"Multi-physics Modelling of a Compliant Humanoid Robot"

Zobova, Alexandra ; Habra, Timothée ; Van der Noot, Nicolas ; Dallali, Houman ; Tsagarakis, Nikolaos ; Fisette, Paul ; Ronsse, Renaud

Abstract

In this paper, we discuss some very important features for getting exploitable simulation results for multibody systems, relying on the example of a humanoid robot. First, we provide a comparison of simulation speed and accuracy for kinematics modeling relying on relative vs. absolute coor- dinates. This choice is particularly critical for mechanisms with long serial chains (e.g. legs and arms). Compliance in the robot actuation chain is also critical to enhance the robot safety and en- ergy efficiency, but makes the simulator more sensitive to modeling errors. Therefore, our second contribution is to derive the full electro-mechanical model of the inner dynamics of the compliant actuators embedded in our robot. Finally, we report our reasoning for choosing an appropriate contact library. The recommended solution is to couple our simulator with an open-source contact library offering both accurate and fast full-body contact modeling.

Document type : *Communication à un colloque (Conference Paper)*

Multi-physics Modelling of a Compliant Humanoid Robot

Alexandra A. Zobova[*], **Timothee Habra**[#], **Nicolas Van der Noot**[#], **Houman Dallali**[†], **Nikolaos G. Tsagarakis**[†], **Paul Fisette**[#], **Renaud Ronsse**[#]

[*] Faculty of Mechanics and Mathematics
Lomonosov Moscow State University
Leninskie Gory, 1
119991, Moscow, Russia
azobova@mech.math.msu.su

[†] Department of Advanced Robotics
Istituto Italiano di Tecnologia
Via Morego, 30, 16163
Genova, Italy
[houman.dallali, nikos.tsagarakis]@iit.it

[#] Center for Research in Energy and Mechatronics
Université catholique de Louvain (UCL)
Place du Levant 2, 1348
Louvain-la-Neuve, Belgium
[timothee.habra, nicolas.vandernoot, paul.fisette, renaud.ronsse]@uclouvain.be

ABSTRACT

In this paper, we discuss some very important features for getting exploitable simulation results for multibody systems, relying on the example of a humanoid robot. First, we provide a comparison of simulation speed and accuracy for kinematics modeling relying on relative vs. absolute coordinates. This choice is particularly critical for mechanisms with long serial chains (e.g. legs and arms). Compliance in the robot actuation chain is also critical to enhance the robot safety and energy efficiency, but makes the simulator more sensitive to modeling errors. Therefore, our second contribution is to derive the full electro-mechanical model of the inner dynamics of the compliant actuators embedded in our robot. Finally, we report our reasoning for choosing an appropriate contact library. The recommended solution is to couple our simulator with an open-source contact library offering both accurate and fast full-body contact modeling.

Keywords: Multibody dynamics, compliant actuators, contact dynamics, humanoid robot, Robotran, Simbody

1 INTRODUCTION

In this article, we present a multibody model of the COMAN humanoid robot[1] [1]. The key features of the proposed model are:

1. an efficient multibody dynamics allowing short simulation computational time;
2. the full electromechanical model of compliant actuators [2] made up with ordinary differential equations of the actuators inner dynamics;
3. a reliable mesh-to-mesh contact processing in order to simulate the robot self-collision and contacts with the environment.

For deriving the multibody equations, the Robotran symbolic generator was selected due to its reliability and efficiency [3]. This work further builds upon [4] by proposing a new actuator modeling and more powerful contact processing. Accurate mesh-to-mesh contacts were obtained through a coupling between the C-code provided by the Robotran generator and the C++ functions provided by the open-source library Simbody [5]. This model proved to be useful to speed up the synthesis and tuning of movement controllers, and can easily be adapted to the simulation of other robots.

[1]This work is supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant 611832 (WALK-MAN).

The structure of the article is the following. First, we describe the multibody system (MBS) of the humanoid robot we have to simulate and control and describe our requirements for a simulator to accurately reproduce the behavior of this MBS. Then we compare three open-source multibody platforms used in robotics – that is the Robotran generator [3], Open Dynamics Engine ODE [6] and Simbody [5]. In the following section, we describe two approaches for contact simulation — i.e. rigid and compliant contact — and show our reasoning for choosing one of them. Based on this analysis, we derive the full MBS model of robot mechanics, which is augmented with its electromechanical model of the compliant actuators. Finally, we share the technical details for coupling the C-code of the Robotran generator with the contact module of the open-source physic engine Simbody.

2 STRUCTURE OF THE ROBOT AND SIMULATOR OVERVIEW

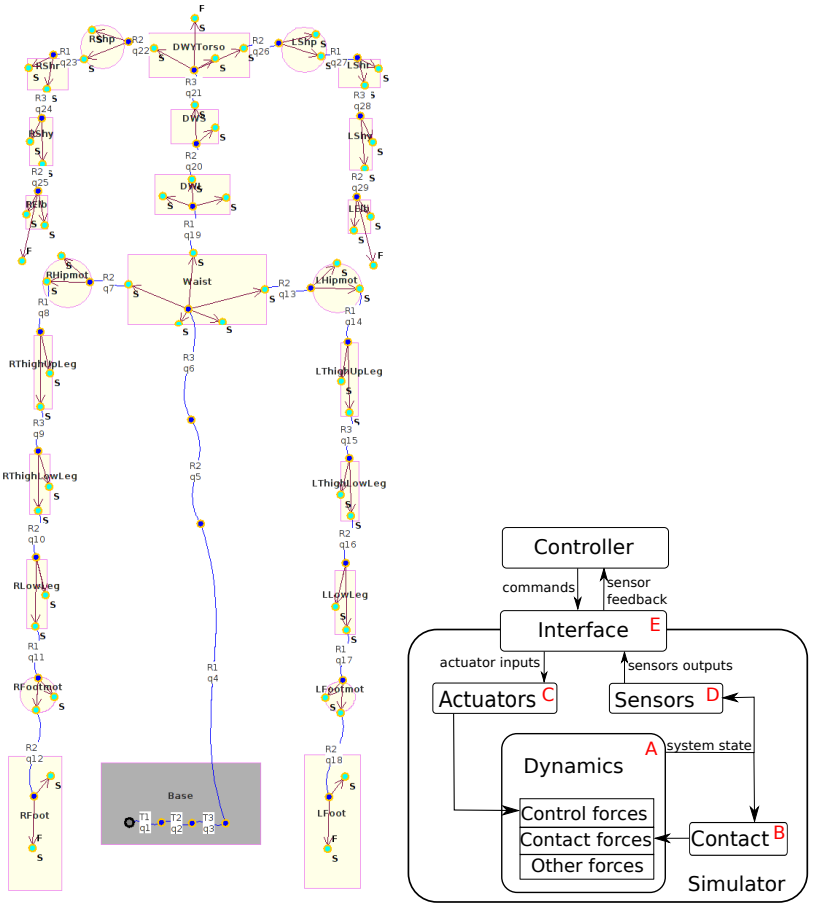

Figure 1. MBS COMAN model (left) and a simulator structure(right).

The multibody model of COMAN consists of 24 absolutely rigid bodies and has a tree-structure shown in Figure 1. The root body Waist is a floating base and has 6 degrees of freedom (DoFs). The other 23 bodies (4 segments for the torso, 4 for each arm and 6 for each leg) are serially attached to their parents by revolute joints. So, the model has 29 mechanical DoFs (among which 6 DoFs for the floating base).

Similarly to the real robot, the model is equipped with 23 series elastic actuators [7] in the revolute joints, each producing a motor torque τ_i, $i = 1 \dots 23$. This torque depends on the corresponding joint angle, joint velocity and the motor angle q_i^m and velocity $\dot{q}_i^m$, $i = 1 \dots 23$. Dynamics of this inner actuator variable q_m is governed by a differential equation involving the current I_i in the motor, which in turn satisfies a DC motor differential equation with a controllable voltage input

V_i. Furthermore, the robot legs are equipped with tunable springs going in parallel to the motors [2]. These springs thus deliver the torques $\boldsymbol{\tau}^p(\boldsymbol{p},\boldsymbol{q})$ at the pitch leg joints, where $\boldsymbol{p} = (p_1,\ldots,p_6)$ is the controllable spring pretensions. Physical model of both types of actuators and differential equations that govern the variables $\boldsymbol{q}^m = (q_1^m,\ldots,q_{23}^m)$ are discussed in details in Section 4.

To accurately simulate this MBS, we need a simulator which combines several modules being connected as represented in Figure 1, on the right. The simulator core is a direct dynamics module (block A) deriving the acceleration of the MBS links based on the MBS structure and forces (external and control) applied to MBS. The way to represent of the MBS structure influences the speed of simulation as well as its accuracy (see Section 3).

A critical component of an integrated robotic simulator is an algorithm for contact processing (block B in Figure.1). Contacts between two segments of a robot (i.e. self-collisions) and with the environment (ubiquitous in walking or grasping), produce external forces that strongly influence the system dynamics. The inputs of block B are the state variables in the general sense, i.e. also including variables capturing the environment state, and the outputs are the contact forces. We discuss this block in Section 5.

To model the actuators used in COMAN, a simulator should provide a tool that allows to integrate the differential equations for the actuators internal state variables together with the MBS mechanical equations (so-called "strong coupling" for multiphysics systems [3]). For the series elastic actuators equipping this robot, these equations are those governing the motor current dynamics and external position. The inputs of this block (C in in Figure 1) are the control input (in our case voltages and spring pretensions) and the state variables. The outputs are the actuator forces and the derivatives of the actuator state variables.

A unified interface between the robot controller and simulator is very helpful for facilitating the development of the controller robot. Ideally, the developed controller should not depend on the specific command and data format of the simulator (Figure 1, block E) (see, e.g., [8, 9]). This would permit to easily transfer the controller code to the real robot or to other simulators, providing connections to a unique interface.

3 DIRECT DYNAMICS MODULE

The direct dynamics module aims at establishing the equation of motion of the MBS using a formalism based on Newton-Euler, virtual work or Lagrange equations (direct dynamics module). Here we focus on the influence of relative vs. absolute and symbolical vs. numerical approaches for the direct dynamics module.

3.1 Relative vs. absolute coordinates

The dynamic equations algorithm computes the MBS velocities and accelerations when the initial positions, velocities and the time evolution of the external and internal forces are given. To this respect, two approaches compete in the literature, namely those based on relative and absolute coordinates.

Most physics engines (e.g., Robotran, MuJoCo [10], and Simbody) use a minimal set of relative coordinates to determine the state of a multibody system and to avoid introducing algebraic constraints when possible. It means that for tree-structured multibody systems – in which bodies are connected without explicit constraints between the generalized coordinates and velocities – the number of generalized coordinates introduced in the system of equations is equal to the number of degrees of freedom of the system. To capture the connection between two adjacent bodies, for example by means of a revolute joint, only one coordinate and one velocity are needed – namely an angle and its time derivative. The set of Newton-Euler equations governing the dynamics of such a system is typically built by a recursive algorithm [11].

Alternatively, in the physics engines that were initially built for video-games and now are widely

used in robotics (e.g. Open Dynamics Engine (ODE) [6], Bullet [12]), absolute coordinates are used to specify the positions of the rigid bodies. It means that for the connection between two adjacent bodies (e.g. by means of a revolute joint), such an engine requires to introduce 6 DoFs per body and 5 bilateral algebraic constraints. This paradigm likely emerged because of the necessity to compute a lot of random impacts, ubiquitous in video games. The impacts were treated as simultaneously added algebraic unilateral constraints. However, this approach may lead to unrealistic behavior of the systems, constraint violations and inaccuracies (about artifacts in video-game engines and their reasons, see chapter 1.2 in [13]).

We carried out a comparison of absolute and relative approaches on a simple example of a compound pendulum (a mass, a length l of a segment between the fixed point and the center of mass, gravity acceleration g normalized to 1, diagonal inertia matrix with principal moments equal to 0.5). These dynamic equations were simulated in the ODE physics engine (absolute coordinates) and Robotran (relative coordinates). Since it is impossible to separate equation generation and numeric integration in ODE, motion was simulated for 25 units of time (one second equals one unit of time multiplied by $\sqrt{\frac{g}{l}}$, where g and l are in meters/sec^2 and meters, respectively). For ODE we used a built-in numeric integrator and the default parameters for the variables governing stability and accuracy (CFM equals 1×10^{-10} and ERP equals 0.2, for more details see ODE wiki [14]). We used equations generated by Robotran in Matlab. The selected numeric integrator was the Runge-Kutta method of 4-5 order with time-varying step. The absolute and relative tolerances were equal to 2.2×10^{-14} (ode45). As ground truth solution, the differential equations of the pendulum were integrated with the Matlab ode45 function with the same parameters. Initial conditions were set to get a continuous rotational motion of the pendulum (pendulum is oriented downward and the angular velocity equals 1.8). The full mechanical energy fluctuations and the angular coordinate error over time for Robotran are around 4×10^{-13}, i.e. the absolute tolerance multiplied by the simulation time. The error of the ODE solution depends on the time-step. The smallest error was achieved for a time-step of about 1×10^{-7}: constraint violation is around 6×10^{-13}; full mechanical energy fluctuations are around 7×10^{-8}. The time-curve for angular coordinate error is shown in Figure 2. The error grows up with time and cannot be decreased with smaller time-steps. Furthermore, it is likely that such errors would propagate over serial joints and segments for systems having more DoFs than the simple one presented here. Finally, this tolerance was achieved at the expense of a greedy simulation time (for ODE it took more than 5 hours to simulate 25 time units on a 3.4 GHz PC, for Robotran in Matlab environment, 7.4 sec only).

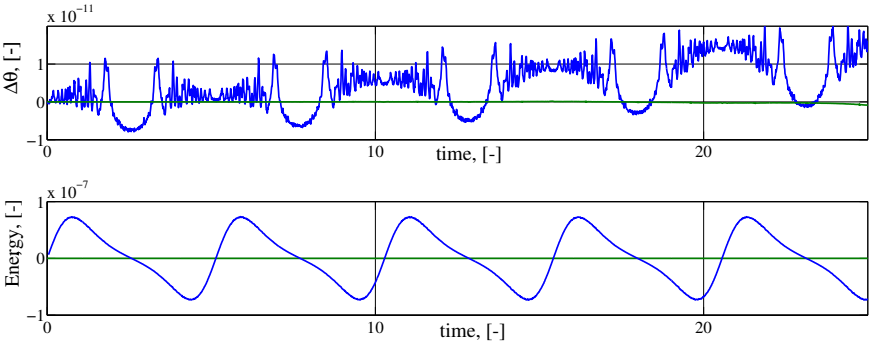

Figure 2. Absolute error of the angular coordinate and full mechenical energy of the compound pendulum for ODE (blue line) and Robotran (green line).

3.2 Symbolic vs. numeric generation

Again, two schools compete regarding the computation of the direct dynamic model, namely symbolic and numerical. A symbolic generation for a given system is run only once. Its output is a symbolic code providing the time derivatives of the state variables in analytical form. Examples of multibody simulators using symbolic algorithms are Robotran and MapleSim [15]. This approach permits to apply automatic symbolic simplifications on the dynamic equations (for example, Robotran performs trigonometric and recursive simplifications) during the equation generation. In the case of sparse mass matrices, the Robotran generator eliminates the useless terms. In the case of recursive simplifications, the full equations that are not used in the subsequent steps are also eliminated (for details, see [3]).

The automatic procedure for numeric generation of derivatives is the same for all multi-body systems, so it is impossible to perform 'system-specific' simplifications, as offered by the symbolic approach. This automatic rebuilding of the derivatives during time-integration is potentially helpful if event-based changes appear in the system, for instance if the system includes unilateral constraints like contacts. We will focus on the connection between the contact algorithm and symbolic/numerical algorithms in the following section.

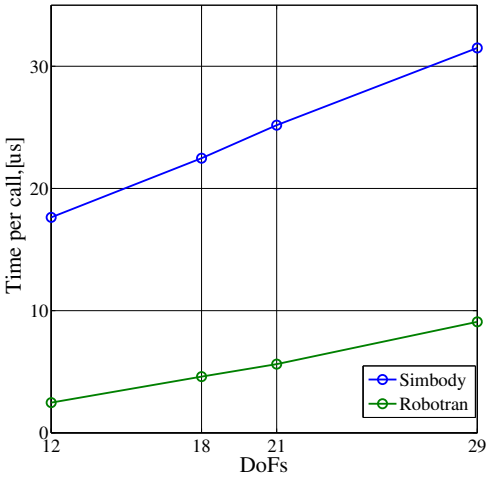

Figure 3. CPU time for the computation of generalized accelerations versus number of DoFs for the COMAN multibody system in Simbody (blue line) and Robotran (green line) physics engines.

As an illustration of the potential superiority of the symbolic vs. numerical approach in direct dynamics module when accuracy and speed are required, we carried out a comparison between two representative algorithms of each family. The tested setup was a tree-like multibody system representing a humanoid robot (full description is provided in Section 2). This MBS was created in two different simulators: Robotran, adopting the symbolic approach, and Simbody, adopting the numerical one (both in relative coordinates), with identical inertial and geometric parameters. For both engines, we provided the same discrete random inputs: 29 joint positions, 29 joint velocities, and 23 generalized torques in the rotational joints. We then compared the produced outputs, i.e. the generalized accelerations, and computation time. The produced accelerations were thus not integrated to get the next velocity, in order to compare the computational time of the direct dynamics module only, disregarding issues related to the selection of a specific time integrator. These simulations revealed that the average relative difference in the generated accelerations was $1.9 \times 10^{-11}\%$. Both physics engines are thus eligible. The computation speed for full COMAN

is approximately 3.5 times faster for the Robotran physics engine as compared to the Simbody one (on the same computer). Figure 3 shows an evolution of the computational time required to compute the joint accelerations, as a function of the number of degrees of freedom in this system. Comparison was performed for 12 (waist and one leg), 18 (waist and two legs), 21 (full torso and two legs, without arms), and 29 degrees of freedom. It shows that for both algorithms, computational time grows up as a linear function of the number of DoFs. Moreover, the Robotran slope is about 2 times smaller than the Simbody one. Therefore, the relative superiority of the symbolic vs. numeric approach further grows up with the number of DoFs.

4 DYNAMIC EQUATIONS AND COMPLIANT ACTUATORS' MODELS

Based on the comparison and arguments provided in Section 3, the Robotran symbolic generator was selected for symbolical generation of MBS dynamic equation for DirectDynamics module together with the compliant contact processing. So, the second-order mechanical multibody model of the robot is provided by Robotran in the symbolic form:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})+\boldsymbol{G}(\boldsymbol{q})+\boldsymbol{N}^c(\boldsymbol{q})\boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}})=\boldsymbol{T}_1(\boldsymbol{q})\boldsymbol{\tau}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{q}^m,\dot{\boldsymbol{q}}^m)+\boldsymbol{T}_2(\boldsymbol{q})\boldsymbol{\tau}^p(\boldsymbol{p},\boldsymbol{q}), \tag{1}$$

where $\boldsymbol{q}=(q_1,\ldots,q_{29})$ is the vector of angular joint positions plus cartesian coordinates of the floating base, $\boldsymbol{M}(\boldsymbol{q})$ is the mass-inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ represents Coriolis and centrifugal forces, $\boldsymbol{G}(\boldsymbol{q})$ represents gravitational forces and torques. Contact forces and torques $\boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}})$ have $6N^c$ components, where N^c is the number of active contacts. For walking, for example, we have N^c equal to 1 or 2 depending on a gate phase. Control is provided by the 23 serial actuators's torques $\boldsymbol{\tau}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{q}^m,\dot{\boldsymbol{q}}^m)$ and 6 parallel actuators' torques $\boldsymbol{\tau}^p(\boldsymbol{p},\boldsymbol{q})$. Control torques and contact forces propagate through the MBS by 29×23, 29×6 and $29\times 6N^c$ matrices $\boldsymbol{T}_1(\boldsymbol{q})$, $\boldsymbol{T}_2(\boldsymbol{q})$, and $\boldsymbol{N}^c(\boldsymbol{q})$ correspondingly (they are all symbolically generated and simplified by Robotran).

Dynamic models of realistic actuators are often missing in existing simulators, where the joint actuators are rather considered as sources of pure torque or position (depending on the mode of control). The motor inertia when reflected to the output of gearbox often has the same order of magnitude as the link inertia. On the other hand, the robotics community recently promoted the use of compliant actuators to enhance the robot safety and energy efficiency, mainly when contacts with the environment are ubiquitous. Typically, these solutions require to design flexible joint robotic systems, where the electric motors are connected in series with a compliant element to mainly provide better force regulation and also shock absorption against environmental impacts. A diagram of such actuators as used in [2] is shown in Figure 4. Therefore the links in robots with flexible joints are indirectly driven by the torque provided by the spring deflections.

For each actuated joint we have an equation that governs motor inner variable q_i^m:

$$J\ddot{q}_i^m+D_m\dot{q}_i^m+\tau_i(q_i,\dot{q}_i,q_i^m,\dot{q}_i^m)=K_t\,I_i^m,\quad i=1,\ldots,23 \tag{2}$$

where J is the motor inertia (including rotor and the moving parts such as gearbox), D_m is the motor's mechanical damping, K_t is torque constant and I^m is the motor current. The motor load torque is given by

$$\tau_i(q_i,\dot{q}_i,q_i^m,\dot{q}_i^m)=K_s(q_i-q_i^m)+D_s(\dot{q}_i-\dot{q}_i^m), \tag{3}$$

where K_s and D_s are stiffness and damping of serial spring respectively.

The current dynamics for i-th motor is modelled as

$$L\dot{I}_i^m+RI_i^m+K_\omega\dot{q}_i^m=V_i \tag{4}$$

where L,R,K_ω and V_i are motor inductance, resistance, back EMF constant and applied input voltage.

If a parallel spring is further added, its torque is given in (5), where r is the radius of the joint's pulley, k_p and d_p are the spring stiffness and damping. The leg pitch joints with a parallel spring are indexed with i where $i = 7, 10, 12, 13, 16, 18$ according to Figure 1.

$$\tau_j^p(p_j, q_i) = \begin{cases} k_p(p_j - rq_i) + d_p(\dot{p}_j - r\dot{q}_i), & \text{if } (p_j - rq_i) < 0 \\ 0, & \text{if } (p_j - rq_i) \geq 0 \end{cases}, \qquad j = 1, \ldots, 6. \tag{5}$$

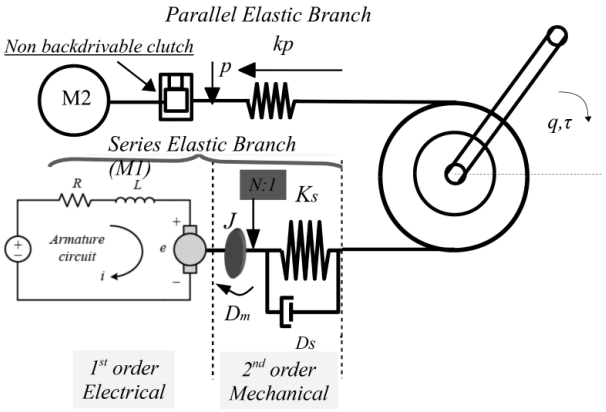

Figure 4. The series and parallel branch of a compliant actuator of the COMAN.

The last term that we need to describe here is the contact term $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$. A contact dynamics module does not exist in the Robotran and has to be implemented. In the next section we provide some details on the algorithms and share our experience of coupling the Robotran simulator with external contact library.

5 CONTACT PROCESSING

The physics of contact is very complicated by itself [16]. Developers of contact modules face different types of problems, from the challenges of establishing the constitutive laws of impacts to the mathematical problems of stability and convergence of numerical methods. Imprecisions of the contact model and parameters introduce the largest inaccuracies in simulation [17]. Contact algorithms are quite expensive regarding computational cost. Moreover, a given algorithm that a simulator uses for contact modeling influences the choice of the associated numeric integrators and the internal representations of the equations of motion. Choosing a contact algorithm is thus challenging. Here we provide some reasoning to help making this choice. It is important to make a distinction between compliant and rigid contacts. Precise definitions and extended comparisons between these two approaches for contact modeling can be found in [18]. Here below, the pros and cons of rigid and compliant contact modeling are overviewed with a global point of view, i.e. with the objective to emphasize connections between the different contact modules with the equation formalism as discussed above.

5.1 Rigid Contact

Briefly speaking, rigid contact means that a contact between two (or more) bodies is treated as an instantaneously imposed unilateral constraint without interpenetration of bodies. So the inputs of such algorithm are the positions and velocities of bodies before contact, the constitutive impact law (i.e. the rule to compute the post-impact velocities – in the simplest case, through a restitution coefficient), and some properties of the system, like inertial matrices and bilateral constraints. The outputs of the algorithm are the velocities of all bodies after impact and, the corresponding

reaction forces. These velocities should satisfy all constraints (unilateral and bilateral). There is thus no interpenetration of the bodies and the mechanical compression and decompression phases are not explicitly calculated. The contact reaction forces – making the contact constraints to be satisfied – depend on other (bilateral or unilateral) constraints that are imposed to the system. Different algorithms exist for rigid contact processing (see [18], [19]): LCP (linear complementarity problem), NLCP methods (non-linear complementarity problem) and others. In fact all of them implement different numeric algorithms for solving the same systems of equations (sometimes called complementary slackness mechanical systems, see [18]).

We identified two bottlenecks related to this approach. First, regarding the constitutive impact law, for complicated systems and multiple impacts, a simple restitution coefficient may vary depending on the impact conditions. Second, rigid algorithms treat unilateral constraints as imposed to points and not to surfaces. So it is necessary to specify the points where the contact constraints (i.e. no interpenetration) are expected to be fulfilled. While this is reasonably easy to do for primitives like spheres and boxes, this is much more challenging for complex and possibly non-convex surfaces covering robots. The localization of the contact points in this case is a difficult mathematical problem. Also, rigid algorithms could introduce inaccuracies when used with some passive compliances like rubber covers of some parts of robots. Nevertheless these algorithms are fast and solve impacts in a single time step, ideally preventing penetration of the bodies. All 'absolute coordinate' engines (ODE, Bullet) and MuJoCo implement rigid contact algorithms.

5.2 Compliant Contact

Compliant contact algorithms integrate contact modelling within the time-dependent multibody equations, through a compression and decompression phase. A repelling force is calculated via a visco-elastic model using the relative distance and velocity between pairs of points on the surfaces of the bodies in contact (elastic foundation model, EFM) [5, 18]. In general, for two bodies bounded by triangular meshes M_1 and M_2, a resulting contact force and torque is a sum:

$$\boldsymbol{f}^{contact}(\boldsymbol{q}^c, \dot{\boldsymbol{q}}^c) = 0.5\sum_1 \left(\boldsymbol{f}_{normal} + \boldsymbol{f}_{tangent}\right) + 0.5\sum_2 \left(\boldsymbol{f}_{normal} + \boldsymbol{f}_{tangent}\right) \tag{6}$$

$$\boldsymbol{\tau}^{contact}(\boldsymbol{q}^c, \dot{\boldsymbol{q}}^c) = 0.5\sum_1 \boldsymbol{r_i} \times (\boldsymbol{f}_{normal} + \boldsymbol{f}_{tangent}) + 0.5\sum_2 \boldsymbol{r_i} \times (\boldsymbol{f}_{normal} + \boldsymbol{f}_{tangent}) \tag{7}$$

The summation $\sum_1$ goes through the triangles of the mesh M_1 that overlap the mesh M_2 and vice versa for $\sum_2$. Forces $\boldsymbol{f}_{normal}$ and $\boldsymbol{f}_{tangent}$ are normal and tangent components of elementary repelling force applied to the current triangle. They depend on relative propagation and velocity of the corresponding triangle with respect to the opposite mesh; $\boldsymbol{r}_i$ is a vector from a center of a contact patch to the center of the triangle. Thus, such an algorithm calculates a force and a torque (applied to the computed center of patch) from the positions and velocities of both bodies in contact,i.e. $\boldsymbol{q}^c$ and $\dot{\boldsymbol{q}}^c$. Models for $\boldsymbol{f}_{normal}$ and $\boldsymbol{f}_{tangent}$ can vary for particular models (vicso-elastic models, viscous, dry, or Stribeck friction, etc).

The main shortcomings of this approach are (i) the difficulties of selecting the appropriate parameters for the elastic layer; and (ii) a ubiquitous trade-off between keeping the interpenetrations of the bodies within reasonable limits and maintaining the computational cost low enough. Indeed, the stiffness of the associated differential equations is directly correlated with the interpenetration magnitude: for high stiffness k, the bodies interpenetrations are of order $O(k^{-2/3})$ and the time-step is of order $O(k^{-1/2})$[18]. So, the computational speed and accuracy critically depends on this parameter. The EFM algorithm is faster and at the same time well approximates finite elements algorithms (FEM), which are considered as an etalon for the deformable bodies [20], [21]. EFM algorithms can simulate interpenetration of the body pairs, micro-slips, and repeated impacts which also occur in the real world.

To summarize, rigid contacts are fast and generate zero or negligible interpenetration of the bodies. This approach is more suitable for perfectly rigid bodies, for example for stainless balls, or railway

wheels. It can be adapted to real-time applications when some lack of accuracy can be permitted. Rigid contact modeling can also produce incorrect output, mainly for complex body shapes. Since it is difficult to separate the direct dynamics module from the contact one, the potential inaccuracy source is usually challenging to identify. This approach can further produce inaccuracies in simulations of robots with passive compliance, for example, covered by a deformable layer. Compliant contact algorithms are more expensive regarding computational load and imply additional stiffness to the numerical solutions of the ordinary differential equations. These algorithms can be easily isolated from the other parts of the simulator and thus coupled to any core of the multibody simulator. They are more accurate for deformable bodies such as rubber foots and hands, if appropriate parameters are well estimated. These algorithms require more computational power to be executed in real-time applications, although not out of the capacities of modern computers.

We propose to use for contacts forces $\boldsymbol{n}$ in Equations (1) the compliant model for contacts (6,7): $\boldsymbol{n} = (\boldsymbol{f}_1^{contact}, \boldsymbol{\tau}_1^{contact}, \ldots, \boldsymbol{f}_{N^c}^{contact}, \boldsymbol{\tau}_{N^c}^{contact})$. Here, we augment Robotran simulator with external open-source library providing compliant contact. This offers a general algorithm for handling contact in the Robotran framework, especially for complex objects whose shape requires to be modeled by a polygonal mesh. In the next section we explain how we coupled the Simbody compliant library with the MBS equations of motion generated by Robotran.

5.3 COUPLING ROBOTRAN SIMULATOR WITH COMPLIANT CONTACT LIBRARY

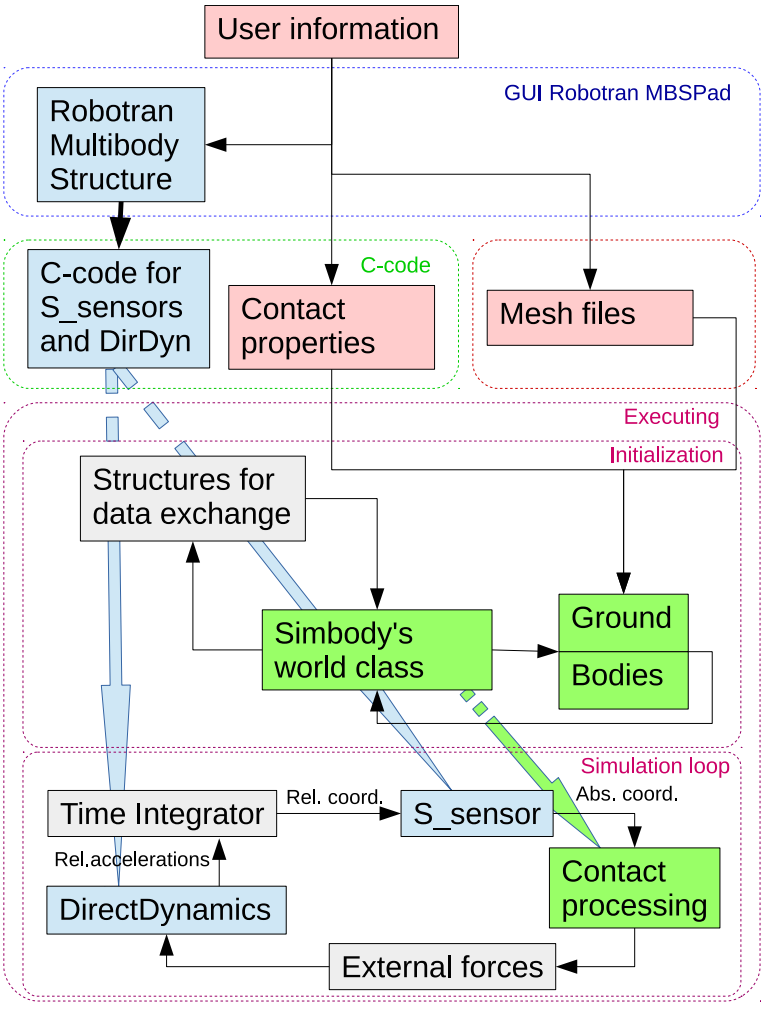

Figure 5. Scheme of coupling between Robotran and Simbody.

Contact processing was made possible by coupling Robotran with an external C++ contact library

developed by the Simbody team [5]. This library was selected based on the following reasons:

- It is open-source and is distributed under permissive Apache 2.0 License;
- It provides compliant contact of two types: Hertz-model and elastic foundation model (EFM, more details can be found in [5, 21]).
- It allows to use WaveFront object files encoding triangular meshes to define complex body shapes;

The scheme of coupling is shown in Figure 5. Concretely, during the building of the multibody structure (that has to be constructed in a the Robotran's GUI-editor MBSPad), the user needs to add S-sensors and F-sensors for each body which will have contact surfaces (i.e. the so-called contact bodies, CBs). A "S-sensor" is a Robotran tool that is used for mapping relative coordinantes and velocities to absolute coordinates and velocities of the CBs. A "F-sensor" is a Robotran tool that allows to define user-external forces. The Robotran server then generates both the MBS equations of motion, and the coordinate tranforms being required to manage contacts through S-sensors and F-sensors. The user further needs to specify the number of CBs, the physical properties governing contact (such as stiffness, viscosity, friction coefficients) for each of them (including ground) and to define the surface of contact, through primitives or meshes, in a C-code template.

The execution starts with the initialization of all the variables that are used during the simulation loop. An instance of the Simbody class world is created. In this world a shadow free body with 6 DoFs for each CB is created. The computation is thus separated between Robotran (in charge of the dynamic integration, complying with the mechanical constraints between the successive bodies) and Simbody (in charge of the computation of the contact forces through the 6-DoFs shadow bodies). Simulating COMAN walking on a rough terrain requires creating only two shadow bodies, i.e. one for both feet, with mesh contact surfaces and a ground in the Simbody world.

In the simulation loop, relative positions and velocities are transformed into absolute positions and velocities, through the code generated for the S-sensors. These positions and velocities are then imposed for each shadow CB in the Simbody world. Then the Simbody library processes the contact and returns values of external forces and moments due to contacts. These forces and moments are then applied to the MBS in Robotran (relying on the F-sensors) and included to update the bodies dynamics.

In Figure 6 two representative tasks that extensively use 3D mesh-to-mesh contact are shown: that are a manipulation task and a locomotion task (with a gait controller presented in [22]).

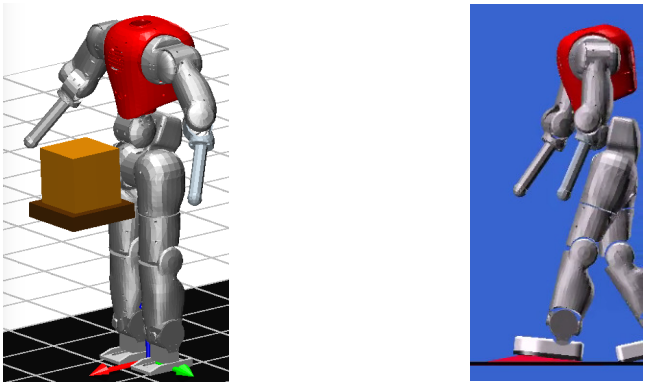

Figure 6. On the left: Manipulation task – COMAN takes a box from a table. On the right: Locomotion task – COMAN goes over a 3D bump .

6 CONCLUSIONS

In the frame of this work, a fast and accurate model of a humanoid robot is derived. Comparison of time-efficiency of different method for direct dynamics module is provided. The full electromechanical models for series compliant actuators and for parallel elastic branch are derived. Coupling of symbolic MBS differential equations with open-source external compliant contact library is made. The simulator that integrates over time the presented multiphysics model shows reliable results for locomotion and manipulation tasks.

REFERENCES

[1] Instituto Italiano di Technologia, COmpliant HuMANoid Platform (COMAN), http://www.iit.it/en/advr-labs/humanoids-a-human-centred-mechatronics/advr-humanoids-projects/compliant-humanoid-platform-coman.html

[2] N. G. Tsagarakis, S. Morfey, H. Dallali, G. A. Medrano-Cerda, D. G. Caldwell. An asymmetric compliant antagonistic joint design for high performance mobility. Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 5512–5517, 2013.

[3] N. Docquier, A. Poncelet, P. Fisette. ROBOTRAN: a powerful symbolic gnerator of multibody models Mechanical Sciences, vol. 4, no. 1, pp. 199–219, 2013.

[4] H. Dallali, M. Mosadeghzad, G. Medrano-Cerda, N. Docquier, P. Kormushev, N. Tsagarakis, Zh. Li, D. Caldwell. Development of a dynamic simulator for a compliant humanoid robot based on a symbolic multibody approach IEEE International Conference on Mechatronics (ICM), pp. 598–603, 2013.

[5] M. A. Sherman, A. Seth, S. L. Delp. Simbody: multibody dynamics for biomedical research Procedia IUTAM, vol. 2, pp. 241–261, 2011.

[6] R. Smith. Open Dynamics Engine (ODE), http://www.ode.org/.

[7] http://www.iit.it/en/advr-labs/humanoids-a-human-centred-mechatronics/advr-humanoids-projects/series-elastic-actuators-seas.html

[8] P. Fitzpatrick, G. Metta, L. Natale. Towards long-lived robot genes. Robotics and Autonomous Systems, Vol. 56, no. 1, pp. 29 – 45.

[9] T.Habra, H.Dallali, A. Cardellino, L. Natale, N. Tsagarakis, P. Fisette, and R. Ronsse, "Robotran-Yarp interface: a framework for real-time controller development based on multibody dynamics simulation," Submitted to ECCOMAS Thematic Conference on Multibody Dynamics 2015.

[10] E. Todorov, T. Erez, Y. Tassa. MuJoCo: A physics engine for model-based control. Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, 2012, p. 5026–5033.

[11] J-C. Samin, P. Fisette. Symbolic Modeling of Multibody Systems. Kluwer Academic Publishers, Dordrecht, 2003.

[12] Bullet Physics library. http://bulletphysics.org/wordpress/

[13] G. van den Bergen, D. Gregorius. Game physics pearls. Natick, Mass.: A.K. Peters, 2010.

[14] Manual - ODE Wiki. http://ode-wiki.org/wiki/index.php?title=Manual/

[15] MapleSim - High Performance Physical Modeling and Simulation - Technical Computing Software, http://www.maplesoft.com/products/maplesim/index1.aspx.

[16] P. J. Blau. Friction science and technology: from concepts to applications. Boca Raton, FL: CRC Press, 2009.

[17] A. Chatterjee, A. Ruina. A new algebraic rigid body collision law based on impulse space considerations ASME J.Appl.Mech.,vol.65, no.4, p. 939–951, 1998.

[18] B. Brogliato, A. ten Dam, L. Paoli, F. Génot, M. Abadie. Numerical simulation of finite dimensional multibody nonsmooth mechanical systems. Applied Mechanics Reviews, vol. 55, no. 2, p. 107, 2002.

[19] E. Drumwright, D. A. Shell An evaluation of methods for modeling contact in multibody simulation Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011, p. 1695–1701.

[20] G. Hippmann. An algorithm for compliant contact between complexly shaped surfaces in multibody dynamics Multibody Dyn. Jorge AC Ambrosio Ed IDMECIST Lisbon Port. Vol. 14, 2003.

[21] A. Pérez-González, C. Fenollosa-Esteve, J. L. Sancho-Bru, F. T. Sánchez-Marín, M. Vergara, and P. J. Rodríguez-Cervantes A modified elastic foundation contact model for application in 3D models of the prosthetic knee. Medical Engineering & Physics, vol. 30, no. 3, pp. 387–398, 2008.

[22] N. Van der Noot, A. Barrea. Zero-Moment Point on a bipedal robot under bio-inspired walking control. Mediterranean Electrotechnical Conference (MELECON), 2014 17th IEEE, pp. 85–90, 2014.