onto the contact space.[54] This method is much faster than the one presented in Chardonnet *et al.* as the algorithm visits only the branches containing contacting bodies. However, implementation is more complex as intermediate variables must be specifically computed.

Here we propose a method that combines these two methods: We introduce an intermediate matrix $\mathbf{\Omega}$ which, here, will represent the inertia matrix projected in the *contacting bodies* space, and will be computed using a modified Featherstone's algorithm. Finally, we project this matrix in the contact space. The idea here is to adapt Chardonnet *et al.*'s method in order to reduce the dependency toward the number of contact points. The implementation of our method is also easier than Chang and Khatib's one.

We want to avoid the computation of intermediate variables. To compute $\mathbf{\Omega}$, we use as for Chardonnet *et al.*'s method a modified Featherstone's algorithm, this time for each contacting body, without any joint torques ($\mathbf{\Gamma} = 0$), any joint velocities ($\dot{\mathbf{q}} = 0$) and without any gravity ($\mathbf{g} = 0$). This algorithm is composed of two loops: the first starting from the contacting bodies to the main body, and the second from the main body to the contacting bodies (an example is represented in Fig. 2). Hence, the free acceleration $\mathbf{a}_{\text{free}}$ becomes null and:

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{J}^{T}\mathbf{f}. \tag{5}$$

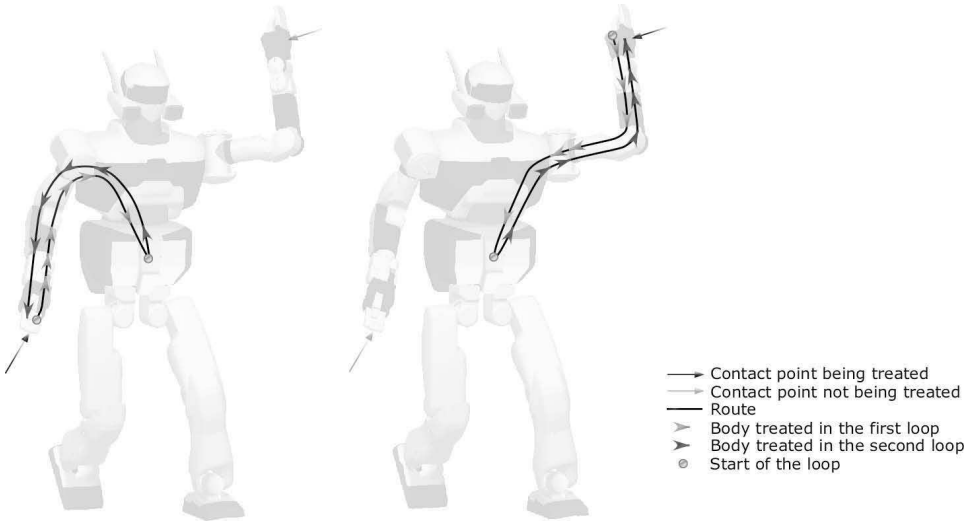


Fig. 2. Loops for the computation of $\mathbf{\Lambda}$. When there are two contacting bodies, on the left, loops for the first contacting body, on the right, loops for the second contacting body. The loops go from the contacting body to the main body (in this example the robot's waist) then go back to the contacting body.

Let us consider we apply six times (corresponding to the six components of force and momentum) a unit force $\mathbf{f}_i = (\mathbf{F}_i, \boldsymbol{\tau}_i)^T$ at the origin of a colliding body i's frame, defined by:

$$\mathbf{F}_i = \begin{matrix} (1,0,0)^T \\ (0,1,0)^T \\ (0,0,1)^T \\ (0,0,0)^T \\ (0,0,0)^T \\ (0,0,0)^T \end{matrix} \quad \boldsymbol{\tau}_i = \begin{matrix} (0,0,0)^T \\ (0,0,0)^T \\ (0,0,0)^T \\ (1,0,0)^T \\ (0,1,0)^T \\ (0,0,1)^T \end{matrix} \quad \begin{matrix} \text{first time} \\ \text{second time} \\ \text{third time} \\ \text{fourth time} \\ \text{fifth time} \\ \text{sixth time} \end{matrix} \tag{6}$$

or, in a condensed form ${}_u\mathbf{f}_i[y] = 1$ if $u = y, 0$ otherwise and $y = \{1,2,3,4,5,6\}$. Then we get:

$$\ddot{q}_{C_i} = \mathbf{M}^{-1}\mathbf{J}_{C_i}^T, \tag{7}$$

where $\mathbf{J}_{C_i}$ is the Jacobian of colliding body i. Transforming into Cartesian space leads to:

$$\mathbf{a}_i = \mathbf{J}_{C_i}\mathbf{M}^{-1}\mathbf{J}_{C_i}^T = \Omega_{i,i}. \tag{8}$$

Recall that the free acceleration is null. Considering the definition of $\Omega_{i,j}$ ($\mathbf{a}_j = \Omega_{i,j}\mathbf{f}_i$) and $\mathbf{f}_i = 1$, we have:

$$\mathbf{a}_j = ({}_1\mathbf{a}_j, \ldots, {}_6\mathbf{a}_j) = \Omega_{i,j}, \tag{9}$$

where ${}_u\mathbf{a}_j$ is the acceleration of body j associated to the unit force ${}_u\mathbf{f}_i$ applied on body i. We project then this equation onto the contact space to obtain $\mathbf{\Lambda}$, the projection is basically a transformation from the colliding body frame to the contact points frames. In the algorithm, only the upper part of $\mathbf{\Lambda}$ is computed, as it is symmetric.

The global complexity of our method is the same as the one of Chang and Khatib. In the worst case, the computation of $\mathbf{\Lambda}$ is in $\mathcal{O}(n^2)$, otherwise it is in $\mathcal{O}(C^2)$ with C the number of contacting bodies. As for Chang and Khatib's method, our algorithm visits only the branches that contain colliding bodies. As for the method of Chardonnet *et al.*, we use a modified Featherstone's algorithm, but we apply it $6C$ times, whereas in Chardonnet *et al.*'s method, the algorithm is applied $3m$ times.

We made a comparison in terms of performance between the three methods (Chardonnet *et al.*'s method, Chang and Khatib's method and the proposed method). We used a desktop PC with a bi-AMD 64 2.5 GHz CPU running under Windows XP. The results are depicted in Fig. 3.

Our method is clearly much faster than Chardonnet *et al.*'s method (about 6.5 times faster). The difference with Chang and Khatib's method is less visible. In most cases, our method will be faster than Chang and Khatib's method but there are some cases, especially when two contacting bodies belong to the same branch, for which Chang and Khatib's method will be faster. Indeed, our method requires to visit the main body by construction of Featherstone's algorithm, whereas Chang and

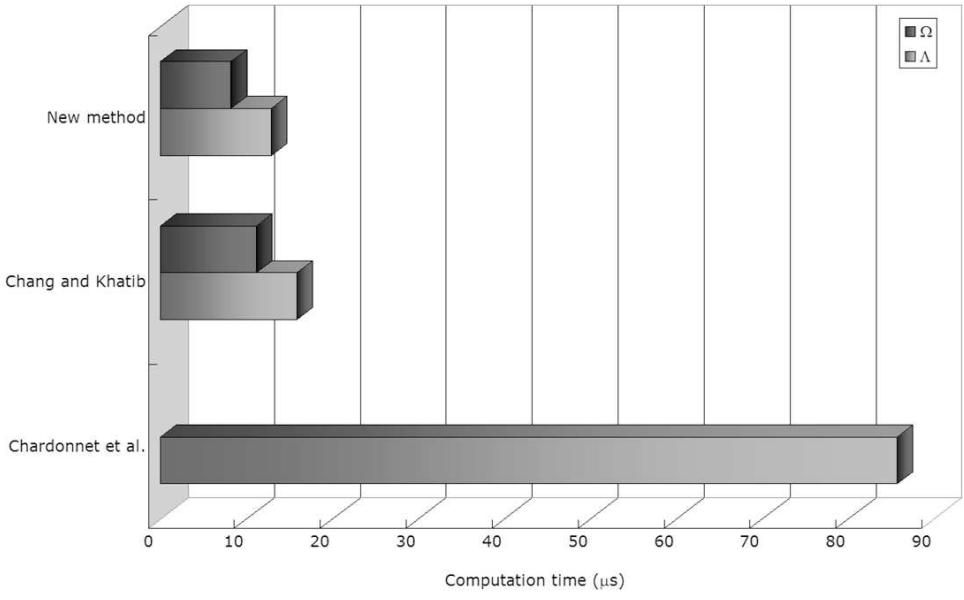


Fig. 3. Comparison between the three methods in terms of computation time over 20 simulations. Light gray bars represent the total time for the computation of $\mathbf{\Lambda}$ (computation of $\mathbf{\Omega}$ and projection into the contact space for Chang and Khatib's method and the proposed method; in Chardonnet *et al.*'s method, there is no intermediate variables, thus there is no dark gray bar).

Khatib's algorithm visits only the branches linking contacting bodies. The advantage however of our method on Chang and Khatib's one is an easier implementation.

3.3. *Collision groups*

The size of $\mathbf{\Lambda}$ depends on the number of contact points. When including friction without any discretization, the size is $3m \times 3m$. This matrix can be block diagonal if the objects are not colliding each other, each block representing the inertia matrix for each object. To obtain a full rank matrix, we define *collision groups*. A collision group is a sequence of unfixed objects in the environment that are contacting each other (see Fig. 4). Each group has then its own reduced $\mathbf{\Lambda}$ matrix, which allows faster computations. This preliminary sorting of contacting bodies at each time step allows a more efficient management of contact forces.

3.4. *Contact forces computation*

Once we compute the Delassus operator, recall the system to be solved:

$$\mathbf{a} = \mathbf{\Lambda f} + \mathbf{a}_{\text{free}} \tag{10}$$

To avoid problems mentioned in Part 2, we worked at a velocity level: we integrated the previous equation once using for example an explicit Euler integration scheme:

$$\mathbf{v}^{t+\delta t} = (\delta t \mathbf{\Lambda})\mathbf{f} + (\delta t \mathbf{a}_{\text{free}} + \mathbf{v}^t) \tag{11}$$

that will be written:

$$\mathbf{v} = \mathbf{W f} + \mathbf{v}_{\text{free}}. \tag{12}$$

We used an iterative method to solve the problem, more specifically a Gauss-Seidel like algorithm originally applied to robotics by Liu and Wang,[48] in nonsmooth mechanics by Moreau,[47] and used for deformable objects by Duriez *et al.*[55] The forces can be found for each contact point by:

$$\mathbf{f}_i^{k+1} = \mathbf{\Lambda}_{ii}^{-1} \quad \mathbf{v}_i^{\text{free}} - \sum_{j=1}^{i-1} \mathbf{\Lambda}_{ij} \mathbf{f}_j^{k+1} - \sum_{j=i+1}^{m} \mathbf{\Lambda}_{ij} \mathbf{f}_j^{k} \Bigg) \tag{13}$$

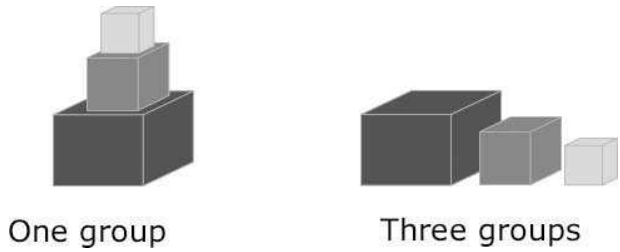


Fig. 4. Examples of collision groups. On the left, the three cubes make one collision group as they are contacting each other. On the right, the three cubes make three collision groups as they are independant from each other, each cube has its own reduced $\mathbf{\Lambda}$ matrix.

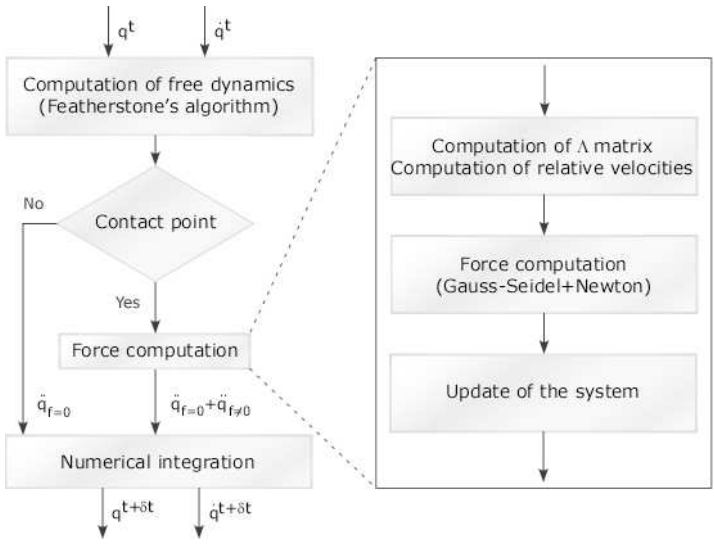


Fig. 5. Global algorithm of the dynamics simulation.

and looping iteratively until convergence, which is proven for dominant matrices.[56] We coupled this process with a Newton method to solve precisely contact forces at each contact point.[55,57,58] The interest of using a Newton method is to converge very quickly if the initial value is close to the solution, thus reducing computation time which is of high interest in our case. We get a small overall computation time and improved accuracy, as we work with matrices of size equal to the number of contact points. Convergence can be improved by choosing an initial value of the forces close to the solution (for instance taking the solution of the previous step). Moreover, compared to LCP solvers, we do not need to wait the end of the calculation to get a solution. Indeed, an estimate of the solution can be enough and thus we can interrupt the resolution process anytime, which is an nice feature for interactive simulations.

The global algorithm is presented in Fig. 5.

4. Haptic Feedback and Interactive Simulation

Interactivity is of major concern as we want to achieve for example collaborative tasks with virtual avatars. But interactivity is also important in the case we want to study the behavior of a virtual avatar against an external perturbation, for example if we want to test the robustness of a command law against unexpected events during the simulation of a task using this law. In OpenHRP, we need to add these perturbations manually for each simulation which is generally difficult and bothersome due to parameters tuning. Thus, we integrated in our simulator a haptic device with force feedback. We chose the Phantom© Omni™ sold by Sensable[b] which includes six

[b] http://www.sensable.com.

degrees of freedom of movement and three of feedback. In our simulator, it is possible to integrate other haptic devices and other interaction devices as its architecture is highly modular.

We will consider two ways of interaction: (i) the haptic probe can penetrate the objects, can be attached to them, thus it can be considered as the application point of bilateral contact forces, this point can be either inside or on the surface of the object (we will call it the attach case), and (ii) the haptic probe cannot penetrate the objects, it is considered as a unilateral interaction point, in this case, the haptic probe allows just tactile exploration (we will call it the touch sensing case). In both cases, the contact point can be determined using the collision detection library included with the haptic device.

For the touch sensing case, the force to be applied to the object is directly given by the haptic device. This force is then multiplied by a user-chosen coefficient.

For the attach case, generally used to manipulate objects for typical pick-and-place tasks, the force is computed using a 6D spring-damper model linking the haptic probe to the point on the object. This force can be written as:

$$\mathbf{f} = k_p(\mathbf{x}_{\text{object}} - \mathbf{x}_{\text{probe}}) + k_v(\dot{\mathbf{x}}_{\text{object}} - \dot{\mathbf{x}}_{\text{probe}}), \tag{14}$$

where $\mathbf{x}_{\text{object}}$ and $\mathbf{x}_{\text{probe}}$ are the positions of the object (the attach point) and of the haptic probe respectively, k_p and k_v are the spring-damper parameters with $k_v = \sqrt{2m_{\text{object}}k_p}$. This method is very fast but needs fine parameter tuning to obtain a realistic feedback. Other methods like constraint-based methods can also be used,[59] however, there are generally time consuming and complex to implement.

In both touch sensing and attach cases, the force is added to the free dynamics equation as a known external force $\mathbf{f}_e$:

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q})(\boldsymbol{\Gamma}(\mathbf{q}) - \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{g}(\mathbf{q})) + \mathbf{M}^{-1}(\mathbf{q})\mathbf{J}_e^{\,T}(\mathbf{q})\mathbf{f}_e. \tag{15}$$

We performed several interactive simulations using the HRP-2 robot. We propose to realize collaborative tasks with HRP-2, e.g., manipulating an object. To achieve such tasks, the robot must be compliant. One way to do it is to control all joints in position and define desired joint positions $\mathbf{q}_d$ by the following equation:

$$\mathbf{M}_d\,\ddot{\mathbf{q}}_d + \mathbf{B}_d\dot{\mathbf{q}}_d = \mathbf{J}_e^{\,T}\mathbf{f}_e, \tag{16}$$

where $\mathbf{M}_d$ and $\mathbf{B}_d$ are positive diagonal matrices corresponding to a virtual inertia and a virtual damping, respectively. The Jacobian $\mathbf{J}_e$ links the joint velocities to the Cartesian velocities of the bodies that are between the main body and the one on which an external force $\mathbf{f}_e$ is applied. We first approach the robot's arm to the object using the device and then close its gripper so that it grasps the object. Then we move the object in the space using the haptic device. We show simulation snapshots for two scenarios in Figs. 6 and 7.

To increase interactivity and the perception of collaboration with a virtual avatar, a visuo-haptic perception module was also added. This module allows a virtual

Fig. 6. HRP-2 sitting in a sofa and taking a can. This simulation shows a realistic scenario in a complex environment with robust multi-contact resolution.

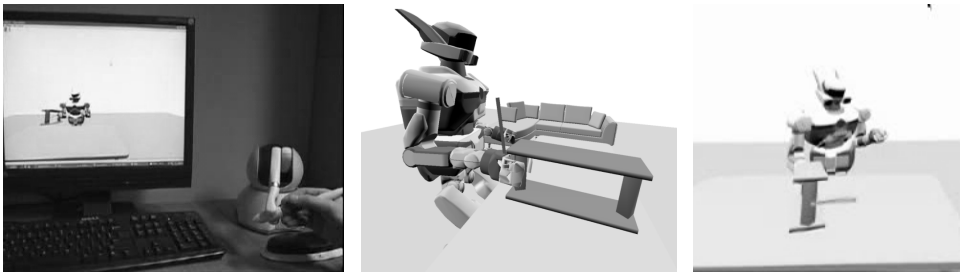


Fig. 7. Collaborative task with the HRP-2 robot. Using the haptic device, we are able to manipulate an object with the HRP-2 robot while sensing force feedback.

avatar to react against a perturbation driven by the user through a haptic interface, for example, touching the robot will notify it that the user wants to perform a task with it. The robot will hence move according to the users' intentions, by looking for instance at the haptic probe. Considering again the scenario of Fig. 7, the simulation results are depicted in Fig. 8.[52]

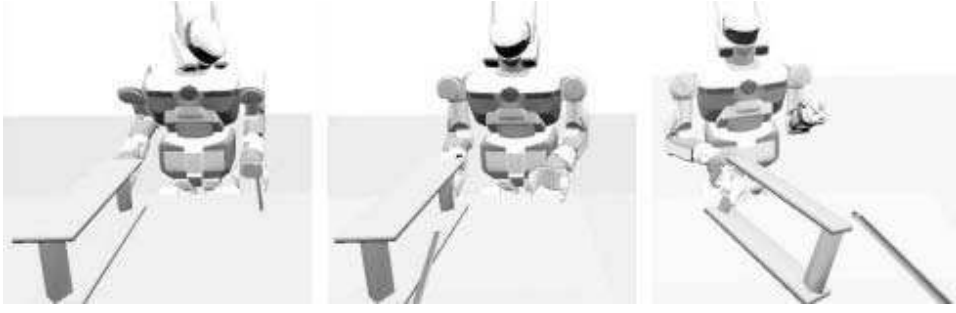


Fig. 8. Collaborative task with the HRP-2 robot with the visuo-haptic perception module. HRP-2 always looks at the haptic probe, which increases the level of interactivity with the virtual avatar.

5. Assessment Through Real Experiments on HRP-2

As far as we know, there is no general methodology to assess a robotic simulator. In many research works, simulators try to provide results that are as much as possible close to real sensors' measures, physical interactions and actuators' models. We could say that the quality of a simulator lies on precise comparisons between the results of a simulated robot and those of a real robot, based on sensors' outputs. However, identifying real parameters such as friction coefficients, impact's coefficients of restitution, joints mechanism parameters,..., is nearly impossible to make for each scenario, even for simple scenarios. Research works on simulation trying to show as much as possible close-to-real results generally deal with very basic examples, such as a ball on a floor. Considering more complex scenarios, for instance including object manipulation, as far as we know, most of the current literature do not present any other comparison between simulation and reality else than a visual one, i.e., comparing simulation and experiment snapshots. Thus, these frameworks do not prove their actual performance. What is important in physical-based simulation, for example a humanoid robot walking on a floor, is not to know the real friction coefficient between the foot and the floor, but to model and to simulate this friction properly, so that the behavior of the robot would be roughly the same if there was such friction in reality, in other words, the time evolution of the measures in simulation and in reality should have the same global characteristics. Moreover, sensors are generally noisy and the most important thing is not to simulate this noise exactly, then compare simulated and real measures, but to get a noise in simulation that has the same global properties. In fact, command laws developed in robotics are robust to these variations, and the goal is to ensure the robustness of these laws by confronting them with different scenarios, so that experiments have more and more chance to be successful with different parameters.

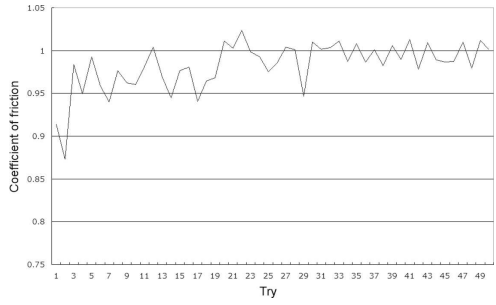
Here, we will present two ways of assessment of our simulator: The first one is based on a precise comparison between simulation and reality on a simple scenario, i.e., identifying real parameters and comparing the outputs, and the second one by showing an example of successful applications that used our simulator.

5.1. *Precise comparison between simulation and reality*

Here we will make our comparison on contact with friction by observing force sensors' outputs: The ones computed in simulation and the ones directly read on the sensors mounted on the robot. We propose here a very simple scenario in a static case implying the HRP-2 robot, made by Kawada Industries, and a fixed object in the environment. We ask the robot to bend its arm and to push with its hand on a pillar just in front of it. The most interesting part is of course when the robot contacts the pillar, especially the contact forces applied on the hand.

To obtain close-to-real results, several parameters must be taken into account: The friction coefficient between all objects, the model of the force sensors, the

(a) Experimental setup for measuring the friction coefficient between the foot and the floor.



(b) Measured values of the friction coefficient between the foot and the floor.

Fig. 9. Measurement of the friction coefficient.

parameters of HRP-2's actuators and finally the flexibilities existing in the knee joints of the robot.

The friction coefficients to be identified are between the feet and the floor, and between the hand and the pillar. To obtain the coefficient between the feet and the floor, we managed to get only one foot of HRP-2, on which we put a mass (enough heavy to get reliable measures), and with a dynamometer, we measured the force f that was necessary to make the foot slip on the floor (see Fig. 9). We can easily deduce the friction coefficient by the following equation:

$$f = \mu f_n = \mu mg \Rightarrow \mu = \frac{f}{mg}. \tag{17}$$

We realized 50 measures and we took the mean value of these measures to get $\mu \approx 0.983$. We proceeded in the same way to get the friction coefficient between the hand and the pillar ($\mu \approx 0.1$).

The next step is to model the force sensors. As the measures read from the real sensors are given in the sensor's frame, we need to know the position and the orientation of the sensor in the frame of the body that contains the sensor. Especially, for the hand, we found that the center of the sensor is nearly the same as the one of the hand and its orientation is 90 degrees. Then, to obtain the real applied forces, the read values should be multiplied by a full rank squared matrix $\mathbf{G}$ which is equal to the identity matrix when the sensor is perfect. $\mathbf{G}$ can be assimilated to an adjustment parameter. In our case, the force sensors that are in HRP-2's hands are old, meaning that $\mathbf{G}$ is not equal to the identity matrix; its parameters were identified from offline measures taken considering different orientations of the hand. Finally the dynamics equation for the sensor is:

$$m\,\ddot{\mathbf{x}} = \mathbf{P} + \mathbf{f}_e + \mathbf{f}_r, \tag{18}$$

where m, $\ddot{\mathbf{x}}$ and $\mathbf{P}$ are respectively the mass, the acceleration and the weight of the body in which the sensor is located, $\mathbf{f}_e$ and $\mathbf{f}_r$ are respectively the external forces

exerted on the sensor and the forces exerted by other bodies of the robot on the sensor (internal forces). This equation is written in the sensor's frame. Here we work in the static case as we want to measure the forces in a static case, thus $\ddot{\mathbf{x}} = 0$. The force given by the sensor $\mathbf{f}_c$ is actually equal to $-\mathbf{f}_r$ and can be written:

$$\mathbf{f}_c = \mathbf{G}\mathbf{f}_l, \tag{19}$$

where $\mathbf{f}_l$ is the force read directly on the sensor. Finally, we add an offset $\mathbf{f}_d$ corresponding to a calibration offset (when the robot's joints are set to zero, the forces read on the sensors are set to zero). Equation (18) becomes then:

$$\mathbf{f}_e = -\mathbf{P} + \mathbf{G}(\mathbf{f}_l + \mathbf{f}_d). \tag{20}$$

We will compare the forces $\mathbf{f}_e$ with the computed ones in simulation. Here we will suppose that the normal to the pillar is along the z-axis of the sensor, and the gravity direction is along the y-axis of the sensor.

We have next to set a command law to actuate joints. We choose a PD-type law:

$$\mathbf{\Gamma} = K_p(\mathbf{q}_d - \mathbf{q}) + K_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) \tag{21}$$

but to be as close as possible to reality, we must take into account different parameters such as dry and viscous frictions in the joints and the coefficients of reduction of the actuators. Considering these parameters, the command law can be written:

$$\mathbf{\Gamma} = R(K_p(\mathbf{q}_d - \mathbf{q}) + K_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}})) + \mu_v\dot{\mathbf{q}} + \mu_d \text{sign}(\dot{\mathbf{q}}), \tag{22}$$

where R are the coefficients of reduction, μ_v and μ_d are respectively the viscous and dry frictions. These parameters were identified by experiments.

The HRP-2 robot has flexibilities in the ankle joints to protect the robot's structure while moving. They create oscillations during walking or standing motions. These oscillations can be reduced using a stabilization control. In our case, we will consider the flexibilities, that are part of the robot's structure, but we will ignore the stabilization control as its expression and the way it acts on the joint references remain unknown. Its integration in our simulator then would be clearly a disturbance for our purpose. This implies the simulation data to account for the oscillations from the flexibilities in the ankle joints during our experiment. We used the model presented in Nakaoka *et al.*,[12] which is a simplified model consisting of virtual joints acting like spring-dampers in each direction. The values of the parameters are tuned according to Nakaoka *et al.*[12]

Once we identified the different parameters needed for our comparison, we obtained the results depicted in Figs. 10 and 11 for the simulation and the experiment, respectively. In Figs. 12 and 13, we show the values of the contact forces computed at the hand's sensor in simulation (in gray) and the ones measured at the real hand's force sensor (in black), along the z and y-axes of the sensor, respectively. Note that the motion of the robot's arm is made in the yz plane of the sensor, thus we always get a null force or a nearly null force in the x-axis of the sensor. We can

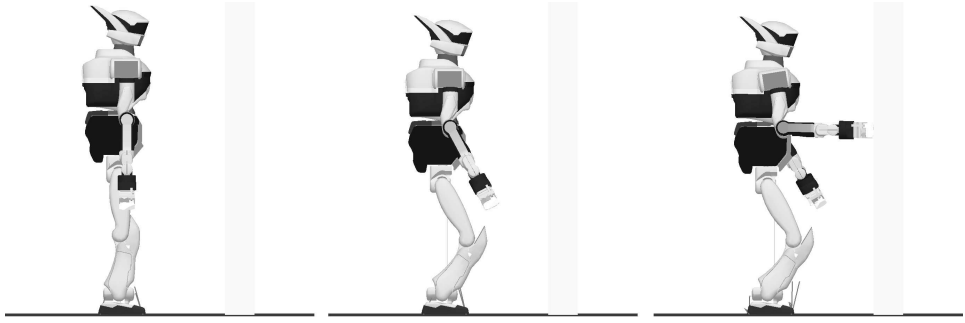


Fig. 10. Simulation screenshots. From left to right: the robot is standing up with all joints set to 0, then it bends the legs and the arms a little bit and finally touches the pillar.

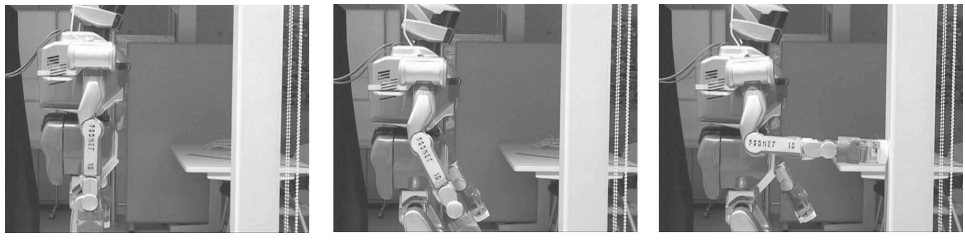


Fig. 11. Experiment screenshots. The motion is the same as in Fig. 10.

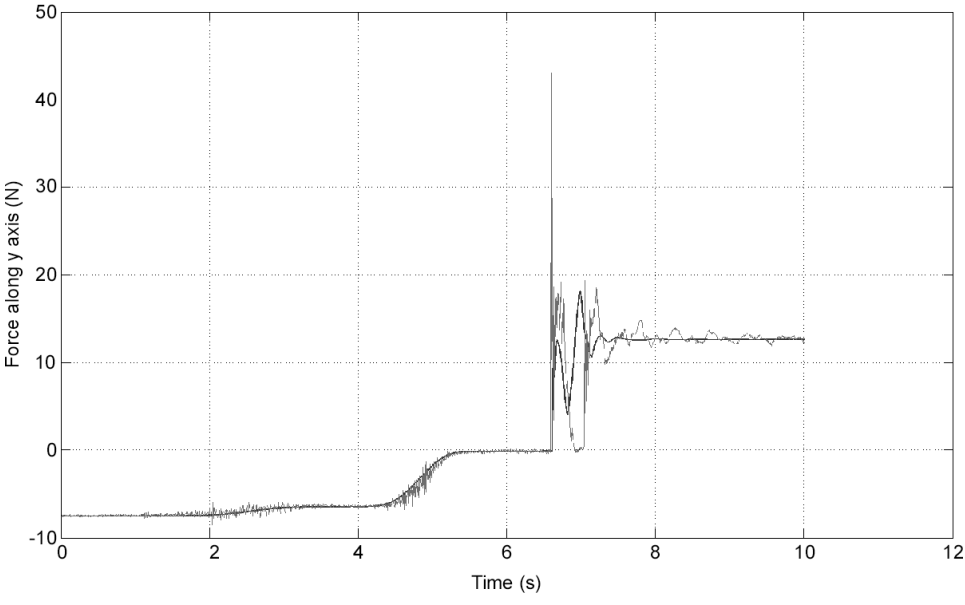


Fig. 12. Evolution of the force read on the force sensor along z axis with respect to time. In black: the computed force. In gray: the measured force.

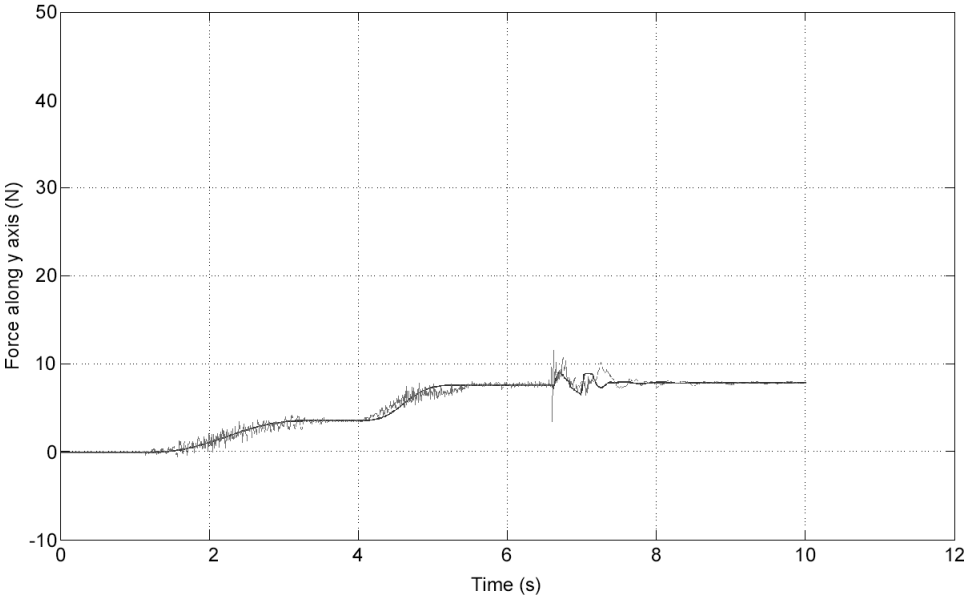


Fig. 13. Evolution of the force read on the force sensor along y axis with respect to time. In black: the computed force. In gray: the measured force.

observe that the values obtained from simulation match very well those of experimental measures. There are slight differences, due to setup errors (e.g., the initial position and orientation of the robot are not exactly the same in simulation and in the experiment). These differences are however negligible against the overall behavior, as we managed to always minimize the setup errors to avoid any kind of drifts between simulation and experimental results, otherwise it altered our simulation results (for instance, differences in position of a few millimeters, e.g., 2 mm, changed completely the overall behavior of the robot). We repeated the experiment several times to guarantee the quality of our simulator and we found similar results as in Figs. 10 and 11.

5.2. *Example of applications*

Our simulator was used in several applications, such as hard manipulation tasks that could be performed by future humanoid robots in collaborative environments. Especially, we asked the HRP-2 robot to lift heavy objects over its head. We used an optimized trajectory generation software based on dynamics calculation taking into account actuators' parameters and energetic considerations.[60] Design of the trajectory was inspired by weight lifting (see Fig. 14). This task is particularly complex and extreme. Indeed, considering the physical properties of the HRP-2 robot, no command law exists to perform such task specifying just initial and final conditions, thus justifying the use of optimized trajectory generation. Our simulator allowed us to

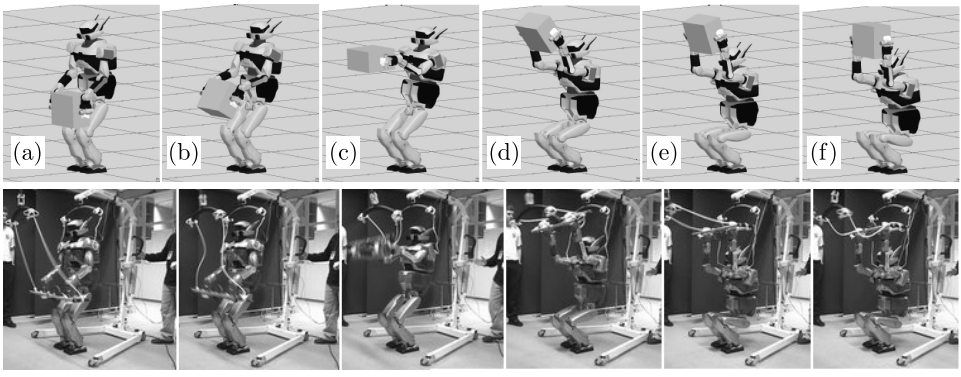


Fig. 14. HRP-2 robot lifting a 8.4 kg object using an optimized trajectory generation software (up: simulation. Down: experiment). From left to right: (a) initial state, (b) accelerating the object upward, (c) switching motion, (d) sliding into under the object, (e) crouching, (f) final state (sitting). The robot also succeeded in lifting a 23.4 kg object.

verify in both cases in simulation that the generated trajectory was indeed feasible on the real robot. Because this kind of applications are extreme and complex, they are typically examples where physical models implemented in simulation must be accurate and robust to avoid severe damages on the real robot. Success of the experiments shows that our simulator is hence valid. Further details on this example and other successful applications using our simulator can be found in the work of Arisumi *et al.*[61–63] Note that in the application depicted in Fig. 14, we also succeeded in making the robot lift a 23.4 kg object,[61] which is, considering the specifications of the HRP-2 robot and the proposed method, a high performance. Indeed, comparing to other existing methods for lifting an object, for example methods using whole-body contact where the robot places the object to be lifted on its arms,[64] we tried to make the robot manipulate an object just using its grippers. Our approach (i) is one of the most difficult way to lift an object, as the robot's motion passes through highly unstable configurations, and (ii) allows a complex manipulation of the object, which is nearly impossible for example with the method proposed by Ohmura and Kuniyoshi.[64]

Note that we tried to perform the same simulations in OpenHRP but we could not get relevant results because of high numerical instabilities caused by the tuning of the spring-dampers parameters used in the penalty-based contact model. Typically we observed divergence in most cases for the reasons we described in Sec. 2.1.

6. Conclusion

We presented an interactive dynamic simulator for multibody systems, including humanoid robots. Our simulator is a successful integration of theoretical models taking into account nonsmooth phenomena such as contact with friction, into a general framework designed for prototyping. Users can interact with the virtual

scene through interfaces, while sensing force feedback. We showed several ways of assessment of our simulator: First by measuring and comparing real measures with simulated ones through the identification of several parameters such as friction coefficients, actuators' models, then by presenting an example of applications of our simulator to complex and extreme manipulation tasks. Using our simulator in concrete applications clearly shows its interest and its effectiveness. Our simulator was also used as the development base for OpenHRP3.[12]

The basics of a high quality simulator are now well defined. We can still improve interactivity. Indeed, real-time interaction works well for scenarios not involving many contact points (less than 100 points). However, dynamics computation gets slower when the number of contact points dramatically increases. This is because we always worked with contact points, rather than contact lines or surfaces or volumes. Working with surfacic or volumic forces would drastically reduce time computation. However, rather than focusing on accelerating computation processes (which can be solved as long as new generations of computers get much faster), we are planning to consider and develop models of other physical phenomena, such as rotational friction, viscous friction, to get even more realistic simulations.

Acknowledgments

This work was partially supported by grants from the Immersence EU CEC project, contract No. 27141, http://www.immersence.info/ (FET-Presence) under FP6.

References

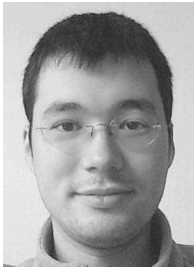
1. D. Baraff, Fast contact force computation for nonpenetrating rigid bodies, in *SIGGRAPH* (Orlando, FL, USA, 1994), pp. 23–34.
2. J. K. Hodgins and W. L. Wooten, Animating human athletes, in *International Symposium on Robotics Research* (1998), pp. 356–367.
3. R. Weinstein, J. Teran and R. Fedkiw, Dynamic simulation of articulated rigid-bodies with contact and collision, *IEEE Transactions on Visualization and Computer Graphics* **12**(3) (2006) 365–374.
4. A. Shapiro, D. Chu, B. Allen and P. Faloutsos, The dynamic controller toolkit, in *2nd Annual ACM SIGGRAPH Sandbox Symposium on Videogames* (San Diego, CA, USA, 2007), pp. 15–20.
5. J. Allard, S. Cottin, F. Faure, P.-J. Bensoussan, F. Poyer, C. Duriez, H. Delingette and L. Grisoni, SOFA — An open source framework for medical simulation, in *Medicine Meets Virtual Reality* (2007), pp. 13–18.
6. F. Faure, S. Barbier, J. Allard and F. Falipou, Image-based collision detection and response between arbitrary volumetric objects, in *ACM SIGGRAPH/Eurohaptics Symposium on Computer Animation (SCA)* (Dublin, Ireland, 2008), pp. 155–162.
7. J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez and P. Kry, Volume contact constraints at arbitrary resolution, *ACM Transaction on Graphics* **29**(4) (2007) 82: 1–82: 10.
8. M. Altomonte, D. Zerbato, D. Botturi and P. Fiorini, Simulation of deformable environment with haptic feedback on GPU, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Nice, France, 2008), pp. 3959–3964.

9. D. C. Ruspini and O. Khatib, Collision/contact models for dynamics simulation and haptic interaction, in *International Symposium on Robotics Research* (Snowbird, UT, USA, 1999), pp. 185−195.
10. D. Ruspini and O. Khatib, A framework for multi-contact multi-body dynamic simulation and haptic display, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Takamatsu Japan, 2000), pp. 1322−1327.
11. F. Kanehiro, H. Hirukawa and S. Kajita, OpenHRP: Open architecture humanoid robotics platform, *Int. J. Robotics Res.* **23**(2) (2004) 155−165.
12. S. Nakaoka, S. Hattori, F. Kanehiro, S. Kajita and H. Hirukawa, Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (San Diego, CA, USA, 2007), pp. 3641−3647.
13. A. T. Miller and H. I. Christensen, Implementation of multi-rigid-body dynamics within a robotic grasping simulator, in *IEEE International Conference on Robotics and Automation (ICRA)* (Taipei, Taiwan, 2003), pp. 2262−2268.
14. W. Son, K. Kim and N. M. Amato, A generalized framework for interactive dynamic simulation for multirigid bodies, *IEEE Transactions on Systems, Man and Cybernetics* **34**(2) (2004) 912−924.
15. J. G. Hale, B. Hohl, S.-H. Hyon, T. Matsubara, E. M. Moraud and G. Cheng, Highly precise dynamic simulator environment for humanoid robots, *Advanced Robotics* **22**(10) (2008) 1075−1105.
16. K. Nagasaka, A. Miyamoto, M. Nagano, H. Shirado, T. Fukushima and M. Fujita, Motion control of a virtual humanoid that can perform real physical interactions with a human, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Nice, France, 2008), pp. 2303−2310.
17. R. Featherstone, *Robot Dynamics Algorithms* (Kluwer Academic Publishers, 1987).
18. R. Featherstone, A divide-and-conquer articulated-body algorithm for parallel $\mathcal{O}(\log(n))$ calculation of rigid-body dynamics, *International Journal on Robotics Research* **18**(9) (1999) 867−892.
19. R. Featherstone, *Rigid Body Dynamics Algorithms* (Springer, New-York, NY, USA, 2007).
20. M. W. Walker and D. E. Orin, Efficient dynamic computer simulation of robotic mechanism, *ASME Transaction, Journal of Dynamic Systems, Measurement Control* **106**(1) (1982) 25−57.
21. B. Mirtich, *Impulse-Based Dynamic Simulation of Rigid Body Systems* (Ph.D. thesis, Univerity of California at Berkeley, 1996).
22. A. Fijany, I. Sharf and G. D'Eleuterio, Parallel $\mathcal{O}(\log(n))$ algorithms for computation of manipulator forward dynamics, *IEEE Transactions on Robotics and Automation* **11**(3) (1995) 389−400.
23. K. Anderson and S. Duan, Highly parallelizable low-order dynamics simulation algorithm for multi-rigid-body systems, *AIAA Journal on Guidance, Control and Dynamics* **23**(2) (2000) 355−364.
24. K. Yamane and Y. Nakamura, Efficient parallel dynamics computation of human figures, in *IEEE International Conference on Robotics and Automation (ICRA)* (Washington, DC, USA, 2002), pp. 530−537.
25. K. Yamane and Y. Nakamura, Automatic scheduling for parallel forward dynamics computation of open kinematic chains, in *Robotics: Science and Systems* (Atlanta, GA, USA, 2007), pp. 25−31.
26. B. Brogliato, A. Ten Dam, L. Paoli, F. Génot and M. Abadie, Numerical simulation of finite dimensional multibody nonsmooth mechanical systems, *Applied Mechanics* **55**(2) (2002) 107−150.

27. V. Acary and B. Brogliato, *Numerical Methods for Nonsmooth Dynamical Systems* (Springer, 2006).
28. O. Khatib, A unified approach for motion and force control of robot manipulators: The operational space formulation, *IEEE Journal of Robotics and Automation* **3**(1) (1987) 43−53.
29. K. Yamane and Y. Nakamura, Stable penalty-based model of frictional contacts, in *IEEE International Conference on Robotics and Automation (ICRA)* (Orlando, FL, USA, 2006), pp. 1904−1909.
30. Y. Hwang, E. Inohara, A. Konno and M. Uchiyama, An order n dynamic simulator for a humanoid robot with a virtual spring-damper contact model, in *IEEE International Conference on Robotics and Automation (ICRA)* (Taipei, Taiwan, 2003), pp. 31−36.
31. D. Turk and G. Wyeth, Semi-analytic method of contact modelling, in *IEEE International Conference on Robotics and Automation (ICRA)* (Orlando, FL, USA, 2006), pp. 1957−1962.
32. S. Hasegawa and M. Sato, Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects, *Computer Graphics Forum* **23**(3) (2004) 529−538.
33. K. G. Murty and F.-T. Yu, *Linear Complementarity, Linear and Nonlinear Programming* (Internet edition, http://ioe.engin.umich.edy/people/fac/books/murty/linear_complementarity_webbook/edition, 1999).
34. J. E. Lloyd, Fast implementation of Lemke's algorithm for rigid body contact simulation, in *IEEE International Conference on Robotics and Automation (ICRA)* (Barcelona, Spain, 2005), pp. 4549−4554.
35. M. Anitescu and F. A. Potra, Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems, *Nonlinear Dynamics* **14**(3) (1997) 231−247.
36. E. Kokkevis, Pratical physics for articulated characters, in *Game Developers Conference* (San Jose, CA, USA, 2004).
37. H. Olsson, K. J. Aström, C. C. de Wit, M. Gäfvert and P. Lischinsky, Friction models and friction compensation, *Eur. J. Control* **4**(3) (1998) 176−195.
38. D. Stewart and J. C. Trinkle, An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction, *Int. J. Numer. Meth. Eng.* **39**(15) (1996) 2673−2691.
39. D. Stewart and J. C. Trinkle, An implicit time-stepping scheme for rigid body dynamics with Coulomb friction, in *IEEE International Conference on Robotics and Automation (ICRA)* (San Francisco, CA, USA, 2000), pp. 162−169.
40. J. C. Trinkle, J. S. Pang, S. Sudarsky and G. Lo, On dynamic multi-rigid-body contact problems with Coulomb friction, *Zeitschrift für Angewandte Mathematik und Mechanik* **77**(4) (1997) 267−279.
41. M. Anitescu and F. A. Potra, A time-stepping method for stiff multibody dynamics with contact and friction, *International Journal for Numerical Methods in Engineering* **55**(7) (2002) 753−784.
42. J. Sauer and E. Schömer, A constraint-based approach to rigid body dynamics for virtual reality applications, in *ACM Symposium on Virtual Reality Software and Technology* (Taipei, Taiwan, 1998), pp. 153−162.
43. F. A. Potra, M. Anitescu, B. Gavrea and J. Trinkle, A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints, and friction, *Int. J. Numer. Meth. Eng.* **66**(7) (2006) 1079−1124.
44. M. Renouf, V. Acary and G. Dumont, 3D frictional contact and impact multibody dynamics. A comparison of algorithms suitable for real-time applications, in *ECCOMAS Thematic Conference* (Madrid, Spain, 2005).

45. D. M. Kaufman, T. Edmunds and D. K. Pai, Fast frictional dynamics for rigid bodies, in *SIGGRAPH* (Los Angeles, CA, USA, 2005), pp. 946−956.
46. D. M. Kaufman, S. Sueda, D. L. James and D. K. Pai, Staggered projection for frictional contact in multibody systems, *ACM Transactions on Graphics* **27**(5) (2008) 164:1−164:11.
47. J.-J. Moreau, Unilateral contact and dry friction in finite freedom dynamics, *Nonsmooth Mech. Appl.* **302** (1988) 1−82.
48. T. Liu and M. Y. Wang, Computation of three-dimensional rigidbody dynamics with multiple unilateral contacts using time-stepping and Gauss-Seidel methods, *IEEE Transactions on Automation Science and Engineering* **2**(2) (2005) 19−31.
49. V. Acary and F. Pérignon, *An introduction to Siconos* (Technical report TR-0340, INRIA, http://hal.inria.fr/inria-00162911/en/, 2007).
50. F. Pfeiffer and C. Glocker, *Multibody Dynamics with Unilateral Contacts* (John Wiley and Sons, New-York, NY, USA, 1996).
51. C. W. Studer, *Augmented Tme-Stepping Integration of Non-Smooth Dynamical Systems*, (Ph.D. thesis, ETH Zürich, 2008).
52. P. Evrard, F. Keith, J.-R. Chardonnet and A. Kheddar, Framework for haptic interaction with virtual avatars, in *17th IEEE International Symposium on Robot and Human Interactive Communication* (München, Germany, 2008), pp. 15−20.
53. J.-R. Chardonnet, S. Miossec, A. Kheddar, H. Arisumi, H. Hirukawa, F. Pierrot and K. Yokoi, Dynamic simulator for humanoids using constraint-based methods with static friction, in *IEEE International Conference on Robotics and Biomimetics (ROBIO)* (Kunming, China, 2006), pp. 1366−1371.
54. K.-S. Chang and O. Khatib, Efficient algorithm for extended operational space inertia matrix, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Kyongju, Korea, 1999), pp. 350−355.
55. C. Duriez, F. Dubois, A. Kheddar and C. Andriot, Realistic haptic rendering of interacting deformable objects in virtual environments, *IEEE Transactions on Visualization and Computer Graphics* **12**(1) (2006) 36−47.
56. F. Jourdan, P. Alart and M. Jean, A Gauss-Seidel like algorithm to solve frictional contact problems, *Comput. Meth. Appl. Mech. Eng.* **155**(1) (1998) 31−47.
57. P. Alart and A. Curnier, A mixed formulation for frictional contact problems prone to Newton like solution methods, *Comput. Meth. Appl. Mech. Eng.* **92**(3) (1991) 353−375.
58. M. Jean, Numerical methods for three dimensional dynamical problems, in *Conference Contact Mechanics* (Southampton, UK, 1993), p. 71.
59. C. B. Zilles and J. K. Salisbury, A constraint-based god-object method for haptic display, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Pittsburgh, PA, USA, 1995), pp. 146−151.
60. S. Miossec, K. Yokoi and A. Kheddar, Development of a software for motion optimization of robots — Application to the kick motion of the HRP-2 robot, in *IEEE International Conference on Robotics and Biomimetics (ROBIO)* (Kunming, China, 2006), pp. 299−304.
61. H. Arisumi, S. Miossec, J.-R. Chardonnet and K. Yokoi, Dynamic lifting by whole body motion of humanoid robots, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Nice, France, 2008), pp. 668−675.
62. H. Arisumi, J.-R. Chardonnet, A. Kheddar and K. Yokoi, Dynamic lifting motion for humanoid robots, in *IEEE International Conference on Robotics and Automation (ICRA)* (Rome, Italy, 2007), pp. 2661−2667.
63. H. Arisumi, J.-R. Chardonnet and K. Yokoi, Whole-body motion of a humanoid robot for passing through a door — Opening a door by impulsive force -, in *IEEE/RSJ International*

Conference on Intelligent Robots and Systems (IROS) (St. Louis, MO, USA, 2009), pp. 428−434.
64. Y. Ohmura and Y. Kuniyoshi, Humanoid robot which can lift a 30 kg box by whole body contact and tactile feedback, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (San Diego, CA, USA, 2007), pp. 1136−1141.



Jean-Rémy Chardonnet received his M.S. degree in mechatronics from the National Engineering Institute in Mechanics and Microtechnologies in Besançon, France, in 2005 and his Ph.D. degree in robotics from the University of Montpellier II, France, in 2009. From 2009 to 2010, he was post-doc in the Evasion team at Jean Kuntzmann Laboratory and INRIA Grenoble, France, then was Teaching Assistant at the Industrial Engineering School of Grenoble INP, France, working on the development of a hands-on interaction peripheral device for virtual object manipulation called HandNavigator. He published several papers that were awarded Best Paper. Now he is Associate Professor at Arts et Métiers ParisTech, Cluny, and CNRS Le2i Institut Image, Chalon-sur-Saône, France.

His research interests include interactive physical simulation, humanoid robots, multimodal interaction, virtual reality. He was involved in the European project IMMERSENCE and in the French ANR project ROMMA. He is currently involved in several projects with industrial partners dealing with virtual reality and in the European project Visionair.