Continuously satisfying constraints with contact forces in trajectory optimization for humanoid robots

Benjamin Chrétien, Adrien Escande, Abderrahmane Kheddar

▶ **To cite this version:**

Benjamin Chrétien, Adrien Escande, Abderrahmane Kheddar. Continuously satisfying constraints with contact forces in trajectory optimization for humanoid robots. IROS: Intelligent Robots and Systems, Sep 2015, Hamburg, Germany. pp.3956-3961, 10.1109/IROS.2015.7353934 . lirmm-01253539

HAL Id: lirmm-01253539

https://hal-lirmm.ccsd.cnrs.fr/lirmm-01253539

Submitted on 11 Jan 2016

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

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

Continuously satisfying constraints with contact forces in trajectory optimization for humanoid robots

Benjamin Chrétien, Adrien Escande and Abderrahmane Kheddar

***Abstract*— Humanoid robots being underactuated, they need to interact with their environment to move. When optimizing trajectories, the contact forces must therefore be taken into account. In this paper, we first highlight an issue encountered when using parametrized functions in the presence of equality constraints which need to be satisfied continuously over a time interval. We then propose a parametrization of contact forces and a formulation of the constraints which allow to write a tractable optimization program whose solution verifies at any instant all the constraints where those forces appear. We finish by exemplifying the approach with preliminary results obtained on a HRP-2 humanoid robot.**

I. Introduction

Trajectory optimization is a powerful tool for generating motions on a humanoid robot. It allows a large freedom in the problem formulation while leveraging the capacities of state-of-the-art optimization solvers. In control, it is typically used on a reduced robot model to give rise to model predictive control methods (see *e.g.* [1]). When used on a full model, it becomes a planning tool. In this paper, we are interested in the latter case. The trajectory optimization problem considers a robot with a configuration space $\mathcal{Q}$, and aims at finding a C^2 function $\boldsymbol{q} : [0, T] \longmapsto \mathcal{Q}$ minimizing a given criterion, and such that the robot realizes a set of goals expressed as constraints. T is the duration of the movement. This function must further obey physical laws and respect the robot limitations at all times. The movement is thus subject to the following set of constraints:

$$\forall t \in [0, T]\,,$$

$$M(\boldsymbol{q}(t))\ddot{\boldsymbol{q}}(t) + \mathbf{n}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) = L\boldsymbol{\tau}(t) + J(\boldsymbol{q}(t))^T \boldsymbol{f}(t) \tag{1}$$

$$\boldsymbol{q}(t) \in \mathcal{Q}_{\text{free}}, \qquad \dot{\boldsymbol{q}}(t) \in \mathcal{V}, \qquad \ddot{\boldsymbol{q}}(t) \in \mathcal{A} \tag{2}$$

$$\boldsymbol{\tau}(t) \in \mathcal{T} \tag{3}$$

$$\boldsymbol{f}(t) \in \mathcal{F} \tag{4}$$

where $\boldsymbol{\tau}$ is the vector of torques and $\boldsymbol{f}$ is the concatenation of force vectors $\boldsymbol{f}_i$ applied on points $\boldsymbol{p}_i$ of the robot[1]. Eq. (1) is the fundamental equation of dynamics (FED) with M the inertia matrix, $\mathbf{n}$ the nonlinear and gravity terms, L a selection matrix mapping the torques to the active joints, and J the Jacobian matrix of all points $\boldsymbol{p}_i$. $\mathcal{Q}_{\text{free}}$ is the subset of $\mathcal{Q}$ where the robot is within its joint limits and free of collision, $\mathcal{V}$ and $\mathcal{A}$ are the sets of acceptable joint speeds and accelerations, $\mathcal{T}$ is the set of feasible torques, and $\mathcal{F}$ is the product of the Coulomb cones of friction at each $\boldsymbol{p}_i$. Because they must be valid for every t, eqs. (1) - (4) define an infinite set of constraints.

Goal constraints are either of the form $g(\boldsymbol{q}, t_0) = 0$ (resp. $g(\boldsymbol{q}, t_0) \geq 0$) for a particular $t_0 \in [0, T]$ or $g(\boldsymbol{q}, t) = 0$ (resp. $g(\boldsymbol{q}, t) \geq 0$) over a time interval $[t_1, t_2] \subseteq [0, T]$.

Given a cost function c, typically linked to the energy consumption or duration minimization, we can write the problem of finding $\boldsymbol{q}$ as the program

$$\min_{\boldsymbol{q}, \boldsymbol{\tau}, \boldsymbol{f}, T} c(\boldsymbol{q}, \boldsymbol{\tau}, \boldsymbol{f}, T) \tag{5}$$
$$\text{subj.to goal constraints and eqs.}(1) - (4)$$

Note that the duration T is part of the variables.

Such a program can be solved with the Pontryagin maximum principle in the simplest cases, but is otherwise intractable since some variables are functions. The classical approach is then to restrict the search to classes of parametric functions: the variables become the parameters of such functions, and the search space has a finite dimension. Since the number of constraints is still infinite, the resulting program is coined *semi-infinite program* (SIP).

The most complete implementation of eq. (5) to date is found in [2], building on the work of [3]. Several specializations of the problem have been considered. In [4], [5], $\boldsymbol{q}$ is parametrized as a linear interpolation between keypoints, and only the path is considered with no consideration of time (and thus of dynamics). In [6], the path is given and the unknown is the speed profile along it which needs to be adjusted to satisfy the dynamic constraints. The duration T is minimized.

The handling of SIP in trajectory optimization has usually been done by only verifying the constraints at sampling times rather than along the whole trajectory [3]. In [7], constraints are verified continuously by the mean of interval analysis. However, the formulation can be overly conservative, and a tighter formulation is proposed in [2] based on Taylor expansion on small time intervals.

A difficult issue in the trajectory optimization program is the handling of contact forces. In [2], the authors compute forces $\boldsymbol{f}(t)$ for a given trajectory $\boldsymbol{q}$ by solving analytically for each time interval an equality-constrained quadratic program in which the constraint (4) is incorporated as a weighted least-square objective. The approach has two drawbacks: (i) (4) is not guaranteed to be satisfied. It has to be checked

This work was partially supported by JSPS (Grant-in-Aid for JSPS Fellows P13786) and by FP7 European project RoboHow.Cog (www.robohow.eu).

B. Chrétien is with the CNRS-UM LIRMM, France, A. Escande is with the CNRS-AIST JRL UMI3218/RL, Japan, A. Kheddar is with the CNRS-AIST JRL UMI3218/RL, Japan and the CNRS-UM LIRMM, France.

[1]The number of points and forces might vary with time; $\boldsymbol{f}$ must be defined for each time interval where this number is constant; same goes for $\mathcal{F}$.

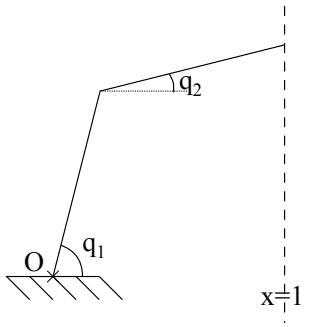

Fig. 1. A simple robot with two unit-length links, whose end effector must stay on the line $x = 1$.

upon termination of the trajectory optimization and in case of failure, a new optimization must be started with new weights. (ii) Nothing prevents the solver from converging to a trajectory $\boldsymbol{q}$ for which no feasible forces exist.

The two above problems are addressed by [8] where $\boldsymbol{f}(t)$ is found by solving a linear program which includes a discretized version of (4) as constraints, and uses a slack variable to measure the possible infeasibility of a given $\boldsymbol{q}(t)$ to drive the overall optimization to a feasible solution. However, feasible forces are only found at sampling times.

To the best of our knowledge, there is no method which finds forces that satisfy (4) continuously over a time interval. The main contribution of this paper is to propose one. We also discuss the problem of parametrization in the presence of equality constraints, a problem that was empirically observed in part, but not explained, in the 2D case in [9], and provide an improvement of the inequality constraint handling proposed in [2].

The remaining of the paper is organized as follows: in Sec. II, we exemplify and discuss the interaction between equality constraints and parametrization, in Sec. III we offer a parametrization of contact forces specifically designed to verify the FED at any instant and in Sec. IV we describe a new way for satisfying continuously inequality constraints. We then show some preliminary optimization results in Sec. V before concluding.

II. Equality constraint and parametrization

To solve problem (5), it is mandatory to discretize the search space. This can be achieved by choosing a set of parametrized functions. In this section we show how, in presence of equality constraints, that choice can dramatically reduce the number of possibilities.

Consider the 2-link arm in 2D shown in Fig. 1, and suppose that we want to constrain its end effector to stay on the line $x = 1$. At any instant, its configuration must satisfy the constraint $\cos(q_1) + \cos(q_2) = 1$. If we chose to parametrize q_1 and q_2 by polynomials of t, it is easy to show that the only solutions to the satisfaction of this constraint over a time interval are $q_1(t) = c$ and $q_2(t) = \cos^{-1}(1 - \cos(c))$, for $-\pi/2 \leq c \leq \pi/2$, *i.e.* constant solutions. Clearly, this is not satisfying as, physically, the robot has the freedom to move its end effector. But here despite the constraint being one-dimensional and while the robot has only 2 degrees of freedom, there is no possible movement in the null space of the constraint. This is due to the choice of parametrization: the chosen parametric functions cannot correctly describe a movement along the constraint. Note that with any other choice of (piecewise) polynomial parametrization (splines, Lagrange polynomials, ...), the same problem occurs.

This problem can be generalized: given a constraint $f(x(t), y(t)) = 0$ over an interval, with $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ and f a n-valued function, a poor choice of parametrization will lead to only constant solutions, despite the constraint letting m degrees of freedom to move.

In certain cases, it is possible to solve the constraint explicitly, *i.e.* write $y(t) = \phi(x(t))$. Then, we only need to parametrize x with any parametrization, and replace y by the composed expression. In our simple example above, we would have $q_2(t) = \cos^{-1}(1 - \cos(q_1))$, and the constraint would be satisfied whatever the parametrization of q_1 is, *provided we add constraints to enforce that* $-\pi/2 \leq q_1(t) \leq \pi/2$ *over the interval*. In the general case however, such a solution is out of reach.

In the trajectory optimization problem, two main types of equality constraints appear: the FED and the Cartesian positions of given bodies (generally the hands and feet) over time intervals. In the next section, we show how we chose to parametrize the variables to avoid the issue caused by the FED and one position constraint. For the remaining equality constraints, the problem is still open and, for want of a better alternative, we transform a constraint $g = 0$ into $-\epsilon \leq g \leq \epsilon$. ϵ should not be too small, to avoid impeding the convergence, but not too big either, to approximate correctly the original equality. A typical value we take on the translation part of a position constraint is 10^{-4}m.

III. FED-compatible force parametrization

For robots with a free-flyer but otherwise fully actuated joints, such as humanoid robots, the configuration space writes $\mathcal{Q} = SE(3) \times \mathcal{Q}'$, where $\mathcal{Q}'$ is the space of joint configurations, and the FED is decomposed accordingly as

$$\begin{bmatrix} M_{ff} & M_{fj} \\ M_{jf} & M_{jj} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_f \\ \ddot{\boldsymbol{q}}_j \end{bmatrix} + \begin{bmatrix} \mathbf{n}_f \\ \mathbf{n}_j \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix} \boldsymbol{\tau} + \begin{bmatrix} J_{ff}^T & J_{fj}^T \\ J_{jf}^T & J_{jj}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_f \\ \boldsymbol{f}_j \end{bmatrix} \tag{6}$$

with f and j denoting the (underactuated) free-flyer part and internal joints part.

We restrict to movements where the robot has always at least one body f fixed and in contact with its environment (in particular we do not consider ballistic phases). We consider this body as the root for kinematics and dynamics models. Since contacting bodies can change over time, we might need to choose different roots, and thus different representations, over the movement duration.

Body f is fixed, so that $\ddot{\boldsymbol{q}}_f = 0$ and eq. (6) becomes

$$\begin{bmatrix} M_{fj} \\ M_{jj} \end{bmatrix} \ddot{\boldsymbol{q}}_j + \begin{bmatrix} \mathbf{n}_f \\ \mathbf{n}_j \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix} \boldsymbol{\tau} + \begin{bmatrix} J_{ff}^T & J_{fj}^T \\ J_{jf}^T & J_{jj}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_f \\ \boldsymbol{f}_j \end{bmatrix} \tag{7}$$

This approach has several advantages: it reduces the number of variables, spares us from working with quantities

in SE(3), and solves exactly a position constraint for one body. It was also adopted in [2]. Note that the forces applied on body f still appear.

$\boldsymbol{\tau}$ is uniquely defined by the lower part of eq. (7). Therefore we do not need to parametrize it and can perform an immediate variable reduction. On the contrary, $\boldsymbol{q}$ and $\boldsymbol{f}$ are non-trivially related through the upper part of the equation so that they must be parametrized carefully if we want eq. (7) to be verified at all times while allowing non-constant solutions.

The upper part of eq. (7) is an expression of the 6 equations of Euler-Newton, up to a multiplication by a nonsingular matrix depending on the choice of frame and point to express the forces and moments [10]. If the moments in the Euler equations are written at a fixed point, J_{ff} is independent of $\boldsymbol{q}$. Furthermore, it is full row rank provided the contact points of body f are not collinear, which can be assumed for body f to remain fixed. These two properties are pivotal in the following derivations.

Regrouping all the terms on the left, and insisting on the dependencies w.r.t $\boldsymbol{q}$, the upper part of eq. (7) can be written

$$\mathbf{d}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = J_{ff}^T \boldsymbol{f}_f + J_{fj}(\boldsymbol{q})^T \boldsymbol{f}_j \tag{8}$$

We now show that $\boldsymbol{f} = \begin{bmatrix}\boldsymbol{f}_f^T & \boldsymbol{f}_j^T\end{bmatrix}^T$ can be written as $Y\mathbf{d}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) + Z(\boldsymbol{q})\tilde{\boldsymbol{f}}$ with $Y\mathbf{d}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ a particular solution to eq. (8), $Z(\boldsymbol{q})$ a basis of the null space of $\begin{bmatrix}J_{ff} & J_{fj}\end{bmatrix}^T$ and $\tilde{\boldsymbol{f}}$ a vector in this null space.

Because J_{ff}^T is full rank, its QR decomposition is $J_{ff}^T \begin{bmatrix}\Pi_1 & \Pi_2\end{bmatrix} = Q \begin{bmatrix}R_1 & R_2\end{bmatrix}$ with Q a 6×6 orthogonal matrix and R_1 6×6 full rank and thus invertible upper-triangular matrix. $\begin{bmatrix}\Pi_1 & \Pi_2\end{bmatrix}$ is a permutation matrix and R_2 has no particular properties.
Denoting $\boldsymbol{f}_1 = \Pi_1^T \boldsymbol{f}_f$ and $\boldsymbol{f}_2 = \Pi_2^T \boldsymbol{f}_f$, eq. (8) is rewritten:

$$Q^T \mathbf{d}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = R_1 \boldsymbol{f}_1 + R_2 \boldsymbol{f}_2 + Q^T J_{fj}(\boldsymbol{q})^T \boldsymbol{f}_j \tag{9}$$

which can be solved for $\boldsymbol{f}_1$:

$$\boldsymbol{f}_1 = R_1^{-1} Q^T (\mathbf{d}_f \left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right) - J_{fj}(\boldsymbol{q})^T \boldsymbol{f}_j) - R_1^{-1} R_2 \boldsymbol{f}_2 \tag{10}$$

Then

$$\begin{aligned} \boldsymbol{f} &= \begin{bmatrix}\Pi_1 \boldsymbol{f}_1 + \Pi_2 \boldsymbol{f}_2 \\ \boldsymbol{f}_j\end{bmatrix} \\ &= \begin{bmatrix}\Pi_1 R_1^{-1} Q^T \\ 0\end{bmatrix} \mathbf{d}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \\ &\quad + \begin{bmatrix}\Pi_2 - \Pi_1 R_1^{-1} R_2 & -\Pi_1 R_1^{-1} J_{fj}(\boldsymbol{q})^T \\ 0 & I\end{bmatrix} \begin{bmatrix}\boldsymbol{f}_2 \\ \boldsymbol{f}_j\end{bmatrix} \end{aligned} \tag{11}$$

from which we deduce Y, Z and $\tilde{\boldsymbol{f}}$.

Note that only $\mathbf{d}_f$ and the upper-right part of Z depend on the trajectory. Other quantities can be precomputed once and for all for a given choice and position of a root body f. This is what makes our approach tractable.

Choosing any parametrization of $\boldsymbol{q}$ and $\tilde{\boldsymbol{f}}$, eq. (11) gives us a parametrization of $\boldsymbol{f}$ satisfying eq. (7) automatically. Note that if k is the total number of contact points considered, then $\tilde{\boldsymbol{f}}$ is of size $3k - 6$. In the following, we use B-splines for $\boldsymbol{q}$ and $\tilde{\boldsymbol{f}}$.

IV. Tight inequality constraint satisfaction

Contact forces need to be in their cones of friction, *i.e.* $Y\mathbf{d}_f + Z\tilde{\boldsymbol{f}} \in \mathcal{F}$. This can be written as a set of inequality constraints $\forall t,\ g(\boldsymbol{q}(t), \tilde{\boldsymbol{f}}(t)) \geq 0$. $\boldsymbol{q}(t)$ and $\tilde{\boldsymbol{f}}(t)$ being parametrized functions of the time, such constraints are eventually expressed as

$$\forall t \in [0, T], \quad g(\boldsymbol{x}, t) \geq 0 \tag{12}$$

where $\boldsymbol{x}$ is the vector of all parameters.
In this section, we explain how we handle general inequality constraints to satisfy them continuously over a time interval. We successively reformulate eq. (12) to ultimately obtain a finite set of constraints over $\boldsymbol{x}$ only.

For the sake of simplicity, we consider hereafter that g is real-valued. The following developments extend readily to vector-valued functions by considering their elements one by one.
First, we rewrite eq. (12) as

$$\min_{t \in [0,T]} g(\boldsymbol{x}, t) \geq 0 \tag{13}$$

Therefore, we get rid of the time dependency, but end up having to compute the global minimum of a function over an interval, which is intractable in the general case. We then cut the time interval $[0, T]$ in N small intervals $[t_{i-1}, t_i]$ with $0 = t_0 < t_1 < \ldots < t_N = T$, where the t_i are expressed as fixed fractions of T and further rewrite eq. (12) as N constraints:

$$\min_{t \in [t_{i-1}, t_i]} g(\boldsymbol{x}, t) \geq 0, \quad i = 1..N \tag{14}$$

Note that both reformulations are equivalent to the original.

If the interval i is small enough, we can accurately approximate g on it by a low-order Taylor expansion $\tilde{g}_i$ around the middle point $(t_{i-1} + t_i)/2$. Since $\tilde{g}_i(t)$ is a polynomial in t, we can apply root-finding techniques to its derivative to locate its extrema and thus its global minimum over $[t_{i-1}, t_i]$. We note $t_i^{\min}(\boldsymbol{x})$ the corresponding minimizer. It is a very good estimate of the minimizer of g over $[t_{i-1}, t_i]$ and we define $g_i^{\min}(\boldsymbol{x}) = \tilde{g}(\boldsymbol{x}, t_i^{\min}(\boldsymbol{x}))$. The inequality constraint eq. (12) is ultimately rewritten as

$$g_i^{\min}(\boldsymbol{x}) \geq 0, \quad i = 1..N \tag{15}$$

This last reformulation is not equivalent to eq. (12) anymore, but is an approximation of it that can be made as accurate as necessary by adjusting the sizes of the intervals and the order of the Taylor expansion, to the expense of the computational speed. In our current implementation, we take time intervals of about 0.2s and Taylor expansion of order 5. This last choice let us use analytical formulae to find the roots of the derivative of $\tilde{g}_i$. Together with the interval size, we get a maximum inaccuracy of magnitude 10^{-6} between g and $\tilde{g}_i$.

The present constraint handling is an improvement over [2], where the constraints are also split into small intervals and a Taylor expansion is used, but then $\tilde{g}_i$ is transformed into an equivalent B-spline, with coefficients $b_{i,j}$, and the constraint is rewritten $\forall (i,j)\ b_{i,j} \geq 0$, using

the fact that a B-spline lies in the convex hull of its control points (a property also used in simpler settings in [11]). This last step can be overly conservative. Our approach is slightly more expensive, but approximates the constraint more closely, giving more freedom to solve the problem.

The fastest state-of-the-art solvers not only require to evaluate the constraints but also their derivatives w.r.t. the optimization parameters. For the above constraint, the derivation goes as follows:

$$\frac{\partial}{\partial \boldsymbol{x}}\left(g_i^{\min}(\boldsymbol{x})\right) = \frac{\partial}{\partial \boldsymbol{x}}\left(\tilde{g}_i(\boldsymbol{x}, t_i^{\min}(\boldsymbol{x})\right) = \frac{\partial \tilde{g}_i}{\partial \boldsymbol{x}}(\boldsymbol{x}, t_i^{\min}) + \dot{\tilde{g}}_i(\boldsymbol{x}, t_i^{\min})\frac{\partial t_i^{\min}}{\partial \boldsymbol{x}} \quad (16)$$

where $\dot{\tilde{g}}_i$ is the derivative of $\tilde{g}_i$ w.r.t. its time dependency. We face 3 possibilities:

- $t_i^{\min}$ lies strictly in the interval $[t_{i-1}, t_i]$ so that we have $\dot{\tilde{g}}_i(\boldsymbol{x}, t_i^{\min}) = 0$ since $t_i^{\min}$ is a critical point.
- $t_i^{\min}$ is on the boundaries of the interval, yet $\dot{\tilde{g}}_i(\boldsymbol{x}, t_i^{\min}) = 0$ (this is a pathological case).
- $t_i^{\min}$ is on the boundaries of the interval and $\dot{\tilde{g}}_i(\boldsymbol{x}, t_i^{\min}) \neq 0$, meaning that $t_i^{\min}$ would change if we move the boundaries: $t_i^{\min}$ is independent of $\boldsymbol{x}$ and thus $\frac{\partial t_i^{\min}}{\partial \boldsymbol{x}} = 0$.

The second term of eq. (16) is then always 0 and we simply compute

$$\frac{\partial}{\partial \boldsymbol{x}}\left(g_i^{\min}(\boldsymbol{x})\right) = \frac{\partial \tilde{g}_i}{\partial \boldsymbol{x}}(\boldsymbol{x}, t_i^{\min}) \quad (17)$$

As usual with pointwise minimum, the issue of continuity must be considered: $t_i^{\min}(\boldsymbol{x})$ is not a continuous function of $\boldsymbol{x}$ as it can jump from one minimum to another when a local minimum becomes the global minimum. However, $t_i^{\min}(\boldsymbol{x})$ is piecewise C^0, and from eq. (17) we get that $g_i^{\min}(\boldsymbol{x})$ is piecewise C^1.

Most solvers require that functions are at least C^1 (everywhere), but they are robust to derivative discontinuities as long as they do not occur at the optimum (a case which can occur in robotics, see for example [12]). Our early experiments did not show any problem with this discontinuity. We however implemented and tested an alternative solution which gets rid of most of the problem by considering not only $\tilde{g}_i(\boldsymbol{x}, t_i^{\min}) \geq 0$ but also all the constraints $\tilde{g}_i(\boldsymbol{x}, t^c) \geq 0$ for all "critical" points t^c: all (local) minimizers and maximizers of $\tilde{g}_i$ over interval i as well as the real part of complex roots of $\dot{\tilde{g}}_i$. The possible robustness increase is not worth the additional computational burden, but we can use this solution as a backup if we encounter a problem one day.

V. Examples

A. Description

The following examples were implemented with the RobOptim framework [13] using Python bindings, and the resolution was achieved thanks to its IPOPT [14] plugin. The model used here is the HRP-2 humanoid robot. In order to isolate our approach within the whole optimization problem, we fix the trajectory of the robot and only optimize on the forces parameters. The free-flyer part of the FED and the cone of friction constraints seen in eq. (4) need to be satisfied continuously over the course of the motion. We do not consider torque limits and thus can disregard the torque variables and the lower part of the FED. The trajectories are generated with our GPU-based whole-body dynamics motion planner that will later integrate the proposed contact forces formulation.

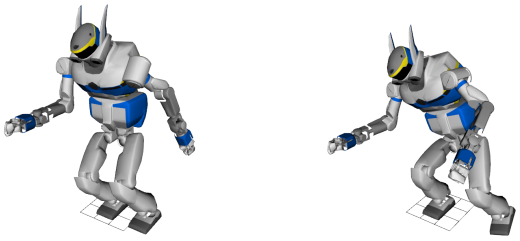

Fig. 2. Single-support motion

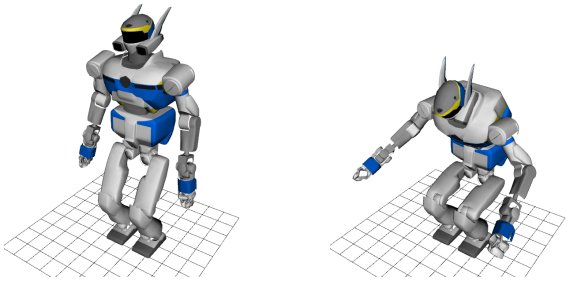

Fig. 3. Double-support motion

We consider two different examples: in the first one (Fig. 2), a single contact surface is considered, and the robot simply needs to lower its center of mass (CoM) while keeping a constant orientation for its right wrist; in the second example (Fig. 3), a double-support motion is considered, and the robot also lowers its CoM from its initial half-sitting posture. Here, we use $3k-6$ cubic B-splines as a parametrization of $\tilde{\boldsymbol{f}}$, and their control points will form the optimization parameter vector. The contact points are taken at the corner of the feet. Hence we have $k = 4$ in the first case and $k = 8$ in the second.

B. Results

In both cases, the solver was able to find contact forces control points such that the continuous constraints are satisfied throughout the motion in a few iterations.

In Fig. 6 and 7, we can indeed see that the contact forces (expressed in the contact surface frame) are on the right side of the contact surfaces (i.e. $\forall t \in [0, T], f_z(t) \geq 0$), while Fig. 4 and 5 confirm that the contact forces remain within the cones of friction (i.e. $\forall t \in T, \alpha(t)/\mu < 1$ where α is the angle of the force with the normal to the contact and μ is the friction coefficient). The FED is satisfied up to numerical errors *i.e.* at any instant t, $\mathbf{d}_f - \begin{bmatrix} J_f f^T & J_f \dot{j}^T \end{bmatrix} \boldsymbol{f}$ has no components greater in absolute value than 10^{-13}. Note that time intervals are visible on the figures.

The examples were generated without any regard to contact forces. If we slightly modify these movements to

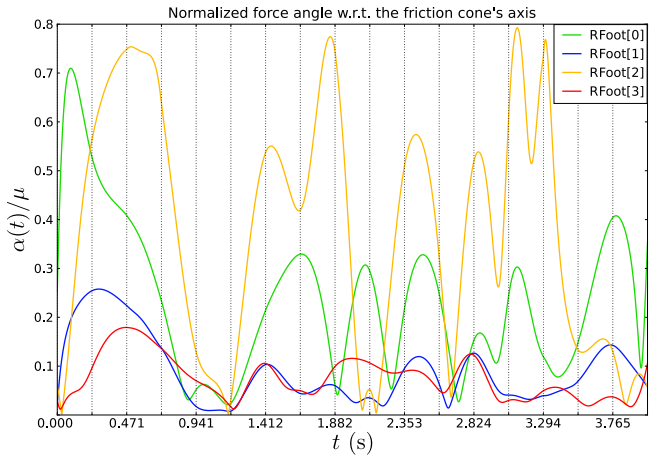

Fig. 4. Normalized angular distance $\alpha(t)/\mu$ of contact forces to the cone of friction over time in the single-support case. Constraint limit is 1.

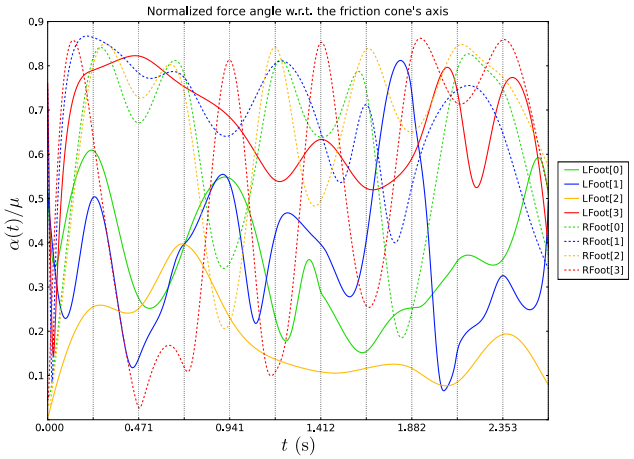

Fig. 5. Normalized angular distance $\alpha(t)/\mu$ of contact forces to the cone of friction over time in the double-support case. Constraint limit is 1.

make them impossible from a contact forces perspective, the solver will fail to converge, either after explicitly flagging the problem as infeasible, or after reaching the user-defined iteration limit. This is because the trajectory is imposed. However, because our approach uses explicitly the forces as optimization variables, it makes it possible to write tractable optimization problems with both trajectory and forces as variables. This is mathematically immediate and implementation is underway.

Currently, most of the constraints computation is done in Python. While this made it easy to prototype and validate the ideas presented here, it yields long computation times (especially due to the polynomial matrix manipulation in NumPy): for the double-support example, with $N = 11$ intervals, 360 variables, 176 constraints, the whole optimization process takes about 15 minutes. However, based on our experience on porting such problems to C++ and in particular of exploiting with GPGPU the important inherent parallelization potential of constraints evaluation in such problems, we anticipate to solve the whole optimization problem with both trajectory and forces in a few seconds. This estimate comes from the current computation times of our GPU-based motion planning library: the joint trajectories used in these examples were generated in less than a second on a laptop.

VI. Conclusion

This paper presented a parametrization of contact forces paired with a constraint formulation designed for the resolution of optimization programs that need to ensure the continuous satisfaction of constraints for robotics problems. The method relies on the QR decomposition of the free-flyer's Jacobian matrix, and allows to choose any parametrization of contact forces that best suit the problem being solved.

In future work, we plan to implement this method in our GPU-based motion planning library to efficiently treat contact forces and torque constraints in addition to the kinematics and simple dynamics constraints that it currently handles.

References

[1] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online Walking Motion Generation with Automatic Foot Step Placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[2] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of Whole-body Optimal Dynamic Multi-Contact Motions," *International Journal of Robotics Research, Special Issue on Motion Planning for Physical Robots Interaction*, vol. 32, no. 9-10, pp. 1104–1119, Apr. 2013.

[3] S. Miossec, K. Yokoi, and A. Kheddar, "Development of a software for motion optimization of robots-application to the kick motion of the HRP-2 robot," *Robotics and Biomimetics*, 2006.

[4] H.-D. R. in Dynamic Environments Using Incremental Trajectory Optimization, "Chonhyon park and jia pan and dinesh manocha," *International Journal of Humanoid Robotics*, vol. 11, no. 2, June 2014.

[5] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, August 2014.

[6] K. Hauser, "Fast interpolation and time-optimization with contact," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1231–1250, August 2014.

[7] S. Lengagne, N. Ramdani, and P. Fraisse, "Guaranteed computation of constraints for safe path planning," *Humanoid Robots (Humanoids), 2007 7th IEEE-RAS International Conference on*, 2007.

[8] K. Hauser, "Fast Dynamic Optimization of Robot Paths under Actuator Limits and Frictional Contact," in *ICRA*, 2014.

[9] S. Lengagne, P. Mathieu, A. Kheddar, and E. Yoshida, "Generation of dynamic multi-contact motions: 2d case studies," in *IEEE-RAS International conference on Humanoid robots*, 2010, pp. 14–20.

[10] P.-B. Wieber, "Some comments on the structure of the dynamics of articulated motion," in *Fast Motions in Biomechanics and Robotics*, Heidelberg, Germany, 2005.

[11] J. E. Bobrow, B. Martin, G. Sohl, E. Wang, F. C. Park, and J. Kim, "Optimal robot motions for physical criteria," *Journal of Robotic Systems*, vol. 18, no. 12, pp. 785–795, December 2001.

[12] A. Escande, S. Miossec, M. Benallegue, and A. Kheddar, "A strictly convex hull for computing proximity distances with continuous gradients," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 666–678, June 2014.

[13] T. Moulard, F. Lamiraux, K. Bouyarmane, and E. Yoshida, "RobOptim: an Optimization Framework for Robotics," in *The Robotics and Mechatronics Conference (ROBOMEC)*, 2013.

[14] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Apr. 2005, vol. 106, no. 1.

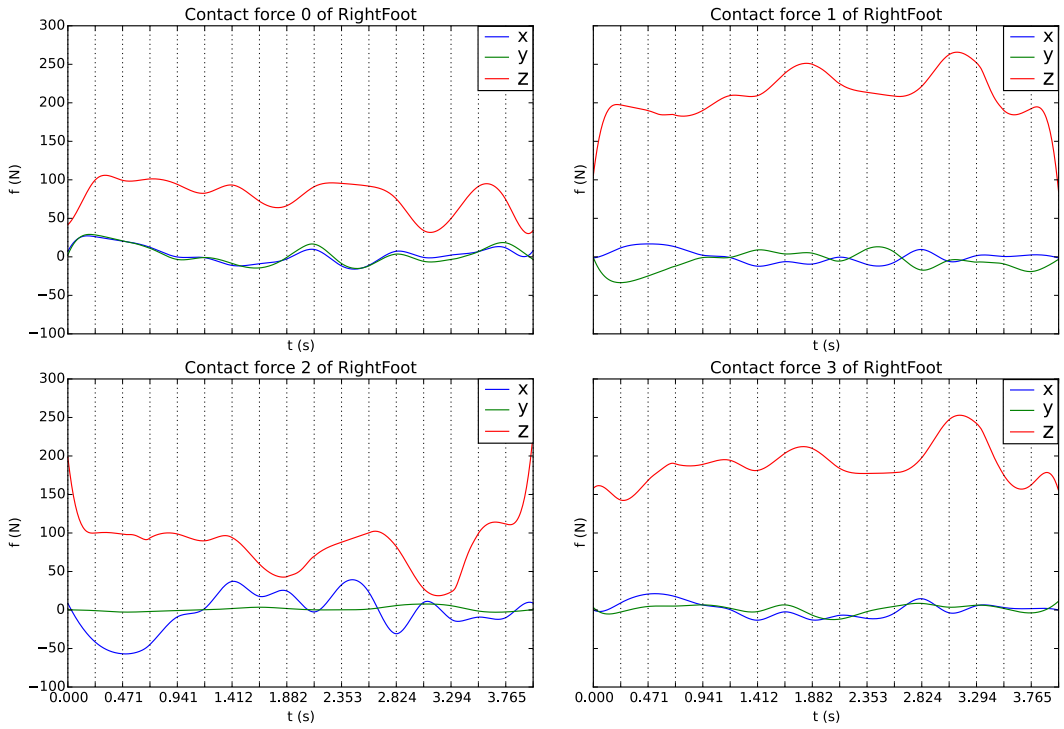

Fig. 6. Evolution of the contact forces (expressed in the contact surface frame) over time in the single-support case.

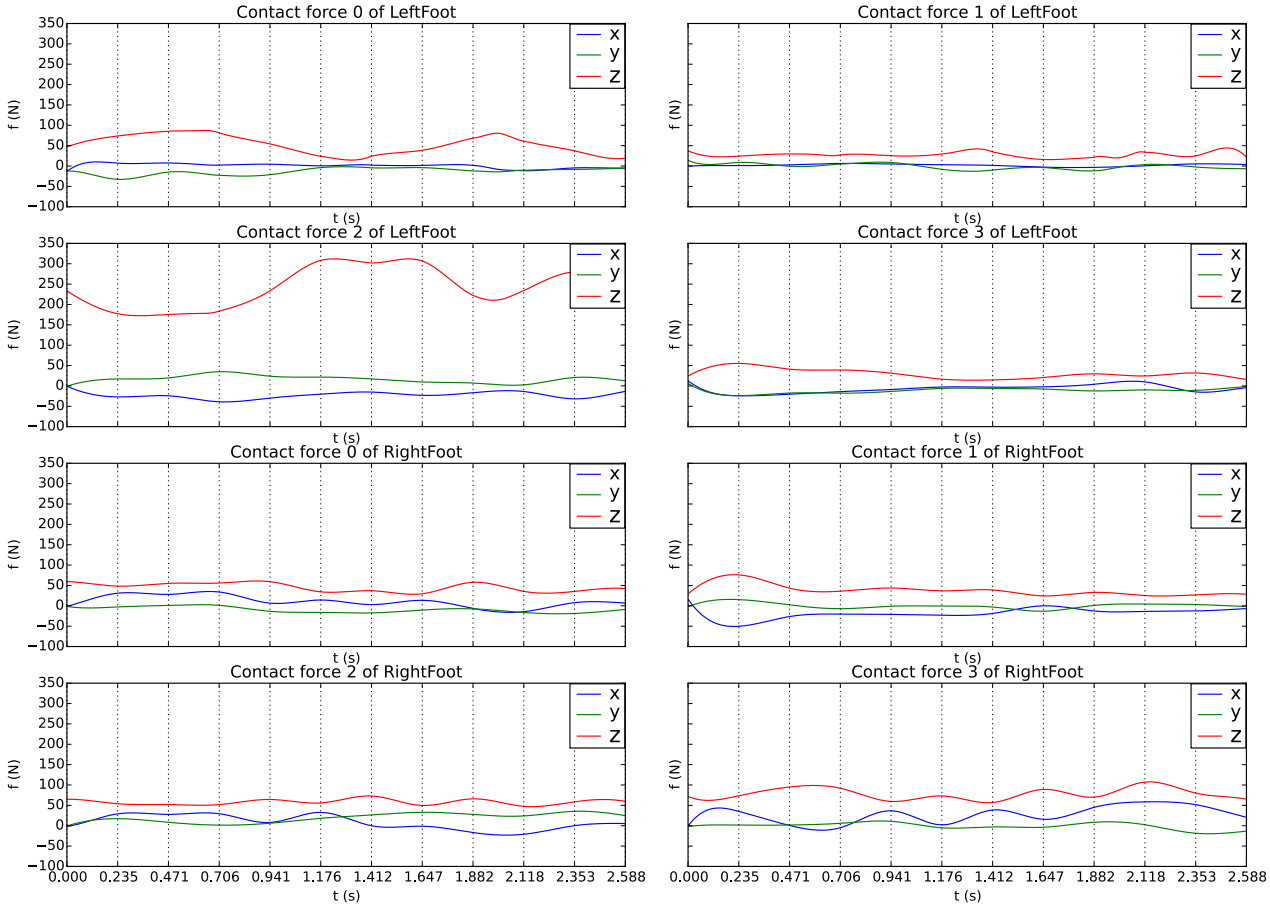

Fig. 7. Evolution of the contact forces (expressed in the contact surface frames) over time in the double-support case.