Planning COM trajectory with variable height and foot position with reactive stepping for humanoid robots

Kirill Van Heerden

***Abstract*—This paper presents a trajectory generator for humanoid robots that can simultaneously consider 3D position tracking of the center-of-mass (COM), reactive stepping (foot positions are non-constant), zero-moment-point (ZMP) constraints as well as position constraints.**

The variable COM height leads to nonlinear ZMP constraints. This work demonstrates that depending on how strict the constraints are, it is possible to generate 1.6 second long trajectories in less than 5 ms on a 3.4 Ghz CPU.

This is done by expressing the problem as a quadratically constrained quadratic program (QCQP) and solving it via sequential quadratic programing (SQP). Computation time is improved by providing the analytical derivatives of all constraints. In particular the nonlinear ZMP constraint is expressed in a quadratic form and its derivative is provided.

***Index Terms*—Model Predictive Control, Humanoid Robot, ZMP**

I. Introduction

Humanoid robots may be useful in a wide area of applications such as elderly care, disaster management, industrial work, etc.

A number of advances have been made in this field, the robots from Honda, Toyota and Boston Dynamics [1], [2] have caught the public attention. However the trajectory generators for humanoid robots are still limited with regards to variable center-of-mass height trajectories.

The reason for this is that the zero-moment-point (ZMP) equation, which is used to create dynamically feasible trajectories, becomes nonlinear when the center-of-mass (COM) height is non-constant, thus it is difficult to generate such trajectories on-line in real-time.

Trajectory generation methods such as [3], [4], [5] are able to realize a variety of linear constraints (such as ZMP constraints, kinematic constraints, etc) while planning COM and foot trajectories (reactive stepping) in real-time. These methods are based on quadratic programming (QP) and they produce an optimal trajectory by minimizing a quadratic goal function that is subjected to a variety of motion constraints.

However, a limitation of the QP based methods is that they are only able to handle linear constraints, thus they cannot directly handle the nonlinear ZMP equation that arises due to variable COM height.

Kirill is with the Department of Robotics, Ritsumeikan University, Kusatsu, Japan, e-mail: kirillatfc-.-ritsumei-.-ac-.-jp.

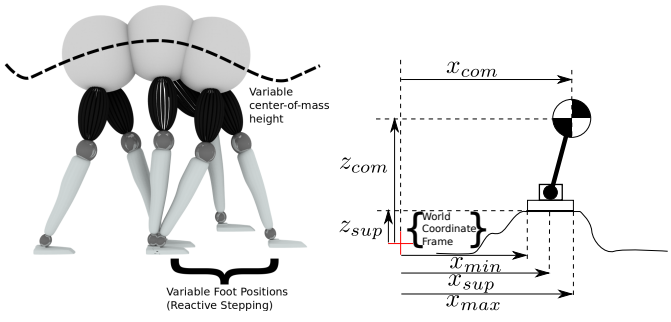
This work was supported by JSPS KAKENHI Grant Number 14536649.

Recently there are some specialized methods [6], [7] that allow a variable COM height to be planned in real-time but these methods have difficulty in dealing with arbitrary constraints and they have not yet demonstrated reactive foot placement.

Sequential Quadratic Programming (SQP) is a popular tool in the aerospace industry for dealing with non-linear trajectory generation problems, it functions by iteratively linearizing the goal function and constraints then solving the resulting quadratic program. There are some works on SQP for humanoid robots [8], [9], [10]. In particular [8] proposes a method of using SQP to solve a model-predictive-control (MPC) problem involving variable COM height. However, this method did not deal with reactive stepping and computational performance was not real-time. In general SQP methods are often considered to be slow and limited to off-line computations.

In this paper, it will be shown that SQP can be used to generate humanoid robot trajectories with variable COM height in less than 5 milliseconds. This is done by expressing the nonlinear ZMP constraints analytically in a quadratic form and then differentiating them to find the closed form expression of the constraint gradients. These analytic gradients reduce the effort required by the SQP solver compared to the finite difference method that would have to be used in the absence of such derivatives.

It should be pointed out that this work is limited to trajectory generators but for a complete humanoid robot



(a) Concept: Trajectory generator for variable COM height and reactive stepping

(b) Visualization of Parameters

Figure 1.

system, other computational costs such as inverse kinematics should also be considered.

This paper extends the author's prior work [11] by including reactive foot positioning (said in another way, adaptive foot positioning), these modifications required the analytical expression of the ZMP equation to be re-derived. Furthermore, the code was reimplemented in C++ to demonstrate computation times under 5 ms. The task realized by this project is conceptually depicted in fig 1a.

II. Theory

In order for a humanoid robots motion to be physically realizable, it is necessary to keep the ZMP within its support polygon.

$$x_{min}(j) < x_{zmp}(j) - x_{sup}(j) < x_{max}(j) \tag{1}$$

Here j, x_{zmp}, x_{sup} ,x_{max} and x_{min} are the current time-step, the ZMP position, center of the support polygon, the upper bound of the support polygon and the lower bound of the support polygon, respectively.

The ZMP can be expressed as follows by assuming that angular momentum can be ignored due to a constant upper body orientation and legs that have negligible mass compared to the upper body.

$$x_{zmp}(j) = x_{com}(j) - \frac{z_{com}(j) - z_{sup}(j)}{\ddot{z}_{com}(j) + g}\ddot{x}_{com}(j) \tag{2}$$

Here x_{com}, $\ddot{x}_{com}$, z_{com}, $\ddot{z}_{com}$, x_{sup}, z_{sup} and g are the COM position and acceleration in the x and z axes, the center of the support polygon in x and z axes and the gravity constant, respectively. These parameters are also visualized in fig. 1b and it should be noted that this equation is slightly different to the normal ZMP equation because of the variable foot height.

By substituting (2) into (1) and only considering the upper bound of the ZMP constraint, the following quadratic equation can be derived.

$$\begin{aligned} &x_{com}(j)\ddot{z}_{com}(j) + x_{com}(j)g - \ddot{x}_{com}(j)z_{com}(j) \\ &\quad +\ddot{x}_{com}(j)z_{sup}(j) - x_{sup}(j)\ddot{z}_{com}(j) \\ &-x_{sup}(j)g - x_{max}(j)\ddot{z}_{com}(j) - x_{max}(j)g < 0 \end{aligned} \tag{3}$$

The lower bound of the ZMP constrain can also be considered but this will be discussed later.

It is possible to parameterize a COM trajectory in terms of its jerk [12], [3], [4]. If this jerk parameterized trajectory was plugged into (3), then the left hand side of the constraint in (3) can be expressed compactly as the following quadratic function for time periods $i \in \{0, \ldots, N\}$ (N is the total number of time-steps):

$$h_i(\chi) = \chi^T P_i \chi + p_i^T \chi + \gamma_i, \tag{4}$$

here χ is a state vector that encodes the jerks and the support polygon center at each time period.

As long as P_i is symmetric, the gradient and Hessian of this quadratic function is known from matrix calculus to be the following:

$$\nabla_\chi h(\chi) = 2P_i\chi + p_i \quad , \quad \nabla^2_\chi h(\chi) = 2P_i \tag{5}$$

It has been demonstrated in other works that model predictive control can be applied to find the optimal set of jerks as well as the support polygon positions by minimizing an appropriate quadratic equation [3], [4]. Thus combining this quadratic goal from MPC with the quadratic ZMP constraints as well as other constraints related to the allowable foot positions (to be discussed later) leads to a quadratically constrained quadratic program (QCQP):

$$\begin{aligned} \min_{\chi}: \quad & f_{goal}(\chi) \\ \texttt{s.t}: \quad & h_i(\chi) < 0 \\ & i \in \{1, \ldots, k\} \end{aligned} \tag{6}$$

Here $f_{goal}(\chi)$ represents the previously mentioned quadratic goal function and k represents the total number of constraints.

This QCQP can be solved with SQP and this implementation would allow a COM trajectory sequence as well as foot positioning to be generated in 3D while also keeping the ZMP within the support polygon. Also, the gradients and Hessians of the quadratic functions f_{goal} and h_i that are required by SQP can easily be found via (5) and can be used to speed up the SQP calculation.

Thus, to facilitate this process, the method of describing a COM trajectory as a number of jerks will be explained in the next section.

A. Parameterizing COM Trajectory with jerks

First consider a COM state vector, $\bar{x}_{com}(k) = [x_{com}(k)\, \dot{x}_{com}(k)\, \ddot{x}_{com}(k)]$. Then, the discrete time relationship between a series of jerks and the COM state and COM position can be expressed as follows.

$$\begin{aligned} \bar{x}_{com}(k+1) &= A\bar{x}_{com}(k) + B\dddot{x}_{com}(k), \\ x_{com}(k) &= C_p\bar{x}_{com}(k) \end{aligned} \tag{7}$$

where A, B and C_p are as follows:

$$A = \begin{bmatrix} 1 & \Delta T & \frac{1}{2}\Delta T^2 \\ 0 & 1 & \Delta T \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{6}\Delta T^3 \\ \frac{1}{2}\Delta T^2 \\ \Delta T \end{bmatrix}, \tag{8}$$

$$C_p = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \tag{9}$$

and ΔT is the integration time.

MPC techniques can be applied to the linear system in (7) to form the following discrete system for N time steps.

$$\bar{X}_{com} = \begin{bmatrix} x_{com}(1) \\ x_{com}(2) \\ \vdots \\ x_{com}(N) \end{bmatrix} = P_{ps}\bar{x}_0 + P_{pu}\dddot{\bar{X}}_{com}, \tag{10}$$

where $\bar{x}_0$ is the initial state vector, a known constant and the matrices P_{ps} and P_{pu} are defined as follows

$$P_{ps} = \begin{bmatrix} C_pA \\ C_pA^2 \\ \vdots \\ C_pA^N \end{bmatrix}, \tag{11}$$

$$P_{pu} = \begin{bmatrix} C_pA^0B & 0 & \dots & 0 \\ C_pA^1B & C_pA^0B & \dots & 0 \\ \vdots & & \ddots & \\ C_pA^{N-1}B & C_pA^{N-2}B & \dots & C_pB \end{bmatrix}. \tag{12}$$

From (10) it can be seen that there is a linear relationship between the jerks and the COM positions of the robot.

Similar systems to (10) can be constructed for the y and z axes, i.e $\bar{Y}_{com}$ and $\bar{Z}_{com}$ and in this case they would use the same P_{ps} and P_{pu} matrices. Likewise for velocity, $\dot{\bar{X}}_{com}$, and acceleration, $\ddot{\bar{X}}_{com}$, systems can be constructed by replacing the selection matrix C_p with

$$C_v = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \quad C_a = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{13}$$

for the velocity and acceleration systems, respectively.

This process would produce the matrices P_{vs}, P_{vu}, P_{as} and P_{au} for the velocity and acceleration systems, respectively.

So, far only the x axis was considered but the y and z axes can be considered in a similar manner with jerks $\dddot{\bar{Y}}_{com}$ and $\dddot{\bar{Z}}_{com}$ in the y and z axes respectively.

Now in order to consider all axes simultaneously, lets define the state vector,

$$\chi = [\dddot{\bar{X}}_{com}^T \; \dddot{\bar{Y}}_{com}^T \; \dddot{\bar{Z}}_{com}^T L_x^T L_y^T L_z^T]^T \tag{14}$$

where $L_x \in \Re^{2M\times 1}$, $L_y \in \Re^{2M\times 1}$ and $L_z \in \Re^{2M\times 1}$ are the vectors that describe the foot positions at every one of the M support phases, they are stacked vectors of the left and right foot positions at each phase, i.e $L_x = [x_{left}^T \; x_{right}^T]$ with $x_{left}, x_{right} \in \Re^{M\times 1}$ being the foot positions at each phase.

Next, lets define the following matrices

$$\begin{array}{c} P_{exju} = \begin{bmatrix} P_{euj} & 0_{1\times N} & 0_{1\times N} & 0_{1\times 6M} \end{bmatrix}, \\ P_{eyju} = \begin{bmatrix} 0_{1\times N} & P_{euj} & 0_{1\times N} & 0_{1\times 6M} \end{bmatrix}, \\ P_{ezju} = \begin{bmatrix} 0_{1\times N} & 0_{1\times N} & P_{euj} & 0_{1\times 6M} \end{bmatrix}, \\ e \in \{p, v, a\}. \end{array} \tag{15}$$

Here P_{euj} is the j'th row of the matrix P_{eu} (i.e if $e = p$ then it is the j'th row of P_{pu}) and $0_{a\times b}$ is zero matrix of dimensions $a \times b$.

Then, the COM position, velocity and acceleration can be defined in terms of the state vector and the initial COM state.

$$\begin{array}{c} e_{com}(j) = P_{psj}\bar{e}_0 + P_{peju}\chi, \\ \dot{e}_{com}(j) = P_{vsj}\bar{e}_0 + P_{veju}\chi, \\ \ddot{e}_{com}(j) = P_{asj}\bar{e}_0 + P_{aeju}\chi, \\ e \in \{x, y, z\}, \\ j \in \{1, \dots, N\}. \end{array} \tag{16}$$

This allows (3) to be parameterized in terms of jerks.

B. Model predictive control of COM and foot positions

Lets define a goal function as follows

$$\begin{array}{c} f_{goal}(\chi) = \sum_{e=x,y,z} (\sum_{j=1}^{N} \frac{R}{2} \dddot{e}_{com}(j) \\ + \frac{\alpha}{2}\dot{e}_{com}(j) + \frac{\beta}{2}(e_{com}(j) - e_{com}^{goal}(j))^2 + \frac{\gamma}{2}(\bar{L}_e - L_e^{goal})^2) \end{array} \tag{17}$$

Minimizing this goal function would be equivalent to minimizing the jerk and velocity of the COM, finding a set of jerks that make the COM follow a desired COM trajectory along the three axes, $\bar{X}_{com_{Goal}} = \begin{bmatrix} x_{com}^{goal}(1) & \dots & x_{com}^{goal}(N) \end{bmatrix}$,$\bar{Y}_{com_{Goal}} = \begin{bmatrix} y_{com}^{goal}(1) & \dots & y_{com}^{goal}(N) \end{bmatrix}$ and $\bar{Z}_{com_{Goal}} = \begin{bmatrix} z_{com}^{goal}(1) & \dots & z_{com}^{goal}(N) \end{bmatrix}$, while also trying to make the foot positions follow their referential positions $\bar{L}_{x_{Goal}}$,$\bar{L}_{y_{Goal}}$ and $\bar{L}_{z_{Goal}}$. The values R, α, β and γ are the jerk, velocity, COM position error and foot position error costs. Substituting linear systems such as (10) into (17) allows every element to be expressed in terms of the state vector and the initial COM state, thus the following quadratic goal function can be constructed (please refer to works such as [4] for more details).

$$\begin{array}{c} f_{goal}(\chi) = \quad \chi^T Q_{goal}\chi + q_{goal}^T\chi \; , \\ Q_{goal} = \begin{bmatrix} \varphi & 0 & 0 & 0 & 0 & 0 \\ 0 & \varphi & 0 & 0 & 0 & 0 \\ 0 & 0 & \varphi & 0 & 0 & 0 \\ 0 & 0 & 0 & \phi & 0 & 0 \\ 0 & 0 & 0 & 0 & \phi & 0 \\ 0 & 0 & 0 & 0 & 0 & \phi \end{bmatrix}, \\ q_{goal} = \begin{bmatrix} \alpha P_{vu}^T P_{vs}\hat{x}_k + \beta_x P_{pu}^T P_{ps}\hat{x}_k - \beta_x P_{pu}^T \bar{X}_{com_{Goal}} \\ \alpha P_{vu}^T P_{vs}\hat{x}_k + \beta_y P_{pu}^T P_{ps}\hat{x}_k - \beta_y P_{pu}^T \bar{Y}_{com_{Goal}} \\ \alpha P_{vu}^T P_{vs}\hat{x}_k + \beta_z P_{pu}^T P_{ps}\hat{x}_k - \beta_z P_{pu}^T \bar{Z}_{com_{Goal}} \end{bmatrix}, \\ \varphi = \frac{R}{2}\mathbf{I}_{\mathbf{N}\times\mathbf{N}} + \frac{\alpha}{2}P_{vu}^T P_{vu} + \frac{\beta}{2}P_{pu}^T P_{pu}, \phi = \frac{\gamma}{2}\mathbf{I}_{\mathbf{N}\times\mathbf{N}}, \end{array} \tag{18}$$

C. ZMP constraints in quadratic form

The center of the support polygon can be extracted from the state vector as follows

$$x_{sup} = U_{sup}S_x\chi \tag{19}$$

Where S_x selects the foot positions in the x axis from the state vector, i.e $L_x = S_x\chi$. Also, U_{sup} is a matrix that converts the foot positions into the center of the support polygon, it gets determined at each MPC time-step based on the remaining support phase sequence. For example, if $M = 3$ and the future supporting phase sequence starts with left support then switches to double support and finally moves to right support, then the center support polygon can be described as

$$\begin{array}{rcc} x_{sup}(1) & = & x_{left}(1), \\ x_{sup}(2) & = & \frac{1}{2}(x_{left}(2) + x_{right}(2)) \\ x_{sup}(3) & = & x_{right}(3). \end{array} , \tag{20}$$

A U_{sup} which would implement the above equations would be

$$U_{sup} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

Now by substituting (16) and (19) into (3), the terms of (3) can be found as follows.

$$\begin{gathered} -x_{max}(j)\ddot{z}_{com}(j) = -x_{max}P_{ajs}\bar{z}_0 - x_{max}P_{azju}\chi, \\ -gx_{max}(j) = -gx_{max}, \\ gx_{com}(j) = gP_{pjs}\bar{x}_0 + gP_{pxju}\chi, \\ x_{com}(j)\ddot{z}_{com}(j) = P_{pjs}\bar{x}_0\bar{z}_0^T P_{ajs}^T \\ +(P_{pjs}\bar{x}_0 P_{azju} + P_{ajs}\bar{z}_0 P_{pxju})\chi \\ \chi^T P_{pxju}^T P_{azju}\chi, \\ -z_{com}(j)\ddot{x}_{com}(j) = -P_{pjs}\bar{z}_0\bar{x}_0^T P_{ajs}^T \\ +(-P_{pjs}\bar{z}_0 P_{axju} - P_{ajs}\bar{x}_0 P_{pzju})\chi \\ -\chi^T P_{pzju}^T P_{axju}\chi, \\ z_{sup}(j)\ddot{x}_{com}(j) = \chi^T P_{axju}^T U_{sup} S_z \chi + x_0^T P_{ajs}^T U_{sup} S_z \chi, \\ -x_{sup}(j)\ddot{z}_{com}(j) = -\chi^T P_{azju}^T U_{sup} S_x \chi - z_0^T P_{ajs}^T U_{sup} S_x \chi, \\ -x_{sup}(j)g = -gU_{sup}S_x\chi, \\ j \in \{1, \ldots, N\}. \end{gathered} \tag{22}$$

Next, by collecting the terms, the following quadratic equations can be found

$$\chi^T \psi_j \chi + p_j^T \chi + \gamma_j$$

$$\begin{array}{rcc} \psi_j & = & \chi^T P_{pxju}^T P_{azju} - P_{pzju}^T P_{axju} \\ & & P_{axju}^T U_{sup} S_z - P_{azju}^T U_{sup} S_x, \\ p_j & = & -x_{max} P_{azju} + gP_{pxju} \\ & & P_{pjs}\bar{x}_0 P_{azju} + P_{ajs}\bar{z}_0 P_{pxju} \\ & & -P_{pjs}\bar{z}_0 P_{axju} - P_{ajs}\bar{x}_0 P_{pzju} \\ & & +x_0^T P_{ajs}^T U_{sup} S_z - z_0^T P_{ajs}^T U_{sup} S_x \\ & & -gU_{sup}S_z, \\ \gamma_j & = & -x_{max}P_{ajs}\bar{z}_0 - gx_{max} + gP_{pjs}\bar{x}_0 \\ & & P_{pjs}\bar{x}_0\bar{z}_0^T P_{ajs}^T - P_{pjs}\bar{z}_0\bar{x}_0^T P_{ajs}^T, \\ & & j \in \{1, \ldots, N\}. \end{array} \tag{23}$$

These equations can be computed in parallel for each timestep. In order to be able to easily find the gradient or Hessian of the quadratic form, $\chi^T \psi_j \chi + p_j^T \chi + \gamma_j$, it is necessary for ψ_j to be symmetric. However ψ_j is in fact not symmetric. This can be rectified by noting that any matrix can be split into a combination of symmetric and anti-symmetric matrices. The "symmetric part" of ψ_j is

$$P_j = \tfrac{1}{2}(\psi_j + \psi_j^T) \qquad j \in \{1 \ldots N\} \ . \tag{24}$$

Also note that $\chi^T \psi_j \chi = \chi^T P_j \chi$, thus ψ_j can be replaced with P_j in (23) and doing so defines every term of (4).

So far only the upper bound of the ZMP has been considered, the lower bound can be considered by multiplying (4) by -1 and substituting $-x_{min}$ for x_{max}.

Finally, please note that one such quadratic constraint can be constructed for each value of j (i.e each MPC prediction step) in both the x and y axes (the z axis has no ZMP constraint so it does not need to be considered).

D. Foot motion constraints

1) Preventing supporting foot from moving: Reactive foot positioning can make the feet move during single support. However this may not be physically realizable, so to prevent this behavior the foot positions need to be constrained in such a way that they can only move during their swinging phase.

For example if the phase sequence was right ,double, left then the following constraints would be needed

$$\begin{gathered} x_{right}(1) = x_{right}(2) \\ x_{right}(2) = x_{right}(3),\ x_{left}(2) = x_{left}(3) \\ x_{left}(3) = x_{left}(4) \end{gathered} \tag{25}$$

These constraints can be enforced via the following equation[1]

$$\begin{gathered} \lambda S_x \chi = \mathbf{0}, \\ \lambda = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \\ \lambda = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{gathered} \tag{26}$$

This is a linear constraint that can be expressed in the form of (4) by using

$$\begin{gathered} P_j = 0, \quad p_j^T = (\lambda S_x)_{j-N-1}, \quad \gamma_j = 0, \\ j \in \{N+1, \ldots, N+1+w\} \end{gathered} \tag{27}$$

where $(\lambda S_x)_j$ is the j'th row of λS_x and w is the number rows of λS_x.

Furthermore, the left and right foot positions during the first phase should be considered as something that is already committed to, thus they should not be allowed to change.

$$\begin{gathered} x_{right}(1) = x_{right}^* \\ x_{left}(1) = x_{left}^* \end{gathered} \tag{28}$$

Here x_{left}^* and x_{right}^* can be foot positions computed for the current support phase during the previous support phase. This constraint also be expressed in the form of (4) .

[1]Another way would be to use $\lambda S_x \chi < \mathbf{0}$ and $-\lambda S_x \chi < -\mathbf{0}$ simultaneously.

2) Following floor shape: Considering variable COM height is not very useful if only flat terrains are considered and reactive foot positioning on non-flat terrains requires the foot height to be constrained to that of the terrain.

The foot height constraints can be expressed as a quadratic constraint as was done in (4) but this would not be very interesting because only flat, inclined or parabolic floor shapes could be expressed in this way. Thus a nonlinear floor constraint will be added. This can be done by adding the nonlinear constraint $H_i(\chi) < 0,\ i \in \{1, \ldots, 2M\}$ for each support phase of the left and right foot. In this work a sinusoidal floor constraint with a height of 0.3 meters was chosen but other choices are also applicable.

$$H_i(\chi) = 0.3\sin(L_x(i)) - L_z(i) \quad i \in \{1, \ldots, 2M\} \tag{29}$$

3) Preventing feet from over extending: The difference between the left and right foot at each phase needs to be constrained to prevent the legs from over extending or colliding with each other.

This constraint was discussed in [4] and will only be mentioned briefly in this work. Basically the following linear constraint needs to be implemented for each support phase

$$\alpha_{min} < x(i)_{left} - x(i)_{right} < \alpha_{max} \quad i \in \{1, \ldots, M\} \tag{30}$$

where α_{min} and α_{max} is the minimum and maximum distance between the feet, respectively.

E. Sequential Quadratic Programming

SQP works by linearizing the QCQP around a starting point to get a QP which can then be solved and the result would be used to move the starting point closer to the correct result. Then the process would be iterated until the solution is reached. The linearized QP problem can be expressed as follows

$$\begin{aligned} \min_{\delta} \quad & \tfrac{1}{2}\delta^T \nabla^2_\chi (f_{goal}(\chi))\delta + \nabla_\chi f_{goal}(\chi)\delta \\ \text{s.t} \quad & \nabla_\chi (h_j(\chi_k)^T)\delta + h_j(\chi_k) < 0 \\ & \nabla_\chi (H_i(\chi_k)^T)\delta + H_i(\chi_k) < 0 \\ & j \in \{1, \ldots, k\}, \quad i \in \{1, \ldots, 2M\} \end{aligned} \tag{31}$$

Where $f_{goal}(\chi)$ is as defined in (18), $h_j(\chi)$ represents (4) which encodes the ZMP and foot position constraints, $H_i(\chi)$ is the foot height constraint that was discussed in the previous section and δ is a new variable for which the linearized sub problem is to be solved.

Again, the gradients and Hessian can be found by applying (5) (this can be done in parallel) and doing so would yield faster and more accurate results than using a finite difference method.

In this work (31) is solved via the active set method in [13] and to further improve computational performance, the Cholesky decomposition was calculated only once and stored. The solution vector χ can then be updated as follows

$$\chi_{k+1} = \chi_k + \delta \ . \tag{32}$$

This procedure is repeated until the maximum number of iterations is reached or the SQP method converges. In this work the maximum number of iterations is set to a number low enough to allow real-time performance. Also, each time the SQP algorithm is run, it can be "hot-started" with the previous time-steps solution in order to allow it to improve the previous result, although this is not necessary.

III. Simulation Results

A simulation was conducted with pendulum cart dynamics where the algorithm outlined in section II was implemented in a MPC loop and a 15 second long trajectory was generated. The parameters are shown in table I and II, note that the preview horizon was chosen to be $N\Delta T = 1.6$ seconds which is similar to what was used in [12] and the average COM height and ZMP bound were chosen to be similar to that of a human (i.e $||x_{max,min}|| = 0.1$ corresponds to a 20 cm long foot and an average COM height of 0.8 meters corresponds to the middle of a 160 cm tall human)

Also, the COM goal position was set to the center of the resulting support polygon by doing $\bar{X}_{com_{goal}} = U_{sup}L_x$ and $\bar{Y}_{com_{goal}} = U_{sup}L_y$ at every MPC time-step based on the previous time-steps L_x and L_y vectors.

Firstly fig. 2a shows the left foot positions that were given as reference to the trajectory generator and the resultant foot positions that the trajectory generator produced. As can be seen the resultant foot positions vary a bit from the desired ones and this demonstrates that the algorithm is able to modify the foot positions in order to find a more optimal foot placement. It can also be seen that the foot height is non-constant.

Next, fig. 3 demonstrates the algorithms ability to track a referential COM position in all three axes while keeping the ZMP within the support polygon. The COM references were set to the support polygon centers, thus they change abruptly but it can be seen that the COM response is smooth (using a zig-zag type of reference may be beneficial but not necessary in this case). It can also be noticed that the ZMP sometimes slightly leaves the support polygon for a short period of time, this may be remedied with more SQP iterations but it would increase computation time. However if we repeat the simulation with tighter ZMP bounds then we can see from the result in fig. 4 that the new ZMP response is smaller than the ZMP of the previous simulation(the ZMP response of the previous simulation was superimposed on the figure as "ZMP X Sim 1" and "ZMP Y Sim 1") and that the new ZMP response no longer touches the edges of the previous support polygon. Thus if we are concerned about ensuring that the ZMP does not leave the real support polygon, then, as suggested in fig. 2b, we can make x_{max} and x_{min} sufficiently smaller than the real support polygon. Of

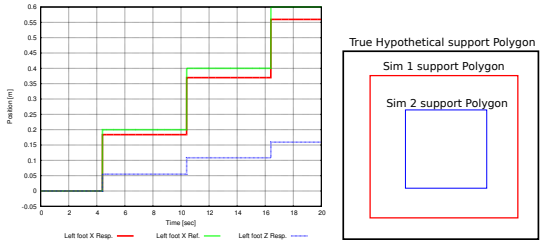
course, unrealistically tight ZMP bounds will not be realized regardless how many SQP iterations are used.

Lastly, the computational time for generating the QCQP problem and solving it is shown in table II. For reference, all simulations were performed on a 3.4 Ghz Intel i7-4770 CPU, low latency Linux kernel 3.13.0, networking and Xserver services were disabled and the Eigen matrix library[14] with vectorization was used.

It can be seen that both simulations could be performed on average in about 4 milliseconds and at most in 4.8 milliseconds, this is arguably real-time performance.

It should be noted though, that the number of SQP iterations (therefore the computation time) that are required to produce a stable trajectory depends on how tight the ZMP bounds are, the COM state and how aggressive the goal trajectory is.

The conclusion that can be drawn from this is that real-time performance is feasible on current hardware as long as the desired task is not too demanding.



(a) Left Foot positions, reactive step-ping

(b) True vs. simulated support polygons

Figure 2.

IV. Conclusions

This paper presented the ZMP equations for variable COM height with variable foot step positions as a closed form quadratic equation. This was then treated as a motion constraint and combined with 3-D model predictive control of a robots center of mass to form a QCQP problem that could be solved via SQP to generate a sequence of foot positions and a center of mass trajectory. Furthermore, the problem gradients that SQP requires can be calculated efficently by analytic differentiation of the constraints in the QCQP. Additional constraints were included to prevent the feet from sliding on the floor while they were supporting the robots weight, to make the foot height track a sinusoidal floor and to prevent the feet from over-extending.

It was found that real-time performance (solution times under 5 ms) is feasible on current hardware as long as the ZMP bounds are not too tight.

Preview Steps N	Preview time-steps ΔT	MPC time-step	R,α,β,γ	Avg. $z_{com_{goal}}$ [m]
16	100ms	5ms	1E-6,5,1E3,1E2	0.8

Table I
Parameters

	ZMP Bound [m] $\|\|x_{max}\|\|$ and $\|\|x_{min}\|\|$	SQP Runs	Computation Time [ms] Avg	Min	Max
Sim 1.	0.1	2	3.83	2.31	4.51
Sim 2.	0.04	2	4.01	2.51	4.81

Table II
ZMP bound and calculation time

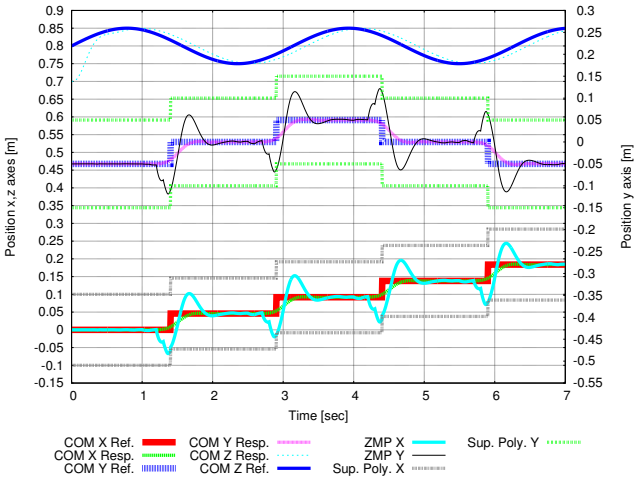


Figure 3. Simulation 1: COM Tracking, ZMP Positions and Support Polygon

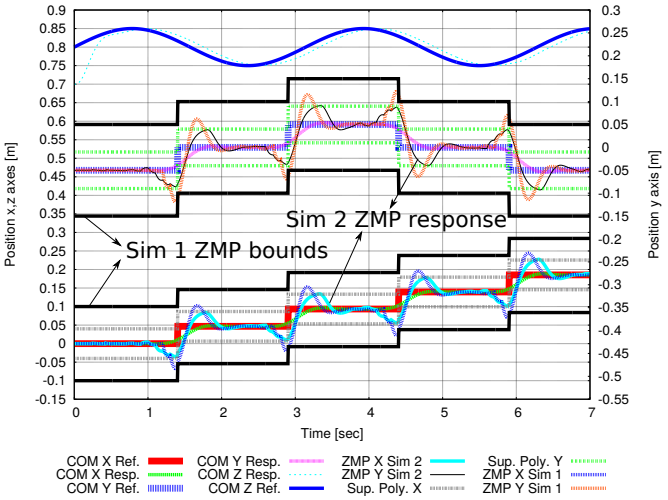


Figure 4. Simulation 2: COM Tracking, ZMP Positions and Support Polygon

References

[1] R. Tajima, D. Honda, and K. Suga, "Fast running experiments involving a humanoid robot," in ICRA, 2009, pp. 1571–1576.

[2] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The development of honda humanoid robot," in ICRA, 1998, pp. 1321–1326.

[3] B. Stephens, "Push recovery control for force-controlled humanoid robots," Ph.D. dissertation, Carnegie Mellon University, 2011.

[4] H. Diedam, D. Dimitrov, P.-B. Wieber, K. D. Mombaur, and M. Diehl, "Online walking gait generation with adaptive foot positioning through linear model predictive control," in IROS, 2008, pp. 1121–1126.

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

[6] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control using divergent component of motion," in IROS, 2013, pp. 2600–2607.
[7] Y. Zhao and L. Sentis, "A three dimensional foot placement planner for locomotion in very rough terrains," in IEEE-RAS International Conference on Humanoid Robots, 2012.
[8] A. Herdt, "Model predictive control of a humanoid robot," Ph.D. dissertation, Paris institute of technology, 2012.
[9] A. Konno, T. Myojin, T. Tsujita, and M. Uchiyama, "Optimization of impact motions for humanoid robots," in IROS, 2008, pp. 647–652.
[10] 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, 2014.
[11] K. van Heerden and R. Ozawa, "An investigation on model predictive control for variable height walking of humanoid robots by using sequential quadratic programming," Robotics Societey of Japan, 2014.
[12] 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 IEEE Int. Conf Robotics and Automation, 2003, pp. 1620–1626.
[13] A. I. D. Goldfarb, "A numerically stable dual method for solving strictly convex quadratic programs," Mathematical Programming, vol. 27, pp. 1–33, 1983.
[14] G. Guennebaud, B. Jacob et al., "Eigen v3," http://eigen.tuxfamily.org, 2010.