Online Generated Kick Motions for the NAO Balanced Using Inverse Dynamics

Felix Wenk[1] and Thomas Röfer[2]

[1] Universität Bremen, Fachbereich 3 – Mathematik und Informatik,
Postfach 330 440, 28334 Bremen, Germany
E-Mail: fwenk@informatik.uni-bremen.de
[2] Deutsches Forschungszentrum für Künstliche Intelligenz,
Cyber-Physical Systems, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
E-Mail: thomas.roefer@dfki.de

Abstract. One of the major tasks of playing soccer is kicking the ball. Executing such complex motions is often solved by interpolating key-frames of the entire motion or by using predefined trajectories of the limbs of the soccer robot. In this paper we present a method to generate the trajectory of the kick foot online and to move the rest of the robot's body such that it is dynamically balanced. To estimate the balance of the robot, its Zero-Moment Point (ZMP) is calculated from its movement using the solution of the Inverse Dynamics. To move the ZMP, we use either a Linear Quadratic Regulator on the local linearization of the ZMP or the Cart-Table Preview Controller and compare their performances.

1 Introduction

To play humanoid soccer, two essential motion tasks have to be carried out: walking over a soccer field and kicking the ball. Both motions have to be both flexible and robust, i.e. the robot has to be able to walk in different directions at different speeds and kick the ball in different directions with different strengths, all while maintaining its balance to prevent falling over.

To design and execute motions to kick the ball, different methods have been developed. A seemingly obvious approach to motion design is to manually set up some configurations, i.e. sets of joint angles called key-frames, which the robot shall assume during the motion, and then interpolate between these key-frames while the motion is executed. This quite popular method has been used to design kick motions by a number of RoboCup Standard Platform League (SPL) teams including B-Human [12] and Nao Team HTWK [13].

Because it completely determines the robot's motion, the interpolation between fixed sets of joint angles precludes any reaction to changing demands or to external disturbances. Therefore, Czarnetzki *et al.* [3] specify key-frames of the motion of the robot's limbs in Cartesian space instead of joint space. This leaves the movement of the robot under-determined, so the Cartesian key-frame approach can be combined with a controller to maintain balance.

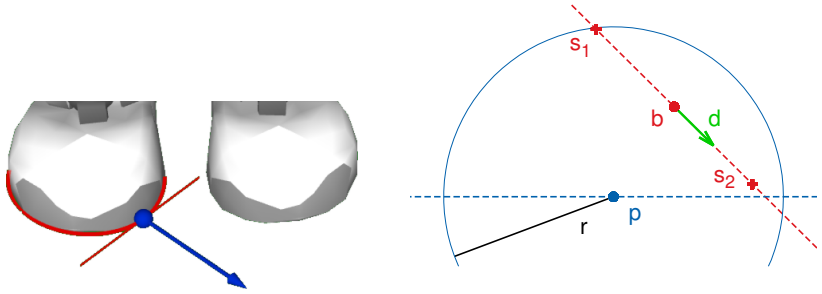
To design more flexible kick motions, Müller *et al.* [11] model the trajectories of the robot's hands and feet in Cartesian space using piecewise Bezier curves. Depending on the position of the ball relative to the kicking robot and the desired kick direction, the Bezier curves are modified such that the kick foot actually hits the ball in the desired direction. The modification of the Bezier curves is constrained such that the resulting curve is always continuously differentiable, which results in a smooth trajectory. In addition, a balancing controller is included to maintain static balance by tilting the robot such that its center of mass (COM) stays within the support polygon, i.e. the contour of the support foot. To make sure the robot gets properly tilted, the angular velocity measured by gyroscopes in the robot's torso is used as feedback.

In this work, the approaches mentioned in this introduction are combined to a motion engine that generates and executes dynamically balanced kick motions, but neither requires prior modeling of trajectories of limbs nor needs prerecorded key-frames to interpolate. Instead of searching the path of the kick foot to the ball like Xu *et al.* [15], the trajectory of the kick foot is an interpolation between a number of reference poses inferred from the ball position, the kick direction and the kick strength. The points are interpolated using a spline that is continuously differentiable twice so that the trajectory of the kick foot has no sudden jumps in acceleration [6]. The limbs of the robot, which are not part of the kick leg and whose motion is therefore not determined by the kick foot trajectory, are moved to maintain dynamic balance, i.e. to keep the Zero-Moment Point (ZMP) [14] within the support polygon. To achieve this, the ZMP is calculated via the solution of the Inverse Dynamics based on an estimate of the motion of the joints. Two different methods to move the ZMP are implemented and compared: first a Linear Quadratic Regulator (LQR) [9, 4], which modifies the joint angles directly using a linearization of the ZMP depending on the motion of the joints, and second a Preview Controller [10] which generates a trajectory for the COM depending the current and the preview of the future ZMP [8]. The COM trajectory is then translated to joint angles using inverse kinematics.

The rest of the paper is organized as follows. The generation of the trajectory of the kick foot is treated in Sect. 2. Section 3 covers the motion of the robot to maintain dynamic balance. The latter includes the calculation of the balance criterion, the ZMP, and therefore the solution of the Inverse Dynamics. Experiments and their results including a comparison of the two balancing methods make up Sect. 4. Section 5 finally concludes this work.

2 Generating the Kick Foot Motion

To generate the kick foot motion, a number of reference points have to be calculated. At first it has to be determined which part of the contour of the foot should hit the ball. Because the contour of the foot is round and the kick foot will not be rotated during the kick, the kick is approximately a collision between two spheres. So the tangent at the contact point on the foot contour has to be



(a) (b)

Fig. 1. a) The kick foot is to collide with the ball at the point where the foot contour's tangent is orthogonal to the kick direction. b) Geometric construction of the strikeout s_1 and swing s_2 reference points.

orthogonal to the kick direction. To calculate the tangent, we approximate the contour of the front of the foot with a cubic Bezier curve.

A cubic Bezier curve of dimension d is described by four control points $\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3 \in \mathbb{R}^d$, which jointly determine the resulting curve

$$\boldsymbol{C}(t) = \sum_{i=0}^{3} B^3_{i,[\alpha,\beta]}(t) \cdot \boldsymbol{p}_i \quad \text{with } t \in [\alpha, \beta], \alpha, \beta \in \mathbb{R} \ . \tag{1}$$

$B^n_{i,[\alpha,\beta]}(t) = \binom{n}{i} \cdot \left(\frac{t-\alpha}{\beta-\alpha}\right)^i \cdot \left(\frac{\beta-t}{\beta-\alpha}\right)^{n-1}$ is the i-th Bernstein polynomial of degree n defined on the interval $[\alpha, \beta]$ [6]. The derivative of such a curve is

$$\dot{\boldsymbol{C}}(t) = \frac{3}{\beta-\alpha} \cdot \sum_{i=0}^{2} (\boldsymbol{p}_{i+1} - \boldsymbol{p}_i)\, B^2_{i,[\alpha,\beta]}(t) \ . \tag{2}$$

As shown in Fig. 1a, the ball contact point on the contour $\boldsymbol{C}_{\text{foot}}$ is then

$$\boldsymbol{C}_{\text{foot}}(t_{\text{bc}}) \quad \text{with } \dot{\boldsymbol{C}}_{\text{foot}}(t_{\text{bc}}) \cdot \boldsymbol{d} = 0 \ , \tag{3}$$

where $\boldsymbol{d} \in \mathbb{R}^2$ is the normalized kick direction vector in the plane of the soccer field. Since the support foot (hopefully!) does not move during a kick, its coordinate system serves as the reference for all coordinates related to the kick foot trajectory. The resulting curve describes the trajectory of the origin of the kick foot. Since $\boldsymbol{C}_{\text{foot}}(t_{\text{bc}})$ is relative to the origin of the kick foot, the ball contact reference point $\boldsymbol{b}$ satisfies $\boldsymbol{b} + \boldsymbol{C}_{\text{foot}}(t_{\text{bc}}) = \boldsymbol{p}_{\text{ball}}$, where $\boldsymbol{p}_{\text{ball}} \in \mathbb{R}^2$ is the ball position on the pitch.

The points $\boldsymbol{s}_1 \in \mathbb{R}^2$ and $\boldsymbol{s}_2 \in \mathbb{R}^2$ to strike out and swing out the kick foot are the intersections of the line in kick direction through the ball reference point with a circle of radius r around the pelvis joint at $\boldsymbol{p} \in \mathbb{R}^2$ of the support leg. The construction is pictured in Fig. 1b. $\boldsymbol{p}, \boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are all in a plane parallel to the field. The height of the plane is a parameter to be tuned by the user. r

is determined manually so that the kick foot can reach all points on the circle. If $\boldsymbol{s}_2$ gets too close to the support leg side of the pelvis or even enters it, it is pulled back on the kick direction line as pictured in Fig. 1b, so that a small safety margin to the support leg remains and collisions between the kick foot and the support leg are avoided.

Aside from the ball position and the kick direction, the user of the kick engine also specifies the duration T of the kick foot motion and the speed v the kick foot should have when the ball is hit.

This information is used to determine the durations of the individual pieces of the curve. The curve must pass through the six points $\boldsymbol{r}_j$ with $0 \leq j < 6$ in order. The i-th curve segment starts at $\boldsymbol{r}_i$, ends at $\boldsymbol{r}_{i+1}$ and takes the duration Δ_i. The segment around the ball from $\boldsymbol{r}_2 = \boldsymbol{b} - \lambda_1 \boldsymbol{d}$ to $\boldsymbol{r}_3 = \boldsymbol{b} + \lambda_2 \boldsymbol{d}$ shall be passed in $\Delta_2 = \frac{\lambda_1 + \lambda_2}{v}$. It turned out that a good choice for λ_1 and λ_2 is if they sum up to a little more than one diameter of the ball. The speed at which to move the kick foot back to the end position of the trajectory is set to $\frac{v}{4}$, so the duration for the last phase is $\Delta_4 = \frac{4 \cdot \|\boldsymbol{r}_5 - \boldsymbol{r}_4\|}{v}$. The other durations are calculated such that they are proportional to the square root of the length between their end points and sum up to $T - \Delta_2 - \Delta_4$. This is also called centripetal parameterization [6].

The durations Δ_i with $0 \leq i < 5$ are equivalent to a knot sequence of a cubic B-Spline curve with the clamped end condition [6]. The clamped end condition requires that the derivative of the curve is 0 at the ends, i.e. in this case $\dot{\boldsymbol{B}}(0) = \dot{\boldsymbol{B}}(T) = 0$, meaning physically that the kick foot does not move. B-Spline curves are a generalization of Bezier curves. As a cubic Bezier curve, a cubic B-Spline curve is a linear combination of control points $\boldsymbol{d}_j$, which are also called de Boor points [6, 2].

$$\boldsymbol{B}(u) = \sum_{j=0}^{L} N_j^3(u) \boldsymbol{d}_j \tag{4}$$

$N_j^n(u)$ are called the B-Spline basis functions or just B-Splines and are recursively defined as

$$N_j^n(u) = \frac{u - u_{j-1}}{u_{j+n-1} - u_{j-1}} N_j^{n-1}(u) + \frac{u_{j+n} - u}{u_{j+n} - u_j} N_{j+1}^{n-1}(u) \tag{5}$$

$$\text{with } N_j^0(u) = \begin{cases} 1 & \text{if } u_{j-1} \leq u \leq u_j \\ 0 & \text{otherwise} \end{cases}.$$

L in (4) is the index of the last control point. For a B-Spline curve of degree n, this is $L = K + 1 - n$, where $K + 1$ is the number of knots. A B-Spline curve is defined over the interval $[u_{n-1}, u_L]$, i.e. in the cubic case over $[u_2, u_L]$. Each reference point is associated with a knot. Since we want the curve to start at $\boldsymbol{r}_0$ and end at $\boldsymbol{r}_5$, this means that $\boldsymbol{B}(u_{i+2}) = \boldsymbol{r}_i$. This implies that $L = 7$, so there are 8 control points and $K + 1 = L + n = 10$ knots in the knot sequence $u_0, \ldots, u_9$. The clamped end condition defines the knots that are not associated to reference points to be $u_0 = u_1 = u_2$ and $u_9 = u_8 = u_7$. Given the durations

between the reference points and that we want the curve to be defined on the interval $[0, T]$, the knots are

$$u_2 = 0 \quad \text{and} \quad u_{i+2} = u_{i+2-1} + \Delta_{i-1} \quad \text{for } 0 < i < 5 \ . \tag{6}$$

Due to the clamped end condition the first two control points collapse with the first reference point and the last two control points collapse with the last reference point, so $\boldsymbol{d}_0 = \boldsymbol{d}_1 = \boldsymbol{r}_0$ and $\boldsymbol{d}_6 = \boldsymbol{d}_7 = \boldsymbol{r}_5$. By inserting the association between knots and reference points $\boldsymbol{B}(u_{i+2}) = \boldsymbol{r}_i$ into (4) and by replacing the known control points with the corresponding reference points, we get a system of four equations for the remaining four control points. Figure 2 shows two different curves calculated by using this scheme. Since the B-Spline curves are infinitely often continuously differentiable between the knots and $n - r$ times continuously differentiable at a knot occurring r times in the knot sequence [6], the resulting curve is always at least $3 - 1$ times continuously differentiable.

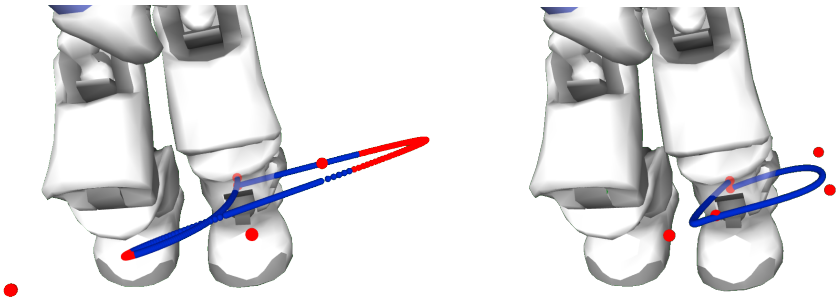


Fig. 2. Trajectories (blue) of the kick foot with the corresponding de Boor points (red dots). On the left, the speed of the kick foot is too large, so that some parts of the trajectory (red) are unreachably far away from the hip. On the right this is fixed by lowering the speed of the foot, resulting in a completely reachable trajectory.

If the speed v of the kick foot is too high relative to the kick duration T, the segment leading up to the ball contact segment may get too long, so that some parts of the curve are not reachable by the kick foot. To check whether this is the case, we calculate the point $\boldsymbol{B}(u_{\max})$ on that segment with the maximum (squared) distance from the pelvis joint of the kick leg and test whether this is larger than the maximum distance allowed $d_{\max}$, i.e. whether $\boldsymbol{B}(u_{\max}) \cdot \boldsymbol{B}(u_{\max}) > d_{\max}^2$. If this is true we define the error function

$$e(v) = \|\boldsymbol{B}(u_{\max}) \cdot \boldsymbol{B}(u_{\max}) - d_{\max}^2\| \tag{7}$$

by making the speed v variable and keeping all other parameters of the curve construction fixed. Knowing that the current v is too large, we calculate the largest acceptable kick foot speed as $v' = \operatorname{argmin}_v e(v)$ using (7). The effect of this procedure is pictured in Fig. 2, where an infeasible curve with a high kick foot speed is turned into a feasible one.

3 Dynamically Balanced Kick Foot Motion

Now that the kick foot trajectory is generated, it needs to be executed while the robot is dynamically balanced.

3.1 Balance Estimation

To determine how balanced the robot is, we estimate the difference between the current ZMP $\boldsymbol{p}_0$ and a demanded ZMP $\boldsymbol{p}_{0,\mathrm{d}}$. The ZMP is defined as the point in the support polygon, i.e. the sole of the support foot, at which the (vertical) reaction force the ground exerts on the foot must act, such that all (horizontal) torques are canceled out [14]. If $\boldsymbol{p}_0$ does not exist and the then-fictional ZMP lies outside of the foot sole, the robot will begin to rotate about the edge of the sole and probably fall over eventually. With $\boldsymbol{f}_{\mathrm{r,z}}$ being the vertical component of the reaction force, $\boldsymbol{\tau}_{\mathrm{xy}}$ the total torque of the robot relative to the projection of the support foot's origin on the ground, $\boldsymbol{n}$ the normal vector on the ground and $\times$ denoting the cross product of two vectors, the ZMP $\boldsymbol{p}_0$ satisfies

$$\boldsymbol{p}_0 \times \boldsymbol{f}_{\mathrm{r,z}} + \boldsymbol{\tau}_{\mathrm{xy}} = 0 \quad \text{and therefore} \quad \boldsymbol{p}_0 = \frac{\boldsymbol{n} \times (-\boldsymbol{\tau}_{\mathrm{xy}})}{\boldsymbol{n} \cdot \boldsymbol{f}_{\mathrm{r,z}}} = \frac{1}{f_{\mathrm{r,z}}} \begin{bmatrix} \tau_{\mathrm{y}} \\ -\tau_{\mathrm{x}} \\ 0 \end{bmatrix} . \quad (8)$$

If the ZMP exists as in Fig. 3, the robot is said to be dynamically balanced.

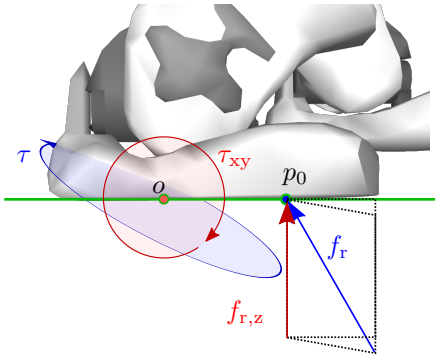


Fig. 3. The vertical component $f_{\mathrm{r,z}}$ of the reaction force f_{r} acts on the Zero-Moment point p_0 to cancel out the horizontal component τ_{xy} of the total torque τ relative to the origin of the sole of the support foot.

To calculate the ZMP, we need to know $\boldsymbol{f}_{\mathrm{r,z}}$ and $\boldsymbol{\tau}_{\mathrm{xy}}$. To infer these from the robot motion, we estimate the angle θ_j, angular velocity $\dot{\theta}_j$ and angular acceleration $\ddot{\theta}_j$ of each joint j and solve the Inverse Dynamics problem using a variation [5] of the Recursive Newton-Euler Algorithm (RNEA) [7]. As the joint angle measurements are too noisy for simple numerical differentiation, we obtain

the estimates for $\dot{\theta}_j$ and $\ddot{\theta}_j$ by numerically differentiating the target joint angles instead. Using Kalman filters [9] for each non-leg joint and one Kalman filter for each support leg, we filter the differences $\Delta\theta_j$, $\Delta\dot{\theta}_j$, and $\Delta\ddot{\theta}_j$ between the actual and commanded joint motions similar to the work of Belanger [1]. These are added to the commanded joint motions obtained by numerical differentiation. This allows us to use very small process variances in the filters, so that the estimates do not jump wildly while still having the estimates respond properly to changes of the joint motions due to our own software.

The motions of the joints of a single leg are filtered in a combined state instead of using individual filters, because we use the rotation of the gyroscope in the NAO's torso as a measurement model to correct the angular velocities of the joints of the support leg.

Since the support foot has no velocity and acceleration during the kick, it serves as the root of the kinematic tree, which is used by the RNEA to calculate the torques and forces across the joints. We also assume an imaginary 'joint' with zero degrees of freedom between the ground and the support foot. Because the support foot does not move, the torque and force exerted on the support foot via the imaginary joint must be the reactions to the torque and force of the support foot and thus $-\boldsymbol{\tau}$ and $\boldsymbol{f}_{\mathrm{r}}$ in (8). Since we do not need to know the forces transmitted by the real joints of the robot, we use a simplified version of the RNEA by Fang *et al.* [5], which only propagates the aggregated momenta and forces from the leaves of the tree to the root. From Fang *et al.* [5], we also implemented taking the derivates

$$-\frac{\partial}{\partial\theta_j}\boldsymbol{\tau}, -\frac{\partial}{\partial\dot{\theta}_j}\boldsymbol{\tau}, -\frac{\partial}{\partial\ddot{\theta}_j}\boldsymbol{\tau}, \frac{\partial}{\partial\theta_j}\boldsymbol{f}_{\mathrm{r}}, \frac{\partial}{\partial\dot{\theta}_j}\boldsymbol{f}_{\mathrm{r}}, \frac{\partial}{\partial\ddot{\theta}_j}\boldsymbol{f}_{\mathrm{r}} \quad \text{for each joint } j. \tag{9}$$

Using these partial derivatives of (9), the partial derivatives of the ZMP $\boldsymbol{p}_0$ from (8) with respect to the joint motion are calculated by

$$\frac{\partial}{\partial\phi_j}\boldsymbol{p}_0 = \frac{\left[\boldsymbol{n}\times\left(-\frac{\partial}{\partial\phi_j}\boldsymbol{\tau}\right)_{\mathrm{xy}}\right]\left[\boldsymbol{n}\cdot\boldsymbol{f}_{\mathrm{r,z}}\right] - \left[\boldsymbol{n}\times(-\boldsymbol{\tau})_{\mathrm{xy}}\right]\left[\boldsymbol{n}\cdot\left(\frac{\partial}{\partial\phi_j}\boldsymbol{f}_{\mathrm{r}}\right)_{\mathrm{z}}\right]}{\left(\boldsymbol{n}\cdot\boldsymbol{f}_{\mathrm{r,z}}\right)^2} \tag{10}$$

with $\phi_j \in \left\{\theta_j, \dot{\theta}_j, \ddot{\theta}_j\right\}$ and j again being a joint of the robot.

$\boldsymbol{p}_{0,\mathrm{d}}$ is moved from its initial position between both feet on the ground to a position within the sole of the support foot before the kick foot is moved, stays there while the kick foot is moved, and is moved back to its initial position when the kick foot reached its final position.

3.2 Balancing with Linear Quadratic Regulation

We implemented a balancer by directly modifying the joint angles of the support leg. To control the bearing of the robot's torso, both the pitch joints of hip and ankle as well as the roll joints of hip and ankle are coupled by a factor β, so that if the pitch pair is modified by $\Delta\theta'_{\mathrm{pitch}}$, the hip joint of the pair will be modified

by $\Delta\theta_{\text{hippitch}} = \Delta\theta'_{\text{pitch}}$ and the ankle joint by $\Delta\theta_{\text{anklepitch}} = \beta\Delta\theta'_{\text{pitch}}$. The roll pair is analogous. With these joint pairs, the change of the ZMP is determined by applying the chain rule using (10):

$$\frac{\partial}{\partial\phi'_{\text{pitch}}}\boldsymbol{p}_0 = \frac{\partial}{\partial\phi_{hippitch}}\boldsymbol{p}_0 + \beta\cdot\frac{\partial}{\partial\phi_{anklepitch}}\boldsymbol{p}_0 \quad \text{with } \phi \in \{\theta, \dot{\theta}, \ddot{\theta}\} \tag{11}$$

$\frac{\partial}{\partial\phi'_{\text{roll}}}p_0$ is analogous.

We use (11) to construct a linear model to be used in a discrete-time, finite-horizon LQR [4]. If the robot was perfectly balanced, the ZMP $\boldsymbol{p}_0$ would coincide with the demanded ZMP $\boldsymbol{p}_{0,\text{d}}$ and the error $\boldsymbol{e} = \boldsymbol{p}_0 - \boldsymbol{p}_{0,\text{d}}$ would be zero. Assuming the demanded ZMP is constant, the linearization of the error over the time interval between time T_t and T_{t+1} with respect to the motions of the joint pairs is $\boldsymbol{e}_{t+1} = \boldsymbol{e}_t + \Delta\boldsymbol{e}$ with

$$\begin{aligned}
\Delta\boldsymbol{e} = {} & \dot{\theta'}_{\text{roll}}\overbrace{\frac{\partial\boldsymbol{p}_0}{\partial\theta'_{\text{roll}}}\Delta T}^{a_1} + \ddot{\theta'}_{\text{roll}}\overbrace{\left(\frac{\partial\boldsymbol{p}_0}{\partial\theta'_{\text{roll}}}\frac{1}{2}(\Delta T)^2 + \frac{\partial\boldsymbol{p}_0}{\partial\dot{\theta'}_{\text{roll}}}\Delta T\right)}^{a_2} \\
& + \dddot{\theta'}_{\text{roll}}\overbrace{\left(\frac{\partial\boldsymbol{p}_0}{\partial\theta'_{\text{roll}}}\frac{1}{6}(\Delta T)^3 + \frac{\partial\boldsymbol{p}_0}{\partial\dot{\theta'}_{\text{roll}}}\frac{1}{2}(\Delta T)^2\frac{\partial\boldsymbol{p}_0}{\partial\ddot{\theta'}_{\text{roll}}}\Delta T\right)}^{b_1} \\
& + \dot{\theta'}_{\text{pitch}}\overbrace{\frac{\partial\boldsymbol{p}_0}{\partial\theta'_{\text{pitch}}}\Delta T}^{a_3} + \ddot{\theta'}_{\text{pitch}}\overbrace{\left(\frac{\partial\boldsymbol{p}_0}{\partial\theta'_{\text{pitch}}}\frac{1}{2}(\Delta T)^2 + \frac{\partial\boldsymbol{p}_0}{\partial\dot{\theta'}_{\text{pitch}}}\Delta T\right)}^{a_4} \\
& + \dddot{\theta'}_{\text{pitch}}\overbrace{\left(\frac{\partial\boldsymbol{p}_0}{\partial\theta'_{\text{pitch}}}\frac{1}{6}(\Delta T)^3 + \frac{\partial\boldsymbol{p}_0}{\partial\dot{\theta'}_{\text{pitch}}}\frac{1}{2}(\Delta T)^2\frac{\partial\boldsymbol{p}_0}{\partial\ddot{\theta'}_{\text{pitch}}}\Delta T\right)}^{b_2},
\end{aligned} \tag{12}$$

and $\Delta T = T_{t+1} - T_t$. This leads to the linear model $\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t$ with

$$\boldsymbol{x} = \begin{bmatrix}\boldsymbol{e}\ \dot{\theta'}_{\text{roll}}\ \ddot{\theta'}_{\text{roll}}\ \dot{\theta'}_{\text{pitch}}\ \ddot{\theta'}_{\text{pitch}}\end{bmatrix}^\top \quad \boldsymbol{u} = \begin{bmatrix}\dddot{\theta'}_{\text{roll}}\ \dddot{\theta'}_{\text{pitch}}\end{bmatrix}^\top \tag{13}$$

and

$$A = \begin{bmatrix}
1 & 0 & a_{1,x} & a_{2,x} & a_{3,x} & a_{4,x} \\
0 & 1 & a_{1,y} & a_{2,y} & a_{3,y} & a_{4,y} \\
0 & 0 & 1 & \Delta T & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \Delta T \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \quad B = \begin{bmatrix}
b_{1,x} & b_{2,x} \\
b_{1,y} & b_{1,y} \\
\frac{1}{2}(\Delta T)^2 & 0 \\
\Delta T & 0 \\
0 & \frac{1}{2}(\Delta T)^2 \\
0 & \Delta T
\end{bmatrix}. \tag{14}$$

With this model, the controls to be applied at time T_t are

$$\boldsymbol{u}_t = -\left(R + B^\top P_k B\right)^{-1} B^\top P_k A\boldsymbol{x}_t \tag{15}$$

with

$$P_k = Q + A^\top P_{k-1}A - A^\top P_{k-1}B\left(R + B^\top P_{k-1}B\right)^{-1} B^\top P_{k-1}A\ , \tag{16}$$

where $P_0 = Q$, k is the horizon parameter, and Q and P are the diagonal, positive-definite matrices of the quadratic cost function $C(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}^\top Q \boldsymbol{x} + \boldsymbol{u}^\top R \boldsymbol{u}$.

We calculate the error $\boldsymbol{e}_t$ and the linearization of the ZMP in each execution cycle. We also keep track of θ'_t, $\dot{\theta}'_t$ and $\ddot{\theta}'_t$. From that, we build the model of (13) and (14) and compute $\boldsymbol{u}_t$, which is the rate of change of the acceleration and which is constant until the next motion cycle. The next angles to be set for the hip and ankle joints are computed from $\theta'_{t+1} = \theta'_t + \Delta T \dot{\theta}'_t + \frac{1}{2}(\Delta T)^2 \ddot{\theta}'_t + \frac{1}{6}(\Delta T)^3 \dddot{\theta}'_t$ according to factor β of the joint pairing. The remaining joints of the support leg are not changed.

To determine the angles of the kick leg, we calculate the pose of the torso using forward kinematics, evaluate the kick foot trajectory to the new kick foot pose, calculate the kick foot pose relative to the torso and use inverse kinematics to solve for the angles of the kick leg joints.

3.3 Balancing with a Cart-Table Controller

In addition to the LQR we implemented the Cart-Table controller by Kajita *et al.* [8]. Instead of a linearization of the ZMP, we use a truly linear model and use an extension of a linear-quadratic controller [10], which responds more quickly to changes of the demanded ZMP, because it also considers the N future demands of the ZMP instead of only the currently demanded ZMP.

According to the Cart-Table model, the ZMP depends on the motion of the COM $\boldsymbol{c}$. With the 2-dimensional COM restricted to a plane of height c_z, the 2-dimensional ZMP $\boldsymbol{p}_0$ on the ground is

$$\boldsymbol{p}_0 = \boldsymbol{c} - \frac{c_z}{g} \ddot{\boldsymbol{c}} \ . \tag{17}$$

With the ZMP, which is calculated using the inverse dynamics method explained above, and a preview series of ZMPs that will be demanded, we calculate the control output $\dddot{\boldsymbol{c}}$, the jerk of the COM. With an initial position the COM this effectively defines a trajectory for the COM. Using the kick foot pose from the evaluation of the kick foot trajectory, we implemented a simple numerical method to move the robot's torso relative to the support foot, such that the COM is moved to the position demanded by the controller and the kick foot to the pose relative to the support foot as demanded by the kick foot trajectory.

The downside of the Cart-Table model is that it only works at a constant COM height, which has to be approached before the actual balancing can start. Also, the effects of the angular motion of the robot around the COM are not modeled. But the advantages outweigh the downsides. As we will see in Sec. 4, the ZMP as modeled by the Cart-Table model is not too different from the ZMP calculated using the inverse dynamics. The computation is also much faster, because the gain matrices of the controller, including a Riccati equation such as (16), can be calculated once in advance and then repeatedly used while balancing.

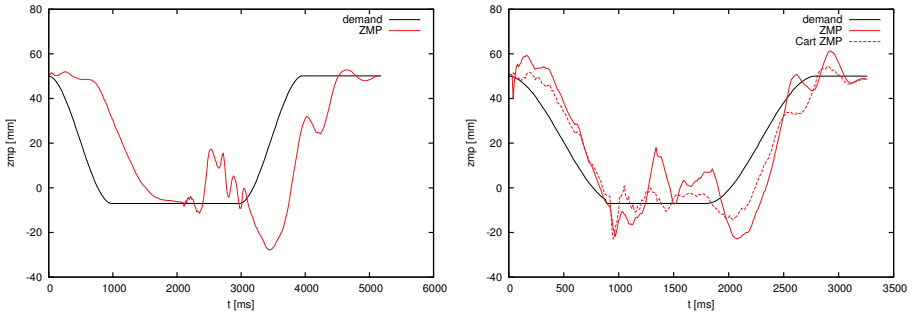


Fig. 4. ZMP tracking in y-direction of the LQR (left) and the Cart-Table controller (right). The bathtub-like plots are the ZMP references to be tracked by the robot, which leans to the side, kicks, and straightens up again. For the Cart-Table controller, the ZMP calculated using the Cart-Table model is also plotted as a dashed line.

4 Results

To evaluate the controllers, we performed different kicks and plotted the resulting estimated ZMPs. A typical example is pictured in Fig. 4. As expected, the Cart-Table controller responds more quickly to changes to the ZMP. As a consequence, the overall kick time is shorter, since the motion engine has to wait less for the robot to become balanced before and after the kick foot is moved. No matter which controller is used, the ZMP gets considerably perturbed while the kick foot is moved. To assess how well the Cart-Table model fits, we also plotted the ZMP as it would result from that model. Although the Cart-Table does not consider rotational effects, the Cart-Table-ZMP matches the ZMP calculated using inverse dynamics quite well. Overall, if the fixed height of the COM is not an issue, the Cart-Table controller appears to be the better choice.

To evaluate the kick foot trajectory, we let the NAO perform kicks in different directions using the Cart-Table controller: kicking straight ahead, and at 25°, 50°, and 80° angles. The results for the 50° kicks and the 80° kicks are plotted in Fig. 5. The trajectory duration (900ms) and the demanded kick foot speed ($1\frac{\mathrm{m}}{\mathrm{s}}$) were the same for all trials.

On average, the demanded angle is almost reached by the 80° kicks. The kicks performed with the left foot scatter with standard deviation $\sigma_{80,\mathrm{left}} = 9°$ around the mean $\mu_{80,\mathrm{left}} = 83°$, the right kicks with $\sigma_{80,\mathrm{right}} = 10°$ around $\mu_{80,\mathrm{right}} = 75°$. The results of the 50° kicks are a lot worse. On average, the kicks reach $\mu_{50} = 32°$. Only some left kicks come close to the demanded angle, but also show a high standard deviation of $\sigma_{50,\mathrm{left}} = 13°$ compared to $\sigma_{50,\mathrm{right}} = 7°$.

We observed similar results for the 25° kicks. As the 50° kicks, on average they only reached roughly half of the demanded angle. The straight kick was the most accurate with $\mu_{0,\mathrm{left}} = -3.6°$ and $\mu_{0,\mathrm{right}} = -3°$, scattering only with $\sigma_{0,\mathrm{left}} = 3.7°$ and $\sigma_{0,\mathrm{right}} = 3.1°$.

We also measured the distances reached by the kicks. The straight kick reached 268cm on average, with 51cm standard deviation. By reducing the tra-

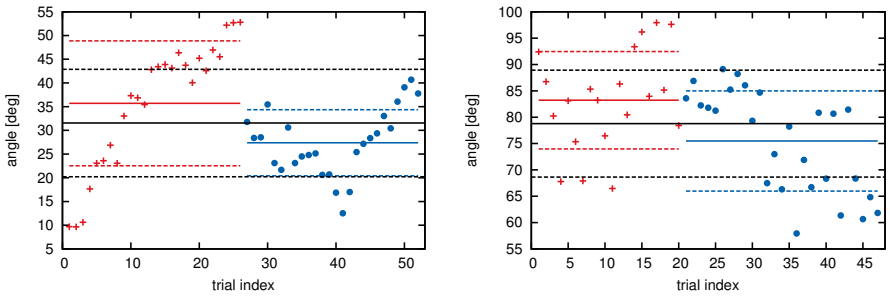


Fig. 5. The measured angles of the 50° kicks (left) and the 80° kicks (right). The means are plotted as solid lines, the dashed lines are one standard deviation away from the corresponding mean. Angles of kicks performed with left are plotted with crosses on the left side of each diagram, kicks performed with right on the right sides with circles.

jectory duration to 750ms and increasing the kick foot speed to $1.3\frac{\mathrm{m}}{\mathrm{s}}$, the average distance increased to 395cm, with 82cm standard deviation. The maximum distance reached was 560cm.

5 Conclusion

In the previous sections, we presented a method to calculate a smooth trajectory for the kick foot given the ball position, the kick direction, the desired duration for the kick and the speed the kick foot should have when hitting the ball, and two methods to execute the trajectory by a dynamically balanced robot.

The results are mixed. On the one hand, the angles reached for 80° and the straight kick are acceptable, while the kicks for 25° and 50° undershoot considerably. This is probably due to our approximation of the foot contour being not accurate enough.

We also observed large variations both in the angles and distances reached. As to why these variations happen, we currently suspect that minor variations in the ball positions have major effects on the resulting trajectories. In addition, small deviations from the trajectory probably cause large deviations of the kick direction if the ball is hit with a highly curved region of the foot.

We conclude that one can use the methods described to perform kicks of a soccer robot such as the NAO without prior modeling of trajectories of the robot's limbs or relying on key-framing methods, in particular if we can fix the variation issue.

Acknowledgement

We would like to thank the members of the team B-Human for providing the software framework for this work. This work has been partially funded by DFG through SFB/TR 8 "Spatial Cognition".

References

1. Belanger, P.: Estimation of angular velocity and acceleration from shaft encoder measurements. In: Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on. pp. 585 –592 vol.1 (May 1992)
2. de Boor, C.: On calculating with B-splines. Journal of Approximation Theory 6(1), 50 – 62 (1972)
3. Czarnetzki, S., Kerner, S., Klagges, D.: Combining key frame based motion design with controlled movement execution. In: Baltes, J., Lagoudakis, M., Naruse, T., Ghidary, S. (eds.) RoboCup 2009: Robot Soccer World Cup XIII, Lecture Notes in Computer Science, vol. 5949, pp. 58–68. Springer Berlin Heidelberg (2010)
4. Doya, K.: Bayesian Brain: Probabilistic Aproaches to Neural Coding. Computational Neuroscience Series, MIT Press (2007)
5. Fang, A.C., Pollard, N.S.: Efficient synthesis of physically valid human motion. ACM Trans. Graph. 22(3), 417–426 (Jul 2003)
6. Farin, G.: Curves and Surfaces for CAGD: A Practical Guide. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling (2002)
7. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer (2008)
8. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on. vol. 2, pp. 1620 – 1626 vol.2 (Sept 2003)
9. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering 82, 35–45 (1960)
10. Katayama, T., Ohki, T., Inoue, T., Kato, T.: Design of an optimal controller for a discrete-time system subject to previewable demand. International Journal of Control 41(3), 677–699 (1985)
11. Müller, J., Laue, T., Röfer, T.: Kicking a ball – modeling complex dynamic motions for humanoid robots. In: del Solar, J.R., Chown, E., Ploeger, P.G. (eds.) RoboCup 2010: Robot Soccer World Cup XIV. Lecture Notes in Artificial Intelligence, vol. 6556, pp. 109–120. Springer (2011)
12. Röfer, T., Laue, T., Müller, J., Bösche, O., Burchardt, A., Damrose, E., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Rieskamp, A., Schreck, A., Sieverdingbeck, I., Worch, J.H.: B-Human Team Report and Code Release 2009 (2009), only available online: `http://www.b-human.de/file_download/26/bhuman09_coderelease.pdf`
13. Tilgner, R., Reinhardt, T., Borkmann, D., Kalbitz, T., Seering, S., Fritzsche, R., Vitz, C., Unger, S., Eckermann, S., Müller, H., Bellersen, M., Engel, M., Wünsch, M.: Team research report 2011 Nao-Team HTWK Leipzig (2011), available online: `http://robocup.imn.htwk-leipzig.de/documents/report2011.pdf`
14. Vukobratović, M., Borovac, B.: Zero-moment point — thirty five years of its life. International Journal of Humanoid Robotics 01(01), 157–173 (2004)
15. Xu, Y., Mellmann, H.: Adaptive motion control: Dynamic kick for a humanoid robot. In: Dillmann, R., Beyerer, J., Hanebeck, U., Schultz, T. (eds.) KI 2010: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 6359, pp. 392–399. Springer Berlin Heidelberg (2010)