Whole-Body Motion Synthesis with LQP-Based Controller - Application to iCub

Joseph Salini, Sébastien Barthélémy, Philippe Bidaud, Vincent Padois

Whole-Body Motion Synthesis with LQP-based Controller - Application to iCub

Joseph Salini, Sébastien Barthélemy, Philippe Bidaud and Vincent Padois

Salini, Barthélemy, Bidaud and Padois Institut des Systèmes Intelligents et de Robotique
Université Pierre et Marie / CNRS UMR 7222 Pyramide T55
4, Place Jussieu 75252 Paris Cedex 05 - France.
{salini, barthelemy, bidaud, padois}@isir.upmc.fr

Summary. This paper deals with the dynamic control of humanoid robots interacting with their environment, and more specifically the behavioral synthesis for dynamic tasks. The particular problem that is considered here is the sequencing of elementary activities subjected to physical constraints, both internal as torque limits and external as contacts, within the framework of posture/tasks coordination. For that we propose to convert the set of tasks into weighted quadratic functions and to minimize their cost with a Linear Quadratic Program. The combination of elementary tasks leads to complex actions, and the continuous evolution of the weights ensures smooth transitions over time, as it is shown in the results.

1 Introduction

This paper focuses on the design of a generic and efficient framework for dynamic whole-body motions. Interesting issues appear, like the postural balance control when the robot realizes numerous objectives, the management of a task hierarchy or the transition control over time. The issue addressed in this paper is to find how to build scenarii of complex dynamic activities for humanoid robots, including physical interaction with the environment and ensuring a match between the control and the dynamics of the robot.

Generally, the whole-body dynamic control is solved using linear algebra methods as pseudo-inverses and projectors which give analytic solutions. In computer graphics, these tools give rise to uses for generation of physical simulations [2], and in robotics, some authors as in [11, 10] have performed the control of humanoid robots with these methods. But the integration of physical constraints in the equations is not easy, especially inequality constraints.

To overcome these difficulties and provide a more natural way to take into account the constraints, some researchers propose the use of optimization programs, and especially Linear Quadratic Programs (LQP) which optimize a quadratic cost function subject to linear constraints. The work presented in [4] shows how to control humanoids and find a robust balance behavior with these particular programs. Furthermore, [1, 13] realize the animation of humanoids with a LQP.

Concerning the task hierarchy, numerous works have been done by the past and the problem is still complicated. [11, 12, 9] propose general frameworks for managing several tasks on redundant systems and return analytic solutions. Using LQP, [1] propose to use a weighted-sum objective as cost function which makes a trade-off between the different objectives, whereas [7] describes the hierarchy as a cascade of LQP with no interference between lower level tasks and upper ones. However, continuity in the transitions is not guaranteed during evolution of the hierarchy.

Here, the authors use a quadratic function to regroup the elementary tasks related to the functional elements of the robot which control a single degree of freedom (a joint) or several degrees of freedom simultaneously (a frame). The whole-body activity is tuned by weights related to the tasks which evolve in time, hence it is shown that relatively sophisticated scenarii can be designed while ensuring torque continuity and consequently a good control.

First, this paper exposes the control of humanoid robots with a LQP and explains the construction of the constraints and the cost function. Second, it addresses the issue of the task hierarchy, their advantages and drawbacks. Third, it shows the application in simulation of the method described in this paper to the humanoid robot iCub [8] shown in Fig. 1.

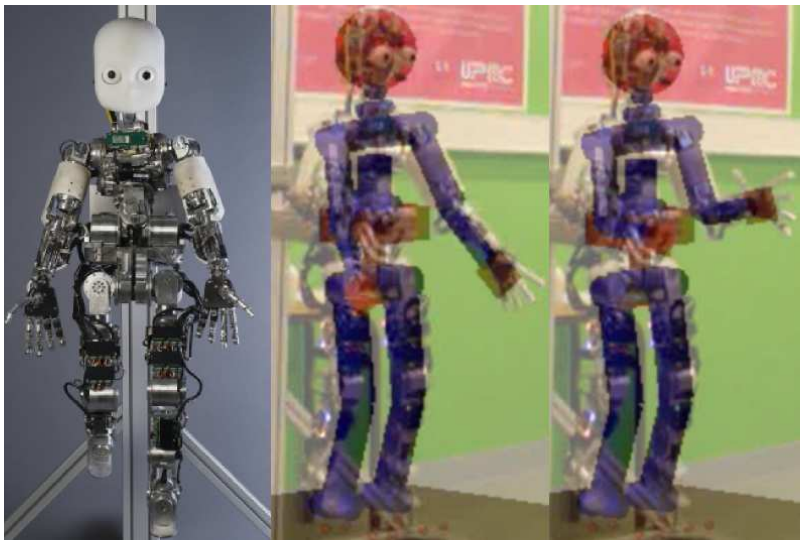

Fig. 1. Left: the real robot iCub. Middle and Right: Superimposition of the real and virtual iCub, performing the same simulation.

2 Definition of constraints

A humanoid robot is a mechanical system composed of several rigid bodies, connected to each other in a tree structure by joints. Each active joint has a limited motion range and is equipped with an actuator that can generate a bounded torque. This kind of robot is basically an under-actuated system, with a trunk, a head, lower and upper limbs, and it interacts with its environment with physical contacts. On these systems, one goal is often to make them perform several basic tasks in a coordinated manner. In this paper, the control of the robot is done by a LQP-based dynamic controller which needs one set of linear constraints and one quadratic cost function to model the problem, and applies the solution for whole-body motion.

The set of constraints used in the LQP generally represents the physical limits of the robot, both internal and external. Some constraints can be added or removed to change the perception of the world by the controller, but they must be carefully selected in particular to deal with variations in the contact conditions.

2.1 Equations of motion

The robot dynamic behavior follows the Euler-Lagrange equations of motion:

$$\mathsf{M}\ddot{\mathbf{q}} + \mathsf{N}\dot{\mathbf{q}} = \mathbf{g} + \mathsf{S}\tau + \mathsf{J}_c^t \mathbf{f}_c \tag{1}$$

where $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathsf{M}(\mathbf{q}), \mathsf{N}(\mathbf{q}, \dot{\mathbf{q}}), \mathbf{g}(\mathbf{q}), \mathsf{S}, \tau, \mathsf{J}_c(\mathbf{q}), \mathbf{f}_c$ are respectively the generalized position, velocity and acceleration vectors, the generalized inertia matrix, the Coriolis and non-linear effects matrix, the gravity vector, the actuation matrix, the torque vector, the contact points Jacobian and finally the contact forces vector. Actuation matrix S allows each degree of freedom to be actuated or not. This equation is linearized around $(\mathbf{q}, \dot{\mathbf{q}})$ to fit the LQP, and it uses $\ddot{\mathbf{q}}, \tau, \mathbf{f}_c$ as variables.

2.2 Joint & actuation limits

As written above, each joint has a limited motion range and the related actuator can generate bounded torque. Furthermore, the linearization of Eq. 1 around $(\mathbf{q}, \dot{\mathbf{q}})$ should be ensured with bounded joint velocities. These constraints are described in the LQP as follows:

$$\tau_{min} \leq \tau \leq \tau_{max} \tag{2}$$
$$\mathbf{q}_{min} \leq \mathbf{q} + \dot{\mathbf{q}}h + \ddot{\mathbf{q}}h^2/2 \leq \mathbf{q}_{max} \tag{3}$$
$$\dot{\mathbf{q}}_{min} \leq \dot{\mathbf{q}} + \ddot{\mathbf{q}}dt \leq \dot{\mathbf{q}}_{max} \tag{4}$$

where $\tau_{(min,max)}, \mathbf{q}_{(min,max)}, \dot{\mathbf{q}}_{(min,max)}$ are torque bounds, joint position and velocity limits, h is the horizon of anticipation, and dt the sampling time. Anticipation avoids sudden braking and collisions with joint mechanical limits. The horizon of anticipation used in Eq. 3 has to be tuned to brake sooner or later.

2.3 Contact & kinematic closure

For all the humanoid bodies interacting with the environment, contacts are described as a set of punctual interactions with friction. Each contact point i has a velocity $\mathbf{v}_{ci} = \mathsf{J}_{ci}(\mathbf{q})\dot{\mathbf{q}} \in \mathbb{R}^3$, and each point develops a force denoted $\mathbf{f}_{ci} \in \mathbb{R}^3$. Four cases may happen

- the contact remains persistent, $\mathbf{v}_{ci} = \mathbf{0}$ and $\mathbf{f}_{ci}$ lies inside the Coulomb cone,
- the contact is lifting, $\mathbf{v}_{ci}.\mathbf{n} > 0$ and $\mathbf{f}_{ci} = \mathbf{0}$,
- there is no contact, $\mathbf{v}_{ci} \in \mathbb{R}^3$ and $\mathbf{f}_{ci} = \mathbf{0}$,
- the contact is sliding $\mathbf{v}_{ci} \times \mathbf{n} \neq \mathbf{0}$ and $\mathbf{f}_{ci} \neq \mathbf{0}$,

where $\mathbf{n}$ is the normal vector of contact. In order to be integrated in the LQP-based dynamic controller, Contact point constraints are expressed in terms of joint accelerations and forces constraints lying inside linearized cones. For instance, the two first cases are expressed as follows:

$$\textit{case } 1\ \mathbf{v}_{ci} = \mathbf{0}\ :\quad \mathsf{J}_{ci}\ddot{\mathbf{q}} + \dot{\mathsf{J}}_{ci}\dot{\mathbf{q}} = \mathbf{0} \tag{5}$$

$$\mathsf{C}\mathbf{f}_{ci} \leq \mathbf{0} \tag{6}$$

$$\textit{case } 2\ \mathbf{v}_{ci}.\mathbf{n} > \mathbf{0}\ :\ \mathsf{J}_{ci}\ddot{\mathbf{q}} + \dot{\mathsf{J}}_{ci}\dot{\mathbf{q}}.\mathbf{n} > \mathbf{0}. \tag{7}$$

In the same way, kinematic loop closure is described in the LQP as a constraint. Two points $i1$ and $i2$ from two kinematic chains are linked to form the kinematic loop, so $\mathbf{v}_{i1} = \mathbf{v}_{i2} \in \mathbb{R}^3$ is expressed in the controller as follows:

$$(\mathsf{J}_{i1} - \mathsf{J}_{i2})\ddot{\mathbf{q}} + (\dot{\mathsf{J}}_{i1} - \dot{\mathsf{J}}_{i2})\dot{\mathbf{q}} = \mathbf{0}. \tag{8}$$

3 Definition of cost function

The quadratic cost function of the LQP represents the objective that the controller may satisfy. Many objectives may lead to conflicting situations, so the selection of a strategy to cope with this issue is discussed in Sec. 4. Here, one objective has to be completed, and its related task is the set composed of the controlled part of the robot, the goal and the way to achieve it.

3.1 Task definition

Each task i is associated to a functional element of the robot. It can be a joint, a frame linked to the robot bodies, the Center of Mass (CoM), etc. One of the main concern in the task definition is to find the appropriate Jacobian matrix J_i and its derivative (with regard to time) $\dot{\mathsf{J}}_i$ related to the task. Hence, the velocity $\mathbf{v}_i$ and acceleration $\dot{\mathbf{v}}_i$ of the task i can be computed as well:

$$\mathbf{v}_i = \mathsf{J}_i\dot{\mathbf{q}} \tag{9}$$

$$\dot{\mathbf{v}}_i = \mathsf{J}_i\ddot{\mathbf{q}} + \dot{\mathsf{J}}_i\dot{\mathbf{q}}. \tag{10}$$

Since the whole-body motion control is dynamic, at each time step a desired acceleration $\dot{\mathbf{v}}_i^{des}$ is computed beforehand, and the cost function of the task becomes $(\delta_i)^2 = \|\dot{\mathbf{v}}_i^{des} - \dot{\mathbf{v}}_i\|^2$.

3.2 Task controller

A task controller computes the desired acceleration mentioned above. A simple one is the tracking of a predefined trajectory. If for the task i the desired pose, velocities and acceleration of reference over the time are respectively $\mathbf{pos}_i^{ref}$, $\mathbf{v}_i^{ref}$ and $\dot{\mathbf{v}}_i^{ref}$, its controller gives the desired acceleration as follows:

$$\dot{\mathbf{v}}_i^{des} = \dot{\mathbf{v}}_i^{ref} + \mathsf{K}_p\mathbf{err}_p + \mathsf{K}_d\mathbf{err}_d \tag{11}$$

where $\mathsf{K}_p, \mathsf{K}_d$ are respectively the stiffness and the damping of the trajectory tracking, and $\mathbf{err}_p, \mathbf{err}_d$ are the proportional and derivative errors. $\mathbf{err}_d$ is the difference between the reference and actual velocities $\mathbf{err}_d = \mathbf{v}_i^{ref} - \mathbf{v}_i$, and $\mathbf{err}_p$ is the difference between the reference and actual pose. Its computation relies on the controlled element and the nature of the movement. For example a joint with linear configuration (like a hinge), or the relative position of a point give a simple error $\mathbf{err}_p = \mathbf{pos}_i^{ref} - \mathbf{pos}_i$. If it is only a translation, the error is computed as an Euclidean distance, but if it incorporates a rotation it becomes more complicated. The authors use the imaginary part of quaternions to compute the proportional rotational error.

Another interesting way to achieve an objective is to use a preview control. It is exploited here to control the center of mass of the robot to generate a gait pattern using a preview control of the zero-moment point (ZMP) [15]. Indeed, the predictive controller returns an optimal path according to a given criterion along a predefined horizon. For example, the use of the inverted pendulum as the approximation of a humanoid explained in [6, 14] allows to predict the trajectory of the center of mass function to a control vector. The first component of this vector is used as the desired acceleration, and the solution is updated at each time step. This preview approach can be extended to other tasks such as compliant motions.

3.3 LQP-based controller

At each time step, the dynamic LQP-based controller solves the following problem:

$$\min_{(\ddot{\mathbf{q}},\tau,\mathbf{f}_c)} \quad ((\delta_i)^2 + \alpha_\tau(\tau)^2 + \alpha_{\mathbf{f}_c}(\mathbf{f}_c)^2) \tag{12}$$

$$s.t.: \qquad Eqs.\ 1-8 \tag{13}$$

where $\alpha_\tau, \alpha_{\mathbf{f}_c}$ are weights. The given torque vector is applied to the robot, and due to the minimization of its norm, the controller generates smooth motions. This method illustrated in Fig. 2 can be compared to optimal quadratic control, where there is a trade-off between the state-feedback and the control vector of the robot. Here the state of the robot is $\ddot{\mathbf{q}}$, the control vector regroups $\tau, \mathbf{f}_c$, and the ratio is given by α_τ and $\alpha_{\mathbf{f}_c}$.

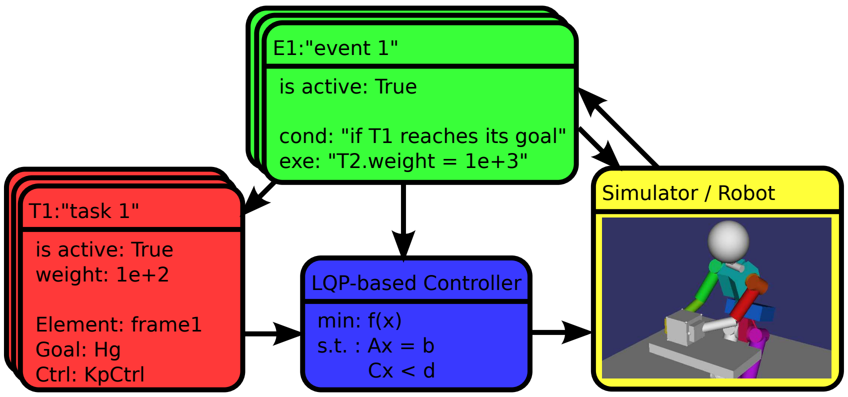

Fig. 2. Relationships between the different parts to perform the control of the humanoid robot.

4 Importance, hierarchy & transition

One of the main difficulties for the implementation of coordination between several tasks on a system with high redundancy is the importance between the various tasks. Indeed, a strict hierarchy should be used when higher priority tasks need to be performed with no error, but a simple trade-off between weighted objectives may offer more flexibility.

4.1 Weighting of tasks

The easiest way to perform several tasks into the LQP-based controller is to optimize a sum of weighted costs which replaces Eq. 12 by $(\Sigma_i(\alpha_i(\delta_i)^2) + \alpha_\tau(\tau)^2 + \alpha_{\mathbf{f}_c}(\mathbf{f}_c)^2)$ where α_i is the weight of task i, and $\Sigma_i\alpha_i = 1$ to keep the same ratio with the command described above. It is a good way to quickly realize several tasks at the same time with one LQP at each time step. As a task j with a bigger weight $\alpha_j \gg \alpha_i \forall i$ will be roughly fulfilled, this method allows to set up a sort of smooth hierarchy between tasks.

4.2 Task hierarchy

A way to perform strict hierarchy is to solve many LQP one after another. Assume that tasks have been sorted by importance: the first level realizes task 1, the second level realizes tasks 2 and 3 according to their weighting coefficients, the third level realizes task 4, and so on. When one level is solved, the related solution is inserted as a new constraints in the next level. For example, assume the first level of the hierarchy is solved by the LQP in Sec. 3.3, hence the optimal solution of task 1 is δ_1^*, the related constraint is $\delta_1 = \delta_1^*$, and the second level becomes:

$$\begin{array}{cc} \min\limits_{(\ddot{\mathbf{q}},\tau,\mathbf{f}_c)} & (\alpha_2(\delta_2)^2 + \alpha_3(\delta_3)^2 + \alpha_\tau(\tau)^2 + \alpha_{\mathbf{f}_c}(\mathbf{f}_c)^2) \\ s.t.: & Eqs.\ 1-8 \\ & \delta_1 = \delta_1^* \end{array}$$

which gives solutions for tasks 2 and 3 and provides new constraints to the next level.

4.3 Tasks transition over time

When performing the synthesis of complex tasks, the set of objectives, their relative importances, their hierarchy and their goals may change during time. A strategy must be set to ensure stable transitions and avoid peaks in the control.

For example, the humanoid robot has to grab a object on a table, it must bend to reach its objective, then it gets up to put the object elsewhere. The set of tasks is:

1. keep robot balance,
2. keep the back upright,
3. grab the object,
4. displace the object.

The balancing task has the most importance (or the highest priority) throughout the example, but grabbing the object is more important than keeping the back upright. When the object is caught, keeping the back upright becomes more important, and the object may finally be displaced. If a hierarchy is set as in Sec. 4.2, the task which keeps the back upright is not continuous because one of its constraint related to higher tasks suddenly vanishes and affects the control. If the importance between coefficients is considered, the transition is done by sliding smoothly their values to their new relative importances, so the trade-offs change continuously over time. This example is described and validated in Sec. 5.2.

5 Experiments

The validation of the method is done through some experiments with the model of the iCub robot. The simulations has been carried out with arboris-python [3], a simulator developed in the ISIR-UPMC laboratory. It aims to perform dynamic simulations quickly and easily with tree-shaped structures under some physical constraints as frictional contacts, joint limits and ball joints. The LQP is solved with cvxopt [5], a convex optimization python library. The sampling time is set to $dt = 0.02\ s$ for all simulations.

5.1 Joint limit avoidance

Here, the robot reaches an objective in front of it with its left hand. The motion is constrained by joint limits, so the controller needs some anticipation to avoid hard braking. Figure 3 shows the displacement of the left arm.

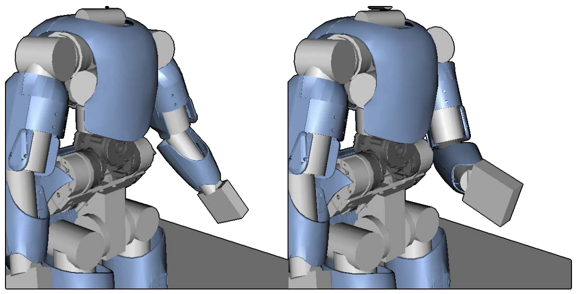

Fig. 3. The initial (left) and final (right) positions of the left hand.

During this simulation, the pitch and roll joints of the shoulder approach their limits. The method presented in Eq. 3 bounds the joint acceleration on a predefined horizon. This approach reduces the sharp evolutions of the torques as shown in Fig. 4, where some peaks appears before $t = 0.5\ s$ due to a short horizon of prediction.

5.2 Box displacement

This experiment is described above. The robot grabs a 3 kg box and displaces it from one place to another, as shown in Fig. 5.

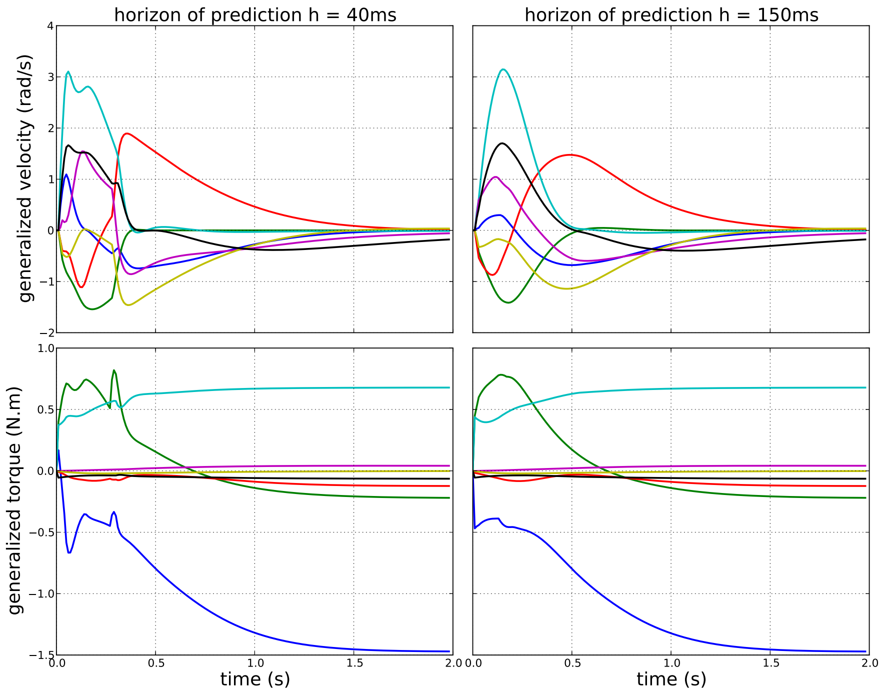

Fig. 4. Left arm behavior when subject to joint limit constraints for different horizons of prediction h.

Here, torques and velocities of the torso joints have been recorded and the results are given in Fig. 6. As the box is relatively heavy — the robot weighs about 20 kg — it cannot be neglected in the dynamic behavior and the postural balance of the humanoid. At $t = 1.5$ s and $t = 4.8$ s, there is two discontinuities in torques applied to the robot, when it lifts and drops the box. Elsewhere, the coefficient of the task which keeps the back upright evolves smoothly to make the grasping task easier, and no sudden change appears during the transition periods.

5.3 Gait pattern

As explain in Sec. 3.2, the approximation of the robot with an inverted pendulum gives an approximated ZMP, which can be controlled with this method. The trajectory of this point is defined a priori and the tasks which realize the steps are synchronized with its path.

Here, the robot gets up from a seated position, initiates a gait pattern, walks 0.5 m and stops. Figure 7 shows the snapshots of this sequence. The duration of one foot step is about 0.8 s and the entire simulation is done in 8 s. To perform this simulation, all joint limits are actives and the horizon of prediction is set to $h = 0.15$ s. To prevent the knees to bend in the wrong way, their limits have been reduced and their positions remain in an admissible set. Furthermore, the feet get closer during the walk, hence the CoM trajectory has less amplitude

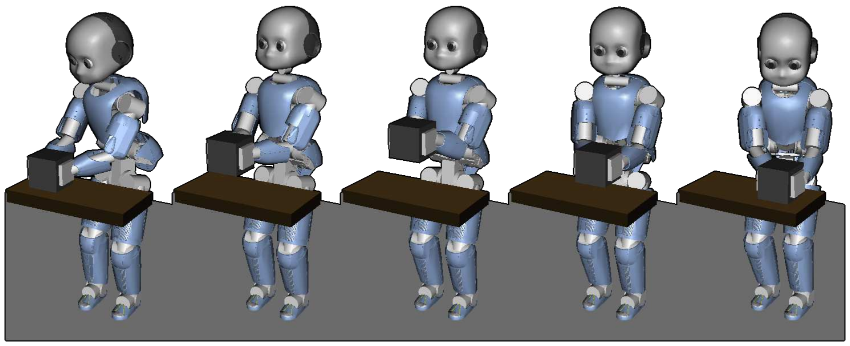

Fig. 5. Sequence of the robot grabbing the box.

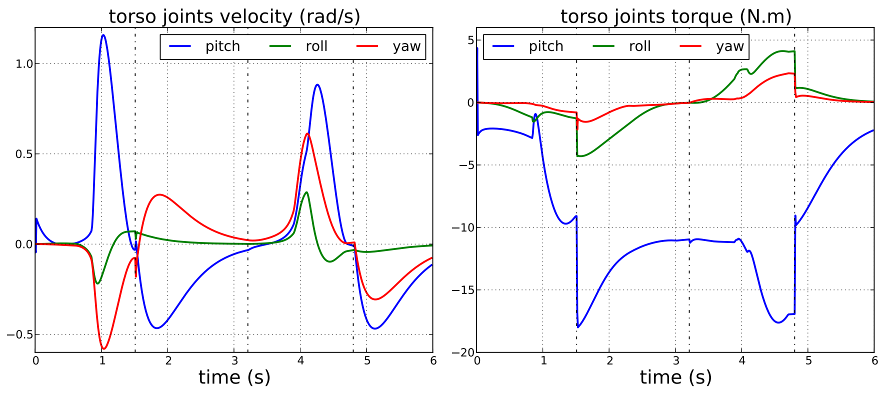

Fig. 6. Torso and velocity joints evolution over the time.

which minimizes the swing of the upper body. The trajectories of the CoM and ZMP from the initiation of the gait pattern to the end are shown in Fig. 8.

6 Conclusions

This controller allows to perform the synthesis of several dynamic tasks on a humanoid robot while interacting with its environment. The robot is subjected to both internal and external constraints, which are described as linear constraints on a Linear Quadratic Program. The tasks are described as a quadratic weighted-sum objective, and the solution of the problem is a general trade-off. The smooth evolution of the weights over the time prevents sharp evolutions of torques applied to the robot. In the future, the authors would like to take into account joint acceleration limits in the solution of the problem. It implies new constraints and

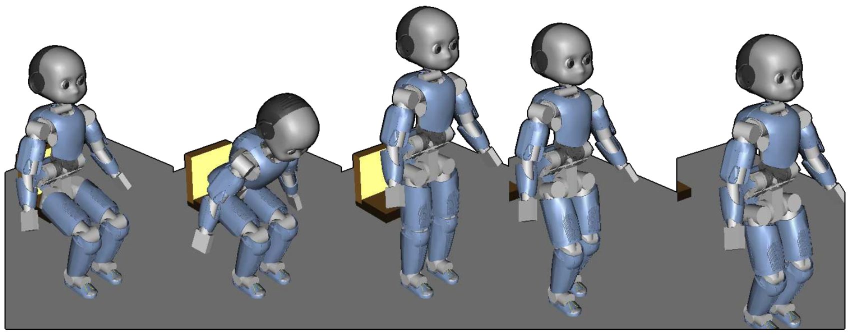

Fig. 7. Sequence of the walk.

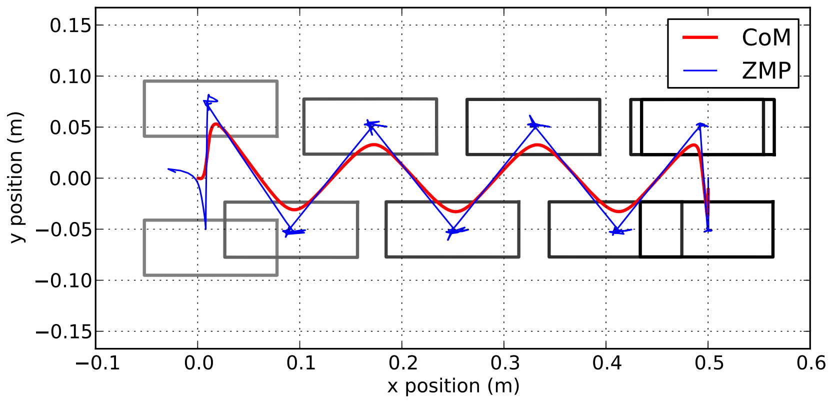

Fig. 8. CoM and ZMP trajectories, with the footprints.

some checks should be done to ensure that they are not overabundant with the joint position and velocity limits.

References

1. Abe, Y., da Silva, M., Popovic, J.: Multiobjective Control with Frictional Contacts. In: Symposium on Computer Animation (SCA) (2007)
2. Baerlocher, P., Boulic, R.: An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. Visual Computer **20**(6), 402–417 (2004)
3. Barthélemy, S., Salini, J., Micaelli, A.: Arboris-python. https://github.com/salini/arboris-pyhton

4. Collette, C., Micaelli, A., Andriot, C., Lemerle, P.: Dynamic Balance Control of Humanoids for Multiple Grasps and non Coplanar Frictional Contacts. In: Humanoids'07 (2007)
5. Dahl, J., Vandenberghe, L.: cvxopt - python software for convex optimization. http://abel.ee.ucla.edu/cvxopt/
6. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. Proceedings of the 2003 IEEE International Conference on Robotics and Automation Taipei, Taiwan (2003)
7. Kanoun, O.: Contribution à la planification de mouvement pour robots humanoïdes. Ph.D. thesis, Université Toulouse III (2009)
8. Metta, G., Sandini, G., Vernon, D., Natale, L., Nori, F.: The iCub humanoid robot: an open platform for research in embodied cognition. In: Permis: performance metrics for intelligent systems workshop. Washington DC, USA (2008)
9. Padois, V.: Enchaînements dynamiques de tâches pour des manipulateurs mobiles à roues. Ph.D. thesis, Institut National Polytechnique, Toulouse, France (2005)
10. Park, J.: Control strategies for robots in contact. Ph.D. thesis, Stanford University (2006)
11. Sentis, L.: Synthesis and control of whole-body behaviors in humanoid systems. Ph.D. thesis, Stanford University (2007)
12. Siciliano, B., J-J., S.: A general framework for managing multiple tasks in highly redundant robotic systems. In: ICAR'91, vol. 2, pp. 1211–1215 (1991)
13. M. da Silva Y. Abe, J.P.: Simulation of human motion data using short-horizon model-predictive control. In: Eurographics (2008)
14. Wieber, P.B.: Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In: IEEE-RAS International Conference on Humanoid Robots. Genova, Italy (2006)
15. Sardain, Ph. and Bessonnet, G.: Forces Acting on a Biped Robot. Center of Pressure - Zero Moment Point. In: IEEE Transactions on Systems, Man, and Cybernetics, Part A. vol. 34 (5), pp. 630–637 (2004)