Noname manuscript No.
(will be inserted by the editor)

Multi-physics Modelling of a Compliant Humanoid Robot

Alexandra A. Zobova · Timothée Habra · Nicolas Van der Noot · Houman Dallali · Nikolaos G. Tsagarakis · Paul Fisette · Renaud Ronsse

Received: date / Accepted: date

Abstract We present a multibody simulator being used for compliant humanoid robot modelling and report our reasoning for choosing the settings of the simulator's key features. First, we provide a study on how the numerical integration speed and accuracy depend on the coordinate representation of the multibody system. This choice is particularly critical for mechanisms with long serial chains (e.g. legs and arms). Our second contribution is a full electromechanical model of the inner dynamics of the compliant actuators embedded in the COMAN robot, since joints' compliance is needed for the robot safety and energy efficiency. Third, we discuss the different approaches for modelling contacts and selecting an appropriate contact library. The recommended solution is to couple our simulator with an open-source contact library offering both accurate and fast contact modelling. The simulator performances are assessed by two different tasks involving contacts: a bimanual manipulation task and a squatting tasks. The former shows reliability of the simulator. For the latter, we report a comparison between the robot behaviour as predicted by our simulation environment, and the real one.

Keywords Multibody dynamics · COMAN · Compliant actuators · Contact dynamics · Humanoid robot · Robotran · Simbody

A.A. Zobova
Faculty of Mechanics and Mathematics, Lomonosov Moscow State University, Leninskie Gory, 1, 119991, Moscow, Russia
E-mail: azobova@mech.math.msu.su

H. Dallali, N.G. Tsagarakis
Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego, 30, 16163, Genova, Italy
E-mail: [houman.dallali, nikos.tsagarakis]@iit.it

N. Van der Noot
Biorobotics Laboratory, Institute of Bioengineering , École polytechnique fédérale de Lausanne (EPFL), EPFL STI IBI BIOROB, Station 9, CH-1015, Lausanne, Switzerland
E-mail: nicolas.vandernoot@epfl.ch

T. Habra, N. Van der Noot, P. Fisette, R. Ronsse
Center for Research in Mechatronics, Institute of Mechanics, Materials, and Civil Engineering, Université catholique de Louvain (UCL), Place du Levant 2, 1348, Louvain-la-Neuve, Belgium
E-mail: [timothee.habra, nicolas.vandernoot, paul.fisette, renaud.ronsse]@uclouvain.be

1 Introduction

In this paper, we investigate the COMAN humanoid robot [1] and develop a multibody simulator with the following key features:

1. an efficient multibody dynamics offering fast computation;
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 C++ functions provided by the open-source library Simbody [5]. This model proved to be useful to support the design phase of the robot [6] providing data for the sizing and selection of actuation systems as well as other mechanical components like bearings and structural parts, and to speed up the synthesis and tuning of control algorithms, like e.g. a locomotion controller [7]. It can further be straightforwardly adapted to the simulation of other robots.

The structure of the article is the following. In Section 2, we describe the multibody system (MBS) of the humanoid robot being simulated and present the general structure of a simulator. In Section 3, we discuss how the MBS formalization (particularly, the choice of the coordinates and the equations' generating procedure) influences the speed and accuracy of the simulation. We compare three open-source multibody platforms used in robotics – which are the Robotran generator [3], Open Dynamics Engine ODE [8] and Simbody [5]. In Section 4, the electromechanical model of the compliant actuators is discussed. In Section 5, we describe two approaches for contact simulation — rigid and compliant contact — and show our reasoning for choosing one of them. Also we share the technical details for coupling the C-code of the Robotran generator with the contact module of the open-source physics engine Simbody. The analysis of a manipulation task's simulation is provided in Section 6. It was performed in order to assess the overall behaviour of the simulator, its real-timeness, and to investigate the influence of some parameters (i.e. the actuators stiffness). Section 7 describes a squatting experiment, emphasizing the comparison between the simulated results and experimental data.

2 Structure of the robot and simulator overview

The multibody model of COMAN consists of 24 absolutely rigid bodies and has a tree-like structure as shown in Fig. 1. A floating base is attached to the waist of the robot and has 6 degrees of freedom (DOF). 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 DOF (6 for the floating base and 23 for the actuated joints).

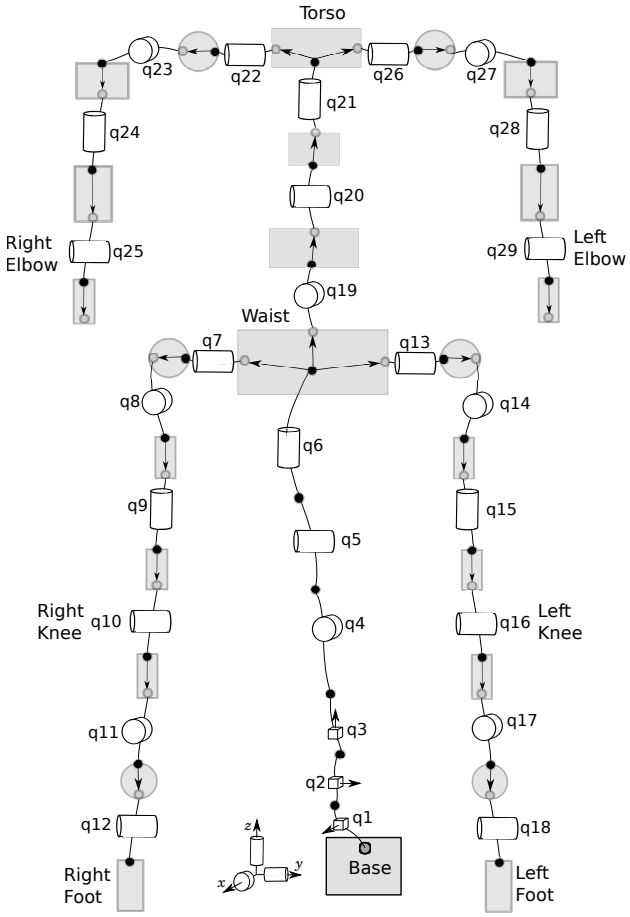

Fig. 1 MBS COMAN model. The cylinders and the cubes show rotational and translational degrees of freedom.

The choice of floating base attachment has a significant impact on the kinematic and dynamic models of a humanoid robot. The most common choices are to attach the floating base either to the feet or to the waist. Attachment of the floating base to the waist displays a number of advantages. First, the mass-inertia matrix structure depends on the floating base attachment and has a smaller condition number if the floating base is attached to the largest mass in the MBS, i.e. the waist in our case. Moreover, the symmetry naturally obtained by putting the floating base to the waist is desirable in developing locomotion algorithms. Consequently, attaching the floating base to the waist is common practice in the literature of humanoid modelling [9,10]. This is also what we performed here.

Similarly to the real robot, the model is equipped with 23 series elastic actuators in the revolute joints, each producing a motor torque. It depends on the corresponding joint angle, joint velocity and the motor angle and velocity. Dynamics of this inner actuator variable is governed by a differential equation involving the current in the motor, which in turn satisfies a DC motor differential equation with

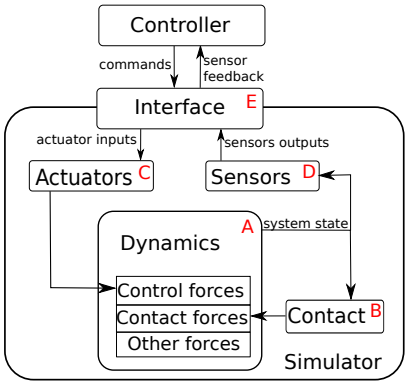

Fig. 2 Simulator structure.

a controllable input voltage . Furthermore, the pitch leg joints can be equipped with tunable springs going in parallel to the motors [2]. These springs deliver additional torques depending on the controllable springs' pretensions. Physical models of both types of actuators and differential equations that govern the variables 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 Fig. 2. 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 the different bodies. The way to represent 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 Fig. 2). 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 multi-physics 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 Fig. 2) 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. Block D transmits bodies' positions and velocities from direct dynamics block A to the controllers. Thus, the block D represents the sensors and inertial measurement units in the simulator.

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 (Fig. 2, block E) (see, e.g., [11]). This can be achieved through the

interacting of the simulator with a middleware that permits the easy transfer of the controller code to the real robot or to other simulators, providing connections to a unique interface [12].

3 The influence of the MBS formalism

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. The modules computes the derivatives with respect to time of the MBS's coordinates and velocities knowing the structure of the MBS and applying forces (internal and external ones). In [16], the mentioned three methods were compared for different types of MBS's structure: in particular, the generation time and memory consumption were analyzed. The analysis was done for symbolic generation in relative coordinates on the Robotran direct dynamics module [3]. Here we focus on the influence of the MBS formalization in the direct dynamics module (in particular, relative vs. absolute and symbolical vs. numerical approaches) on three different engines – Robotran [3], Open Dynamics Engine ODE [8] and Simbody [5]. We chose this three engines because they are open-code and freely available for scientific research. Moreover, they are popular in the robotics community for humanoids modelling, according to [13], and use different approaches to derive and compute the direct dynamics equations (for a review of different simulators, see also [14]).

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 [15], 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 [16].

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) [8], Bullet [17]), absolute coordinates are used to specify the positions of rigid bodies. Absolute coordinates require three Cartesian coordinates for the center of mass of each body, expressed in the inertial frame, and some representation of the orientation matrix for each body again. To represent a connection between two adjacent bodies (e.g. a revolute joint), such an engine introduces 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 behaviour of the systems, constraint violations and inaccuracies (about artefacts in video-game engines and their reasons, see chapter 1.2 in [18]).

We carried out a comparison of absolute and relative approaches on two simple examples: a single pendulum and a double pendulum. A single pendulum is a rigid body that can freely rotates about the fixed horizontal axis in a homogeneous gravity field. The mass, the length l of a segment between the fixed point and the center of mass (COM), gravity acceleration g are normalized to 1 (so time is non-dimensional), the inertia matrix about COM is diagonal with principal moments equal to 5.5. The double pendulum is a chain of two rigid bodies. The first body rotates freely around fixed horizontal axis, while the second one rotates with respect to the first body around the parallel axis. All the parameters for the first body were the same, the distance between two axis equals is $2l$, the distance between the moving rotation axis and the second body's COM equals 2.5, the second body's inertia matrix about COM is diagonal with principal moments equal to 2. Ideally, both models should conserve the mechanical energy; i.e. $E(t) = \text{const}$.

Robotran uses relative coordinates to represent MBS systems: for pendulums, these are the angles of rotation about the horizontal axis. ODE, which uses absolute coordinates, introduces 3 Cartesian coordinates for the COM of each body. Orientation is represented with four numbers to form the quaternion. The quaternion should have unity length (one constraint). The cylindrical joint introduces five additional algebraic constraints.

For ODE, we used a built-in numeric integrator with fixed time-step and the default parameters for the variables governing stability and accuracy: the constraint force mixing parameter (CFM) equals 10^{-10}, the error reduction parameter (ERP) equals 0.2 [19]. Robotran generated equations were integrated by Runge-Kutta 4th-order integrator with fixed time step.

For both models and for different time-steps, Table 1 reports the following parameters: number of coordinates #Q and number of algebraic constraints #AC, maximum of the energy violation $\Delta E = (E(t) - E(0))/E(t)$ during 10 time-units[1] of simulation, maximum of algebraic constraint violation ΔAC (we use the relative error in the distance between the COM and the axis of rotation), and average real-time factor parameter (RTF) — the ratio of the simulated time (10 sec) to the time spent for the simulation. ΔAC is calculated only for ODE, since the Robotran does not use algebraic contraints for these models. Robotran achieves precision of double type for time-step $\Delta t = 10^{-4}$, with RTF around 20. The same precision on ODE was not achieved even for smaller time-steps.

Energy and constraints violation signalize that time evolution of MBS coordinates could be inaccurate. To show it properly, we consider the following example. The lengths and inertia parameters of the double pendulum were selected such that theoretically the pendulum allows the synchronous bodies' movement, if they have any equal initial angular velocities. It means, that the angle between the bodies remains zero. Thus, the solution is one-periodic, and the period can be analytically computed. For the angular velocity equal to 2.1 rad/time-unit, this

[1] One time-unit equals $\sqrt{l/g}$ seconds, where g and l are in meters/sec^2 and meters, respectively.

		Robotran				ODE				
	Δt	#Q	#AC	ΔE	RTF	#Q	#AC	ΔE	ΔAC	RTF
Single Pendulum	10^{-2}	1	0	$3 \cdot 10^{-8}$	800	7	6	$2{\cdot}10^{-3}$	10^{-3}	10^{3}
	10^{-4}			10^{-13}	31			$8{\cdot}10^{-6}$	10^{-7}	26
	10^{-6}			10^{-13}	0.3			$8{\cdot}10^{-8}$	10^{-11}	0.3
Double Pendulum	10^{-2}	2	0	$3 \cdot 10^{-8}$	500	14	12	$6{\cdot}10^{-3}$	10^{-3}	700
	10^{-4}			10^{-13}	18			10^{-5}	10^{-7}	10
	10^{-6}			10^{-13}	0.2			10^{-7}	10^{-11}	0.1

Table 1 Comparison of Robotran and ODE in simulating a single and a double pendulum.

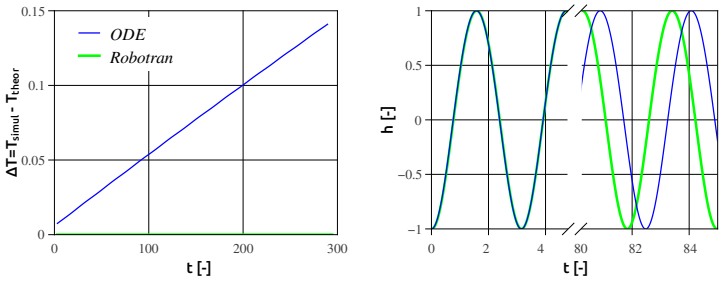

Fig. 3 Double pendulum simulation. On the left: period of rotation versus time for Robotran (green line) and ODE (blue line); on the right: Height of the first segment's COM versus time, calculated by Robotran (solid green line) and ODE

period equals approximately $T_{\text{theor}} = 3.15$ time-units[2]. The left plot on Fig. 3 shows the evolution of the period of simulated rotations versus time for ODE and Robotran for 300 time-units simulation with a time-step of $\Delta t = 10^{-2}$. The period of rotations calculated with the Robotran fluctuates of about $5 \cdot 10^{-9}$ about the theoretical value T_{theor}. For ODE, the same period grows up almost linearly. On the right panel the height of the first segment's COM is shown: at the beginning the time curves coincides, while after 80 time-units the difference between the simulated curves aproaches 50% of maximum height and still continues to increase; at 124 time-unit the pendulums in ODE and Robotran rotate in antiphase.

The models we took as examples are very simple comparing to the real humanoid robot we need to simulate. But even on these two simple models, we see that the simulation accuracy significantly depends on internal representation of the MBS system. The simulations of larger MBS systems require relative coordinates to be accurate and fast.

3.2 Symbolic vs. numeric generation

Again, the symbolic and numerical methods compete regarding the computation of the direct dynamic model. A symbolic generation for a given system is

[2] Analytical result was obtained by Maxima computer algebra system [20], with numerical precision $3.5 \cdot 10^{-14}$.

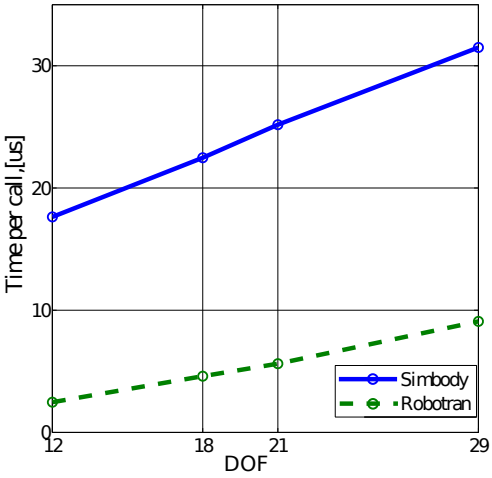

Fig. 4 CPU time for the computation of generalized accelerations versus number of DOF for the COMAN multibody system in Simbody (blue line) and Robotran (green line) physics engines.

run only once. Its output is a symbolic code providing the time derivatives of the state variables in an analytical form. Examples of multibody simulators using symbolic algorithms are Robotran, openSYMORO [21] and MapleSim [22]. This approach permits automatic symbolic simplifications of the dynamic equations during the equation generation. For example, the Robotran symbolic generator performs trigonometric simplifications (like '$\sin^2(q_i) + \cos^2(q_i) = 1$' and '$2\sin(q_i)\cos(q_i) = \sin(2q_i)$') and grouping and factoring in more complicated formulae. In the case of sparse mass matrices, it further disregards the terms which are not necessary for later computations. 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 MBS, 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 rigid contacts. (We will focus on the connection between the contact algorithm and symbolic/numerical algorithms in Section 5.) In the symbolic approach, it is necessary to regenerate the symbolic code, when some structural changes occur. However, in case of compliant contact processing, the structure of the humanoid robot does not change.

As an illustration of the potential superiority of the symbolic vs. numerical approach in the 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 generalized accelerations and computation time. The accelerations were not integrated through time to eliminate issues related to the selection of a specific time integrator. These simulations revealed that the average relative difference in the generated accelerations was about $1.9 \cdot 10^{-11}\%$. 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 that of Simbody (on the same computer). Figure 4 shows an evolution of the computational time required to compute the joint accelerations, as a function of the number of degrees of freedom in the 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, the computational time increases as a linear function of the number of DOF: for inertia matrix decomposition both engines use the algorithms that linearly depends on the number of DOF. However, the Robotran slope is about 2 times smaller than the one of Simbody. Therefore, the relative superiority of the symbolic vs. numeric approach further magnifies with the number of DOF.

4 Dynamic equations and compliant actuators model

Based on the comparison and arguments provided in Section 3, the Robotran symbolic generator was selected for symbolical generation of MBS dynamic equation for the DirectDynamics module together with the compliant contact processing. Thus, 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. During walking, for example, we have N^c equal to 1 or 2 depending on the walking gait phase. Control is provided by the 23 serial actuators 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. All the matrices were symbolically generated and simplified by Robotran by reccurent Newton-Euler method and stored automatically in C-functions.

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 the 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 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 Fig. 5. 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 the motor's 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 mechanical damping, K_t is torque constant and I^m is the motor current. This equation approximates well the motor dynamics if two assumptions hold [23]. They are: A1) the rotor kinetic energy is due to the rotor own rotation and A2) the rotor inertia is symmetric about the rotor axis of rotation. Both of them are justified for COMAN actuators, because of the symmetry of rotors and high gear ratios ($n_i \approx 80 - 100$) [24, Paragraph 13.1.1].

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 the 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 Eq. (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 q_i where $i = 7, 10, 12, 13, 16, 18$ according to Fig. 2.

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

The compliant actuators described here require simultaneous integration of MBS equations (1) and additional differential equations (2), (4) governing the inner variables. This model of the actuators can be easily changed on any other multi-physics model, e.g. hydraulic.

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 Robotran and has to be implemented. In the following section, we provide some details about the algorithms and share our experience of coupling the Robotran simulator with an external contact library.

5 Contact processing

The physics of contact is very complicated by itself [25]. Developers of contact modules face different types of problems, from the challenges of establishing the

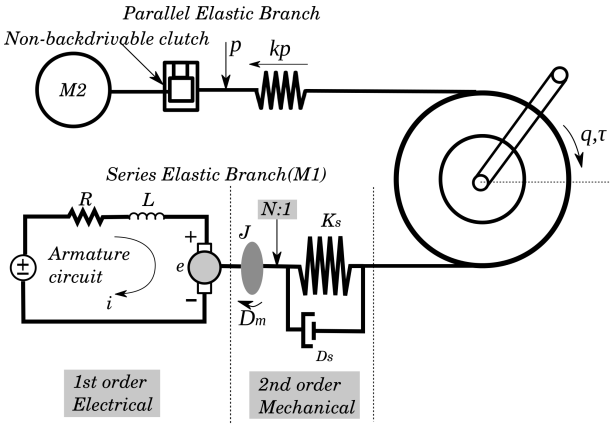

Fig. 5 The series and parallel branch of a compliant actuator of the COMAN.

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 [26]. Contact algorithms are quite expensive in terms of 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 in 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 [27]. 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 algorithms 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 on the system. Different algorithms exist for rigid contact processing (see [27], [28]): LCP (linear complementarity problem), NLCP methods (non-linear complementarity problem) and others. In fact all of them implement different numeric algorithms for solving the

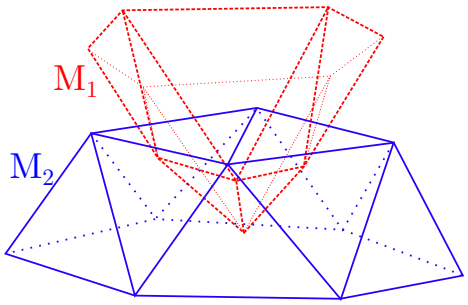

Fig. 6 Compliant contact of two triangular meshes.

same systems of equations (sometimes called complementary slackness mechanical systems, see [27]).

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 primitive shapes 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 M_1 and M_2 of the bodies in contact (elastic foundation model, EFM) [5,27]. In general, for two bodies bounded by triangular meshes M_1 and M_2 (Fig. 6), a resulting contact force and torque is a sum:

$$\boldsymbol{f}^{\text{contact}}(\boldsymbol{q}^c, \dot{\boldsymbol{q}}^c) = \frac{1}{2}\sum\nolimits_1 (\boldsymbol{f}_n + \boldsymbol{f}_\tau) + \tfrac{1}{2}\sum\nolimits_2 (\boldsymbol{f}_n + \boldsymbol{f}_\tau) \tag{6}$$

$$\boldsymbol{\tau}^{\text{contact}}(\boldsymbol{q}^c, \dot{\boldsymbol{q}}^c) = \frac{1}{2}\sum\nolimits_1 \boldsymbol{r_i} \times (\boldsymbol{f}_n + \boldsymbol{f}_\tau) + \tfrac{1}{2}\sum\nolimits_2 \boldsymbol{r_i} \times (\boldsymbol{f}_n + \boldsymbol{f}_\tau) \tag{7}$$

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 $\boldsymbol{f}_n$ and $\boldsymbol{f}_\tau$ are normal and tangent components of the elementary repelling force which are 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 $\boldsymbol{f}_n$ and $\boldsymbol{f}_\tau$ 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 correlates 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 $O(k^{-1/2})$ [27]. Thus, 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 is considered as the etalon for the deformable bodies [29], [30]. 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 module, 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 with 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 capture the contact forces (i.e. $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ in (1)) with a compliant model as provided in Eqs. (6,7): $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = (\boldsymbol{f}_1^{\text{contact}}, \boldsymbol{\tau}_1^{\text{contact}}, \ldots, \boldsymbol{f}_{N^c}^{\text{contact}}, \boldsymbol{\tau}_{N^c}^{\text{contact}})$, where N^c is the number of contacts. Here, we augment the Robotran simulator with an 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 the Robotran simulator with a compliant contact library

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;

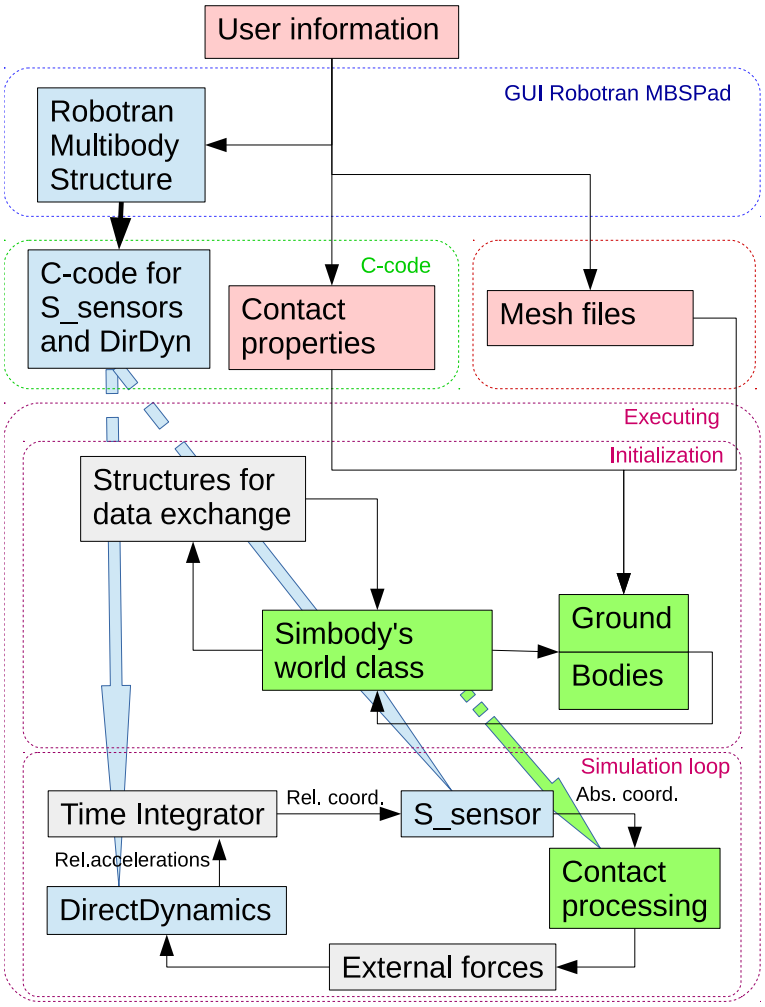

Fig. 7 Scheme of coupling between Robotran (light blue blocks) and Simbody (green blocks). User provides information about contact that are stored in red blocks. Blocks for coupling are grey.

– It provides compliant contact of two types: Hertz-model and elastic foundation model (EFM, more details can be found in [5,30]).
– It allows to use WaveFront object files encoding triangular meshes to define complex body shapes.

The scheme of coupling is shown on Fig. 7. Concretely, during the building of the multibody structure (that has to be constructed in 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 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 floating body with 6 DOF 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-DOF 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.

Figure 8 shows two representative tasks that extensively use 3D mesh-to-mesh contact: a manipulation task and a locomotion task (with a gait controller presented in [6]). Animations for these two tasks are attached as supplementary materials.

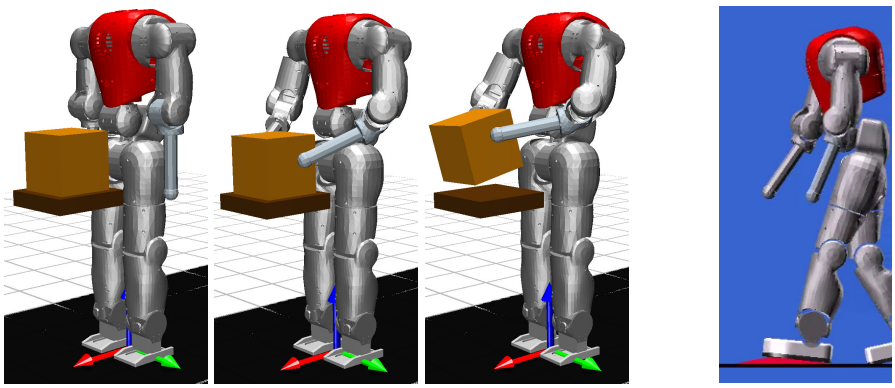

Fig. 8 On the left: Manipulation task – COMAN takes a box from a table. On the right: Locomotion task – COMAN goes over a 3D bump .

6 Manipulation task

To test the model with compliant actuators and mesh contacts, we simulated a bimanual manipulation task (see Fig. 8). Initially, the robot stands in its natural

Name	Type	N of faces	Friction Coefs [-, -, $\frac{N \cdot m}{sec}$]	Young modulus [Pa]	Dissipation [$\frac{sec}{m}$]
Box	Mesh	588	0.9, 1.1, 0.5	10^4	10
Wrisp	Primitive	-	1.05, 1.1, 0.5		
Foot	Mesh	504	1.9, 2.1, 0.5		
Ground	Primitive	-	0.9, 1.1, 0.5		
Table	Mesh	12	0.9, 1.1, 0.5		0.1

Table 2 The contact characteristics.

position. A cubic box (15 cm side length, 1.5 kg weight) lies 20 cm in front of the robot on a table, 47.5 cm above the ground. Three pairs of surfaces are initially in contact: the box with the table, and each of the two feet with the ground. The surface of the box is a mesh automatically generated by Simbody internal procedure, each wrist is a sphere primitive with radius 2 cm, while the feet and the table are defined as mesh objects by Wavefront obj files. Finally, the ground is defined as a one-half-space. The number of faces for meshes, friction coefficients (dynamic, static and viscous respectively), stiffness and dissipation coefficients are shown in the Table 2 (the details about the contact parameters' meaning are given in [30,31]) .

For the arm joints (right arm joints are from 22 to 25, left arm — from 26 to 29, see Fig. 1) the motors' inner variables $q_i^m, i = 22..25, 26..29$ are controlled: in position (i.e. tracking a trajectory). The voltages V_i (see Section 4) are calculated by PI-controllers. The arms move symmetrically, the references for q_i^m are some piece-wise linear functions of time (showed by the dashed lines in Fig. 9). Solid lines shows the simulated joint positions q_i. For all other actuated joints, zero-references are used for the inner motor variables, so that the corresponding joints are requested to avoid moving. We defined the arms' reference trajectories such that after approximately one second of simulation, two new contacts appear (between each of the wrists and the box). The grasping force and dry friction in this contact phase causes the robot to lift up the box.

The MBS equation (1) were integrated by Runge-Kutta 4th-order algorithm with a fixed time-step on a 6-core AMD 3.5 GHz processor under 64-bit Ubuntu OS. The simulation sampling frequency was taken equal to 1 kHz. A 2-sec simulation with $\Delta t = 1 \cdot 10^{-4}$ sec takes 150.6 sec (the real-time factor is 0.013). A 2-sec simulation without the calculation of the contact between the feet and the ground (i.e. considering the feet being locked on the ground) and the same Δt takes 60 sec (the real-time factor is 0.033). The largest time-step for numerically stable simulation with fixed feet is $\Delta t = 9 \cdot 10^{-4}$ sec. In this case, the simulation takes 7.1 sec (the real-time-factor is 0.28). In the following, we present the results involving five bodies with contact surfaces (i.e. both the feet unlocked), simulated with $\Delta t = 1 \cdot 10^{-4}$ sec.

At $t \approx 1.06$ sec, both wrists touch the box, and start to slowly raise it up. In the left upper corner of Fig. 10, the sum of all contact forces, acting on the box in the z-direction is shown. During the first 0.2 sec, we see the relaxation of the oscillation that occurs due to the compliance of the contact between the table and the box; after that time and till $t \approx 1.06$ sec, the gravitational force is

compensated by the contact force, being equal to 14.715 N. After the wrists touch the box, two vertical dry friction forces between the wrists and the box occur, and the box is raised upward. The pressing contact force that acts between the box and each wrist is shown on the right upper panel. The contact force between the foot and the ground is shown in the left lower panel. We can also see the relaxation oscillation till 0.2 sec, and some perturbations after the robot takes the box. Due to the rotation of the box and the non-zero contact patch between each wrist and the box, the dry friction torque in the y-direction appears (in the lower right panel), stopping the rotation of the box.

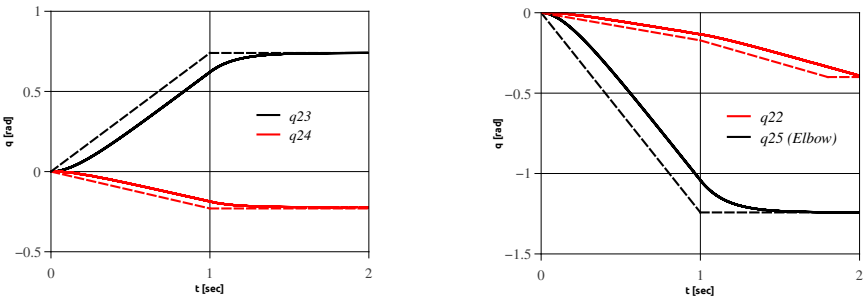

Fig. 9 References for motor's variables (dashed) and simulated curves for right arm joints (solid).

To illustrate the compliance in the elbow actuator, we also performed the same simulation for much smaller K_S coefficients (see Eq. (3)). (For the previous simulation $K_S = 1.2 \cdot 10^2$ N·m, for this one $K_S = 1.2 \cdot 10^{-3}$ N·m.) The left panel of Fig. 11 shows the grasping force for both simulations. Naturally, stiffer springs cause higher grasping forces than the softer springs. The right panel shows the time-curves for q_{25}^m and q_{25} for softer spring: we see that after the wrist touches the box ($t > 1.2$ sec), the inner spring starts to deform, resulting in a decrease of the grasping force.

7 Squatting task

In order to validate the proposed simulator, the WALK-MAN humanoid robot [32] was simulated with it (Fig. 12, top left). The WALK-MAN is an adult-size humanoid robot that has the same MBS structure as COMAN (except for the head, that is attached to its torso by revolute actuated joint), but with different mass and inertia and more powerful compliant actuators. It can thus be modeled with the proposed simulator after tuning some parameters. The data predicted by the model were compared with the one measured from the sensors of the real robot.

A squatting task was selected. As the simulator and the robot platform are both interfaced with the YARP middleware [11], the exact same controller can be run on both. This experiment was initially performed to highlight the ease to switch between a simulator and a real robot [12]. In this task, the controller follows

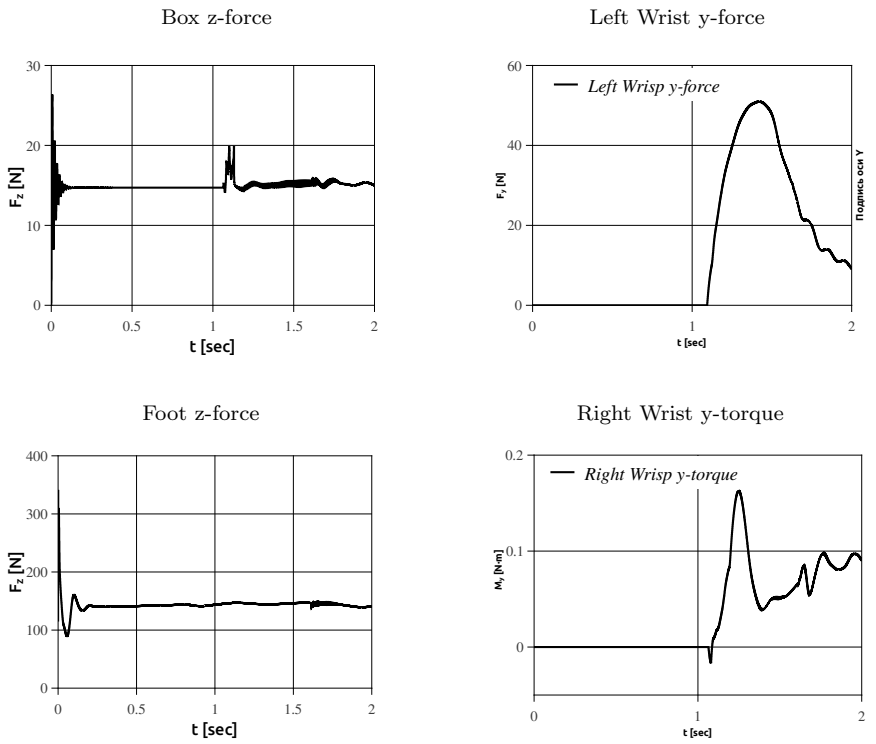

Fig. 10 The contact forces; z is the vertical axis, y the horizontal one in the lateral plane.

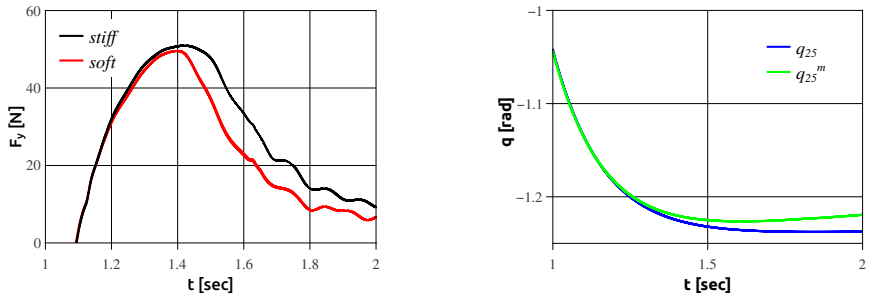

Fig. 11 The softer springs in elbow joints give smaller pressing contact forces between the wrists and the box (left graph) and larger difference between joint angle q_{25} (blue line) and inner motor variable q_{25}^m (green line).

a joints trajectory where the robot squats down by 30 cm and returns to its initial posture. The joints position, torque and the 6 DOF force torque sensor of the feet were logged during 4 cycles of 7 seconds each, see Fig. 12.

The top right panel represents the evolution of the position tracking of the Hip joint (other joints show similar comparison between the experimental and simulated results). It appears that the joints controller properly tracks the desired joints trajectory, both in simulator and with the real robot. However, experimental

time-curve for the joint torque in knee joints is more oscillatory and the average torque profile is slightly higher in the experiment. We suggest two main reasons of this difference: first, the joints torque sensors of the real platform were suffering poor measurement due to backlash in its inner mechanism. This has been improved later on. Second, the weight and inertia of the actual robot's segments is a bit bigger because of unmodelled shock absorbing plastic covering the robot body and electronic boards. The bottom left panel shows the measurement of the vertical force in the foot sensor. During the first second of the task, the simulated data displays fast damping oscillations. This is due to the compliant contact model, compare with left upper panel of Fig. 10, and an initial error of the feet-ground propagation in the simulator. Then the simulated normal force oscillates due to the robot's center of mass acceleration. The experimental curve is also more oscillatory, probably due to sensor imprecisions and poor fitting of the compliant model parameters. However, the simulation results are satisfactory matching the experimental data, validating the use of the proposed simulator for supporting the design and controller tuning of advanced robots.

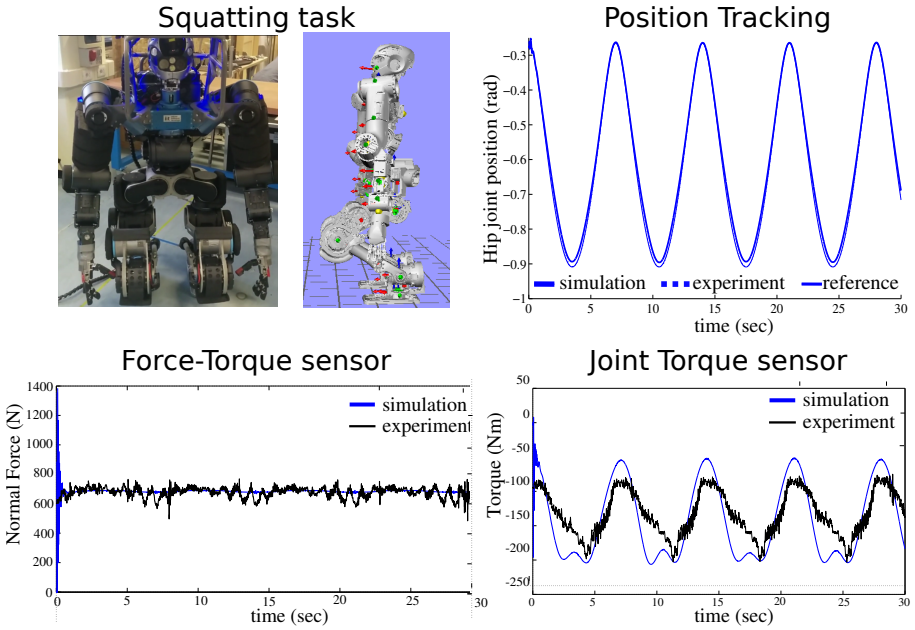

Fig. 12 Squatting task performed by a simulated and real WALK-MAN robot (from [12]).

8 Conclusions

In the frame of this work, a simulator for humanoid robots was developed and the following questions were studied. We investigated, how the inner MBS representation in the direct dynamics module influences the simulator speed and

accuracy. In particular, we showed that for the chain MBS systems, ODE, which adopts the absolute coordinates with algebraic constraints, is less accurate than Robotran with relative coordinates. The code of the symbolically simplified MBS equations takes less time to be calculated than the automatically generated one. Next, a full electromechanical actuators model was built. It requires the simultaneous time-integration of MBS equations and equations governing the inner actuator dynamics. Finally, our reasoning for compliant algorithms of contact processing were reported. Concretely, this was achieved through the coupling of the Simbody open-source contact library and our MBS simulator.

The performance of the simulator was quantified via two examples. First, we analysed the simulation of bimanual manipulation regarding the time evolution of joint angles, contact forces, and inner motors variables. Second, the reliability of the simulator was investigated through by the comparison with a squatting experiment, performed with a simulated and actual WALK-MAN robot.

The simulator offers to simulate at least the COMAN and WALK-MAN robots, and can be adopted for other humanoid robots with custom multi-physics actuators in joints, e.g. compliant, hydraulic, or electrical. The simulator code is publicly available via an open-source platform [33].

Acknowledgements This work is supported by the European Communitys Seventh Framework Programme (FP7/2007-2013) under Grant 611832 (WALK-MAN), by the foundation "Wallonie-Brussel International" (post-doc scholarship awarded to AZ), and by the Belgian F.R.S.-FNRS (Crédit aux Chercheurs #6809010 awarded to RR, Aspirant #16744574 awarded to NVdN).

References

1. Tsagarakis, N.G., Morfey, S., Cerda,G.M., Zhibin,L., Caldwell,D.G., Compliant humanoid coman: Optimal joint stiffness tuning for modal frequency control. IEEE Robotics and Automation (ICRA), 673-678 (2013)
2. Tsagarakis, N.G., Morfey, S., Dallali, H., Medrano-Cerda, G.A., Caldwell, D.G.: An asymmetric compliant antagonistic joint design for high performance mobility. IEEE Intelligent Robots and Systems (IROS), 5512–5517 (2013)
3. Docquier,N., Poncelet,A., Fisette,P.: ROBOTRAN: a powerful symbolic gnerator of multibody models. Mechanical Sciences 4(1), 199–219 (2013)
4. Dallali, H., Mosadeghzad, M., Medrano-Cerda, G., Docquier, N., Kormushev, P., Tsagarakis, N., Li, Zh., Caldwell, D.: Development of a dynamic simulator for a compliant humanoid robot based on a symbolic multibody approach. IEEE International Conference on Mechatronics (ICM), 598–603 (2013)
5. Sherman, M.A., Seth, A., Delp, S.L.: Simbody: multibody dynamics for biomedical research. Procedia IUTAM, 2, 241–261 (2011)
6. Van der Noot, N., Colasanto, L., Barrea, A., Van den Kieboom, J., Ronsse, R., Ijspeert, A.J.: Experimental validation of a bio-inspired controller for dynamic walking with a humanoid robot. Intelligent Robots and Systems (IROS), IEEE, 393–400 (2015)
7. Dallali, H., Mosadeghzad, M., Medrano-Cerda, G., Vo-Gia Loc, Tsagarakis, N., Caldwell, D., Gesino, M: Designing a High Performance Humanoid Robot Based on Dynamic Simulation. 2013 European Modelling Symposium (EMS), IEEE, 359–364 (2013)
8. Smith, R.: Open Dynamics Engine (ODE). http://www.ode.org/. Accessed 21 June 2016
9. Mistry, M., Schaal, S., Yamane, K.: Inertial parameter estimation of floating base humanoid systems using partial force sensing. 9th IEEE-RAS International Conference on Humanoid Robots, IEEE, 492–497 (2009)
10. Sentis,L.: Synthesis and Control of Whole-Body Behaviors in Humanoid Systems. PhD Thesis, Stanford University (2007)

11. Fitzpatrick, P., Metta, G., Natale, L.: Towards long-lived robot genes. Robotics and Autonomous Systems 56(1), 29 – 45 (2008)
12. Habra, T., Dallali, H., Cardellino, A., Natale, L., Tsagarakis, N., Fisette, P., Ronsse, R.: Robotran-YARP Interface: A Framework for Real-Time Controller Developments Based on Multibody Dynamics Simulations. In "Multibody Dynamics: Computational Methods and Applications", Springer International Publishing, 147–164 (2016)
13. Ivaldi, S., Peters, J., Padois, V., Nori, F.: Tools for dynamics simulation of robots: a survey based on user feedback // Proceedings of 2014 IEEE-RAS International Conference on Humanoid Robots, IEEE, 842–849 (2014)
14. Boeing, A., Bräunl, Th.: Evaluation of real-time physics simulation systems. Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia, ACM, 281–288 (2007)
15. Todorov, E., Erez, T., Tassa, Y.: MuJoCo: A physics engine for model-based control, IEEE Intelligent Robots and Systems (IROS), 5026–5033, (2012)
16. Samin, J-C., Fisette, P.: Symbolic Modeling of Multibody Systems. Kluwer Academic Publishers, Dordrecht (2003)
17. Bullet Physics library. http://bulletphysics.org/. Accessed 21 June 2016
18. Van den Bergen, G., Gregorius, D.: Game physics pearls. Natick, Mass.: A.K. Peters (2010)
19. Manual - ODE Wiki. http://ode-wiki.org/wiki/index.php?title=Manual/. Accessed 21 June 2016
20. Maxima, a Computer Algebra System. http://maxima.sourceforge.net/. Accessed 21 June 2016
21. Khalil, W., Vijayalingam, A., Khomutenko, B., Mukhanov, I., Lemoine, Ph., Ecorchard, G.: OpenSYMORO: An open-source software package for symbolic modelling of robots. IEEE/ASME International Conference on Advanced Intelligent Mechatronics Proceedings, IEEE, 1206–1211 (2014)
22. MapleSim - High Performance Physical Modeling and Simulation - Technical Computing Software. http://www.maplesoft.com/products/maplesim/index1.aspx Accessed 21 June 2016
23. Spong, M.W.: Modeling and Control of Elastic Joint Robots, J. Dyn. Sys., Meas., Control 109(4), 310–318 (1987)
24. Siciliano, B., Khatib, O. (Eds.): Springer Handbook of Robotics. Springer-Verlag Berlin Heidelberg (2008)
25. Blau, P. J.: Friction science and technology: from concepts to applications. CRC Press (2009)
26. Chatterjee, A., Ruina, A.: A new algebraic rigid body collision law based on impulse space considerations, ASME J.Appl.Mech., 65(4), 939–951 (1998)
27. Brogliato, B., ten Dam, A., Paoli, L., Génot, F., Abadie, M.: Numerical simulation of finite dimensional multibody nonsmooth mechanical systems. Applied Mechanics Reviews, 55(2), 107–150 (2002)
28. Drumwright, E., Shell, D.A.: An evaluation of methods for modeling contact in multibody simulation. Robotics and Automation (ICRA), 1695–1701 (2011)
29. Hippmann, G.: An algorithm for compliant contact between complexly shaped bodies. Multibody System Dynamics, 12(4), 345–362 (2004)
30. Pérez-González, A., Fenollosa-Esteve, C., Sancho-Bru, J.L., Sánchez-Marín, F.T., Vergara, M., Rodríguez-Cervantes, P.J.: A modified elastic foundation contact model for application in 3D models of the prosthetic knee, Medical Engineering & Physics, 30(3), 387–398 (2008)
31. Simbody: Multibody Physics API https://simtk.org/projects/simbody Accessed 21 June 2016
32. Negrello, F., Garabini, M., Catalano, M.G., Kryczka, P., Choi, W., Caldwell, D., Bicchi, A., Tsagarakis, N.G.: WALK-MAN Humanoid lower body design optimization for enhanced physical performance. IEEE International Conference on Robotics and Automation (ICRA), 1817–1824 (2016)
33. Simulators of the COMAN and WALK-MAN humanoid robots
https://gitlab.robotran.be/walkman/coman_robotran/,
https://gitlab.robotran.be/walkman/walkman_robotran/ Accessed 21 June 2016