

On the Implementation of Model Predictive Control for On-line Walking Pattern Generation

Dimitar Dimitrov, Pierre-Brice Wieber, Joachim Ferreau, Moritz Diehl

▶ **To cite this version:**

Dimitar Dimitrov, Pierre-Brice Wieber, Joachim Ferreau, Moritz Diehl. On the Implementation of Model Predictive Control for On-line Walking Pattern Generation. ICRA 2008 - IEEE International Conference on Robotics & Automation, May 2008, Pasadena, CA, United States. pp.2685-2690, 10.1109/ROBOT.2008.4543617 . inria-00390492

HAL Id: inria-00390492

https://inria.hal.science/inria-00390492

Submitted on 2 Jun 2009

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.

On the Implementation of Model Predictive Control for On-line Walking Pattern Generation

Dimitar Dimitrov*
mitko@roboresearch.net

Pierre-Brice Wieber*
Pierre-Brice.Wieber@inrialpes.fr

*INRIA Rhône-Alpes
38331 St Ismier Cedex, France

Hans Joachim Ferreau**
joachim.ferreau@esat.kuleuven.be

Moritz Diehl**
moritz.diehl@esat.kuleuven.be

**Department of Electrical Engineering,
K.U. Leuven, Belgium

***Abstract*— This article addresses the real-time implementation issues of a model predictive control based walking pattern generation for a humanoid robot. We approximate the multibody dynamic model with a linear discrete time system, and at each step solve a quadratic program in order to keep the output within a predefined set of constraints. The focus is on creating an efficient framework for forming and solving the underlying optimization problem. For that purpose we develop: a) a reliable guess for the active constraints at optimality; b) a fast way of generating an initial feasible point with respect to the set of constraints for each preview interval; c) a variable discretization sampling time. A simple implementation of a standard primal active set algorithm which exploits a "hot start" is used to demonstrate the advantages of the first point, while the latter one is verified using an existing dual solver.**

I. INTRODUCTION

Walking pattern generation for humanoid robots is a challenging field of research. A variety of methods based on knowledge of the full dynamic characteristics of the system are introduced [1], [2], [3]. Such methods strongly rely on the model accuracy, and usually assume that the precomputed trajectories of the state variables, can be executed in a straightforward way in an error-free environment. In the absence of disturbances, and for short trajectories, such approach proves to be successful, however, it is by no means robust, and its application is limited. In practice, feedback errors are always present as a result of the contact model between the feet and the ground (which in most cases cannot be precisely modeled), hence leading to the necessity of a scheme for on-line walking pattern generation. Such scheme should be able to generate motions that perform tracking of a predefined desired state in a fast and efficient way, while considering the limitations of the system and a feedback for the current dynamical state.

In [4] the authors address this issue by developing a Zero Moment Point (ZMP) preview control scheme, that generates dynamically stable motions for a humanoid robot, by approximating it with a linear inverted pendulum (LIP) [5]. With such notation, they formulate the walking pattern generation problem as a typical infinite horizon Linear Quadratic Regulator (LQR). This approach is advantageous, because it explicitly accounts for the system state, hence considering a tracking error, and leads to fast computation. In the case when the Center of Mass (CoM) of the system is constrained to stay on a plane, with constant height, tuning the gains of the controller amounts to solving a Riccati equation (which can be performed off-line [6]).

The disadvantages of the ZMP preview control scheme outlined above are: a) it does not explicitly account for the constraints of the ZMP; b) in case of disturbances the system response can overshoot the boundaries of the admissible region.

The above two disadvantages are mentioned in [7], and for the case when no constraints are imposed for the ZMP an analytical solution is presented. Furthermore, it is proposed that instead of solving an infinite horizon LQR, a receding horizon LQR with constraints can be used. Which is a different way to say linear model predictive control (LMPC).

In this article, we develop a general framework for the application of a LMPC scheme to on-line walking pattern generation. We build up on the results in [4], [7] and discuss issues related to computation efficiency. It is well-known that in the presence of linear constraints on the input and output, a LMPC problem can be set up as a quadratic program (QP) [9]. Hence, the main focus will be on ways to efficiently identify the solution of a QP, exploiting some of the particular characteristics of a humanoid walking process. In general there are several approaches for obtaining this solution. Two of them are "interior point" methods and "active set" methods. A comparison between them in the context of MPC is made in [10], [11]. We adopt the latter (based on the size of our problem, which will be discussed in Section III-A) and develop a reliable guess for the active constraints at optimality, and a feasible starting point. Both of them can be used to facilitate the optimization solver. In addition we discuss a scheme based on a variable sampling time which can greatly decrease the size of the problem, while keeping a constant preview horizon.

The article is organized as follows: in Section II we briefly outline the LMPC scheme presented in [7] and add a small modification to it. In Section III we focus on developing a guess for the active constraints at the optimal solution and feasible initial point, that will be used as a "hot start" for the QP solver. In Section III-E a variable sampling time scheme is discussed, and its advantages demonstrated using

a dual QP solver [8]. Finally, in Section IV we use an implementation of a standard primal active set algorithm in order to illustrate the benefits of using a "hot start".

II. LMPC SCHEME FOR WALKING PATTERN GENERATION

The following *assumptions* are made:

- $\boldsymbol{a}1$) The footsteps are predefined in advance, and cannot change;
- $\boldsymbol{a}2$) each foot is placed on a flat horizontal surface;
- $\boldsymbol{a}3$) the CoM of the system is constrained to move on a horizontal plane;
- $\boldsymbol{a}4$) the time duration of single- and double-support phases are predetermined.

The ZMP preview control scheme proposed in [4] approximates the dynamics of a humanoid robot with that of a 3D linear inverted pendulum (3D-LIP). Such approximation, and with assumption $\boldsymbol{a}3$, result in a decoupled set of equations governing the motion of the ZMP.

$$z^x = x - \frac{h}{g}\ddot{x}, \qquad z^y = y - \frac{h}{g}\ddot{y}, \tag{1}$$

where, $\boldsymbol{z} = [z^x,\ z^y]^T$ are the coordinates of the ZMP on the flat floor, x, y and h are the coordinates of the CoM (note that the altitude h is assumed constant), g is acceleration due to gravity, and a dot over a variable denotes a time derivative.

The trajectories of both the CoM and the ZMP are discretized then as piecewise cubic polynomials, with constant jerks $\dddot{x}$, $\dddot{y}$ and $\dddot{z}$ over time intervals of lengths T_k ($k = 1, 2, \ldots, N$). Choosing a constant discretization sampling time T_k simplifies the analysis below, nevertheless, for the moment we are going to keep the discussion general, as this proves to play a major role in the on-line application of our controller. Due to the decoupled structure of (1), some of the notations will be made only regarding the forward motion x, keeping in mind that the lateral motion y is identical.

Focusing on the state of the system at times $t_k = \sum_{i=1}^{k} T_i$ with the notations:

$$\hat{\boldsymbol{x}}_k = \begin{bmatrix} x(t_k) \\ \dot{x}(t_k) \\ \ddot{x}(t_k) \end{bmatrix},\ \dddot{x}_k = \dddot{x}(t_k),\ z^x_k = z^x(t_k), \tag{2}$$

the trivial integration of the constant jerk $\dddot{x}_k$ over time intervals of lengths T_k leads to the recursive relationship

$$\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{A}_k \hat{\boldsymbol{x}}_k + \boldsymbol{B}_k \dddot{x}_k \tag{3}$$

while equation (1) leads to

$$z^x_k = \boldsymbol{C}\hat{\boldsymbol{x}}_k \tag{4}$$

where,

$$\boldsymbol{A}_k = \begin{bmatrix} 1 & T_k & T_k^2/2 \\ 0 & 1 & T_k \\ 0 & 0 & 1 \end{bmatrix}; \qquad \boldsymbol{B}_k = \begin{bmatrix} T_k^3/6 \\ T_k^2/2 \\ T_k \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & -h/g \end{bmatrix}$$

Equations (3) and (4) can be used to determine the dynamic behavior of the 3D-LIP system during a given interval of time, for a sequence of control inputs $\{\dddot{x}_k\}$. This is a reasonable approximation for the complicated dynamics of a humanoid robot, and provides a computationally efficient way to estimate whether the ZMP of the real system is in the convex hull of the predefined reference footsteps. Hereafter, this is assumed to be the criteria which guarantees that the system will not tip over and fall. It can be expressed as follows:

$$\boldsymbol{E}_k \boldsymbol{z}_k + \boldsymbol{e}_k \geq 0 \tag{5}$$

where, the matrix $\boldsymbol{E}_k \in R^{p^k \times 2}$, and vector $\boldsymbol{e}_k \in R^{p^k}$ describe the geometry of the convex hull (with p^k edges) formed by the feet during the k^{th} sampling period.

As noted in [4], $(\boldsymbol{A}, \boldsymbol{B})$ in equation (3) is controllable, hence the tracking problem of a reference output can be implemented as an infinite horizon LQR, without explicitly considering the ZMP constraints in (5). Hence, minimizing the jerks $\dddot{x}_k$, $\dddot{y}_k$, while maintaining the ZMP as close as possible to the reference one. This approach is suitable for high frequency on-line walking pattern generation, and it guarantees convergence towards the reference state. Nevertheless, it is susceptible to perturbations or even feedback errors (depending on the design of the controller [9]). The reason follows from the fact that after a disturbance, the system response can overshoot the boundaries of the admissible region, hence equation (5) can be easily violated.

For dealing with this problem, the authors of [7] proposed to solve a receding horizon LQR, explicitly considering the linear constraints in (5). It is well known that this amounts to solving a quadratic programming problem at each iteration [9]. The idea behind LMPC, can be summarized as follows: (i) at time t_k and for the current state, determine an optimal control input by solving a QP problem over a fixed future interval, considering the current and future constraints; (ii) apply only the first step of the optimal control; (iii) measure (estimate) the state reached at (t_{k+1}) and repeat step (i).

In order to form a QP problem, equation (3) is iterated N times and combined with N versions of (4), relating at once N values of the jerk of the CoM to N values of $\boldsymbol{z}$.

$$\begin{aligned} z^x_{k+1} &= \boldsymbol{C}\boldsymbol{A}_k \hat{\boldsymbol{x}}_k + \boldsymbol{C}\boldsymbol{B}_k \dddot{x}_k \\ z^x_{k+2} &= \boldsymbol{C}\boldsymbol{A}_{k+1}\boldsymbol{A}_k \hat{\boldsymbol{x}}_k + \boldsymbol{C}\boldsymbol{A}_{k+1}\boldsymbol{B}_k \dddot{x}_k + \boldsymbol{C}\boldsymbol{B}_{k+1} \dddot{x}_{k+1} \\ &\cdots \end{aligned} \tag{6}$$

With the notation:

$$\boldsymbol{Z}^x = \begin{bmatrix} z^x_{k+1} \\ \vdots \\ z^x_{k+N} \end{bmatrix}; \qquad \boldsymbol{U}^x = \begin{bmatrix} \dddot{x}_k \\ \vdots \\ \dddot{x}_{k+N-1} \end{bmatrix},$$

the above relations can be expressed in a more compact way:

$$\boldsymbol{Z}^x = \boldsymbol{P}_s\, \hat{\boldsymbol{x}}_k + \boldsymbol{P}_u\, \boldsymbol{U}^x \tag{7}$$

For convenience the equation for the lateral motion will be written as well, with corresponding meaning of the symbols:

$$\boldsymbol{Z}^y = \boldsymbol{P}_s\, \hat{\boldsymbol{y}}_k + \boldsymbol{P}_u\, \boldsymbol{U}^y \tag{8}$$

Equations (7) and (8) determine the position of the ZMP for a period in the future with length $T^p = \sum_{k=1}^{N} T_k$. This period is referred to as *preview window*. Following from assumptions $\boldsymbol{a}1$ and $\boldsymbol{a}4$, an augmented constraint matrix $\boldsymbol{E}$ and vector $\boldsymbol{e}$ can be defined for the entire *preview window*, leading to an augmented constraint equation:

$$\boldsymbol{E}\boldsymbol{Z} + \boldsymbol{e} \geq 0 \tag{9}$$

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{Z}^x \\ \boldsymbol{Z}^y \end{bmatrix}; \qquad \boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}^x \\ \boldsymbol{U}^y \end{bmatrix}$$

If (7) and (8) are substituted into (9) this leads to a linear constraint for the control variable $\boldsymbol{U}$, which if satisfied will result in a "stable" motion:

$$\bar{\boldsymbol{E}}\boldsymbol{U} + \bar{\boldsymbol{e}} \geq 0 \tag{10}$$

For the formulation of a quadratic program we set the following objective function:

$$\min_{\boldsymbol{U}} \ \frac{1}{2}(\boldsymbol{U}^2 + \alpha\dot{\boldsymbol{S}}^2 + \beta(\boldsymbol{S} - \boldsymbol{S}_{ref})^2) \tag{11}$$

$$\boldsymbol{S} = [x_{k+1}, \ldots, x_{k+N}, y_{k+1}, \ldots, y_{k+N}]^T$$

where, α and β are positive scalar gains, and $\boldsymbol{S}_{ref}$ contains reference values for the position of the CoM during the current *preview window*. The latter two terms in (11) act like a spring and damper to the profile of the CoM, and can be used to influence its behavior within the constraints. This can be beneficial in cases of disturbances, and at the start and end of the trajectory. A standard choice for $\boldsymbol{S}_{ref}$ is in the center of the foot/feet. Equation (11), can be rewritten in the following standard form:

$$\min_{\boldsymbol{U}} \ \frac{1}{2}\boldsymbol{U}^T\boldsymbol{H}\boldsymbol{U} + \boldsymbol{U}^T\boldsymbol{g} \tag{12}$$

with $\boldsymbol{H} \in R^{N\times N}$ and $\boldsymbol{g} \in R^{N\times 1}$ being the Hessian matrix and gradient vector of the objective function. Equations (12) and (10) define a standard QP problem with inequality constraints. With the current problem, the Hessian matrix $\boldsymbol{H}$ is strictly positive definite, and since a set of linear constraints form a (polyhedral) convex set, there exists a unique global minimizer $\boldsymbol{U}^*$ [12].

III. Solving the optimal problem on-line

A. Size of the QP (constant sampling)

The size of the QP defined by (12) and subject to the constraints (10) depends on N, which is related to the length T^p of the *preview window* and the choice of the sequence of sampling times $\{T_k\}$. In the case of HRP2 humanoid robot, for stability of the control scheme, we need at least $T^p = 0.7$ s [7]. However, for robustness and to reduce tracking errors $T^p = 1.5$ s is commonly used. For the moment let us assume constant sampling intervals $T_k = 20$ ms with $k = 1, 2, \ldots, 75$. This leads to a QP with 75 x 2 = 150 state variables.

The number of inequality constraints m^i (that appear in (10)) for the *preview window* T^p depends on the geometry of the feet, and the time duration of the single- (T^s) and double-support (T^d) phases. A typical case is depicted in Fig. 1 ($T^s = 0.7$ s, $T^d = 0.1$ s), where the circles represent what the profile of the ZMP would be if all the controls $\boldsymbol{U}^*$ are applied[1] in the case without perturbation. Let us denote with c_k^j the j^{th} constraint of equation (5), with $j = 1, 2, \ldots, p^k$. Then $m^i = \sum_{k=1}^{N} p^k$. For the *preview window* in Fig. 1, $m^i = 67$ x 4+8 x 6 = 316, where each single-support phase is defined by four constraints and each double-support, by six. Furthermore, it can be observed that in the case without perturbation the number of active constraints (constraints that hold as equalities) is $m^a = 5$ (indicated by arrows). In the case with perturbation (depicted with triangles) $m^a = 12$.

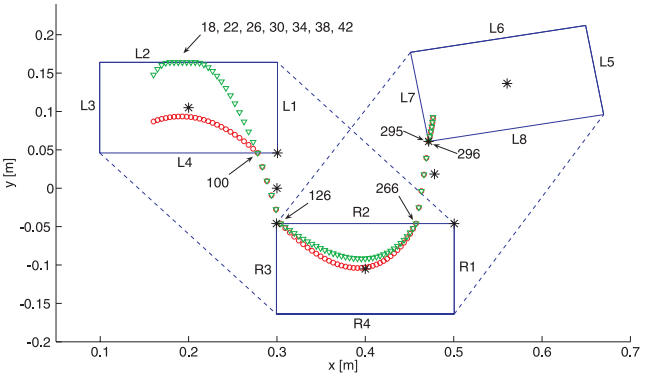


Fig. 1. Constraints for a generic *preview window*. Solid lines and dashed lines represent single- and double-support constraints, respectively. The ZMP profile is denoted by circles and triangles corresponding to cases without and with perturbation, respectively. The remaining symbols are defined in Sections III-C and III-D.

B. Choice of optimization algorithm

The choice of an optimization algorithm that can efficiently solve a sequence of quadratic programs as the one outlined in the previous subsection, is not unique. Fast and reliable solvers based on *interior point* and *active set* strategies, are available. There has been a great deal of research related to the application of both approaches in the context of MPC [10], [11], [13], [14]. In general, the choice depends mostly on: (i) the number of active constraints and state variables; (ii) whether a "hot start" is available; (iii) whether there is a cheap way to determine an initial feasible point. Points (i) and (ii) are essential for choosing between *interior point* and *active set* QP solvers. The reason is that, on problems where the number of active constraints remains small, *active set* solvers can identify them quickly with little computational effort. While in the case of *interior point* methods, one pays a "fixed price" regardless of the number of active constraint. Furthermore, *active set* methods typically gain more from "hot starting" (for reasons that are not yet fully understood [10]). That is why they are best suited for our problem.

Point (iii) influences the choice between alternative *active set* strategies. A primal solver for instance, requires starting

[1]Note that, only the first control input is used by the MPC scheme.

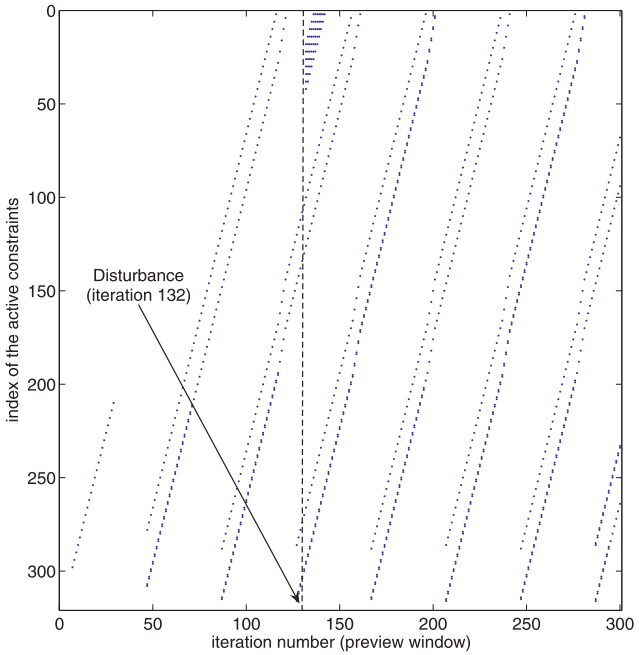


Fig. 2. Sliding active constraints.

with a feasible point (satisfying equation (10)). This could be considered as a drawback, since determining such a point can be costly. For the current application, however, a feasible starting point can be calculated in a straightforward and very efficient way (to be derived in Section III-D), hence rendering the utilization of both primal and dual solvers possible.

C. Guess for the active constraints

Finding the solution of a QP problem as the one defined in (12) and (10) in the case when the set of active constraints at optimality ($\boldsymbol{A}_c(\boldsymbol{U}^*)$) is known, is equivalent to solving (12) with equality constraints:

$$\bar{\boldsymbol{E}}_i\boldsymbol{U} + \bar{\boldsymbol{e}}_i = 0, \qquad i \in \boldsymbol{A}_c(\boldsymbol{U}^*) \tag{13}$$

which amounts to solving a linear system of equations that has a unique solution [12].

In general, prior knowledge of $\boldsymbol{A}_c(\boldsymbol{U}^*)$ is not available, and its identification is the main challenge facing *active set* schemes. The algorithm takes an iterative approach, and generates a sequence of feasible iterates. At each of them a certain subset of the constraints (10) is assumed to be active (referred to as the *working set* $\boldsymbol{W}$), and an equality constrained QP is solved (for a detailed description of an *active set* strategy see [12], p. 462).

The smaller the number of active constraints that need to be identified, the faster the algorithm will converge to the optimal solution $\boldsymbol{U}^*$. A standard "cold" (i.e., no prior information) start, assumes an empty initial working set ($\boldsymbol{W}_0$). If, however, all (or a subset of the) active constraints at $\boldsymbol{U}^*$ are known in advance, they can be included directly in $\boldsymbol{W}_0$ (if their gradients are linearly independent), leading to faster algorithm convergence.

It is possible to form a reliable guess for $\boldsymbol{A}_c(\boldsymbol{U}^*)$ as a result of assumption $\boldsymbol{a}4$, and due to the distinct characteristics of the current problem. The following three *observations* play an important role:

- $\boldsymbol{o}1$) If c_k^j is an active constraint for *preview window* i then (with few exceptions) c_{k-1}^j is an active constraint for *preview window* $i+1$;
- $\boldsymbol{o}2$) with no perturbations, the active constraints appear during the change of the support foot;
- $\boldsymbol{o}3$) when a new set of single-support constraints becomes visible in the *preview window*, at least two new constraints become active.

Observation $\boldsymbol{o}1$ implies that the active constraints are "sliding" with a step p^k. This behavior is depicted in Fig. 2. The x-axis represents 300 preview iterations for a typical set of footprints, and the y-axis denotes the indexes of the resultant active constraints. It can be observed that the evolution pattern of the active constraints is very consistent, hence, predictable.

At iteration 132 the system is perturbed, and an increased number of active constraints can be observed until iteration 142. However, even in this case (except for the iteration when the perturbation occurred) the "sliding" behavior is preserved. The interpretation of the increased number of active constraints is depicted with triangles in Fig. 1, where the ZMP (resulting from the application of all controls in $\boldsymbol{U}^*$) reaches the boundary of the admissible set.

As numerical example for the utilization of the "sliding" behavior, we will consider the case with no perturbations in Fig. 1, where the set of active constraints for the current *preview window* is: $\boldsymbol{A}_c^{132} = [100, 126, 266, 295, 296]$. The guess for the active set of the next iteration can be formed as: $\boldsymbol{G} = [96, 122, 262, 291, 292]$, which coincides with the real set ($\boldsymbol{A}_c^{133}$).

From a practical point of view, for a "reasonable" set of predefined foot placements, and in a case with no perturbations, it seems natural to distinguish between two type of foot constraints: *regular* and *irregular*. The first type ($L4$, $R2$ and $L8$ in Fig. 1) are easy to handle. In the case of observation $\boldsymbol{o}3$, the last two *regular* constraints in the *preview window* will be always active. Once activated, observation $\boldsymbol{o}1$ is valid until the *regular* constraint is dropped from the *preview window*. The behavior of *irregular* constraints on the other hand ($L1$, $R3$, $R1$ and $L7$ in Fig. 1) is more difficult to predict. It depends on the choice of α and β in (11), hence, an efficient guessing scheme needs to be "tuned" for any new set of gains.

Guessing the additional active constraints $\boldsymbol{A}_c^{dist}$ for the first QP after a disturbance cannot be achieved based on the above *observations*. $\boldsymbol{A}_c^{dist}$ depends on the direction and magnitude of the disturbance, which in general should be assumed unknown. In such a case, we can set a maximum time for solving a QP, and try to identify the solution within the time limits. If the number of constraints in $\boldsymbol{A}_c^{dist}$ is large, identifying them and finding $\boldsymbol{U}^*$ within a given time might not be possible. In such case, we can "stop" at a *suboptimal* solution with an estimated set of active constraints $\boldsymbol{A}_{c-est}^{dist}$. During the next QP, the standard guess $\boldsymbol{G}$ in combination

with $\boldsymbol{A}_{c-est}^{dist}$ can be included in $\boldsymbol{W}_0$, and so forth, until optimality is reached [15].

D. Feasible initial point

Once a reliable guess for the active constraints has been formed, generating a corresponding feasible initial point $\boldsymbol{U}_0$ is straightforward. From equations (7) and (8), $\boldsymbol{U}_0$ can be expressed as a function of the current state and a still unknown vector of feasible zero moment points for the next *preview window* $\boldsymbol{Z}^f$:

$$\boldsymbol{U}_0 = \bar{\boldsymbol{P}}_u^{-1} \bar{\boldsymbol{P}}_s \begin{bmatrix} \hat{\boldsymbol{x}}_k \\ \hat{\boldsymbol{y}}_k \end{bmatrix} + \bar{\boldsymbol{P}}_u^{-1} \boldsymbol{Z}^f \tag{14}$$

where,

$$\bar{\boldsymbol{P}}_u = \begin{bmatrix} \boldsymbol{P}_u & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_u \end{bmatrix}; \qquad \bar{\boldsymbol{P}}_s = \begin{bmatrix} \boldsymbol{P}_s & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_s \end{bmatrix}$$

The matrix $\bar{\boldsymbol{P}}_u \in R^{2N \times 2N}$ is of full rank, and its inverse can be precomputed off-line together with the product $\bar{\boldsymbol{P}}_u^{-1} \bar{\boldsymbol{P}}_s$, as will be discussed in the next subsection.

First, let us consider the case with zero active constraints. Forming a feasible $\boldsymbol{Z}^f$ with respect to the known constraints for the next *preview window*, amounts to choosing a point (for instance) in the middle of each constraint. This guarantees that equation (14) will lead to a feasible $\boldsymbol{U}_0$.

In case of a nonempty guess $\boldsymbol{G}$, its elements should be a subset of the active constraints at $\boldsymbol{U}_0$. Forming such $\boldsymbol{U}_0$ can be easily achieved by choosing each entry in $\boldsymbol{Z}^f$ that correspond to an active constraint in $\boldsymbol{G}$ on the same constraint. Example are depicted with stars in Fig. 1.

E. Variable sampling time

With the constant sequence of sampling times $\{T_k\}$ adopted in Section III-A, the Hessian matrix $\boldsymbol{H}$ of the QP is constant for the entire motion of the system. This is advantageous from the viewpoint of computational efficiency, since $\boldsymbol{H}$ can be formed and pre-factorized off-line. Furthermore, for each *preview window*, computational savings can be gained when forming the gradient $\boldsymbol{g}$ and constraints in (10), since some of the multiplications appearing inside contain the constant matrices $\boldsymbol{P}_s$ and $\boldsymbol{P}_u$, hence can be precomputed off-line as well.

Using constant sampling time for the entire *preview window*, however, can impose limitations. Two of them will be discussed below:

1) Fast sampling: In many cases, due to the characteristics of the system, smaller sampling times could be required by the control module in order to obtain a desired behavior. In the presence of perturbations is could be desirable to estimate the state of the system and update the control inputs with high frequency. If the required frequency of computation is 5 ms for example, discretization of the *preview window* with constant $T_k = 5$ ms, would result in 600 state variables ($T^p = 1.5$ s), hence leading to a QP problem not suitable for on-line implementation. Instead, we can use constant $T_k = 20$ ms and additionally divide only the first interval into four sub-intervals of length 5 ms. Hence,

- at time $t = 0$ ms we obtain the current state of the robot $\boldsymbol{S}_0$, and solve the following QP:

$$\min_{\boldsymbol{U}} \ \frac{1}{2}\boldsymbol{U}^T \boldsymbol{H} \boldsymbol{U} + \boldsymbol{U}^T \boldsymbol{g}(\boldsymbol{S}_0) \tag{15a}$$

$$\bar{\boldsymbol{E}}(\boldsymbol{S}_0)\boldsymbol{U} + \bar{\boldsymbol{e}}(\boldsymbol{S}_0) \geq 0 \tag{15b}$$

- after applying $\boldsymbol{U}_0^*$, we obtain the new state $\boldsymbol{S}_1$ at time $t = 5$ ms and solve (with additional equality constraint):

$$\min_{\boldsymbol{U}} \ \frac{1}{2}\boldsymbol{U}^T \boldsymbol{H} \boldsymbol{U} + \boldsymbol{U}^T \boldsymbol{g}(\tilde{\boldsymbol{S}}_0) \tag{16a}$$

$$\bar{\boldsymbol{E}}(\tilde{\boldsymbol{S}}_0)\boldsymbol{U} + \bar{\boldsymbol{e}}(\tilde{\boldsymbol{S}}_0) \geq 0 \tag{16b}$$

$$\boldsymbol{U}_0 = \boldsymbol{U}_0^* \tag{16c}$$

  where, $\tilde{\boldsymbol{S}}_0 = f(\boldsymbol{S}_1, -\boldsymbol{U}_0^*, \boldsymbol{\eta})$ is the estimated state at time $t = 0$ ms, starting from $\boldsymbol{S}_1$ and using $-\boldsymbol{U}_0^*$, and considering noise $\boldsymbol{\eta}$ (if $\boldsymbol{\eta} = 0$, $\tilde{\boldsymbol{S}}_0 = \boldsymbol{S}_0$).
- next apply the optimal inputs $\boldsymbol{U}_0^*$ and $\boldsymbol{U}_1^*$ to reach the new state 5 ms later, and so forth until $t = 20$ ms. Then clear the additional equality constraints, and repeat the above steps for the next time interval of 20 ms.

The above formulation solves essentially the same problem four times, if $\boldsymbol{\eta} = 0$. The reasoning comes from the fact that if at any of the 5 ms sampling intervals there is a perturbation, the system will be able to react promptly to it. This is in contrast with the case when $T_1 = 20$ ms is used, where in the worst scenario, the system will react with a 20 ms delay. The auxiliary equality constraints in the above formulation, prevent us from dropping the already passed 5 ms sampling intervals, hence keeping the Hessian matrix and N constant.

2) Reduced number of states: Based on observation $\boldsymbol{o}2$ in Section III-C, it is desirable to chose the sequence $\{T_k\}$ in such a way, that the times of support foot change are not "skipped". However, with "reasonably long" constant sampling intervals this would be possible only if additional restrictions are imposed on the time duration of the single- and double-support phases.

Inspired by observation $\boldsymbol{o}2$, we can overcome the above limitation and reduce the state variables of the QP by designing a variable sampling time sequence by following the three criteria below:

- $\boldsymbol{c}1$) The variation of $\{T_k\}$ is finite, and follows a predefined circular pattern;
- $\boldsymbol{c}2$) the times of support foot change are not skipped;
- $\boldsymbol{c}3$) T^p, T^s, T^d and N are kept constant;

Criterion $\boldsymbol{c}1$ limits the number of different sequences $\{T_k\}$ that can be used. This results in a finite number of Hessian matrices, that can be formed and pre-factorized off-line. For $T^p = 1.5$ s ($T^s = 0.7$ s, $T^d = 0.1$ s) and a set of sampling intervals $\{20, 40, 60\}$ ms, a circular pattern for $\{T_k\}$ can be obtained using 40 sequences, with each sequence containing 32 sampling times. Hence, resulting in 64 state variables. Fig. 3 depicts a comparison of QP computation time between the cases when constant and variable sampling times are used. The simulation is implemented in C++ and

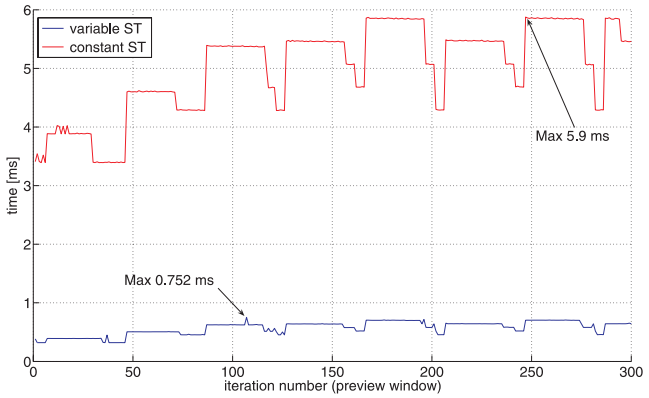


Fig. 3. QP computation time for the cases with constant (150 states, red line) and variable sampling times (64 states, blue line).

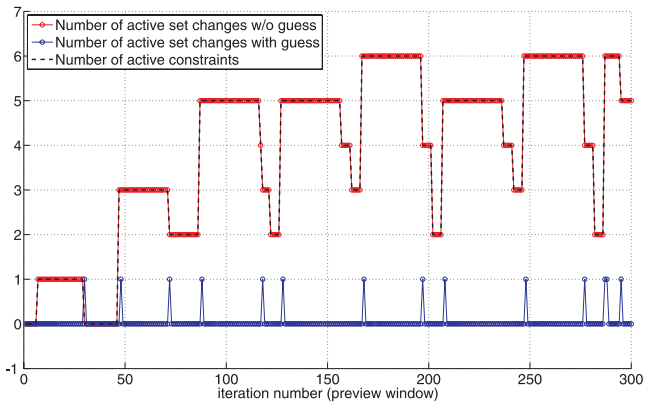


Fig. 4. Number of active set changes to convergence of the QP solver with and without guess, and number of active constraints.

performed on a PC with Intel Core 2 Duo Processor, 2GHz. Optimization flag $-O2$ is used. The simulation conditions are identical to the ones used to generate Fig. 2 in the case without disturbance with gains $\alpha = 10, \beta = 2$. The dual QP solver QL [8] was utilized.

IV. Efficiency of the "Hot Start"

In order to demonstrate the computational savings as a result of the discussion in Section III-C, we implemented a standard primal active set algorithm and compared cases with and without "hot start". The comparison is made on the number of active set changes necessary for the algorithm to converge to the optimal solution. Fig. 4 depicts the results of a simulation with conditions identical to the one in the previous section.

The case when a guess is supplied to the solver is depicted with a blue line. With a perfect guess no active set changes are made. As can be seen, for most of the preview windows, the guessing scheme is able to identify all the active constraints at optimality. In less than 5% of the cases, a single wrong guess is made, which results in one active set change. It was confirmed that the inacuracy of the guess was due to *irregular* constraints (see Section III-C). When a guess is not supplied, the number of active set changes coincide with the number of active constraints at $\boldsymbol{U}^*$ (red and black line). Note however, that in general, for the identification of n^a active constraints a primal algorithm might need more than n^a active set changes [12].

V. Conclusion

In this paper, we presented the results from the analysis of the application of linear model predictive control for walking pattern generation for humanoid robots. The focus was on creating an efficient framework for forming and solving the underlying quadratic programming problem. For that purpose we presented a way to make a reliable guess for the active constraints at optimality, based on the observation that the active constraints "slide" with the change of the *preview window*. It was noted that the "sliding" behavior is retained even after the system state is perturbed. Furthermore, we demonstrated a fast way of generating an initial feasible point with respect to the set of constraints for each preview interval, which makes possible the utilization of any *active set* solver. Finally, a discussion of the application of a variable sampling time was made. It was shown that the utilization of variable sampling time can greatly reduce the size of the optimization problem, and can be useful when update of the control inputs is required with high frequency.

References

[1] K. Hirai, M. Hirose, Y. Haikawa, and T. Takaneka, "The development of Honda humanoid robot," in *Proc. of the IEEE Int. Conf. on Robot. & Automat.,* pp. 1321-1326, 1998.

[2] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi and K. Tanie, "Planning walking patterns for a biped robot," *IEEE Trans. on Robotics and Automation,* Vol.17, No.3, June, pp.280-289, 2001.

[3] J. Park, and H. Cho, "An on-line trajectory modifier for the base link of biped robots to enhance locomotion stability," in *Proc. of the IEEE Int. Conf. on Robot. & Automat.,* pp.3353-3358, 2000.

[4] 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 *Proc. of the IEEE Int. Conf. on Robot. & Automat.,* pp.1620-1626, 2003.

[5] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Yokoi, and H. Hirukawa, "A realtime pattern generator for biped walking," in *Proc. of the IEEE Int. Conf. on Robot. & Automat.,* pp.31-37, 2002.

[6] K. Nishiwaki, and S. Kagami, "High frequency walking pattern generation based on preview control of ZMP," in *Proc. of the IEEE Int. Conf. on Robot. & Automat.,* pp.2667-2672, 2006.

[7] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *Proc. of IEEE-RAS Int. Conf. on Humanoid Robots,* pp.137-142, 2006.

[8] K. Schittkowski, "QL: A Fortran code for convex quadratic programming - User's guide," *Department of Mathematics, University of Bayreuth*, Report, Version 2.11, 2005.

[9] G. C. Goodwin, M. M. Seron, and J. A. De Doná, "Constrained control and estimation," *Springer, 1st edition*, September, 2004.

[10] S. Wright, "Applying new optimization algorithms to model predictive control," in *Proc. of CPC-V*, 1996.

[11] R. A. Bartlett, A. Wächter, and L. T. Biegler, "Active set vs. interior point strategies for model predictive control," in *Proc. of the American Control Conference,* pp.4229-4233, June, 2000.

[12] J. Nocedal, and S. J. Wright, "Numerical optimization," *Springer Series in Operations Research, 2nd edition*, 2000.

[13] C. Rao, J. B. Rawlings, and S. Wright, "Application of interior point methods to model predictive control," *J. Opt. Theo. Applics.,* pp. 723-757, 1998.

[14] I. Das, "An active set quadratic programming algorithm for real-time predictive control," *Optimization Methods and Software,* Vol.21, No.5, pp.833-849, 2006.

[15] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *Int. J. of Robust and Nonlinear Control*, (in press).