A closed-form solution for real-time ZMP gait generation and feedback stabilization

Russ Tedrake[1], Scott Kuindersma[2], Robin Deits[1], Kanako Miura[3]

Abstract— **Here we present a closed-form solution to the continuous time-varying linear-quadratic regulator problem for zero-moment point (ZMP) tracking. This generalizes previous analytical solutions for gait generation by allowing "soft" tracking (with a quadratic cost) of the desired ZMP, and by providing the feedback gains for the resulting time-varying optimal controller. This enables fast $\mathcal{O}(n)$ computation, with n the number of piecewise polynomial segments in the desired ZMP trajectory. Results are presented using the Atlas humanoid robot where dynamic walking is achieved by recomputing the optimal controller online.**

I. Introduction

Zero-moment point (ZMP) approaches have become a standard tool for achieving dynamic balance for walking and manipulation in humanoid robots. Conceptually, the ZMP defines the point on the ground plane at which the moment produced by inertial and gravitational forces is parallel to the surface normal (i.e. the robot is not tipping) [1]. The success of ZMP stems from the fact that, under certain reasonable assumptions, the ZMP dynamics are linear and, given a sequence of desired footsteps, prescribing ZMP trajectories can be done efficiently with trajectory optimization or heuristic methods.

ZMP tracking controllers are often implemented on robots by directly tracking the sensed center of pressure, which corresponds to the ZMP when the ground is flat, or stabilizing the robot's center of mass (COM) along a trajectory consistent with the planned ZMP trajectory. Computing this COM trajectory is sometimes referred to as the walking pattern generation problem [2]. Various methods have been proposed to design such controllers and to compute COM trajectories given ZMP trajectories (Section II-A).

In this paper we show that an optimal time-varying linear-quadratic regulator (LQR) for the continuous ZMP trajectory has a closed-form solution that can be computed efficiently with an iterative algorithm. The COM trajectory corresponding to the reference ZMP trajectory can be solved for in a similar manner. We describe an implementation on the Boston Dynamics Atlas that demonstrates online ZMP replanning and stabilization with sub-millisecond computation times.

*This work was supported by DARPA via Air Force Research Laboratory award FA8750-12-1-0321, the Office of Naval Research via awards N00014-12-1-0071 and N00014-10-1-0951, NSF awards IIS-0746194 and IIS-1161909, MIT, and MIT CSAIL. Robin Deits is supported by the Fannie and John Hertz Foundation.

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, {russt,rdeits}@csail.mit.edu

[2]Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, scottk@seas.harvard.edu

[3]Humanoid Research Group, Intelligent Systems Research Institute, National Institute of Advanced Industrial Science & Technology (AIST), 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan, kanako.miura@aist.go.jp

II. Background

In this section we outline related work on ZMP planning and controller design for bipedal walking robots and briefly describe the linear ZMP dynamics and trajectory representation.

A. Related work

There are multiple examples in the literature of algorithms that efficiently compute COM trajectories, ZMP feedback controllers, or both. Harada et al. [3] derived an analytical expression for the COM trajectory given a piecewise polynomial ZMP trajectory. In this solution, the ZMP tracking was exact, so care is required in designing desired ZMP trajectories to avoid large COM motions. Kajita et al. [2] proposed a preview control approach where a discrete-time, infinite-horizon LQR problem was solved numerically to stabilize the ZMP. This method required considerably more computation, but generalized the specification as the optimization of a quadratic cost which could balance ZMP tracking against COM acceleration, giving more robust COM trajectory output. Urata et al. [4] observed that the analytical results of [3] could be interpretted as the limit where the input cost $\mathbf{R} \to 0$ (the input in this case is the COM jerk), and provided an explicit solution for the Riccati equation. Here we demonstrate that the full preview control approach has a closed-form solution (no limiting arguments required) and give an efficient algorithm which outputs the feedback gains as well as the nominal COM trajectory.

Wieber et al. [5], [6] have proposed sparse model-predictive control (MPC) algorithms that solve finite-horizon formulations of the discrete-time ZMP LQR problem. Feng et al. [7] use differential dynamic programming (DDP) to compute and stabilize COM trajectories online for receding horizon control. In our approach we solve the full continuous-time ZMP LQR problem over long (tens of seconds) time horizons online, which would permit easy integration with sophisticated receding horizon ZMP optimization algorithms.

B. ZMP Dynamics

The planar COM and ZMP dynamics of a legged rigid body system on flat ground can be written in state space form as

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{u} \\ &= \begin{bmatrix} 0_{2\times2} & \mathbf{I}_{2\times2} \\ 0_{2\times2} & 0_{2\times2}\end{bmatrix}\mathbf{x}+\begin{bmatrix} 0_{2\times2} \\ \mathbf{I}_{2\times2}\end{bmatrix}\mathbf{u} \qquad (1)\\ \mathbf{y} &= \mathbf{C}\mathbf{x}+\mathbf{D}(\mathbf{x},\mathbf{u}) \\ &= \begin{bmatrix} \mathbf{I}_{2\times2} & 0_{2\times2}\end{bmatrix}\mathbf{x}+\frac{-z_{\text{com}}}{\ddot{z}_{\text{com}}+g}\mathbf{I}_{2\times2}\mathbf{u}, \qquad (2)\end{aligned}$$

where $\mathbf{x} = [x_{\text{com}}, y_{\text{com}}, \dot{x}_{\text{com}}, \dot{y}_{\text{com}}]^T$, $\mathbf{u} = [\ddot{x}_{\text{com}}, \ddot{y}_{\text{com}}]^T$, $\mathbf{y} = [x_{\text{zmp}}, y_{\text{zmp}}]^T$, g is a constant gravitational acceleration, and z_{com} is the COM height. Note that unlike the formulations commonly found in the literature (e.g., [2], [4]), we do not include the 3rd-order derivatives of the COM. Assuming the COM height, z_{com}, remains constant, the term $\mathbf{D}(\mathbf{x},\mathbf{u})$ becomes $\mathbf{D}\mathbf{u}$, and the ZMP outputs become linear (resulting in the well-known linear inverted pendulum dynamics). In practice, this is often a reasonable assumption to make despite violations that occur during operation. We use this simplification to derive the closed-form LQR solution in Section III.

C. Piecewise-Polynomial ZMP Trajectories

Reference ZMP trajectories are typically defined with respect to a sequence of desired footsteps. One approach is to use a simple piecewise linear trajectory, where the centers of each footstep define the knots and the timing is a function of the desired walking velocity. More generally, we assume that desired ZMP trajectories, $\mathbf{y}_d(t)$, can be described by *continuous* piecewise polynomial of degree k with n breaks at t_j (with $t_0 = 0$ and $t_n = t_f$):

$$\mathbf{y}_d(t) = \sum_{i=0}^{k} \mathbf{c}_{j,i}(t-t_j)^i, \qquad (3)$$

for $j = 0, ..., n-1$ and $\forall t \in [t_j, t_{j+1})$. Figure 1 illustrates example piecewise linear and cubic spline ZMP trajectories defined with respect to a sequence of planned footsteps.

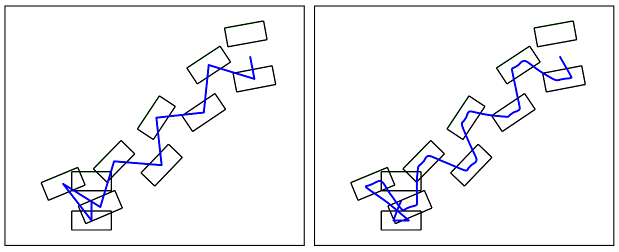


Fig. 1. Reference ZMP trajectories for a footstep plan. The ZMP trajectory (blue line) passes through each stance foot in sequence and can be represented as a piecewise linear function (left), cubic spline (right), or other piecewise polynomial.

III. LQR Design

We formulate the optimal ZMP tracking controller by solving a continuous-time LQR problem. Given desired ZMP trajectory, $\mathbf{y}_d(t)$, we formulate:

$$\begin{aligned}\underset{\mathbf{u}(t)}{\text{minimize}} \quad & \int_0^\infty \left(\|\mathbf{y}(t)-\mathbf{y}_d(t)\|^2_{\mathbf{Q}} + \|\mathbf{u}(t)\|^2_{\mathbf{R}}\right) dt \\ \text{subject to} \quad & \mathbf{Q} = \mathbf{Q}^T > 0 \\ & \mathbf{R} = \mathbf{R}^T > 0 \qquad (4)\\ & \mathbf{y}_d(t) = \mathbf{y}_d(t_f), \quad \forall t \ge t_f \\ & \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t)+\mathbf{B}\mathbf{u}(t) \\ & \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t)+\mathbf{D}\mathbf{u}(t)\end{aligned}$$

and initial conditions $\mathbf{x}(0) = \mathbf{x}_0$. Here $\mathbf{Q}$ and $\mathbf{R}$ explicitly trade off ZMP tracking performance against cost of accelerating the COM. Note that we have placed a more restrict requirement on $\mathbf{Q}$ that it be positive definite. As shown below, this is important to guarantee that our closed-form solution exists. Observing the third constraint in (4), this problem can be rewritten with a cost on state *in coordinates relative to the final conditions*:

$$\begin{aligned}\bar{\mathbf{x}}(t) &= \mathbf{x}(t) - \begin{bmatrix}\mathbf{y}_d(t_f) \\ 0_{2\times1}\end{bmatrix} \qquad (5)\\ \bar{\mathbf{y}}_d(t) &= \mathbf{y}_d(t)-\mathbf{y}_d(t_f). \qquad (6)\end{aligned}$$

We then have the LQR problem:

$$\begin{aligned}\underset{\mathbf{u}(t)}{\text{minimize}} \quad & \int_0^\infty g(\bar{\mathbf{x}}(t),\mathbf{u}(t))dt \\ \text{subject to} \quad & \mathbf{Q} = \mathbf{Q}^T > 0 \\ & \mathbf{R} = \mathbf{R}^T > 0 \qquad (7)\\ & \mathbf{y}_d(t) = \mathbf{y}_d(t_f), \quad \forall t \ge t_f \\ & \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t)+\mathbf{B}\mathbf{u}(t) \\ & \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t)+\mathbf{D}\mathbf{u}(t)\end{aligned}$$

where

$$\begin{aligned}g(\bar{\mathbf{x}}(t),\mathbf{u}(t)) = \; & \bar{\mathbf{x}}^T(t)\mathbf{Q}_1\bar{\mathbf{x}}(t) + \bar{\mathbf{x}}^T(t)\mathbf{q}_2(t) + \qquad (8)\\ & q_3(t) + \mathbf{u}^T(t)\mathbf{R}_1\mathbf{u}(t) + \\ & \mathbf{u}(t)^T\mathbf{r}_2(t) + 2\bar{\mathbf{x}}^T(t)\mathbf{N}\mathbf{u}(t)\end{aligned}$$

and

$$\begin{aligned}\mathbf{Q}_1 &\equiv \mathbf{C}^T\mathbf{Q}\mathbf{C} \qquad (9)\\ \mathbf{q}_2(t) &\equiv -2\mathbf{C}^T\mathbf{Q}\bar{\mathbf{y}}_d(t) \qquad (10)\\ q_3(t) &\equiv \|\bar{\mathbf{y}}_d(t)\|^2_{\mathbf{Q}} \qquad (11)\\ \mathbf{R}_1 &\equiv \mathbf{R}+\mathbf{D}^T\mathbf{Q}\mathbf{D} \qquad (12)\\ \mathbf{r}_2(t) &\equiv -2\mathbf{D}\mathbf{Q}\bar{\mathbf{y}}_d(t) \qquad (13)\\ \mathbf{N} &\equiv \mathbf{C}^T\mathbf{Q}\mathbf{D}. \qquad (14)\end{aligned}$$

Note that this implies that $\lim_{t\to\infty}\bar{\mathbf{x}}(t) = 0$ in order for the cost to be finite. From standard LQR theory [8], we know that the optimal cost-to-go for this problem has the general form

$$J(\bar{\mathbf{x}}(t),t) = \bar{\mathbf{x}}^T(t)\mathbf{S}_1(t)\bar{\mathbf{x}}(t) + \bar{\mathbf{x}}^T(t)\mathbf{s}_2(t) + s_3(t). \qquad (15)$$

The optimal controller is defined as

$$\mathbf{u}^*(t) = -\mathbf{R}_1^{-1}(\mathbf{N}_B\bar{\mathbf{x}}(t) + \mathbf{r}_s(t)), \qquad (16)$$

where $\mathbf{N}_B = \mathbf{N}^T + \mathbf{B}^T\mathbf{S}_1$ and $\mathbf{r}_s(t) = \frac{1}{2}(\mathbf{r}_2(t) + \mathbf{B}^T\mathbf{s}_2(t))$ and the $\mathbf{S}_1$, $\mathbf{s}_2$, and s_3 terms are computed via the Riccati differential equation:

$$\begin{aligned}
\dot{\mathbf{S}}_1 &= -\mathbf{Q}_1 + \mathbf{N}_B^T\mathbf{R}_1^{-1}\mathbf{N}_B - \mathbf{S}_1\mathbf{A} - \mathbf{A}^T\mathbf{S}_1 && (17)\\
\dot{\mathbf{s}}_2(t) &= -\mathbf{q}_2(t) + 2\mathbf{N}_B^T\mathbf{R}_1^{-1}\mathbf{r}_s(t) - \mathbf{A}^T\mathbf{s}_2(t) && (18)\\
\dot{s}_3(t) &= -q_3(t) + \mathbf{r}_s(t)^T\mathbf{R}_1^{-1}\mathbf{r}_s(t). && (19)
\end{aligned}$$

The key observation here is that there are *no time-dependent terms in* (17), so $\mathbf{S}_1$ is a constant, given by the steady-state solution of the algebraic Riccati equation. Furthermore, it depends on $\mathbf{Q}$, $\mathbf{R}$, and z_{com}, but *does not* depend on the ZMP trajectory, so it can be computed offline and simply used as a constant at runtime. The optimal feedback controller (16) can therefore be expressed as

$$\mathbf{u}^*(t) = \mathbf{K}_1\bar{\mathbf{x}}(t) + \mathbf{k}_2(t), \qquad (20)$$

where the feedback matrix $\mathbf{K}_1$ is a constant and

$$\mathbf{k}_2(t) = -\mathbf{R}_1^{-1}\left(\frac{1}{2}\mathbf{B}^T\mathbf{s}_2(t) - \mathbf{D}\mathbf{Q}\bar{\mathbf{y}}_d(t)\right). \qquad (21)$$

We discuss an efficient method for computing the $\mathbf{s}_2(t)$ trajectory in the next section.

A. Computing $\mathbf{s}_2(t)$

The time-varying linear term in the cost-to-go is given by the linear differential equation:

$$\dot{\mathbf{s}}_2(t) = \mathbf{A}_2\mathbf{s}_2(t) + \mathbf{B}_2\bar{\mathbf{y}}_d(t), \quad \mathbf{s}_2(t_f) = 0 \qquad (22)$$

with

$$\begin{aligned}
\mathbf{A}_2 &= \mathbf{N}_B^T\mathbf{R}_1^{-1}\mathbf{B}^T - \mathbf{A}^T && (23)\\
\mathbf{B}_2 &= 2(\mathbf{C}^T - \mathbf{N}_B^T\mathbf{R}_1^{-1}\mathbf{D})\mathbf{Q}. && (24)
\end{aligned}$$

Assuming $\bar{\mathbf{y}}_d(t)$ is described by a continuous piecewise polynomial, this system has an explicit solution given by:

$$\mathbf{s}_2(t) = e^{\mathbf{A}_2(t-t_j)}\boldsymbol{\alpha}_j + \sum_{i=0}^{k}\boldsymbol{\beta}_{j,i}(t-t_j)^i, \qquad (25)$$

for all $t \in [t_j, t_{j+1})$ with $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}_{j.i}$ vector parameters to be solved for. Taking

$$\begin{aligned}
\dot{\mathbf{s}}_2(t) &= \mathbf{A}_2 e^{\mathbf{A}_2(t-t_j)}\alpha_j + \sum_{i=0}^{k}\mathbf{A}_2\boldsymbol{\beta}_{j,i}(t-t_j)^i\\
&\quad + \sum_{i=0}^{k}\mathbf{B}_2\mathbf{c}_{j,i}(t-t_j)^i && (26)\\
&= \mathbf{A}_2 e^{\mathbf{A}_2(t-t_j)}\boldsymbol{\alpha}_j + \sum_{i=1}^{k} i\boldsymbol{\beta}_{j,i}(t-t_j)^{i-1} && (27)
\end{aligned}$$

requires that

$$\begin{aligned}
\boldsymbol{\beta}_{j,0} &= \mathbf{B}_2\mathbf{c}_0\\
\mathbf{A}_2\boldsymbol{\beta}_{j,i} + \mathbf{B}_2\mathbf{c}_{j,i} &= (i+1)\boldsymbol{\beta}_{j,i+1}, \quad i = 0, ..., k-1\\
\mathbf{A}_2\boldsymbol{\beta}_{j,k} + \mathbf{B}_2\mathbf{c}_{j,k} &= 0.
\end{aligned}$$

The remaining term for the controller (20) can be straightforwardly computed given the solution to $\mathbf{s}_2(t)$:

$$\mathbf{k}_2(t) = -\frac{1}{2}\mathbf{R}_1^{-1}\mathbf{B}^T e^{\mathbf{A}_2(t-t_j)}\boldsymbol{\alpha}_j + \sum_{i=0}^{k}\boldsymbol{\gamma}_{j,i}(t-t_j)^i, \qquad (28)$$

where

$$\boldsymbol{\gamma}_{j,i} = \mathbf{R}_1^{-1}\mathbf{D}\mathbf{Q}\mathbf{c}_{j,i} - \frac{1}{2}\mathbf{R}_1^{-1}\mathbf{B}^T\boldsymbol{\beta}_{j,i}. \qquad (29)$$

Algorithm 1 solves for the parameters of $\mathbf{s}_2(t)$ and $\mathbf{k}_2(t)$ backwards in time. The algorithm complexity is $\mathcal{O}(nk)$. Therefore, if the trajectory is piecewise linear ($k = 1$) with one segment per footstep, the computation time is linear in the number of footsteps.

For this algorithm to work, $\mathbf{A}_2$ must be full rank. We can rewrite $\mathbf{A}_2$ in block diagonal form as

$$\mathbf{A}_2 = \begin{bmatrix} 0 & (\mathbf{S}_{1,2} + \mathbf{Q}\mathbf{D})\mathbf{R}_1^{-1} \\ -\mathbf{I} & \mathbf{S}_{1,3}^T\mathbf{R}_1^{-1} \end{bmatrix}, \qquad (30)$$

where $\mathbf{S}_1 = \begin{bmatrix} \mathbf{S}_{1,1} & \mathbf{S}_{1,2} \\ \mathbf{S}_{1,2}^T & \mathbf{S}_{1,3} \end{bmatrix}$. The inverse can be expressed as,

$$\mathbf{A}_2^{-1} = \begin{bmatrix} \left[(\mathbf{S}_{1,2} + \mathbf{Q}\mathbf{D})\mathbf{R}_1^{-1}\right]^{-1}\mathbf{S}_{1,3}^T\mathbf{R}_1^{-1} & -\mathbf{I} \\ \left[(\mathbf{S}_{1,2} + \mathbf{Q}\mathbf{D})\mathbf{R}_1^{-1}\right]^{-1} & 0 \end{bmatrix}. \qquad (31)$$

We then have that $\mathbf{A}_2$ is full rank if and only if $(\mathbf{S}_{1,2} + \mathbf{Q}\mathbf{D})$ is invertible. Recall that $\mathbf{Q} = \mathbf{Q}^T > 0$ and $\mathbf{D}$ is a positive scaling of the identity matrix. Therefore, $\mathbf{A}_2$ is invertible if $\mathbf{S}_{1,2}$ is full rank. Since $\mathbf{S}_1$ is a constant, this property can be ensured at design time (empirically $\mathbf{S}_{1,2}$ is always full rank for admissible costs, but a formal statement is difficult to make since the continuous algebraic Riccati equation does not have a closed-form solution).

```
Data: $\mathbf{A}_2$, $\mathbf{B}_2$, degree $k$ piecewise polynomial $\bar{\mathbf{y}}_d(t)$
      with $n$ breaks
Result: $\boldsymbol{\alpha}_j$, $\boldsymbol{\beta}_{i,j}$, $\boldsymbol{\gamma}_{i,j}$, $\forall j \in \{1,\dots,n\}$, $\forall i \in \{0,\dots,k\}$
for $j = n,\dots,1$ do
    $\boldsymbol{\beta}_{j,k} = -\mathbf{A}_2^{-1}\mathbf{B}_2\mathbf{c}_{j,k}$;
    $\boldsymbol{\gamma}_{j,k} = \mathbf{R}_1^{-1}\mathbf{D}\mathbf{Q}\mathbf{c}_{j,k} - \frac{1}{2}\mathbf{R}_1^{-1}\mathbf{B}^T\boldsymbol{\beta}_{j,k}$;
    for $i = k-1,\dots,0$ do
        $\boldsymbol{\beta}_{j,i} = \mathbf{A}_2^{-1}\left((i+1)\boldsymbol{\beta}_{j,i+1} - \mathbf{B}_2\mathbf{c}_{j,i}\right)$;
        $\boldsymbol{\gamma}_{j,i} = \mathbf{R}_1^{-1}\mathbf{D}\mathbf{Q}\mathbf{c}_{j,i} - \frac{1}{2}\mathbf{R}_1^{-1}\mathbf{B}^T\boldsymbol{\beta}_{j,i}$;
    end
    if $j = n$ then
        $\boldsymbol{\alpha}_j = e^{\mathbf{A}_2(t_n-t_j)}\backslash\left(-\sum_{i=0}^{k-1}\boldsymbol{\beta}_{j,i}(t-t_j)^i\right)$;
    else
        $\boldsymbol{\alpha}_j = e^{\mathbf{A}_2(t_{j+1}-t_j)}\backslash$
            $\left(\boldsymbol{\alpha}_{j+1} + \boldsymbol{\beta}_{j+1,1} - \sum_{i=0}^{k-1}\boldsymbol{\beta}_{j,i}(t-t_j)^i\right)$;
    end
end
```

Algorithm 1: Solve for parameters of $\mathbf{s}_2(t)$ and $\mathbf{k}_2(t)$.

B. Computing the COM Trajectory

It is often useful to compute the COM trajectory, $\mathbf{x}(t)$, that corresponds to the desired ZMP trajectory to design consistent whole-body walking motions. The COM trajectory can be solved for in a similar manner as the affine terms in the optimal controller (20).

Substituting (20) into the dynamics (1), we have

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\left(\mathbf{K}_1\mathbf{x}(t) + \mathbf{k}_2(t)\right) \\ &= (\mathbf{A} + \mathbf{B}\mathbf{K}_1)\mathbf{x}(t) + \mathbf{B}\mathbf{k}_2(t). \end{aligned} \quad (32)$$

Since the solution $\mathbf{k}_2(t)$ is the result of another linear system cascaded in front of this one, it is conceptually simplest to solve them jointly. We define:

$$\mathbf{z}(t) = \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{s}_2(t) \end{bmatrix} \quad (33)$$

$$\dot{\mathbf{y}}(t) = \mathbf{A}_z\mathbf{z}(t) + \mathbf{B}_z\bar{\mathbf{y}}_d(t), \quad (34)$$

where

$$\mathbf{A}_z = \begin{bmatrix} \mathbf{A} + \mathbf{B}\mathbf{K}_1 & -\frac{1}{2}\mathbf{B}\mathbf{R}_1^{-1}\mathbf{B}^T \\ 0 & \mathbf{A}_2 \end{bmatrix} \quad (35)$$

$$\mathbf{B}_z = \begin{bmatrix} \mathbf{B}\mathbf{R}_1^{-1}\mathbf{D}\mathbf{Q} \\ \mathbf{B}_2 \end{bmatrix}. \quad (36)$$

The solution to this system, as in the above, has the general form:

$$\mathbf{z}(t) = e^{\mathbf{A}_z(t-t_j)}\mathbf{a}_j + \sum_{i=0}^{k} \mathbf{b}_{j,i}(t-t_j)^i. \quad (37)$$

Algorithm 2 solves for the coefficients of (37) forward in time. Note that it is possible to reuse the parameters, $\boldsymbol{\beta}_{i,j}$, returned from Algorithm 1 and thereby only solve for the top half of $\mathbf{b}_{j,i}$.

```
Data: $\mathbf{x}(0)$, $\mathbf{A}_z$, $\mathbf{B}_z$, degree $k$ piecewise polynomial
      $\bar{\mathbf{y}}_d(t)$ with $n$ breaks
Result: $\mathbf{a}_j$, $\mathbf{b}_{i,j}$, $\forall j \in \{1,\dots,n\}$, $\forall i \in \{0,\dots,k\}$
$\mathbf{x} = \mathbf{x}(0)$;
for $j = 1,\dots,n$ do
    $\mathbf{b}_{j,k} = -\mathbf{A}_z^{-1}\mathbf{B}_z\mathbf{c}_{j,k}$;
    for $i = k-1,\dots,0$ do
        $\mathbf{b}_{j,i} = \mathbf{A}_z^{-1}\left((i+1)\mathbf{b}_{j,i+1} - \mathbf{B}_z\mathbf{c}_{j,i}\right)$;
    end
    $\mathbf{a}_j = \begin{bmatrix} \mathbf{x} - \mathbf{b}_{j,1} \\ \boldsymbol{\alpha}_j \end{bmatrix}$;
    $\mathbf{x} = \begin{bmatrix} \mathbf{I} \\ 0 \end{bmatrix} e^{\mathbf{A}_z(t_{j+1}-t_j)}\mathbf{a}_j + \sum_{i=0}^{k-1}\mathbf{b}_{j,i}(t_{j+1}-t_j)^i$;
end
$\mathbf{b}_{j,1}[1:2] = \mathbf{b}_{j,1}[1:2] + \mathbf{y}(t_f)$;
```

Algorithm 2: Solve for the COM trajectory, $\mathbf{x}(t)$.

IV. Experiments

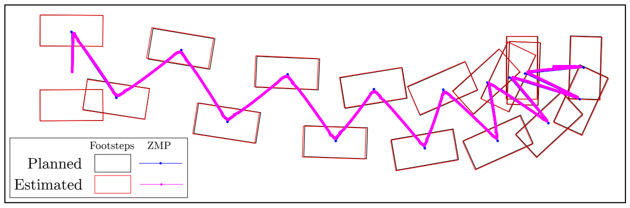
Below we describe an application to ZMP stabilization and trajectory replanning with the Atlas humanoid robot designed by Boston Dynamics (Figure 2). More information about the controller implementation [9], [10] and footstep planner [11], [12] are available in previous publications. In addition, code that implements Algorithms 1 and 2, along with a variety of Atlas walking examples, is available in the Drake software library [13].

Fig. 2. The Atlas humanoid built by Boston Dynamics, Inc.

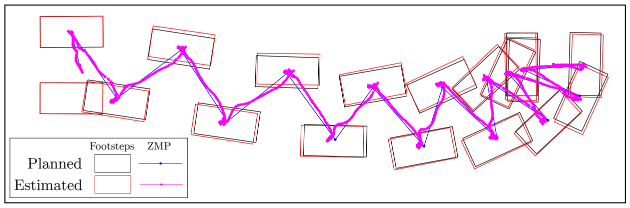
A. Atlas ZMP tracking and online adjustment

To use the ZMP tracker on Atlas, we first optimize a footstep plan from the robot's current pose to a goal pose. From this footstep plan we generate a reference ZMP trajectory, $\bar{\mathbf{y}}_d(t)$, that linearly interpolates between the centers of the desired foot locations. We compute $\mathbf{S}_1$ from the algebraic Riccati equation once offline and solve for $\mathbf{s}_2$ online using Algorithm 1. The controller attempts to stabilize the reference ZMP trajectory by descending the cost-to-go function (15) subject to the instantaneous dynamics and input constraints of the system in a quadratic program (QP) [10]. This separation of planning and execution is sufficient when the desired foot locations and COM trajectory can be tracked accurately, which is the case in simulation or in hardware when walking at moderate speeds on flat ground.

However, the real world inevitably causes some error in the positioning of the feet, and that error increases as the robot moves its feet faster or traverses uneven terrain. Error in the foot locations can result in the reference ZMP trajectory moving to the edge of the foot or even leaving the support polygon entirely. Fortunately, the closed-form solution for $\mathbf{s}_2(t)$ allows the system to recover from foot placement error online. Any time a foot is in contact with the ground, we adjust the corresponding foot placement in the footstep plan to match the current estimated position of that foot. From this modified foot placement we compute a new ZMP trajectory, $\bar{\mathbf{y}}_d(t)$, and a new solution for $\mathbf{s}_2(t)$. We do not need to recompute $\mathbf{S}_1$ since it is time invariant and does not depend on $\bar{\mathbf{y}}_d(t)$. For a ZMP trajectory corresponding to 16 footsteps (with 35 linear ZMP trajectory segments), this entire process requires approximately 300 microseconds on an Intel i7 at 3 GHz, allowing us to recompute the optimal controller alongside our control loop.



(a) Simulation



(b) Hardware

Fig. 3. Comparison of desired vs achieved footstep and ZMP tracking in simulation and hardware. In each example, the ZMP trajectory and optimal tracking controller are being recomputed online after every step.

We demonstrate our approach with a plan consisting of 16 footsteps during which Atlas moves forward 2 meters and turns 90 degrees to the left. Figure 3 shows the planned and sensed footstep locations and ZMP trajectory for simulation and hardware. The sensed positions of the feet were computed from the forward kinematics of the robot's estimated state, which fuses information from its onboard IMU and leg odometry [10]. The robot's estimated COM position, $\mathbf{x}$, and velocity, $\dot{\mathbf{x}}$, were sampled at 60 Hz and filtered using LOESS smoothing with a span of 30 samples [14]. The smoothed COM trajectory was used to compute COM acceleration, $\ddot{\mathbf{x}}$, from which ZMP position, $\mathbf{y}$, was computed using (2).

To demonstrate the effectiveness of the online recomputation of $\bar{\mathbf{y}}_d(t)$ and $\mathbf{s}_2(t)$, we intentionally turn down the gains that control the position of the robot's feet by a factor of 10. This results in very poor foot placement tracking, shown in Figure 4. By continually recomputing new ZMP trajectories and optimal controllers for the estimated positions of the feet, the robot is able to complete the entire walking plan despite a mean error of 10 cm between the planned and sensed positions of the feet. Without the online recomputation of $\bar{\mathbf{y}}_d(t)$ and $\mathbf{s}_2(t)$, this level of footstep tracking error causes the robot to fall after the first footstep.

V. Conclusions

We derived a closed-form solution for the optimal time-varying LQR for ZMP trajectories and showed that the closed-form COM trajectory could be computed with few additional operations. Sub-millisecond solve times allow optimal feedback controllers to be recomputed online as the underlying ZMP trajectory changes due to tracking errors or changes in the walking pattern. We described an application to stable walking control on a physical and simulated Atlas humanoid robot.

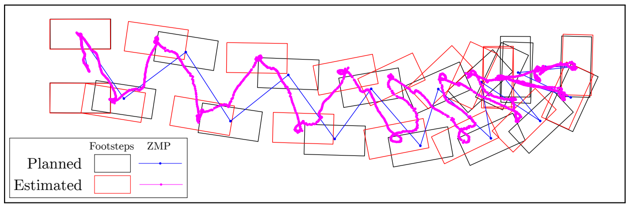


Fig. 4. Walking with intentionally poor foot tracking. The tracking gains on the x and y positions of the feet were reduced by a factor of 10 to introduce substantial error into the footstep positions. By continually recomputing the ZMP trajectory and tracking controller, we maintain stable walking despite this disturbance.

References

[1] P. Sardain and G. Bessonnet, "Forces acting on a biped robot. Center of pressure-zero moment point," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 34, no. 5, pp. 630–637, 2004.

[2] 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 *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, Sept. 2003.

[3] K. Harada, S. Kajita, K. Kaneko, and H. Hirukawa, "An Analytical Method for Real-Time Gait Planning for Humanoid Robots," *International Journal of Humanoid Robotics*, vol. 03, no. 01, pp. 1–19, 2006.

[4] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba, "Online Decision of Foot Placement using Singular LQ Preview Regulation," in *IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, Oct. 2011.

[5] P. B. Wieber, "Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. IEEE, Dec. 2006, pp. 137–142.

[6] D. Dimitrov, A. Sherikov, and P.-B. Wieber, "A sparse model predictive control formulation for walking motion generation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, USA, Sept. 2011, pp. 2292–2299.

[7] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson, "3D walking based on online optimization," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Atlanta, GA, Oct. 2013.

[8] R. Tedrake, "Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832)." Tech. Rep., 2014.

[9] S. Kuindersma, F. Permenter, and R. Tedrake, "An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014.

[10] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for Atlas," *Autonomous Robots (in press)*, 2015.

[11] R. L. H. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Proceedings of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*. Istanbul: Springer International Publishing, 2014, pp. 109–124.

[12] ——, "Footstep Planning on Uneven Terrain with Mixed-Integer Convex Optimization," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 279–286.

[13] R. Tedrake, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2015.

[14] W. G. Jacoby, "Loess: a nonparametric, graphical tool for depicting relationships between variables," *Electoral Studies*, vol. 19, pp. 577–613, 2000.