LQP controller design for generic whole body motion

Joseph Salini, Sébastien Barthélemy, Philippe Bidaud

Institut des Systèmes Intelligents et de Robotique
Université Pierre et Marie / CNRS UMR 7222
Pyramide T55
4, Place Jussieu 75252 Paris Cedex 05 - France
email: salini@robot.jussieu.fr, barthelemy@robot.jussieu.fr, bidaud@robot.jussieu.fr

This paper deals with robotic control through optimization tools, as Linear Quadratic Program (LQP). This method allows whole-body control, takes account contacts between the robot and the environment, and achieves tasks simultaneously. Each task can have a priority level, so hierarchy is possible. The method is applied on a virtual robot Icub, designed from the real one.

Keywords: Whole-body control; environment interaction; LQP; Hierarchy

1. Introduction

We considers the problem of task/posture coordination of humanoid robots interacting physically with their environment. The focus of the paper is on the creation of an efficient framework for solving the underlying control problem in a generic form. Controlling the postural balance when the robot realizes a complex task is a very challenging problem which has received a limited number of contributions until now. A nice introduction to this problematic can be found in Wieber[9] [10] . The particular problem of gait planning for full-body humanoid robots is treated in[2] and the control of physical interactions in Park[6]). Beyond, Luis Sentis[8] approaches the problem of whole-body motion of humanoids as a hierarchy of tasks, each task of lower priority being applied in the null space of higher priority task. One of the major drawbacks in this method is that it does not consider explicitly contact constraints. An alternative mean using Linear Quadratic Program (LQP) have been investigated by Barthélemy,[1] Da Silva[4] and Abe.[11]

In this paper, models, equations and LQP will be introduced in the first section. In the second section, the LQP will be written explicitly with its associated constraints and the function to optimize, and controls will be

detailed to drive the robot properly. In the third section, authors will apply these methods on a virtual robot (designed like the real one, Icub[5]). Finally, conclusions and perspectives for further works are given on the last section.

2. Modelling

2.1. *Whole Body Dynamics*

The issue in this paper is to control the whole body motion of a humanoid interacting with the environment. It has to achieve tasks with its limbs while conserving postural equilibrium. A virtual Icub robot, with about the same mechanical properties than the real one, has been used in this paper. It has 38 degrees of freedom (dof): 6 dof are unactuated, and are used to locate its trunk in space, and the 32 dof left are its joints dof, which are actuated. The robot is considered as a multibody system, modelled with rigid bodies, and apply bounded torque on each joint. It follows Euler-Lagrange motion equations.

$$M(q)\ddot{q} + N(q,\dot{q})\dot{q} = G(q) + \tau + J_c(q)^t f_c \tag{1}$$

$$\tau_{min} \leq \tau \leq \tau_{max} \tag{2}$$

$M, N, \ddot{q}, \dot{q}, G, \tau, J_c, f_c$ are respectively the generalized mass matrix, the Coriolis and non-linear effects matrix, the generalized acceleration and velocity vectors, the gravity vector, the contact points jacobian and finally the contact forces vector. Eq. (1) represents the dynamic equations, and Eq. (2) represents the joint torque boundaries.

2.2. *Contact Description with the Environment*

Contact needs to be explained to model properly the robot when interacting with the environment. The robot is under-actuated, and it uses the contact forces to achieve tasks. The velocity of the contact point is given by $\dot{x}_c = J_c(q)\dot{q}$. It is assumed that sliding may not occur at each contact point. For each contact point, there are two cases left: the point does not move, or it takes off. In the first case the velocity is null $\dot{x}_c = 0$, and by derivation we find Eq. (3), where H_c is the hessian matrix of contact (derivative from the jacobian). Furthermore, according to Coulomb's law, the force must stay in the friction cone, which avoids sliding. This cone approximated by a linear cone leads directly to Eq. (4). In the second case, the contact point takes off, hence the velocity along the contact normal n is greater than 0. The minimal constraint is to set the reaction forces negative, as written in

Eq. (5). The two cases for the contact point i are described as follow:

$$case\ 1:\ \dot{x}_{ci} = 0$$

$$J_{ci}(q)\ddot{q} + H_{ci}(q,\dot{q})\dot{q} = 0 \tag{3}$$

$$Cf_{ci} \leq 0 \tag{4}$$

$$case\ 2:\ \dot{x}_{ci}.n > 0$$

$$f_{ci} \leq 0 \tag{5}$$

2.3. *Control Law*

The robot consists of several bodies, each body being located in space by a body frame. This frame describes the position and attitude of the body from a reference frame. In order to achieve tasks with the robot, we should select some frames we want to control, which are linked to robot bodies. These frames will be controlled independently, and next will be integrated in the whole body motion. A frame is defined in relation to an other frame by a position and an attitude. Let's assume that a reference frame linked to the world is set. Then, all other frames are located from this one during control. The position is defined by a 3x1 vector p, and the attitude is defined by a quaternion Q. Yuan gives a good introduction about quaternion control in.[12] The frame will be controlled in acceleration, and quaternion is used to control in rotation. Two ways are explored now.

First, the frame can follow an acceleration trajectory defined *a priori*. From the pose trajectory, the derivation gives the velocity trajectory, and other derivation gives the acceleration trajectory.

The second way is a proportional derivative control law. One chooses the proportional gain Kp, which represents the stiffness of the movement, and the derivative gain Kd, which represents the damping. Let's assume frame I has to reach the goal pose defined by a desired position p_I^{des} and a quaternion Q_I^{des} which represents the desired attitude. The pose error x_I^{err} is the concatenation of the position error $\delta p = p_I^{des} - p$ and the attitude error δQ. The frame twist is defined by $\dot{x}_I = J_I\dot{q}$ where J_I is the jacobian of frame I, and will be used in damping. Notice that $\dot{x}_I$ concatenates translational and rotational velocity. The desired acceleration $\ddot{x}_I^{des}$ is finally computed in Eq. (6). This method can be transposed to joint control. If one wants the joint q_i to reach the value q_i^{des}, the command law is defined in Eq. (7). Again, Kp_q and Kd_q are the proportional and derivative gains for joint control. These control laws are used in this paper on every controlled frames

or joints.

$$\ddot{x}_I^{des} = Kpx_I^{err} + Kd(0 - \dot{x}_I) \quad (6)$$

$$\ddot{q}_i^{des} = Kp_q(q_i^{des} - q_i) + Kd_q(0 - \dot{q}_i) \quad (7)$$

2.4. *Linear Quadratic Program*

The purpose here is to control a robot with criteria minimization. LQP should be a good mean, as it is designed to resolve a problem with criteria to minimize. Suppose the criteria is $Fopti(x)$, a linear quadratic function. LQP find the optimal solution x^* which minimize $Fopti(x)$ under a set of equality and inequality constraints. In summary:

$$\begin{aligned} find: \; & x^* = argmin_{(x)}(Fopti(x)) \\ u.c.: \; & Ax = b \\ & Cx \leq d \end{aligned}$$

If the problem is over-constrained or if many solutions exist, the program will return no solution.

3. LQP-based controller design

3.1. *Designing LQP*

The problem here is to cast the physic laws and the desired motions into a suitable LQP. The model of the robot is defined by the equations Eq. (1)–(5). As they must be respected, it could be interesting to set them as equality and inequality constraints in the LQP. But Eq. (1) is generally non-linear. The idea is to linearize this equation around $(q, \dot{q})$, and to solve the problem at each time step. The parameters $M, N, \dot{q}, G, J_c, H_c$ and C can be obtained by simulation, computation or measurement. $\ddot{q}, \tau, f_c$ will be used in the LQP as variables. When a solution exists, the optimal torque τ^* found by LQP will finally be applied to the robot.

The optimization function should now be set. In order to generate smooth motion, τ^2 is minimized. This is the first step to design the LQP: minimize τ^2 under the constraints Eqs. (1)–(5). But such a system is not sufficient and will collapse. One needs to feed the LQP with more information.

3.1.1. *Frame Control*

In section 2.3, frame control has been explained. Let I be the controlled frame. Its twist is given by $\dot{x}_I = J_I\dot{q}$, where J_I is the jacobian of frame I.

This equation does not allow controlling the robot because it is not function to the LQP variables. The idea is to get the derivation of this expression, that is why section 2.3 defined a acceleration control. This leads to Eq. (8):

$$J_I\ddot{q} + H_I\dot{q} = \ddot{x}_I \tag{8}$$

Where H_I is the hessian matrix of frame I. In Eq. (8), control is possible because it is function to $\ddot{q}$. The desired acceleration $\ddot{x}_I^{des}$ is defined by a controller like a proportional derivative or a trajectory defined *a priori*. The goal is to find $\ddot{q}$ which can solve $\ddot{x}_I = \ddot{x}_I^{des}$. Two means are explored: the first is a "hard control", which consists in setting this equality as a constraint, and allows no error. The second is a "soft control" and consists in minimizing the distance between $\ddot{x}_I$ and $\ddot{x}_I^{des}$, where error may occur.

$constraint$:

$$J_I\ddot{q} + H_I\dot{q} = \ddot{x}_I^{des} \tag{9}$$

$minimize$:

$$\min_{(\ddot{q})}(\|\ddot{x}_I - \ddot{x}_I^{des}\|^2) \Leftrightarrow \min_{(\ddot{q})}(J_I.\ddot{q} + H_I.\dot{q} - \ddot{x}_I^{des})^2 \tag{10}$$

This function will be called a task. In this paper, the second solution is preferred.

3.1.2. *Hierarchy*

Here, the problem is to minimize two functions: on one hand torque minimization $\min_{(\tau)}(\tau^2)$, and in the other hand task achievement $\min_{(\ddot{q})}(\|\ddot{x}_I - \ddot{x}_I^{des}\|^2)$. The first idea is to sum the two functions, but they are not homogeneous, and there is no information about their relative importance. That is why the second idea is to weight each function by a coefficient of importance. For example, let's assume that 3 functions should be minimize: the torque and the control of 2 frames I and J by the method described in Eq. (10). So optimizing function become:

$$F_{opti} = \alpha_\tau.\tau^2 + \alpha_I.\|\ddot{x}_I - \ddot{x}_I^{des}\|^2 + \alpha_J.\|\ddot{x}_J - \ddot{x}_J^{des}\|^2 \tag{11}$$

Where α_I represent the importance coefficient of task I. Here, homogeneous aspect is removed (α make all tasks homogeneous), and it allows prioritization. Imagine that to control point I is more important than to control point J. Then one should choose $\alpha_I \gg \alpha_J$. If there is a mean to control I and J independently, so whatever α_I and α_J, the LQP will minimize the both tasks I and J. But if they both need some degrees of freedom, then the program will minimize first task I because it "costs" more, and then

the task J with the leaving degrees of freedom. This approach is close (but not equivalent!) to work on null space of application. If one wants to work on null space, "hard control" described on Eq. (9) should be used to control points. The "soft control" developed in Eq. (10) may offer more flexibility. Let's assume a task with low level of importance has no more degree of freedom. If one uses "hard control", Eq. (9) cannot be solved, then blocks the LQP and return "no solution". But if one uses "soft control", the task will be ignored because tasks which cost more exist and have to be minimized beforehand. It does not block the LQP and a solution generally exists. This method allows a kind of hierarchy between tasks.

Last point concerns the priority of the torque minimization task. As the robot is redundant, there are many ways to achieve a task. Torque minimization just gives a specific solution. Hence, every task should have more priority:

$$\alpha_\tau \ll \alpha_I \ \forall I \tag{12}$$

3.2. *Maintaining the robot standing upright*

Numerous points should be control to maintain the robot standing upright. The first interesting point is the center of gravity. It is controlled to reach a position inside the base of support center at a given altitude. Although the robot stands upright for a moment, it pitches and falls. Indeed, the LQP minimizes its function at one time step, but it does not mean that it minimizes the function during the whole movement. An idea is to control the spine and the pelvis to stay straight, and the head attitude (the position is not controlled). Hence, the robot stands but it uses its arms to realize the different tasks. Finally, a posture control could be set to make the robot gait to look like human gait when it is idle.

3.3. *Chaining Up Tasks*

Thanks to previous paragraph, the robot can stand. Methods developed in section 3.1.1 allow controlling hand, elbow, knee, etc. The priority of each task has to be set relatively to other tasks defined above. Of course, it is possible to change these priorities at each time step. An interesting point is about chaining up tasks. Indeed, during a move, the goal could change. Let's Assume it is the case for task I. The idea is to change directly the desired pose H_I^{des}. But this will produce a discontinuity in the acceleration $\ddot{x}_I^{des}$ due to controller, and thus a discontinuity in the applied torque. This is not close to reality and could lead to instability. In this paper, a tracking

point is created, which goes smoothly from a goal to an other, and command law uses its pose as the desired pose H_I^{des}. This ensures the acceleration and torque continuity.

3.4. *LQP Design*

All elements are set to control our robot. Here is the LQP, with command law described in Eq. (6)–(7). If other frames or joints are controlled, the tasks should be added to the function $Fopti$.

$$\begin{aligned}
&\min_{(\ddot{q},\tau,f_c)} \; Fopti \\
&\quad u.c.: \quad M.\ddot{q} + N.\dot{q} = G + \tau + J_c^t.f_c \\
&\qquad\quad \tau_{min} \leq \tau \leq \tau_{max} \\
&\qquad\quad J_{ci}.\ddot{q} + H_{ci}.\dot{q} = 0 \quad \forall i \; in \; contact \\
&\qquad\quad C.f_{ci} \leq 0 \quad \forall i \; in \; contact \\
&\qquad\quad f_{ci} \leq 0 \quad \forall i \; taking \; off
\end{aligned}$$

$$\begin{aligned}
Fopti \quad &= \alpha_\tau.\tau^2 \\
&+\alpha_{post}.\|\ddot{q}_{post} - \ddot{q}_{post}^{des}\|^2 \\
&+\alpha_{cog}.\|\ddot{x}_{cog} - \ddot{x}_{cog}^{des}\|^2 + \alpha_{spn}.\|\ddot{q}_{spn} - \ddot{q}_{spn}^{des}\|^2 \\
&+\alpha_{pel}.\|\ddot{\omega}_{pel} - \ddot{\omega}_{pel}^{des}\|^2 + \alpha_{head}.\|\ddot{\omega}_{head} - \ddot{\omega}_{head}^{des}\|^2 \\
&... \\
&+\alpha_{hand}.\|\ddot{x}_{hand} - \ddot{x}_{hand}^{des}\|^2 + \alpha_I.\|\ddot{x}_I - \ddot{x}_I^{des}\|^2 + ...
\end{aligned}$$

4. Application to the virtual iCub

The authors have applied this method to a virtual robot according to the real one, Icub. Geometry, inertia, articulations ranges and torque limits have been defined. The goal is, on one hand to check if the command described in this paper gives good results, and in the other hand if it can be applied further on the real robot. The simulator used to perform these simulations is Arboris,[7] a robotic simulator designed for the Matlab software. The simulation time step is set to $\delta t = 2.5e^{-3}s$. To resolve our LQP at each time step, the software Yalmip[3] is used. Importance coefficients α are set as follow. center of gravity control is very important, so $\alpha_{cog} = 1$. For the spine and pelvis control, less important tasks, their coefficients are set to $\alpha_{spn} = \alpha_{pel} = 1e^{-2}$. Pointing, vision and other tasks are next, that's why $\alpha_{head} = \alpha_{hand} = \alpha_{foot} = 1e^{-4}$. Finally, torque minimization and pos-

ture control come last, so $\alpha_{post} = \alpha_\tau = 1e^{-9}$. Notice that a task which minimizes the contact force must be set $(min : \ \alpha_{f_c} f_c^2)$ with $\alpha_{f_c} = 1e^{-13}$, otherwise the solver does not converge and needs more iteration. About the control law, Kp is set to $20s^{-2}$ and $Kp_\omega = \pi Kp$, so $Kd = 2\sqrt{20}s^{-1}$ and $Kd_\omega = 2\sqrt{20\pi}s^{-1}$, and it commands all tasks.

The first simulation makes the robot to stand upright, thanks to the LQP described in section 3.4. The result of the simulation shows the robot which does not fall after 4 seconds. The next step is to realize pointing task. On Fig. 1, the robot reaches the goals with its two hands. The center of gravity control, the most important task, is fulfilled during whole simulation with a maximal error of $\|\ddot{x}_{cog} - \ddot{x}_{cog}^{des}\| = 3e^{-5}ms^{-2}$. The hierarchy between tasks is respected, which is confirmed by other experience where the priority between an elbow task and a hand task are changed. Coefficients are set to $\alpha_{high} = 1e^{-4}$ and $\alpha_{low} = 1e^{-6}$. On Fig. 2, final states after 2 seconds of simulation are shown: when two tasks cannot be done in the same time, the LQP commands the robot to realize the tasks with most priority first, and commands other tasks to approach their goals as close as possible.

The last simulation is about standing on one foot. The first task is to command the center of gravity to stand above the base of support of left foot, and when it is complete, to take off the right foot to desired height. As zero moment point (ZMP) is not controlled, dynamic motions cannot be done. Final state after 2 seconds is shown on Fig. 3, and error in acceleration of the center of gravity is again inferior to $3e^{-5}m.s^{-2}$. On the same figure on right side, numerous tasks are executed simultaneously, and after 2 seconds of simulation, the robot stand on left foot and realized all tasks.

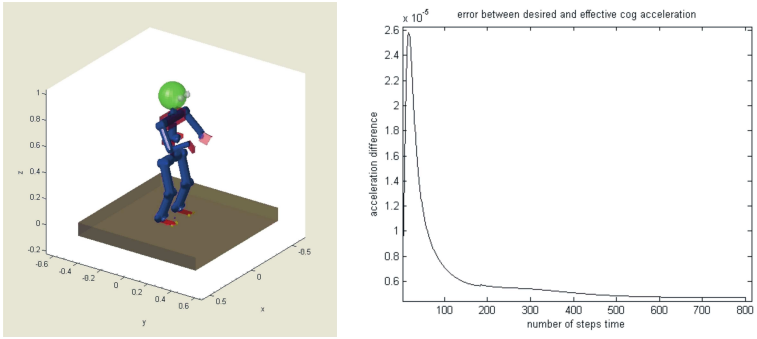

Fig. 1. Pointing tasks

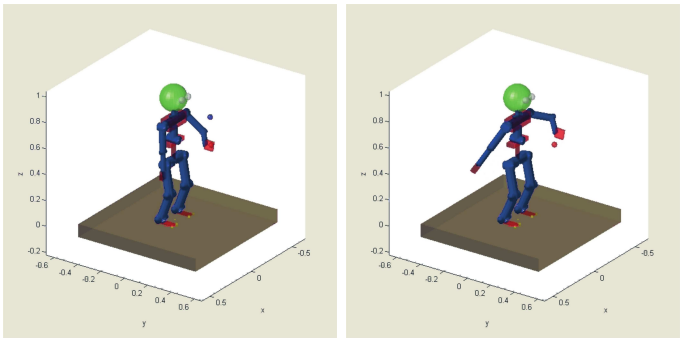

Fig. 2. Hierarchy. On left: priority to hand task ; On right: priority to elbow task

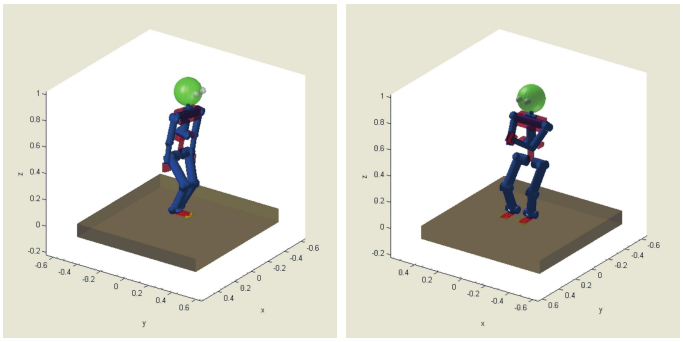

Fig. 3. On left: stand on left foot ; On right: numerous tasks in the same time

5. Conclusion

In this paper, we have presented a method to control humanoid robots. They do not want to reproduce human control, but want to use some aspect as criteria minimization. The chosen method is based on LQP resolution at each time step. The problem to solve is the execution of tasks, represented by quadratic functions, under linear constraints. It should respect the dynamic equations, torque limitations, contact laws, etc., which are represented as constraints. To execute tasks, motions are controlled in acceleration, defined by PD control laws. The move could be a constraint, which allows no error, or LQP can minimize the norm between a desired and an applied acceleration, and this allow error. Thanks to this method, hierarchy is possible: importance coefficients define priority between each

task. Hence, whole body control is possible, and simulations have been done on a virtual robot inspired from a real one, Icub.

From this paper, many perspectives occur. First, it would be interesting to get ZMP information in order to control the dynamic balance of the robot. This balance should be measured to quantify it. This would lead to dynamic walk. An other point is about robustness. The results obtain on other robots with the same controller should be compared. Other tests should be done: if the geometric or inertial data are not exactly the same between the virtual robot and those given to the LQP, or if the generalized torque τ is noisy, it would be interesting to know if efficient control is still possible. These issues will be addressed in further studies.

References

1. S. Barthélemy and P. Bidaud. Stability measure of postural dynamic equilibrium. In *11th International Symposium of Advances in Robot Kinematics (ARK'08)*, 2008.
2. F. Kanehiro K. Fujiwara K. Kaneko S. Kajita M. Nakamura K. Harada, H. Hirukawa. Dynamical balance of a humanoid robot grasping an environment. *IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, pp. 1167–1173*, 2004.
3. J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.
4. J. Popovic M. da Silva, Y. Abe. Simulation of human motion data using short-horizon model-predictive control. In *Eurographics*, 2008.
5. G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori. The iCub humanoid robot: an open platform for research in embodied cognition. In *PerMIS: Performance Metrics for Intelligent Systems Workshop*, Washington DC, USA, August 2008.
6. J. Park. *Control Strategies for Robots in Contact*. PhD thesis, Stanford University, 2006.
7. A. Micaelli S. Barthélemy. https://vizir.robot.jussieu.fr/trac/arboris.
8. L. Sentis. *Synthesis and Control of Whole-Body Behaviors in Humanoid Systems*. PhD thesis, Stanford University, 2007.
9. P.-B. Wieber. Constrained dynamics and parametrized control in biped walking. *International Symposium on Mathematical Theory of Networks and Systems. Perpignan, FR*, 2000.
10. P.-B. Wieber. *Modelisation et Commande d'un Robot Marcheur Anthropomorphe*. PhD thesis, Ecole des Mines de Paris, 2000.
11. J. Popovic Y. Abe, M. da Silva. Multiobjective control with frictional contacts. In *Eurographics/ACM siggraph*, 2007.
12. J. Yuan. Closed-loop manipulator control using quaternion feedback. *IEEE Journal of Robotics and Automation. vol. 4, no. 4, pp. 434-440*, August 1988.