A Reactive Walking Pattern Generator Based on Nonlinear Model Predictive Control

Maximilien Naveau, Manuel Kudruss, Olivier Stasse, Christian Kirches, Katja Mombaur, Philippe Souères

A Reactive Walking Pattern Generator Based on Nonlinear Model Predictive Control

M. Naveau[1], M. Kudruss[2], O. Stasse[1], C. Kirches[2], K. Mombaur[2], P. Souères[1]

***Abstract*— The contribution of this work is to show that real-time nonlinear model predictive control (NMPC) can be implemented on position controlled humanoid robots. Following the idea of "walking without thinking", we propose a walking pattern generator that takes into account simultaneously the position and orientation of the feet. A requirement for an application in real-world scenarios is the avoidance of obstacles. Therefore the paper shows an extension of the pattern generator that directly considers the avoidance of convex obstacles. The algorithm uses the whole-body dynamics to correct the center of mass trajectory of the underlying simplified model. The pattern generator runs in real-time on the embedded hardware of the humanoid robot HRP-2 and experiments demonstrate the increase in performance with the correction.**

I. Introduction

The recent DARPA robotics challenge have shown the need for humanoid robots with an increased level of functionality enabled by proper control. Such complex robots must provide a simple interface for humans and handle as much as possible the motion generation autonomously. A general scheme is to use a motion planner to find an optimal path over a discrete set of foot-step transitions between two quasi-static poses [1], [2]. The foot-steps transition are given by a statistical exploration of a whole-body controller together with a walking pattern generator. The planner then finds a feasible sequence of quasi-static poses and foot-step transitions which minimizes a cost function and avoids obstacles. This solution is then improved online while ensuring feasibility, see for instance [3]. In general it is not possible to realize real-time motion planning by directly using the controller itself because it is not possible to run more than one or two instances of the same controller before collision. Therefore, when the planner fails it is necessary to solve a continuous local problem which will provide a feasible solution different from the precomputed one [1]. The statistical exploration can be advantageously used to cast an optimization problem to find an initial guess [1]. Recently, *Deits* proposed to define the area of convergence for a local convex problem with linear constraints [4] for a template model. With template models the inertia related to the whole-body motion is ignored, regulated to zero or corrected. In this paper it is corrected by means of a dynamic filter. It is shown in the experimental section that it is drastically improving the performances over [5] on the same robot. The use of template model is a practical solution on platforms with limited computation capabilities. Even if advanced whole-body motion controllers are now closer to real-time feasibility, e.g. the one proposed by *Todorov* which was recently applied to HRP-2 [6], they still need powerful multi-core CPUs which limit their integration on humanoid robots due to heat and power consumption.

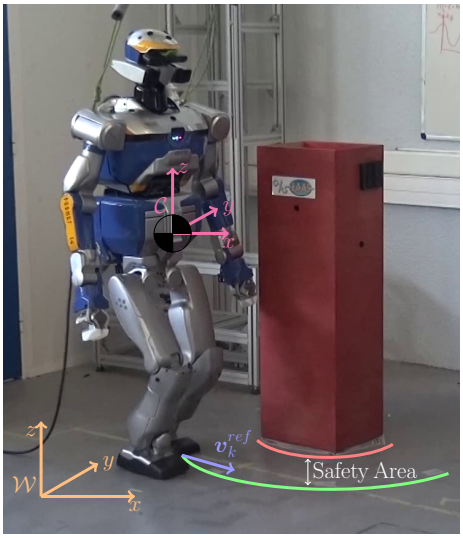

Fig. 1: HRP-2 avoiding reactively on an obstacle, even if the reference velocity v_k^{ref} drives it into it. The upper body geometry is taken into account by setting a constraint (in green) such that the robot is sufficiently away from the obstacle (in red).

Another improvement of this paper over the method developed in [5] is the nonlinear formulation which here allows to deal with obstacle. More precisely [5] integrates the information provided by statistical exploration of the controller feasibility between two foot-step transitions. It makes possible to correct foot-steps while having a guarantee over their feasibility. It is realized by reformulating the optimization problem to generate balanced Center-of-Pressure (CoP) and Center-of-Mass (CoM) trajectories where the free variables are the jerk of the CoM as well as foot-step positions and orientations. The feasible foot-steps, i.e. free of self-collision and singularities, are specified through linear constraints. This works well for level ground walking, unfortunately integrating obstacles with linear constraints implies a pre-processing of the environment or to use a different solver.

The present paper shows that obstacles can be dealt

[1]CNRS, LAAS, Université de Toulouse, 7 av. du Colonel Roche, F-31400, Toulouse, FRANCE. {mnaveau, ostasse, psoueres}@laas.fr

[2]Interdisciplinary Center for Scientific Computing, Heidelberg University, Im Neuenheimer Feld 368, 69120 Heidelberg, GERMANY. {manuel.kudruss,christian.kirches,katja.mombaur}@iwr.uni-heidelberg.de

This work has been conducted in the frame of the European project KoroiBot.

with in real-time using a nonlinear scheme. Although not demonstrated in this paper, it can be coupled with a real-time planner. The proposed method would provide a local feasible solution while the planner is looking for a global feasible solution [3].

A. State of the Art

Previous works have proposed to apply Model Predictive Control (MPC) to humanoid robots walking by considering either the whole body or a template model. When a model is available for a robot, MPC has several advantages. It can be very fast when using analytical solutions [7], [8], [9]. However such formulation makes generally some specific assumptions to find the derivation. This makes difficult the extension to other walking functionalities. On the other hand MPC schemes formulated as an optimization problem with a finer discretization grid can be more easily modified to include various walking modes inside a single formulation [10]. In addition MPC as an optimization problem is becoming increasingly popular [11], because for a given class of problems it allows using efficient off-the-shelf solvers. Moreover several methods exist to increase the efficiency of solvers for NMPC problems. For instance, it is possible to use warm-starts or use a sub-optimal solution while maintaining feasibility [12]. The goal in humanoid robotics is to find a problem formulation which realizes all the needed functionalities and copes with the robot capabilities. The locomotion problems described in [13], that include multi-contacts and consider the whole robot model over a time horizon, are not yet solvable in real-time and strongly depend on the models used to represent the physics.

Despite numerous efforts to address this large scale nonlinear problem with roughly ten thousand variables [6], [14], no solution yet exists to generate physically consistent controls in real-time using humanoid robot embedded computers. On the other hand template models projecting the overall robot dynamics to its CoM are used in research works [15], [16], [17], and already showing promising performance. Motion generation with template models can sometimes be solved analytically, and in such cases provide fast solutions that are particularly well suited for platforms with limited computational power. However, when increased CPU power is available, MPC-based solutions with the whole model are much more complete and reliable. Furthermore, as they can be easily modified, they provide more adaptive functionalities. In this paper, with a bottom-up approach, we are trying to increase the functional level of a control architecture that already works on an existing humanoid robot, HRP-2 [5]. The point of this paper is to present extensions of the linear MPC scheme presented in [5], that allows automatic foot placement in real-time. For instance, the problem depicted in Fig. 1 shows the humanoid robot HRP-2 driven by a desired velocity provided by the user. The former scheme was specifically formulated as a cascade of two quadratic programs (QPs). Foot-step orientations are solution of the first problem, while the second solution of the second QP provides the CoM trajectory and foot-step positions. This separation is efficient because the constraints are linear. If an obstacle has to be taken into account then the constraints have the shape depicted in Fig. 2, which is not convex anymore. To maintain the convexity, the solution would be to pre-process the obstacle and the feasibility area of the foot-steps. However a linearization at the a point of the obstacle boundary is equivalent to adding a linear constraint as depicted in Fig. 2. The algorithm proposed in this paper is doing a similar operation and therefore no pre-processing is necessary. This is one of the major contribution of this paper in comparison to [5]. The proposed nonlinear extension

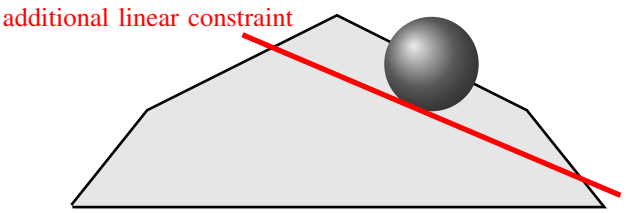

Fig. 2: Walkable zone distorted by a convex obstacle

takes into account the exact expression of constraints such as, for instance, locally avoiding a convex obstacle. Other formulations for walking motion generation have already been proposed. [4] is using mixed-integer convex optimization for planning foot-steps with Atlas. [18] is using a mixed-integer convex optimization for MPC control and foot steps timing, but this approach is not real-time feasible. In this work we introduce three nonlinear inequalities to handle balance, foot step orientation and obstacle avoidance. This new real time walking pattern generator has been successfully tested on the humanoid robot HRP-2 as depicted in Fig. 1. A key ingredient for achieving real-time performance was the following observation: *one real-time iteration of the nonlinear scheme is enough to find a reasonable solution.*

B. Contribution of the article

- It proposes a nonlinear reformulation of classical walking pattern generator able to find simultaneously foot-step positions and orientations.
- It introduces nonlinear constraints able to cope with obstacles in the environment.
- It shows experimentally that one iteration of the nonlinear iterative scheme provides a suboptimal but sufficient solution for practical cases.
- Thanks to the use of a dynamical filter that corrects the CoM trajectory to compensate the limitation of the template model as in [19] the whole body dynamics can be taken into account. This technical implementation has a strong impact on the robot performances.
- The whole algorithm runs in real time on the embedded hardware of the human-size humanoid robot HRP-2.

The theoretical contribution is the formulation of the problem as a sequence of locally linearized quadratic problems and the real-time feasible solution by applying the idea of the so called ”real-time” iteration. Details are described in Sec. III. Our practical contribution, showing that the algorithm can be implemented in real-time on the humanoid robot HRP-2, is detailed in Sec. V. A particular treatment of the dynamical filter is given in Sec. IV.

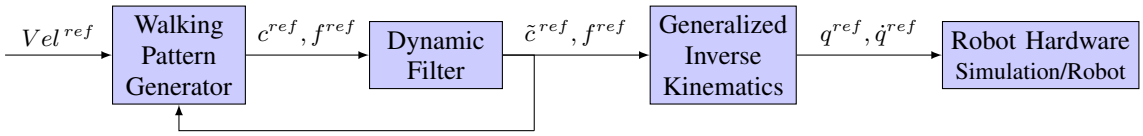

Fig. 3: The control scheme: Vel^{ref} is the input velocity. c^{ref} and f^{ref} are respectively the CoM and the feet 3D trajectories $\tilde{c}^{ref}$ is the CoM trajectory filtered. q^{ref}, $\dot{q}^{ref}$ denote respectively the generalized coordinate vector and its derivation.

II. Derivation of the dynamics

In this work the well known Linear Inverted Pendulum Model (LIPM) from [17] is used as the template model of the robot's dynamics and the following assumptions are made: 1) the angular momentum produced by the rotations of all the robot parts is supposed to be zero, 2) the robot CoM evolves on a horizontal plane, 3) the normals of the contact forces have to be collinear. As a consequence, each quantity can be expressed as a function of three degrees of freedom (DoFs), which are the projection of the robot CoM (x, y)-position on the ground plane and its free-flyer orientation θ around the vertical axis z. The reader is kindly referred to [5] for a detailed description of the several terms omitted due to the lack of space in the following section.

A. Discretization of CoM dynamics

In order to obtain a smooth trajectory, one controls the robot CoM through its jerk $\dddot{c}^{\nu}$ on a preview horizon, where c denotes the position of the CoM in the world frame and $\nu \in \{x, y\}$ is used to simplify the notation. This is done by applying a constant sampling period T and by assuming a piecewise constant jerk on each interval, i.e. $\dddot{c}^{\nu}_k(t) \equiv constant$, $t \in [kT, (k+1)T]$, $k \in \{0, 1, ..., N\}$, where N is the length of the preview horizon.

The following time-stepping scheme maps the current state of the frame c^{ν}_k to the future states by

$$\hat{c}^{\nu}_{k+j} = A^j \, \hat{c}^{\nu}_k \; + \sum_{i=0}^{j-1} A^i \, B \, \dddot{c}^{\nu}_{k+i} \;\; , \;\; j \in [0, N], \qquad (1)$$

$$\hat{c}^{\nu}_k = \begin{bmatrix} c^{\nu}_k \\ \dot{c}^{\nu}_k \\ \ddot{c}^{\nu}_k \end{bmatrix}, \; A = \begin{bmatrix} 1 & T & {}^{T^2}/_2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}, \; B = \begin{bmatrix} {}^{T^3}/_6 \\ {}^{T^2}/_2 \\ T \end{bmatrix}. \qquad (2)$$

To express the CoM over the preview horizon the vector C^{ν}_{k+1} of size $\mathcal{R}^N$ and its derivatives are defined as

$$\begin{aligned} C^{\nu}_{k+1} &= \begin{bmatrix} c^{\nu}_{k+1} & \dots & c^{\nu}_{k+N} \end{bmatrix}^T, \\ \dot{C}^{\nu}_{k+1} &= \begin{bmatrix} \dot{c}^{\nu}_{k+1} & \dots & \dot{c}^{\nu}_{k+N} \end{bmatrix}^T, \\ \ddot{C}^{\nu}_{k+1} &= \begin{bmatrix} \ddot{c}^{\nu}_{k+1} & \dots & \ddot{c}^{\nu}_{k+N} \end{bmatrix}^T, \\ \dddot{C}^{\nu}_{k+1} &= \begin{bmatrix} \dddot{c}^{\nu}_{k+1} & \dots & \dddot{c}^{\nu}_{k+N} \end{bmatrix}^T. \end{aligned}$$

Using eq. (1), the above vectors can be expressed as a function of the initial state $\hat{c}^{\nu}_k$ and the CoM jerk $\dddot{C}^{\nu}_{k+1}$. The latter belongs to the free-variable vector of the optimization problem described in section III.

B. Linear inverted pendulum dynamics

In this paper the balance criteria used is to have the center of pressure (CoP) in the convex hull of the robot's support polygon, which is defined by the contacts with the ground [17] (see Sec. III-C). Hence, the CoP has to be expressed in terms of the system's free variables, i.e. the CoM jerk. Using the assumptions made in the introduction of Sec. II, the robot CoP can be expressed as a linear function of the CoM, i.e.

$$z^{\nu}_{k+n} = \begin{bmatrix} 1 & 0 & -{}^{h}/_{g} \end{bmatrix} \hat{c}^{\nu}_{k+n} \, , \;\; \nu \in \{x, y\} \, , n \in [0, N-1],$$

with $h = c^z - z^z$ being the height of the CoM with respect to the ground and g the norm of the gravity vector. Using eq. (1), a recursive expression for the future evolution of the CoP for a fixed horizon of N sampling steps is given by

$$z^{\nu}_{k+n} = \begin{bmatrix} 1 & 0 & -{}^{h}/_{g} \end{bmatrix} \left[A^n \, \hat{c}^{\nu}_k \; + \sum_{i=0}^{n-1} A^i \, B \, \dddot{c}^{\nu}_{k+i} \right]. \qquad (3)$$

As in Sec. II-A, the vector $Z^{\nu}_{k+1} = \begin{bmatrix} z^{\nu}_{k+1} & \dots & z^{\nu}_{k+n} \end{bmatrix}^T$, of size $\mathcal{R}^N$, is used to describe the CoP on the preview horizon. This vector can then be expressed in terms of $\hat{c}^{\nu}_k$ and $\dddot{C}^{\nu}_{k+1}$.

C. Automatic foot step placement

The adaptive placement of the feet, with the aim to ensure balance of the robot even under external perturbations, is a key-feature of the algorithm. To this end, consider a frame $\mathcal{F}$ attached to the support foot, with its current position and orientation on the ground given by f^{η}_k, with $\eta \in \{x, y, \theta\}$. The future steps, also free variable of the optimization problem, are denoted by

$$\begin{aligned} F^{\eta}_{k+1} &= \begin{bmatrix} f^{\eta}_{k+1} & f^{\eta}_{k+2} & \dots & f^{\eta}_{k+N} \end{bmatrix}^T \\ F^{\eta}_{k+1} &= v_{k+1} f^{\eta}_k + V_{k+1} \tilde{F}^{\eta}_{k+1} \end{aligned} \qquad (4)$$

with F^{η}_{k+1} of size $\mathcal{R}^N$ representing the foot support position at each time step and $\tilde{F}^{\eta}_{k+1}$ of size $\mathcal{R}^{nf}$ the actual free variables of the problem. The vector $v_{k+1} \in \mathbb{R}^N$ and matrix $V_{k+1} \in \mathbb{R}^{N \times nf}$ indicate which step falls in the sampling interval (see [5] for more details). Sampling times correspond to rows, steps to columns, and nf is the maximum number of double support phases in the preview.

In theory, the usage of a single point mass as model prevents the definition of an orientation. In [5] a frame attached to the center mass is defined and the orientation of this frame and the feet directions are optimized. In this work only the foot step orientations are optimized, and the orientation of the robot free-flyer is computed from this solution. Let ff$^{\theta}(t)$, $f^{\theta,L}(t)$ and $f^{\theta,R}(t)$ be respectively the

orientation of the free-flyer, the left foot and the right foot at any time t. Hence $\text{ff}^\theta(t)$ is by convention :

$$\begin{bmatrix} \text{ff}^\theta(t) \\ \dot{\text{ff}}^\theta(t) \\ \ddot{\text{ff}}^\theta(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(f^{\theta,L}(t) + f^{\theta,R}(t)) \\ \frac{1}{2}(\dot{f}^{\theta,L}(t) + \dot{f}^{\theta,R}(t)) \\ \frac{1}{2}(\ddot{f}^{\theta,L}(t) + \ddot{f}^{\theta,R}(t)) \end{bmatrix}.$$

III. Nonlinear Model Predictive Control

Solving the orientation problem separately from the position problem is a workaround to linearize the CoP (eq. (13)) and foot position (eq. (15)) constraints derived below. However, computing separately the orientation and then injecting the solution into the position QP amounts to solve a different problem than the nonlinear combination of both. In the following the nonlinear problem will be derived, analyzed and an appropriate approach is proposed, allowing the real-time execution of the algorithm on the robot.

A. The controller

A scheme of the controller is shown in Fig. 3. This open-loop controller is used for tracking respectively a referenced linear and angular velocity. In the first step, the walking pattern generator (WPG) computes the foot steps and CoM jerk from the given velocity $Vel_{k+1}^{ref} = \begin{bmatrix} Vel_{k+1}^{x,ref} & Vel_{k+1}^{y,ref} & Vel_{k+1}^{\theta,ref} \end{bmatrix}$. Then it uses an Euler integration scheme to compute the CoM trajectory from its jerk and polynomials of fifth order to retrieve 3D trajectories for the feet from the foot step planning. The CoM computed by the WPG is then filtered (see Sec. IV) and sent altogether with the feet trajectory to a generalized inverse kinematics algorithm. The output is a whole-body walking trajectory that can be applied directly on the robot. The WPG is then reinitialized with the current reference velocity input and with the corrected initial states of the dynamic filter.

B. The cost function

The cost function used in the NMPC is given by

$$\min_{U_k} \frac{\alpha}{2} J_1(U_k) + \frac{\alpha}{2} J_2(U_k) + \frac{\beta}{2} J_3(U_k) + \frac{\gamma}{2} J_4(U_k) \quad (5)$$

with α, β and γ being the weights of the cost function and U_k the free variables of the problem defined as

$$U_k^{x,y} = \begin{bmatrix} \dddot{C}_k^x \\ \tilde{F}_k^x \\ \dddot{C}_k^y \\ \tilde{F}_k^y \end{bmatrix}, \; U_k^\theta = \tilde{F}_k^\theta, \; U_k = \begin{bmatrix} U_k^{x,y} \\ U_k^\theta \end{bmatrix}. \quad (6)$$

$J_1(U_k)$ is the cost function related to the linear velocity tracking

$$J_1(U_k) = \|\dot{C}_{k+1}^x - Vel_{k+1}^{x,ref}\|_2^2 + \|\dot{C}_{k+1}^y - Vel_{k+1}^{y,ref}\|_2^2.$$

$J_2(U_k)$ is the cost function related to the angular velocity tracking

$$J_2(U_k) = \|F_{k+1}^\theta - \int Vel_{k+1}^{\theta,ref} dt \,\|_2^2.$$

Experiments have shown that a different weight between linear and angular velocities was not necessary at this stage.

$J_3(U_k)$ is the cost function minimizing the distance between the CoP and the projection of the ankle on the sole

$$J_3(U_k) = \|F_{k+1}^x - CoP_{k+1}^x\|_2^2 + \|F_{k+1}^y - CoP_{k+1}^y\|_2^2. \quad (7)$$

$J_4(U_k)$ is the cost function minimizing the norm of the control

$$J_4(U_k) = \|\dddot{C}_{k+1}^x\|_2^2 + \|\dddot{C}_{k+1}^y\|_2^2.$$

The above minimization function can then be express in a canonical form

$$\min_{U_k} \quad \frac{1}{2} \; {U_k}^T \, Q_k \, U_k \;\; + \;\; {p_k}^T \, U_k \,, \quad (8)$$

$$\text{with } Q_k = \begin{bmatrix} Q_k^{x,y} & 0 \\ 0 & Q_k^\theta \end{bmatrix}, \quad p_k = \begin{bmatrix} p_k^{x,y} \\ p_k^\theta \end{bmatrix}, \quad Q_k^\theta = \alpha \, \mathbb{I}_{nf},$$

$$p_k^\theta = \alpha \left(\begin{bmatrix} 1 & \dots & nf \end{bmatrix} \, T_{step} \, Vel_{k+1}^{\theta,ref} + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} f_k^\theta \right).$$

The reader is kindly referred to [5] for the defintion of $Q^{x,y}$ and $p^{x,y}$. The matrix Q_k^θ and p_k^θ are derived because we use a slightly different method than [5] to deal with the orientation.

C. The constraints

First of all the balance of the robot has to be ensured, then the feasibility of the foot step needs to be verified. Finally, the nonlinear constraint which implements the obstacle avoidance is described. It is one of the contribution introduced by this paper. The following exposition is based on [5].

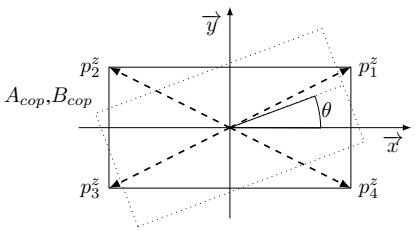

Fig. 4: Shape of the foot with the position vector p_i^z describing the support polygon and θ representing its orientation. The 4×2 matrix A_{cop} and the 4×1 vector B_{cop} are the linear algebra representation of the edges.

1) Balance constraint: The CoP has to remain inside the support polygon [20]. This polygon is depicted in Fig. 4. The set of linear inequalities representing the convex polygon is denoted as A_{cop} and B_{cop}. Only one foot is modeled as a support polygon for two reasons: 1) HRP-2 feet are symmetrical, 2) the sampling period of the problem is designed in a way that no iteration of the optimization problem falls into a double support phase. The CoP at instant k, ($z_k = [z_k^x \;\; z_k^y]^T$), see Sec. II-B) lies inside the support polygon if and only if

$$A_{cop} R(f_k^\theta) \, (z_k - f_k) \leq B_{cop} \quad (9)$$

$$\begin{bmatrix} A_{cop,k}^{x,\theta} & A_{cop,k}^{y,\theta} \end{bmatrix} (z_k - f_k) \leq B_{cop} \quad (10)$$

$$R(f_k^\theta) = \begin{bmatrix} \cos(f_k^\theta) & \sin(f_k^\theta) \\ -\sin(f_k^\theta) & \cos(f_k^\theta) \end{bmatrix}, \quad (11)$$

where $f_k = [f_k^x \;\; f_k^y]^T$, $A_{cop,k}^{x,\theta}$ is the left column of $A_{cop} R(f_k^\theta)$ and $A_{cop,k}^{y,\theta}$ is the right one. Using eq. (4) the

constraint for each time step of the preview horizon is defined by

$$D_{k+1}(U_k^\theta)\begin{bmatrix} Z^x_{k+1} - v_{k+1}f^x_k - V_{k+1}\tilde{F}^x_{k+1} \\ Z^y_{k+1} - v_{k+1}f^y_k - V_{k+1}\tilde{F}^y_{k+1} \end{bmatrix} \leq b_{cop\ k+1} \tag{12}$$

With $b_{cop\ k+1} = [B_{cop} \dots B_{cop}]^T$ and $D_{k+1}(U_k^\theta) =$

$$\begin{bmatrix} A^{x,\theta}_{cop,k+1} & & 0 & A^{y,\theta}_{cop,k+1} & & 0 \\ & \ddots & & & \ddots & \\ 0 & & A^{x,\theta}_{cop,k+N} & 0 & & A^{y,\theta}_{cop,k+N} \end{bmatrix}.$$

From eq. (12), the canonical form of the constraint is

$$A_{cop,k}(U_k^\theta)\ U_k^{x,y} \ \leq\ \overline{U_{cop,k}}, \tag{13}$$

where $A_{cop,k}(U_k^\theta)$ is a matrix depending on U_k^θ which makes this constraint nonlinear. And $\overline{U_{cop,k}}$ is the upper bound vector. The last steps of the derivation are detailed in [5].

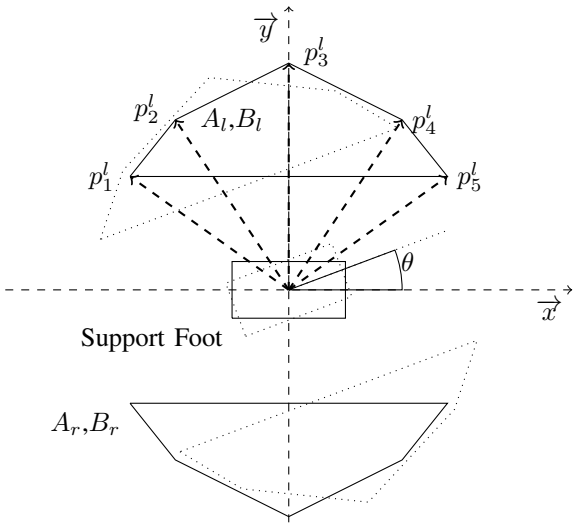

Fig. 5: Shape of the selected convex polygon boundary of the foot placement. The 5×2 matrix $A_{\mathrm{r,l}}$ and the 5×1 vector $B_{\mathrm{r,l}}$, define the convex hull as a set of linear inequalities.

2) Foot step feasibility constraint: This constraint uses the same convex hull as in [5] to ensure the feasibility of the steps. For HRP-2 this convex hull is shown in Fig. 5. The set of linear inequalities representing this convex polygon is defined by A_{foot} and B_{foot}. Instead of r or l the lower index $foot$ is used because the problem is symmetrical. The constraint, representing the fact that the swing foot has to land inside the convex hull, is given as

$$A_{foot}R(\theta)(f_{k+1} - f_k) \leq B_{r,l}. \tag{14}$$

In the exact manner as in eq. (13), the vector and matrices depicted in Sec. II are used to express this constraint for each previewed foot step. More details are presented in [5]. The canonical form of the constraint is

$$A_{foot,k}(U_k^\theta)\ U_k^{x,y} \ \leq\ \overline{U_{foot,k}}, \tag{15}$$

where $A_{foot,k}(U_k^\theta)$ depends on U_k^θ like $A_{cop,k}(U_k^\theta)$, which makes this constraint nonlinear. And $\overline{U_{foot,k}}$ is the upper bound vector.

3) Foot orientation constraint: One additional feasibility constraint considers the maximum and minimum angle between both feet

$$-\theta_{thresh} \leq F^\theta_{k+1} - F^\theta_k \leq \theta_{thresh}, \tag{16}$$

with the canonical form

$$\underline{U_{\theta,k}} \leq A_\theta U_k^\theta \leq \overline{U_{\theta,k}} \tag{17}$$

$$\text{with}: A_\theta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \ddots & 0 & -1 & 1 \end{bmatrix},$$

$$\overline{U_{\theta,k}} = \begin{bmatrix} \theta_{thresh} + f^\theta_k & \theta_{thresh} & \dots & \theta_{thresh} \end{bmatrix}^T,$$

$$\underline{U_{\theta,k}} = \begin{bmatrix} -\theta_{thresh} + f^\theta_k & -\theta_{thresh} & \dots & -\theta_{thresh} \end{bmatrix}^T.$$

In practice the bound $\theta_{thresh} = 0.05rad$ takes into account the hardware limits. At this stage, the optimization problem allows the robot to place its feet anywhere inside the convex hull at any moment. In [5], the velocity of the foot is limited by bounding the feasible foot step area that corresponds to a maximum velocity. We chose to use the same idea extended to all the foot steps degrees of freedom. This significantly decreases the variation of accelerations before foot landing.

D. Additional constraint : local obstacle avoidance

Here, only the convex obstacles are considered. For simplification the obstacle is defined as a circle $C = \{(p^x, p^y) \in \mathcal{R}^2, (p^x - x_0)^2 + (p^y - y_0)^2 = R^2\}$ Where x_0 and y_0 are its center coordinates in the world frame and R its radius. The previewed foot steps are feasible if they are outside the circle. This constraint does not depend on the orientation of the foot steps. For the j^{th} previewed step, at iteration $k + j$ the constraint is expressed by

$$\left(f^x_{k+j} - x_0\right)^2 + \left(f^y_{k+j} - y_0\right)^2 \geq R^2 + m^2 \tag{18}$$

$$\iff \quad U_k^T H_{obs,j} U_k + A_{obs,j} U_k \ \geq \underline{U_{obs,j}} \quad , \tag{19}$$

with $H_{obs,j}$ a selection matrix, $A_{obs,j}$ a vector depending on x_0 and y_0, and m a security margin taking into account the swept volume of the robot.

E. The solver

This paragraph presents the method used to solve the problem detailed in the previous sections. The non-linearity of the constraint and the still quadratic objective classifies the former LQR scheme as a nonlinear least squares optimization problem, which has the general form

$$\min_{U_k} \quad \frac{1}{2}\|l(U_k)\|_2^2 \tag{20a}$$

$$\text{s.t.} \quad \underline{h} \leq h(U_k) \leq \overline{h}. \tag{20b}$$

In general, derivative-based methods in the form of sequential quadratic programming SQP can be used for nonlinear optimization problems. These methods are called SQP because at each iteration a second order approximation of the nonlinear problem is calculated. Here, the least squares

structure can be exploited to solve eq. (20) more efficiently using a generalized Gauß-Newton method. Starting with an initial guess U_{k-1} the method iterates $U_k = U_{k-1} + \Delta U_k$, where the increment ΔU_k is obtained from the solution of the following QP approximation under the following form

$$\min_{\Delta U_k} \quad \frac{1}{2}\|l_{k-1} + (\nabla_{U_k} l_k|_{U_{k-1}})^T \Delta U_k\|_2^2 \tag{21a}$$

$$\text{s.t.} \quad \underline{h} - h_{k-1} \leq (\nabla_{U_k} h_k|_{U_{k-1}})^T \Delta U_k \leq \overline{h} - h_{k-1} \tag{21b}$$

with

$$l_k := l(U_k), \;\; h_k := h(U_k)\,.$$

Reformulating eq. (21) as a QP in canonical form, we get

$$\min_{\Delta U_k} \quad \frac{1}{2}{\Delta U_k}^T \tilde{Q}_k \Delta U_k + \tilde{p_k}^T \Delta U_k \tag{22a}$$

$$\text{s.t.} \quad \underline{\tilde{U_k}} \leq \tilde{A}_k \Delta U_k \leq \overline{\tilde{U_k}} \tag{22b}$$

with

$$\tilde{Q_k} = Q_k, \quad \tilde{p_k} = \begin{bmatrix} \frac{1}{2}(U_{k-1}^{x,y})^T Q_k^{x,y} + p_k^{x,y} \\ \frac{1}{2}(U_{k-1}^{\theta})^T Q_k^{\theta} + p_k^{\theta} \end{bmatrix}$$

$$\tilde{A_k} = \begin{bmatrix} A_{cop,k}(U_{k-1}^{\theta}) & \nabla^T_{U_k^{\theta}} A_{cop,k}|_{U_{k-1}^{\theta}} \; U_{k-1}^{x,y}) \\ A_{foot,k}(U_{k-1}^{\theta}) & \nabla^T_{U_k^{\theta}} A_{foot,k}|_{U_{k-1}^{\theta}} \; U_{k-1}^{x,y}) \\ 0 & A_{\theta} \\ H_{obs,j} U_{k-1} + A_{obs,j} & 0 \end{bmatrix},$$

$$\underline{\tilde{U_k}} = \begin{bmatrix} -\infty \\ -\infty \\ \underline{U_{\theta,k}} \\ \underline{U_{obs,j}} \end{bmatrix} - h_{k-1}\,, \qquad \overline{\tilde{U_k}} = \begin{bmatrix} \overline{U_{cop,k}} \\ \overline{U_{foot,k}} \\ \overline{U_{\theta,k}} \\ +\infty \end{bmatrix} - h_{k-1}\,,$$

$$h_{k-1} = \begin{bmatrix} A_{cop,k}(U_{k-1}^{\theta}) \; U_{k-1}^{x,y} \\ A_{foot,k}(U_{k-1}^{\theta}) \; U_{k-1}^{x,y} \\ A_{\theta} \; U_{k-1}^{\theta} \\ U_{k-1}^T H_{obs,j} U_{k-1} + A_{obs,j} U_{k-1} \end{bmatrix},$$

$$\forall j \in 1, \ldots, nf.$$

In this work the NMPC scheme is based on the idea of the so called "real-time iteration" [21], [22]. At each time instant of the control loop the nonlinear problem resolution requires the use of a SQP method. However by carefully initializing the applied SQP method and by preserving the state from the last iteration, the computational effort can be reduced to solving a single QP (one iteration of the respective SQP method) at each time. Furthermore, the computational process can be separated into three phases, two of which can be completed in advance without knowledge of the actual process state. In this way, the feedback delay can be drastically reduced. Therefore, instead of solving eq. (20) we recalculate its linearization once at each iteration of the control loop and solve a single QP eq. (22) in each iteration. This allows a real-time execution on the robot even for the proposed nonlinear formulation.

IV. Dynamic Filter

Recall that the algorithm presented in this paper and the one presented in [5] assume that the inertial effect of the legs and the arms are neglected. An interesting fact is that the algorithm in [5] that was successfully implemented on the HRP-2 in the Japan Robotic Laboratory (JRL), turned out to be unstable for its first test on another HRP-2 robot located at LAAS-CNRS. In order to cope with this difficulty, we used the dynamic filter introduced by Kajita [17]. This filter aims to correct the difference between the referenced CoP computed by the pattern generator and the CoP reconstructed from the joint trajectories finally realized on the robot. In order to do so, a second model predictive control is used. This technique is often seen as applying a Newton Raphson method on the following equation $z^{ref}(t) = RNEA(q(t))$, where $RNEA$ is the Recursive Newton-Euler Algorithm applied on the multi-body robot model. It computes the multi-body CoP from q, the generalized spatial state vector at time t. In general this method does not guarantee the convergence, and might suffer from numerical instability. However, it has proven its efficiency for this specific problem [23]. Indeed in practice one iteration of the dynamic filter is sufficient to reduce considerably the error on the CoP (see Fig. 6).

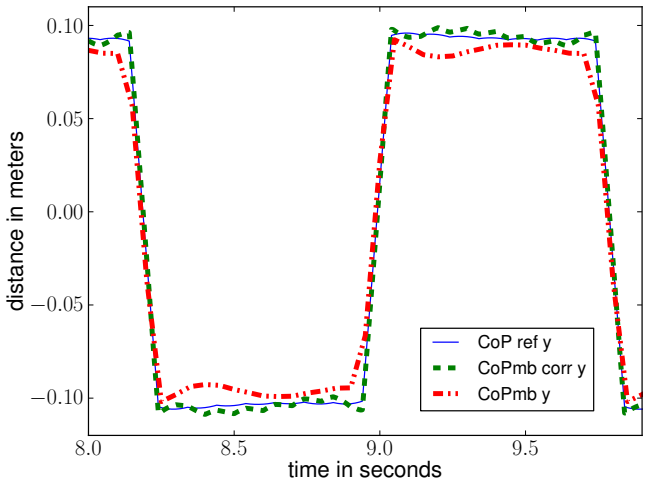

Fig. 6: Result of the dynamic filtering on the CoP. In solid blue, the reference CoP computed by the solver. In dash-dot-dot red, the CoP multi-body. In dashed green, the CoP multi-body recomputed after correction.

V. Experiments with HRP-2

In this section two experiments on the HRP-2 humanoid robot are presented. As described in the introduction they correspond to local situations where a foot-step planner using a discrete set of foot-step transitions may fail. We consider the case where only a reference velocity is given to drive the robot. It corresponds to a sensor-based behavior such as the one presented in [24]. The integration with a reactive planner such as the one presented [3] is left for future work. In the first experiment, the reference velocity drives the robot towards an obstacle which can be avoided thanks to the WPG. The second experiment shows the robot performing a circular trajectory and avoiding an obstacle.

A. Experimental setup

The duration of one full step is $0.8\,s$, including single support ($0.7\,s$) and double support ($T = 0.1s$). During the

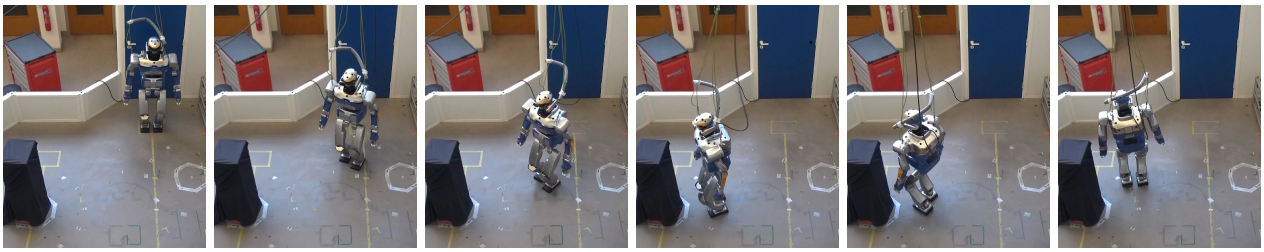

Fig. 7: Experiment on the HRP-2 robot using the setup B.

experiment the preview horizon of the NMPC is two full steps, while the preview horizon of the dynamic filter is equal to one full step in order to insure real time feasibility on HRP-2.

Fig. 8 depicts the two experimental setups. The upper figure and Fig. 1 show the output of the algorithm in the situation *A*. The forward velocity is set to $Vel^{ref}_{k+1} = [0.2, 0, 0]$ and the obstacle to avoid is the red box. In Fig. 8 the box is represented by the inner red circle while the security margin is represented by the outer green circle. The robot is allowed to step on the green circle but not inside it. This margin prevents the upper body from colliding with the obstacle. The setup *B*, depicted in Fig. 7 and 8 is quite similar. A constant velocity $Vel^{ref}_{k+1} = [0.2, 0, 0.2]$ including rotation around the vertical axis is sent to the walking pattern generator. The robot starts to describe a circle and get stuck in front of the obstacle. As the constraint is locally linearized, and because the reference velocity in translation is going towards the constraint, the robot is blocked in translation. Thus, it stops moving forward and continues to turn on spot, as the angular velocity is not conflicting with the constraint. Once the robot has passed the obstacle, it can freely move forward and describe a circle again.

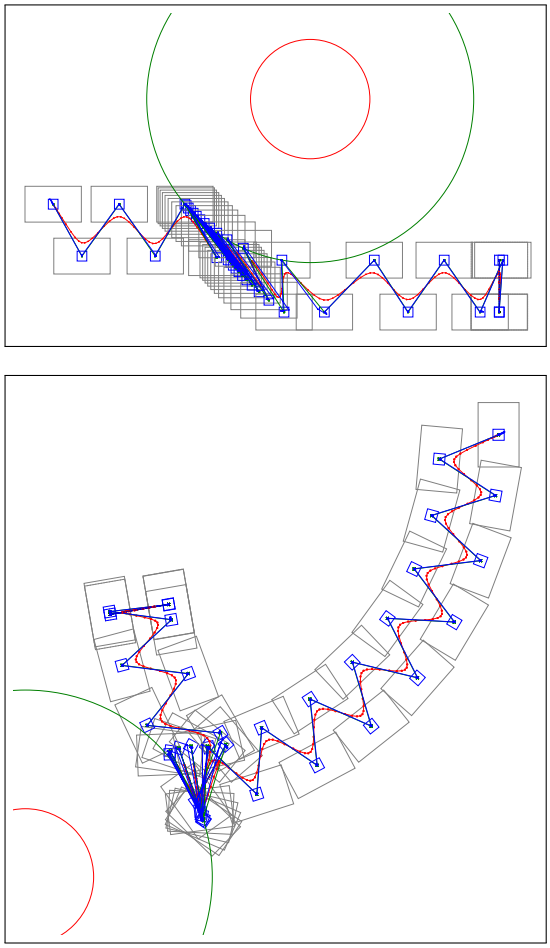

Fig. 8: Center-of-mass and center-of-pressure trajectories for obstacle avoidance and foot-step orientation using NMPC. Situation A *(up): Constant forward velocity. Situation* B *(bottom): Constant forward and angular velocity.*

B. Robustness to perturbation

A disturbance test case has been performed in simulation. The disturbance is introduced as a force added to the CoM acceleration in the walking pattern generator. This force is applied during $100\,ms$. Two kind of disturbances were considered: on the sagittal plane (both directions) and on the coronal plane (both directions). In both cases, we considered two walking situations: forward and on spot.

On the coronal plane, the maximum lateral force that can be handled is $90\,N$, equivalent to $-0.63\,J$, and $-45\,N$, equivalent to $0.675\,J$. The asymmetry comes from the fact that the robot might be in a different walking situation during the push. The push may occur when the robot can perform a step without collision, or when it cannot. In the latter case, the magnitude of the force that can be rejected is smaller. We found roughly the same values for the two walking situations.

When walking on spot, the maximum forward and backward perturbation is $\pm 115\,N$,equivalent to $\pm 0.86\,J$, as the problem is symmetrical. When walking forward, the maximum disturbance is smaller in the forward direction. The interval found is $[-160; 70]\,N$, equivalent to $[-1.12; 1.54]\,J$.

C. Computation time

This algorithm runs online on the HRP-2 CPU board ($Intel(R)\,Core2(TM)\,Duo\,E7500$, one core used, $2.8\,GHz$, $3\,Mb$ of cache size, on Ubuntu 10.04 LTS). So only counts the iteration when the NMPC is computed. Thus the statistics apply only when the walking pattern generator is computed. The time measurement has been performed on the complete control architecture (see Fig. 3).

Time consumption	experiment A	experiment B
Average (ms)	3.95	4.00
Standard deviation (ms)	0.14	0.18
Minimum (ms)	3.34	3.085
Maximum (ms)	4.34	5.19

The robot is controlled at a period of $5\,ms$. Over all the experiences, there was only one iteration over $5\,ms$. It is due to the stabilizer which consumes more CPU time when the robot is in a configuration leading to a kinematic singularity. The algorithm is still computed every $100\,ms$ to simplify the double support phase handling.

D. Cost function gains

The cost function gains are : $\alpha = 2.5$, $\beta = 10^3$ and $\gamma = 10^{-5}$. As specified in Sec. III-B, α is the reference tracking gain, β is a gain maintaining the CoP close to the center of the foot, and γ is the regularization gain. They were chosen according to their experimental performance. The chosen cost function gives different foot steps compared to [4]. Where the minimization of a cost-to-go criteria in [4] was used, here the robot follows a velocity prescribed by the user and can differ from it locally to avoid an obstacle. This local method runs in real time at a lower level of control which copes with potential evolution of the environment after a first planning.

E. qpOASES solver

The nonlinear problem is linearized analytically (see Sec. III-E) to form a quadratic problem with linear constraints. The off-the-shelf solver qpOases [25] is used to solve the respective QP. This solver is a primal solver implementing an online active set strategy.

VI. Conclusion

In this paper we presented a real-time embedded nonlinear walking pattern generator. Nonlinear inequalities make possible to choose the foot step automatically while considering orientation and local avoidance of convex obstacles. Its performance was demonstrated in two different experiments using the humanoid robot HRP-2. The computational cost of the walking pattern generator is $2\,ms$ on the robot. An extension to our method would be to use a planner in addition to the walking pattern generator.

ACKNOWLEDGMENT

This work was supported by the FP7 Project Koroibot 611909 founded by the European Commission and by the PSPC Project Romeo 2 founded by the French Government. C. Kirches was supported by DFG Graduate School 220 (Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences) funded by the German Excellence Initiative.

References

[1] J. Chestnutt, "Navigation and gait planning," in *Motion Planning for Humanoid Robots*, K. Harada, E. Yoshida, and K. Yokoi, Eds. Springer London, 2010, pp. 1–28.

[2] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *Int. Conf. on Humanoid Robotics (ICHR)*, 2012, pp. 674–679.

[3] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 427–439, 2012.

[4] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *Int. Conf. on Humanoid Robotics (ICHR)*, 2014.

[5] A. Herdt, N. Perrin, and P.-B. Wieber, "Walking without thinking about it," in *IEEE International Conference on Robot Systems (IROS)*, 2010, pp. 190–195.

[6] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body Model-Predictive Control applied to the HRP-2 Humanoid," in *IROS*, 2015.

[7] M. Morisawa, K. Harada, S. Kajita, S. Nakaoka, K. Fujiwara, F. Kanehiro, K. Kaneko, and H. Hirukawa, "Experimentation of humanoid walking allowing immediate modification of foot place based on analytical solution," in *ICRA*, 2007, pp. 3989–3994.

[8] R. Tedrake, S. Kuindersma, R. Deits, and K. Miura, "A closed-form solution for real-time zmp gait generation and feedback stabilization," in *Int. Conf. on Humanoid Robotics (ICHR)*, 2015.

[9] C. G. A. S. Faraji, S. Pouya and A. J. Ijspeert, "Versatile and robust 3d walking with a simulated humanoid robot (atlas): A model predictive control approach." in *ICRA*, 2014.

[10] A. Sherikov, D. Dimitrov, and P.-B. Wieber, "Whole body motion controller with long-term balance constraints," in *Int. Conf. on Humanoid Robotics (ICHR)*, 2014.

[11] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson, "3d walking based on online optimization," in *Int. Conf. on Humanoid Robotics (ICHR)*, 2013.

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[13] I. Mordatch, E. Todorov, and Z. Popovi, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 43, 2012.

[14] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *Int. Conf. on Humanoid Robotics (ICHR)*, 2014, pp. 295–302.

[15] P. M. Wensing and D. E. Orin, "3d-slip steering for high-speed humanoid turns," in *IROS*, 2014.

[16] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robot*, vol. 35, no. 2, 2012.

[17] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *ICRA*, 2003, pp. 1620–1626.

[18] A. Ibanez, P. Bidaud, and V. Padois, *Advance in Robot Kinematics*, 2014, ch. Automatic Optimal Biped Walking as a Mixed-Integer Quadratic Program, pp. 505–516.

[19] K. Nishiwaki and S. Kagami, "Online walking control system for humanoids with short cycle pattern generation," *Int. Jour. of Robotics Research*, vol. 28, no. 6, pp. 729–742, 2009.

[20] P.-B. Wieber, "On the stability of walking systems," in *Proc. of the Inter. Workshop on Humanoid and Human Friendly Robotics*, 2002.

[21] H. G. Bock, M. Diehl, P. Kühl, E. Kostina, J. Schlöder, and L. Wirsching, "Numerical Methods for Efficient and Fast Nonlinear Model Predictive Control," in *Assessment and Future Directions of Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, R. Findeisen, F. Allgöwer, and L. Biegler, Eds., 2007, vol. 358, pp. 163–179.

[22] M. Diehl, *Real-Time Optimization for Large Scale Nonlinear Processes*, ser. Fortschritt-Berichte VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik, 2002, vol. 920.

[23] K. Nishiwaki and S. Kagami, "Walking control on uneven terrain with short cycle pattern generation," in *Int. Conf. on Humanoid Robotics (ICHR)*, 2007, pp. 447–453.

[24] M. Garcia, O. Stasse, J.-B. Hayet, C. Dune, C. Esteves, and J.-P. Laumond, "Vision-guided motion primitives for humanoid reactive walking: decoupled vs. coupled approaches," *Int. Jour. of Robotics Research*, vol. 34, no. 4-5, pp. 402–419, 2015.

[25] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, 2014.