Biologically Inspired Dead-beat controller for bipedal running in 3D

Johannes Englsberger, Pawel Kozlowski, Christian Ott

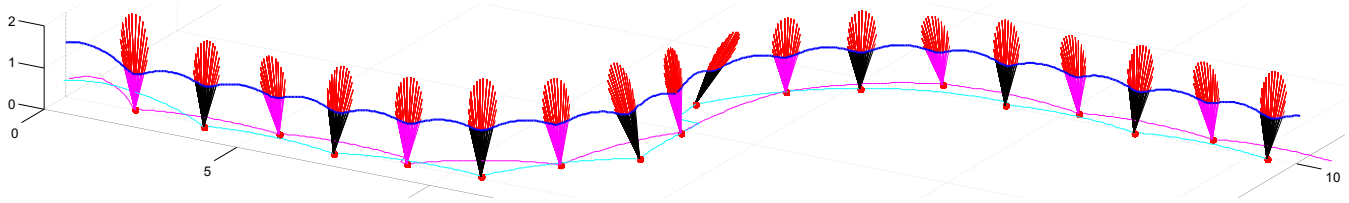

Fig. 1: Bipedal point-mass model runs in 3D based on Biologically Inspired Dead-beat (BID) control.

***Abstract*— This paper introduces a Biologically Inspired Dead-beat (BID) controller for bipedal running in 3D. The controller runs in real-time, is extremely robust against perturbations and allows for versatile running patterns. It is based on the encoding of leg forces and CoM trajectories during stance as polynomial splines, allowing for intuitive and primarily analytical controller design. The performance of the control framework is tested in various simulations for a bipedal point-mass model.**

I. Introduction

Locomotion - as observed in nature - has always aroused enthusiasm and curiosity in human spectators. From an engineering point of view, gaited forms of locomotion - once fully understood - promise highly increased mobility of machines as compared to wheel-based locomotion.

The first efforts in robotic bipedal locomotion have been put in the subdomain of bipedal walking. Over the decades, the field of bipedal walking control has made major progress. Alongside successes in passive dynamics walking [1], one of the major breakthroughs has been the introduction of zero moment point control [2], [3] for bipedal walking. More recently, several successful walking control algorithms have been presented, e.g. [4]–[10], to name but a few. Recently, bipedal walking algorithms have reached a level that is close to actual application in real-world scenarios. Most walking algorithms attempt to keep the robot in a fully controllable state, which facilitates the use of standard control methods.

In contrast, running and hopping by definition contain partially uncontrolled states and are thus seen as challenging tasks. Running provides a number of assets such as high achievable speed and efficiency. It has thus recently been addressed by several research studies. Aside from few exceptions such as [11]–[14], most running algorithms are based on the spring-loaded inverted pendulum (SLIP) [15], [16]. Dadashzadeh et al. [17] present a SLIP-based two-level controller for running simulations of the ATRIAS robot.

The authors are with Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany. E-mail: johannes.englsberger@dlr.de

Vejdani et al. [18] introduce bio-inspired swing leg control for running on ground with unexpected height disturbances. Koepl and Hurst [19] control the stance phase impulse of a planar SLIP model and achieve robust running.

Recently, Wensing and Orin [20] presented an algorithm that computes periodic trajectories of the 3D-SLIP offline and applies an offline computed linearized control law to stabilize the virtual SLIP model around the periodic solutions. The desired leg forces are passed to a whole-body controller and bipedal running of a simulated humanoid robot is achieved. Like many other methods, the SLIP-based approach used for CoM planning and control in [20] comes with two major drawbacks: Firstly, it requires offline computation of periodic SLIP gaits which suffers from the curse of dimensionality. Second, the motions are designed to be periodic, which makes the design of non-periodic running motions such as accelerating and turning a challenge.

These drawbacks are eliminated in this paper. We propose a dead-beat controller, that is real-time capable, allows for versatile running motions and is very robust against external perturbations. It has been inspired by observations from human running experiments (see Fig. 2) and uses polynomial splines to encode the robot's CoM motion and leg forces during stance. Through the use of polynomials, the control design process is very intuitive and comprehensible. One major advantage of the algorithm is that both upcoming foot target locations on the ground are predicted at all times, which facilitates the design of appropriate foot trajectories.

The paper is organized as follows: Section II motivates the use of polynomial splines to approximate observations from human running experiments. Section III introduces our planning and control framework. Section IV presents various simulations for a bipedal point-mass robot with two massless feet. Sections V and VI discuss the proposed control framework's assets and limitations and conclude the paper.

II. Human running experiments as motivation

The main idea in this paper is to *design desired CoM trajectories* that produce *approximately natural GRF profiles*

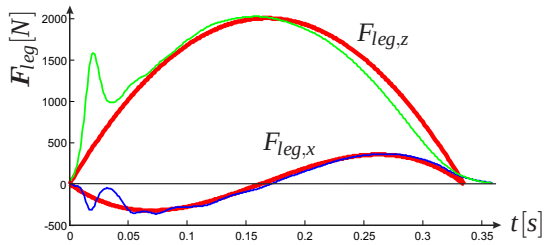

Fig. 2: Comparison of experimentally measured human leg forces (blue/green) and polynomial approximations (red).

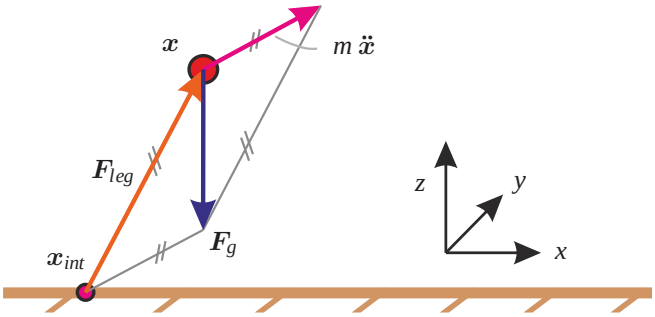

Fig. 3: Forces acting on robot's center of mass (CoM)

while fulfilling several *boundary conditions*. It is well known that some physical template models, such as the SLIP, generate ground reaction forces (GRF) similar to the ones observed in human running. The lack of closed form solutions for the SLIP motivates us to find an alternative way of encoding the leg force ($\boldsymbol{F}_{leg}$, equivalent to GRF) which allows to find closed form solutions to running. Figure 2 shows a typical GRF profile that was recorded during a human running experiment via force plate. The human GRF profiles can be approximated quite well by a polynomial of order 2 in the vertical direction and by a polynomial of order 3 in the $x-$direction. Therefore, our original idea was to approximate the leg force profile during stance via polynomials. The total force $\boldsymbol{F}_{CoM}$ acting on the CoM can be computed from leg force $\boldsymbol{F}_{leg}$ and gravitational force $\boldsymbol{F}_g$ (see Fig. 3)

$$\boldsymbol{F}_{CoM} = \boldsymbol{F}_{leg} + \boldsymbol{F}_g = \boldsymbol{F}_{leg} + m\,\boldsymbol{g} \quad . \tag{1}$$

Here, m is the robot's total mass and $\boldsymbol{g} = [0\;0\;-g]^T$ denotes the gravitational acceleration vector. The constant offset between $\boldsymbol{F}_{CoM}$ and $\boldsymbol{F}_{leg}$ in (1) and Newton's 2nd law (CoM acceleration $\ddot{\boldsymbol{x}} = \frac{\boldsymbol{F}_{CoM}}{m}$) motivate us to use - during stance - a 4th order polynomial to encode the vertical CoM position z and 5th order polynomials to encode the horizontal CoM positions x and y, as this correlates to 2nd and 3rd order polynomials for the CoM accelerations $\ddot{x}$, $\ddot{y}$, $\ddot{z}$ and thus leg forces. This polynomial encoding can be written as:

$$\begin{bmatrix} \sigma(t) \\ \dot{\sigma}(t) \\ \ddot{\sigma}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & t & t^2 & t^3 & t^4 & t^5 \\ 0 & 1 & 2t & 3t^2 & 4t^3 & 5t^4 \\ 0 & 0 & 2 & 6t & 12t^2 & 20t^3 \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{t}_\sigma^T(t) \\ \boldsymbol{t}_{\dot{\sigma}}^T(t) \\ \boldsymbol{t}_{\ddot{\sigma}}^T(t) \end{bmatrix}} \boldsymbol{p}_\sigma, \quad \sigma \in \{x,y,z\} \tag{2}$$

Here, $\boldsymbol{t}_\sigma^T(t)$, $\boldsymbol{t}_{\dot{\sigma}}^T(t)$ and $\boldsymbol{t}_{\ddot{\sigma}}^T(t)$ denote the time-mapping row vectors that - for a given time t - map the polynomial parameter vectors $\boldsymbol{p}_\sigma$ to CoM positions $\sigma(t)$, velocities $\dot{\sigma}(t)$ and accelerations $\ddot{\sigma}(t)$. The last elements of the vectors are greyed out to indicate that they are only used for the horizontal directions, but not for the vertical one.

III. Derivation of Dead-Beat Controller

A. Outlook on our approach for bipedal running control

Running is typically defined as a locomotion pattern, which employs alternate flight and (single leg supporting) stance phases. In this paper, we use a preview of the upcoming two stance and flight phases, as shown in Fig. 4. The desired relative apex and touch-down heights $\Delta z_{apex,des}$ and $\Delta z_{TD,des}$ are used as design parameters. They indicate how high over the floor (at z_{floor}) the apex of the flight curve (i.e. $\dot{z} = 0$) should be and at what CoM height the touch-down (TD) is supposed to happen. Another design parameter, used in this work, is the total stance time T_s, whereas the total flight time T_f results from the boundary conditions chosen in section III-C. To keep track of the current running state, we use a state machine. It switches from flight to stance, if the CoM is below $z_{TD} = z_{floor} + \Delta z_{TD,des}$ and the vertical velocity is negative, and from stance to flight when the total stance time is over. A timer provides the time in stance $t_s \in [0, T_s]$ and the time in flight $t_f \in [0, T_f]$. They are reset at state transitions and provided to the controllers.

Figure 5 shows a computation flow of the proposed running algorithm, which will be derived in the next sections.

B. CoM dynamics during flight

During flight phases, the CoM cannot be controlled, i.e. it follows its natural dynamics (parabolic path through space). For a given time t, the CoM position $\boldsymbol{x}(t) = [x(t), y(t), z(t)]^T$ and velocity $\dot{\boldsymbol{x}}(t) = [\dot{x}(t), \dot{y}(t), \dot{z}(t)]^T$ can be computed as

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \dot{\boldsymbol{x}}_0\, t + \boldsymbol{g}\, \frac{t^2}{2} \quad , \tag{3}$$

$$\dot{\boldsymbol{x}}(t) = \dot{\boldsymbol{x}}_0 + \boldsymbol{g}\, t \, , \tag{4}$$

where $\boldsymbol{x}_0$ and $\dot{\boldsymbol{x}}_0$ are the initial ($t = 0$) CoM position and velocity. One typical task in running control is to achieve a certain apex height. The apex is the highest point in the ballistic flight curve, i.e. the vertical CoM velocity is zero ($\dot{z}_{apex} = 0$). Using this condition and the current vertical CoM velocity $\dot{z}$ instead $\dot{z}_0$ in the third row of (4), we find the current time to apex Δt_{apex} as

$$\Delta t_{apex} = \frac{\dot{z}}{g} \quad . \tag{5}$$

If Δt_{apex} is negative (true for $\dot{z} < 0$), then the CoM is already on the descending path of the ballistic flight curve and thus the time of apex is in the past. In the same way, we find the remaining time until touch-down (TD) as

$$\Delta t_{TD} = \Delta t_{apex} + \sqrt{\Delta t_{apex}^2 + \frac{2}{g}\,(z - z_{TD})} \quad . \tag{6}$$

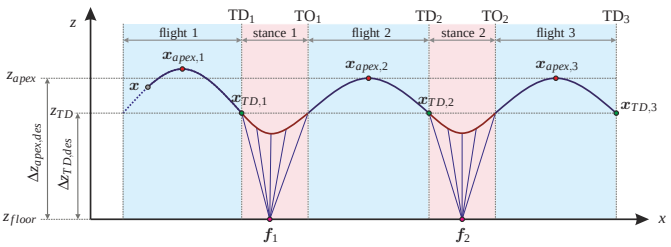

Fig. 4: Preview of upcoming flight and stance phases (planar sketch) - used for design of boundary conditions.

Here, $z_{TD} = z_{floor} + \Delta z_{TD}$ is the CoM height, at which touch-down (flight to stance transition) is previewed to happen. The relative touch-down height Δz_{TD} is computed as

$$\Delta z_{TD} = min(\Delta z_{TD,des},\ z - z_{floor} + \frac{\dot{z}^2}{2g} - \Delta_{apex,TD,min}) \quad . \quad (7)$$

That way, nominally the desired relative touch-down height $\Delta z_{TD,des}$ (*design variable*) is achieved, while for challenging initial conditions or perturbations a minimum positive height difference between apex and touch-down $\Delta_{apex,TD,min}$ is guaranteed and the solution of (6) is assured to be real.

During flight, for every current CoM state ($\boldsymbol{x}$, $\dot{\boldsymbol{x}}$), the predicted touch-down position $\boldsymbol{x}_{TD} = [x_{TD}, y_{TD}, z_{TD}]^T$ and velocity $\dot{\boldsymbol{x}}_{TD} = [\dot{x}_{TD}, \dot{y}_{TD}, \dot{z}_{TD}]^T$ is computed via (3) and (4) by setting $t = \Delta t_{TD}$, $\boldsymbol{x}_0 = \boldsymbol{x}$ and $\dot{\boldsymbol{x}}_0 = \dot{\boldsymbol{x}}$.

C. Vertical planning and boundary conditions

As mentioned above, the vertical CoM trajectory during stance is encoded via a 4th order polynomial, i.e. it has 5 polynomial parameters. These can be derived using 5 boundary conditions. Fig. 4 graphically displays the used preview of upcoming flight and stance sequences and the corresponding boundary conditions. In this work, we make use of four linear vertical boundary conditions:

- initial position equals TD position ($z(t_s = 0) = z_{TD}$)
- initial velocity equals TD velocity ($\dot{z}(t_s = 0) = \dot{z}_{TD}$)
- initial acceleration is minus gravity ($\ddot{z}(t_s = 0) = -g$), i.e. vertical leg force is zero (see (1))
- final acceleration is minus gravity ($\ddot{z}(t_s = T_s) = -g$), i.e. vertical leg force is zero (see (1))

These boundary conditions can be combined to the boundary condition vector $\boldsymbol{b}_z$, which can be linearly related to the vertical polynomial parameter vector $\boldsymbol{p}_z$ via

$$\underbrace{\begin{bmatrix} z_{TD} \\ \dot{z}_{TD} \\ -g \\ -g \end{bmatrix}}_{\boldsymbol{b}_z} = \underbrace{\begin{bmatrix} \boldsymbol{t}_z^T(0) \\ \boldsymbol{t}_{\dot{z}}^T(0) \\ \boldsymbol{t}_{\ddot{z}}^T(0) \\ \boldsymbol{t}_{\ddot{z}}^T(T_s) \end{bmatrix}}_{\boldsymbol{B}_z} \boldsymbol{p}_z \quad . \quad (8)$$

Here, $\boldsymbol{B}_z$ denotes the boundary condition mapping matrix. The general solution of the linear system $\boldsymbol{B}_z\, \boldsymbol{p}_z = \boldsymbol{b}_z$ is

$$\boldsymbol{p}_z = \boldsymbol{p}_{z,0} + \boldsymbol{r}_z\, \tilde{p}_z \, . \quad (9)$$

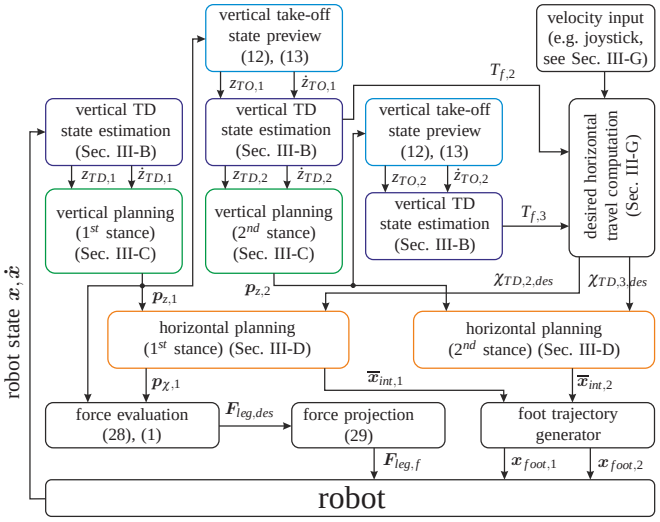

Fig. 5: Computation flow of proposed controller.

The preliminary solution vector $\boldsymbol{p}_{z,0}$ is computed from the boundary conditions via pseudo-inverse of $\boldsymbol{B}_z$ as

$$\boldsymbol{p}_{z,0} = \boldsymbol{B}_z^T (\boldsymbol{B}_z\, \boldsymbol{B}_z^T)^{-1}\, \boldsymbol{b}_z \, . \quad (10)$$

The nullspace base vector $\boldsymbol{r}_z$ in (9) is chosen such that $\boldsymbol{B}_z\, \boldsymbol{r}_z = 0$ holds and thus the whole (one-dimensional) nullspace of $\boldsymbol{B}_z$ is represented by the scalar variable $\tilde{p}_z$. The vector $\boldsymbol{r}_z$ can be computed as

$$\boldsymbol{r}_z = \begin{bmatrix} -\boldsymbol{B}_{z,square}^{-1}\, \boldsymbol{b}_{z,final} \\ 1 \end{bmatrix} \quad , \quad (11)$$

where $\boldsymbol{b}_{z,final}$ is the last column in $\boldsymbol{B}_z$, while $\boldsymbol{B}_{z,square}$ consists of all other columns.

With $\boldsymbol{p}_z$ and (2), the vertical CoM position and velocity at take-off (at the end of the stance phase) are computed as

$$z_{TO} = \boldsymbol{t}_z^T(T_s)\, \boldsymbol{p}_z \quad (12)$$

$$\dot{z}_{TO} = \boldsymbol{t}_{\dot{z}}^T(T_s)\, \boldsymbol{p}_z \quad . \quad (13)$$

Equation (8) encodes the four *linear* previously described vertical boundary conditions. The fifth boundary condition that we aim to fulfill is the apex height of the CoM during the upcoming flight phase (see Fig. 4). Evaluating (5) for $\dot{z} = \dot{z}_{TO}$ and setting $z_0 = z_{TO}$, $\dot{z}_0 = \dot{z}_{TO}$ and $t = \Delta t_{apex}$ in (3, third row), the apex height z_{apex} can be expressed as

$$z_{apex} = z_{TO} + \frac{\dot{z}_{TO}^2}{2g} \, . \quad (14)$$

We are looking for a parameter vector $\boldsymbol{p}_z$ that will result in the desired apex height $z_{apex,des}$, which can be computed as

$$z_{apex,des} = z_{floor} + \Delta z_{apex,des} \, . \quad (15)$$

Combining (9), (12), (13) and (14) and setting $z_{apex} = z_{apex,des}$ results in

$$0 = \frac{t_z^T\, \boldsymbol{r}_z}{2g}\, \tilde{p}_z^2 + (t_z^T\, \boldsymbol{r}_z + \frac{t_z^T\, p_{z,0}\, t_{\dot{z}}^T\, \boldsymbol{r}_z}{g})\, \tilde{p}_z + \frac{(t_{\dot{z}}^T\, p_{z,0})^2}{2g} - z_{apex,des} \quad (16)$$

This is a quadratic equation in the unknown scalar variable $\tilde{p}_z$. It can be shown that the only valid solution (yielding positive vertical take-off velocities) to (16) is

$$\tilde{p}_z = \frac{2\,\dot{z}_{TD} - gT_s - \sqrt{g(gT_s^2 - 4\,\dot{z}_{TD}\,T_s + 8\,(z_{apex,des} - z_{TD}))}}{4\,T_s^3} \quad . \tag{17}$$

Note: finally only (10), (11) and (17) are necessary as inputs for (9), which computes the polynomial parameter vector $\boldsymbol{p}_z$ that fulfills all five desired vertical boundary conditions, i.e. the four linear ones and the quadratic one.

D. Horizontal planning and boundary conditions

Newton's 2nd law $[\ddot{x},\ddot{y},\ddot{z}]^T = \frac{1}{m}\,[f_{com,x}, f_{com,y}, f_{com,z}]^T$ says that each component of the CoM's acceleration only depends on the component of the CoM force $\boldsymbol{F}_{CoM}$ which points in its corresponding direction. Thus, each spatial force component can be derived independently. In our framework, the derivation for the *x*- and *y*-component is equivalent. In all equations in the following derivation, we will use the letter χ to indicate horizontal quantities, i.e. $\chi \in \{x,y\}$.

As in Sec. III-C, we choose - motivated by Fig. 2 - the following four linear horizontal boundary conditions:

- initial position equals TD position ($\chi(t_s = 0) = \chi_{TD}$)
- initial velocity equals TD velocity ($\dot{\chi}(t_s = 0) = \dot{\chi}_{TD}$)
- initial acceleration is zero ($\ddot{\chi}(t_s = 0) = 0$), i.e. horizontal leg force is zero (see (1))
- final acceleration is zero ($\ddot{\chi}(t_s = T_s) = 0$), i.e. horizontal leg force is zero (see (1))

Additionally, it is desirable to specify one more horizontal boundary condition that specifies the horizontal travel (for both horizontal directions *x* and *y*). One simple option would be to specify the horizontal take-off velocity, which might be used by a purely velocity-based controller. In this paper, we present a slightly different approach: We derive the desired CoM position at the second and third touch-down event (see Fig. 4) from a virtual joystick input (see Sec. III-G) and use them as boundary conditions. As compared to pure take-off velocity control, this boundary condition is closer related to CoM position control. It is thus expected to be more suitable for our future running research topics (see Sec. VI). Since the horizontal velocity during flight is constant (and equal to the horizontal take-off velocity $\dot{\chi}_{TO}$), we can propagate the take-off state to the second touch-down state via

$$\chi_{TD,2,des} = \chi_{TO} + T_f\,\dot{\chi}_{TO} = (\boldsymbol{t}_\chi^T(T_s) + T_f\,\boldsymbol{t}_{\dot{\chi}}^T(T_s))\,\boldsymbol{p}_\chi \quad , \tag{18}$$

Here, T_f is obtained by setting $z = z_{TO}$ and $\dot{z} = \dot{z}_{TO}$ in (5) and (6). Note: z_{TO} and $\dot{z}_{TO}$ are computed from the vertical polynomial parameter vector $\boldsymbol{p}_z$. Thus, the vertical boundary conditions are solved before the horizontal ones (see Fig. 5).

Equation (18) - just like the previous four boundary conditions - depends linearly from the horizontal polynomial parameter vector $\boldsymbol{p}_\chi$. The first five horizontal boundary conditions can be combined in the boundary condition vector

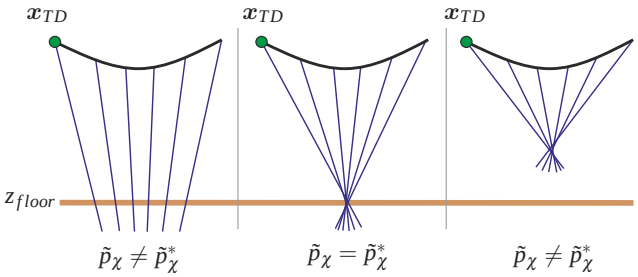

Fig. 6: Effect of $\tilde{p}_\chi$ on force ray focusing (lines of action).

$\boldsymbol{b}_\chi$ that is linearly related to $\boldsymbol{p}_\chi$ via

$$\underbrace{\begin{bmatrix} \chi_{TD} \\ \dot{\chi}_{TD} \\ 0 \\ 0 \\ \chi_{TD,2,des} \end{bmatrix}}_{\boldsymbol{b}_\chi} = \underbrace{\begin{bmatrix} \boldsymbol{t}_\chi^T(0) \\ \boldsymbol{t}_{\dot{\chi}}^T(0) \\ \boldsymbol{t}_{\ddot{\chi}}^T(0) \\ \boldsymbol{t}_{\ddot{\chi}}^T(T_s) \\ \boldsymbol{t}_\chi^T(T_s) + T_f\,\boldsymbol{t}_{\dot{\chi}}^T(T_s) \end{bmatrix}}_{\boldsymbol{B}_\chi} \boldsymbol{p}_\chi \quad . \tag{19}$$

Here, $\boldsymbol{B}_\chi$ denotes the boundary condition mapping matrix. The general solution of the linear system $\boldsymbol{B}_\chi\,\boldsymbol{p}_\chi = \boldsymbol{b}_\chi$ is

$$\boldsymbol{p}_\chi = \boldsymbol{p}_{\chi,0} + \boldsymbol{r}_\chi\,\tilde{p}_\chi \quad . \tag{20}$$

The preliminary solution vector $\boldsymbol{p}_{\chi,0}$ and the nullspace base vector $\boldsymbol{r}_\chi$ are computed in the same way - i.e. via the equivalents of (10) and (11) - as for the vertical direction.

As compared to the vertical direction, the horizontal directions have one more polynomial parameter - and thus one more degree of freedom (DOF) - available. This DOF, which is represented by the remaining free variable $\tilde{p}_\chi$ in (20), has an effect on the geometry of the leg force rays in space (see Fig. 6). Our goal is to find the value for $\tilde{p}_\chi$, which produces the best possible focusing of leg forces, such that these are best feasible for finite-sized (or even point-) feet. We will therefore try to minimize the mean square of the deviation (integral over time) of the leg force ground intersection points[1] x_{int} from their mean value $\overline{x}_{int}$.

To this end, we first compute the time-dependent intersection point $\boldsymbol{x}_{int} = [x_{int}, y_{int}, z_{floor}]$ of the leg force $\boldsymbol{F}_{leg}$ (see Fig. 3) with the floor. For a given time in stance t_s, it's horizontal components $\chi_{int}(t_s)$ are found as

$$\begin{aligned} \chi_{int}(t_s) &= \chi(t_s) - \frac{f_{leg,\chi}(t_s)}{f_{leg,z}(t_s)}\,(z(t_s) - z_{floor}) \\ &= \underbrace{(\boldsymbol{t}_\chi^T(t_s) - \frac{(\boldsymbol{t}_z^T(t_s)\boldsymbol{p}_z - z_{floor})\,\boldsymbol{t}_{\ddot{z}}^T(t_s)}{\boldsymbol{t}_{\ddot{z}}^T(t_s)\boldsymbol{p}_z + g})}_{\boldsymbol{d}^T(t_s)}\,\boldsymbol{p}_\chi \quad . \end{aligned} \tag{21}$$

Here, $f_{leg,\chi}(t_s)$ and $f_{leg,z}(t_s)$ are the horizontal and vertical components of the leg force $\boldsymbol{F}_{leg}$ and $z(t_s)$ is the height

[1]Note: the intersection point of the leg force (originating from CoM) with the ground is also known as Centroidal Moment Pivot point (CMP) [21].

of the CoM. Now, the horizontal components of the mean intersection point $\overline{\boldsymbol{x}}_{int} = [\overline{x}_{int}, \overline{y}_{int}, z_{floor}]$ can be computed via

$$\overline{\chi}_{int} = \frac{1}{T_s} \int_{t_s=0}^{T_s} \chi_{int}(t_s)\, dt_s = \underbrace{\frac{1}{T_s} \int_{t_s=0}^{T_s} \boldsymbol{d}^T(t_s)\, dt_s}_{\boldsymbol{e}^T}\ \boldsymbol{p}_\chi \ . \quad (22)$$

Here, $\boldsymbol{e}^T$ is a constant row vector. The deviation from the mean value can be computed as

$$\Delta\chi_{int}(t_s) = \chi_{int}(t_s) - \overline{\chi}_{int} = \underbrace{(\boldsymbol{d}^T(t_s) - \boldsymbol{e}^T)}_{\boldsymbol{k}^T(t_s)}\ \boldsymbol{p}_\chi \ . \quad (23)$$

The square of the deviation at a given time t_s is

$$\Delta\chi^2_{int}(t_s) = \boldsymbol{p}_\chi^T \boldsymbol{k}(t_s)\, \boldsymbol{k}^T(t_s) \boldsymbol{p}_\chi = \boldsymbol{p}_\chi^T \boldsymbol{L}(t_s)\, \boldsymbol{p}_\chi \ . \quad (24)$$

In order to obtain the mean square of the deviation $\chi_{int,ms}$, we once again integrate and insert (20) to achieve

$$\begin{aligned} \chi_{int,ms} &= \boldsymbol{p}_\chi^T \frac{1}{T_s} \int_{t_s=0}^{T_s} \boldsymbol{L}(t_s)\, dt_s\, \boldsymbol{p}_\chi = \boldsymbol{p}_\chi^T\, \boldsymbol{M}\, \boldsymbol{p}_\chi \quad (25) \\ &= \underbrace{\boldsymbol{r}_\chi^T \boldsymbol{M}\, \boldsymbol{r}_\chi}_{\alpha}\, \tilde{p}_\chi^2 + 2\underbrace{\boldsymbol{r}_\chi^T \boldsymbol{M} \boldsymbol{p}_{\chi,0}}_{\beta}\, \tilde{p}_\chi + \underbrace{\boldsymbol{p}_{\chi,0}^T \boldsymbol{M} \boldsymbol{p}_{\chi,0}}_{\gamma} \ . \end{aligned}$$

Here, α, β and γ denote scalar quantities. Theoretically, differentiation of (25) would yield the optimal value $\tilde{p}^*_\chi$ that minimizes $\chi_{int,ms}$. In practice, however, the integrals in the above equations would be extremely hard to solve due to the non-linearities in (21) and thus, solving for the analytic expressions of α, β and γ would be inefficient. Thus, we will use a trick: Knowing the structure of (25), we will - via discretization of evaluation time and three arbitrary choices for $\tilde{p}_\chi$ (in this work, we chose -1, 0 and 1) - numerically derive three supporting points for (25), which allow us to solve for the unknown parameters α, β and γ. Therefore, - for each of the three arbitrary choices for $\tilde{p}_\chi$ - we evaluate all preceding equations for a discrete set of n evaluation times. These can be combined in an evaluation-time vector $t_{s,eval} = [t_{s,0}, \ldots t_{s,n}]$ that is evenly distributed along the range $[0, T_s]$. That way, the numeric results of equations (21), (23) and (24) become corresponding evaluation vectors (with n elements), while the integrals in (22) and (25) turn into sums. Note: $\boldsymbol{p}_\chi$ is computed via (20) and our arbitrary choices for $\tilde{p}_\chi \in \{-1, 0, 1\}$. That way, we achieve numeric approximations $\varepsilon_{-1} = \hat{\chi}_{int,ms}(\tilde{p}_\chi = -1)$, $\varepsilon_0 = \hat{\chi}_{int,ms}(\tilde{p}_\chi = 0)$ and $\varepsilon_1 = \hat{\chi}_{int,ms}(\tilde{p}_\chi = 1)$ for $\chi_{int,ms}$. As announced above, we can now solve (25) for the unknown parameters α, β and γ:

$$\hat{\alpha} = \frac{\varepsilon_1 + \varepsilon_{-1}}{2} - \varepsilon_0 \qquad \hat{\beta} = \frac{\varepsilon_1 - \varepsilon_{-1}}{2} \qquad \hat{\gamma} = \varepsilon_0 \ . \quad (26)$$

With these approximations of α, β and γ we compute the optimal $\tilde{p}_\chi$ that minimizes $\chi_{int,ms}$ via differentiation of (25):

$$\tilde{p}^*_\chi = \frac{-\hat{\beta}}{2\,\hat{\alpha}} = \frac{\varepsilon_{-1} - \varepsilon_1}{2\,(\varepsilon_{-1} + \varepsilon_1 - 2\,\varepsilon_0)} \ . \quad (27)$$

Finally, setting $\tilde{p}_\chi = \tilde{p}^*_\chi$, we evaluate (20) which results in the horizontal polynomial parameter vector $\boldsymbol{p}_\chi$ that respects all desired horizontal boundary conditions and leads to the best possible leg force focusing.

Note: the quality of this numerical approximation of α, β and γ depends on the number of evaluation times n. We found that for a choice of about $n \geq 10$, the root mean square of the ground intersection point ($\sqrt{\chi_{int,ms}}$) has sufficiently converged as compared to $n \to \infty$.

E. Foot placement

The basic modules of legged locomotion algorithms are typically one that handles the robot's CoM and another one that controls the foot placement. In the previous section, we derived the mean intersection points $\overline{\boldsymbol{x}}_{int}$ of the leg forces with the ground. These are used as foot targets and are updated online. Note that for discontinuous perturbations, these foot targets may jump. Due to the 2-step-preplanning, both future foot targets are known at all times, such that foot trajectory generation is facilitated. In this work, we implemented the foot reference trajectories as polynomial splines. The use of more sophisticated foot trajectories, as e.g. proposed in [18], is part of our future research.

F. State feedback control

In the nominal case (no perturbations), the force profiles and foot target locations as derived in the previous sections assure that after the first stance phase all desired boundary conditions from sections III-C and III-D are perfectly fulfilled (dead-beat control). Therefore, planning once per step (typically at take-off) or even pre-planning a whole sequence of upcoming steps and according leg force profiles would be sufficient. Yet, since in the real world we have to expect perturbations, we propose a feed-back control method, which is based on the continuous re-planning of the desired contact forces (and corresponding polynomial parameters) throughout flight phases. Therefore, based on the current CoM state ($\boldsymbol{x}$, $\dot{\boldsymbol{x}}$), the previewed CoM touch-down state is updated by inserting the remaining time to touch-down Δt_{TD} from (6) in (3) and (4). This continuous re-planning is facilitated by the very low computational demand of the proposed algorithm. In case of perturbations, we observe very high robustness of the controller (see Sec. IV).

The desired three-dimensional force acting on the CoM can be computed for a given time in stance t_s as

$$\boldsymbol{F}_{CoM,des}(t_s) = m \begin{bmatrix} \boldsymbol{t}_{\ddot{x}}^T(t_s)\, \boldsymbol{p}_x \\ \boldsymbol{t}_{\ddot{y}}^T(t_s)\, \boldsymbol{p}_y \\ \boldsymbol{t}_{\ddot{z}}^T(t_s)\, \boldsymbol{p}_z \end{bmatrix} \quad (28)$$

The corresponding desired leg force $\boldsymbol{F}_{leg,des}$ is found by reordering (1). The polynomial parameters were chosen to result in the best achievable focus of the leg forces with the ground. Yet, for physical robots the desired leg forces may not be achievable. One obvious example is when the robot is modeled as point-mass with point feet (as used in the simulations presented in Sec. IV). In that case, the leg force is constraint to point along the direction of the vector $\boldsymbol{u}_{x,f}$ (unit vector pointing from CoM to point foot). As the other two spatial directions are unactuated, the desired leg force $\boldsymbol{F}_{leg,des}$ has to be projected to the feasible direction:

$$\boldsymbol{F}_{leg,f} = \boldsymbol{u}_{x,f} \boldsymbol{u}_{x,f}^T \boldsymbol{F}_{leg,des} \quad . \quad (29)$$

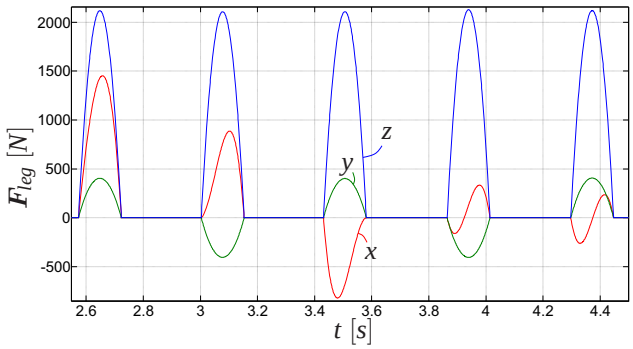

Fig. 7: Force profiles during running simulations.

The feasible leg force $\boldsymbol{F}_{leg,f}$ can be commanded to the point-mass point-foot model. For robots with finite sized foot or angular inertia, this kind of projection may be unnecessary.

G. Steering interface

In this paper, we use a rather position based approach to specify the desired horizontal travel. Higher level controllers are provided a joystick-like interface with three inputs: the desired horizontal velocities $\dot{x}_{joystick}$ and $\dot{y}_{joystick}$ and a yaw rate $\dot{\varphi}_{joystick}$. The latter can be seen as the angular velocity of the robot's virtual hip. The joystick velocities are integrated and result in (sway-free) nominal CoM positions and orientations. Assuming constant joystick velocities, these can (knowing the required touch-down times from Sec. III-C) be extrapolated forward to nominal CoM positions and orientations at the moments of anticipated touch-downs. These nominal trajectories are sway-free, i.e. the CoM shift from left to right foot - required to avoid collisions between the feet - are not considered yet. To create side-wards swaying of the CoM, we design the touch-down positions $\boldsymbol{x}_{TD,2}$ and $\boldsymbol{x}_{TD,3}$ (see Fig. 4) to be at a constant horizontal offset $\Delta x_{com,TD}$ (used as design parameter) from the nominal CoM position in the direction of the virtual hip (left or right). These were used as boundary conditions in Sec. III-D.

IV. Simulations

To test the performance and robustness of the proposed control framework, we performed numerous simulations. The robot model used in this work consists of a point-mass with two massless point-feet. The desired leg forces are projected via (29) to perfectly comply with the point-foot constraint. For all shown simulations, the chosen design parameters were $m = 50kg$, $\Delta z_{apex,des} = 1.1m$, $\Delta z_{TD,des} = 1m$, $T_s = 0.15s$, $\Delta x_{com,TD} = 0.06m$ and $\Delta_{apex,TD,min} = 0.05m$.

Figure 7 shows a set of typical force profiles. In the shown example, the robot first accelerates forward, then backward and then switches to a periodic gait.

Figure 8 shows our controller's tracking performance for varying joystick inputs (see Sec. III-G). Overall, the controller shows very accurate tracking performance. The higher deviations during some phases are caused by continuously changing joystick inputs, since our steering interface assumes

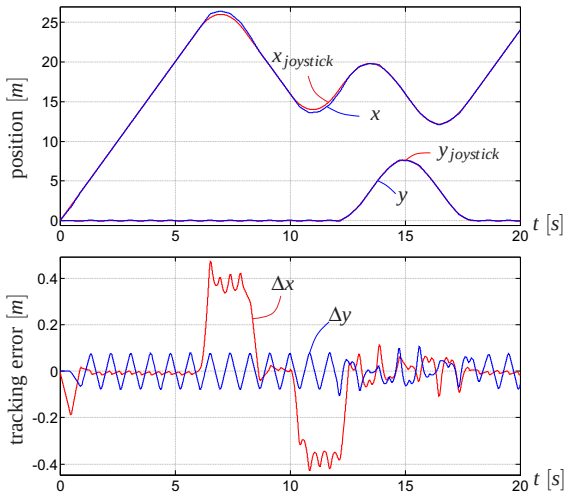

Fig. 8: Tracking performance when steered by joystick input.

constant steering rates. These deviations might be erased by the use of a different steering interface.

Figure 9 shows the results of a robustness examination for three different constant external forces. From top to bottom, the figure shows phase plots for three simulations. Each simulation was setup in the following way: no perturbation during the first 4 seconds, then 4 seconds of constant force acting (magnitude: -50N (corresponding to $\approx 10\%$ of the robot's mass (here 50kg)), force direction: purely x, y and z, respectively), followed by 4 seconds of no perturbation. Here, $\Delta x = x - x_{joystick}$ and $\Delta y = y - y_{joystick}$ denote the errors w.r.t. the nominal horizontal CoM position $\boldsymbol{x}_{joystick} = [x_{joystick}, y_{joystick}]^T$ as provided by the joystick (see Sec. III-G). The stars denote the initial states. The phase plots show that for perturbed and unperturbed phases, the system very quickly converges to corresponding limit cycles. Note: the perturbation force magnitudes in the shown simulations were kept comparably low in order to increase readability of the plots. In other simulations, we increased the continuous unknown lateral perturbation force to $10.000N$ (i.e. 200 times the robot's weight), while still running successfully. The resulting leg length of up to 78m is unrealistic, but the simulation shows the high robustness of the controller. The maximum continuous vertical (downward) force the robot could bear was about $750N$ (i.e. 1.5 times robot's weight). For higher forces, it would finally collide with the ground.

V. Discussion

The trajectory generation and control method described in this paper yields leg force profiles that are independent of the specific hardware design of some particular robot, i.e. the method is generic. The proposed control framework might be used to identify required actuator characteristics for the design of new robots. On the other side, if a specific robot with its predefined hardware limitations and kinematics is to be controlled, other approaches such as optimal control may be required to assure feasibility and good performance.

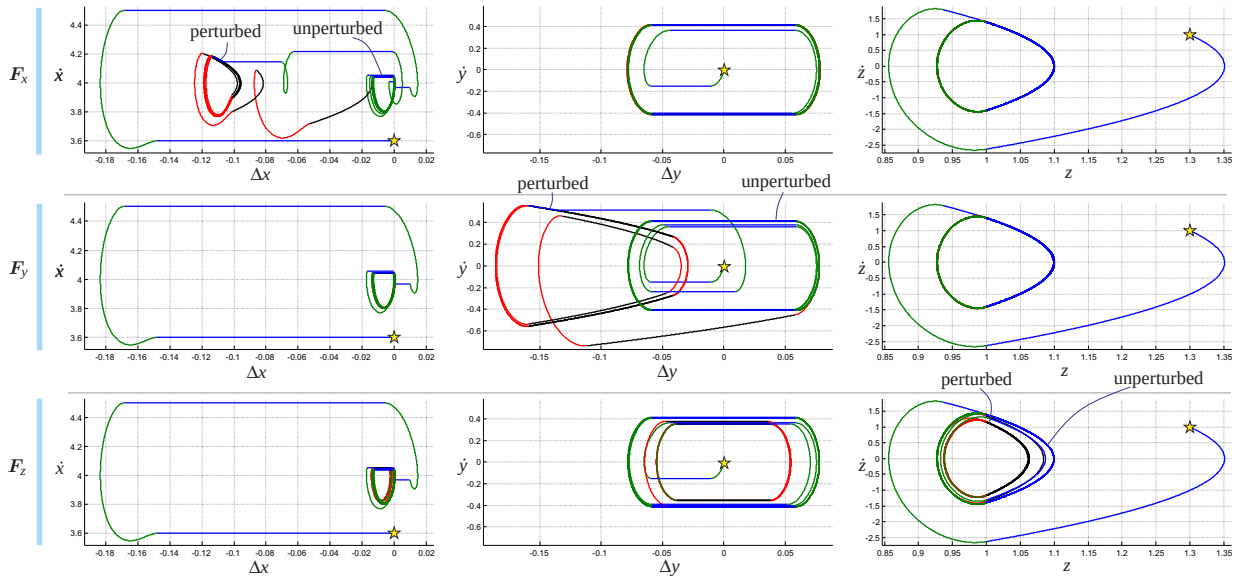

Fig. 9: Robustness examination for different constant external forces.
Perturbation inactive: stance green, flight blue. Perturbation active: stance red, flight blue.

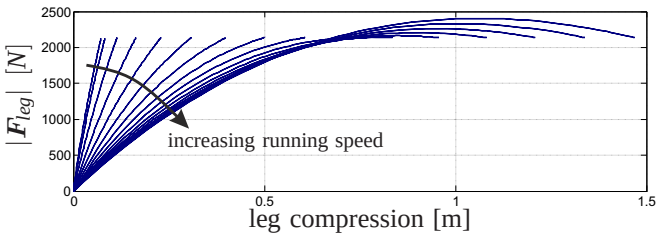

Fig. 10: Force over leg compression ($l_{TD} - l$) for increasing running speed ($0 - 15m/s$). l denotes leg length. ($T_s = 0.15s$)

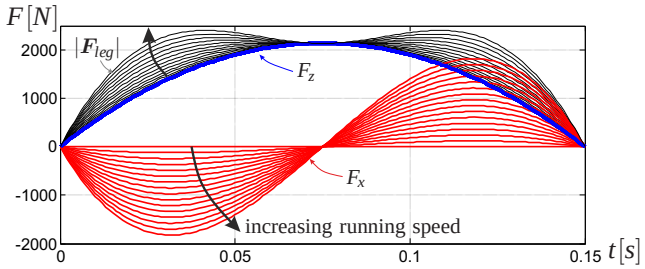

Fig. 11: Comparison of force absolutes for different speeds.

In our control framework, impact-free state transitions are assumed (compare Fig. 2). The impact losses in real systems will cause perturbations. Anyhow, due to its high robustness, we expect good performance of the controller.

Regarding required leg stiffness, we find rather linear stiffness profiles for slow running and more and more degressive non-linear profiles for faster running (see Fig. 10). These degressive stiffness profiles lead to lower maximum leg-force absolutes and are in good accordance with the observations from [22] and [23]. Figure 11 shows the correlating force profiles (for $T_s = 0.15s$); note: here, F_y was set to zero (no sideward CoM sway) for better readability. Due to the linear independence of the spatial directions, the vertical force F_z is equal for all horizontal running speeds. Also note: up to a certain running speed (in the shown example about 10m/s), the maximum in the force magnitude $|\boldsymbol{F}_{leg}|$ is unchanged.

The force profiles as derived in sections III-C and III-D nominally lead to perfect tracking after just one stance phase (deadbeat control), i.e. the controller is nominally perfectly stable. In case of actuation limitations, the control commands may have to be adjusted (e.g. via (29) for point-mass point-feet robots), such that stability can no longer be guaranteed. Yet, our simulations show the high robustness of the controller even in case of constraints.

Figure 12 shows how far the force intersection point $\chi_{int}(t_s)$ deviates from the mean intersection point $\overline{\chi}_{int}$ (i.e. the stance foot position). In the shown simulation, the robot starts at zero speed and then runs at $2\,\frac{m}{s}$. The stance time is set to $150ms$. The initial range of deviation is about $22mm$, while for stationary running it is about $6mm$. This shows that the original (non-projecting) method is well applicable for small-footed robots and that (29) typically has minor influence.

As mentioned before, the proposed algorithm is real-time capable. In our Matlab/Simulink simulation setup, for a sampling time of 1ms, we were able to compute a bit more than 1000 time steps per second. On a real-time operation system, we expect even shorter computation times.

Note: the proposed control framework might almost be called a closed-form solution to running. Only the analytical expressions in the second part of Sec. III-D are so complex, that it is more efficient to solve them numerically.

Note: The algorithm proposed in this paper may also be applied to the problems of hopping and jumping. We also expect that quadrupedal gaits such as galloping and trotting can be achieved by minor extensions of the algorithm.

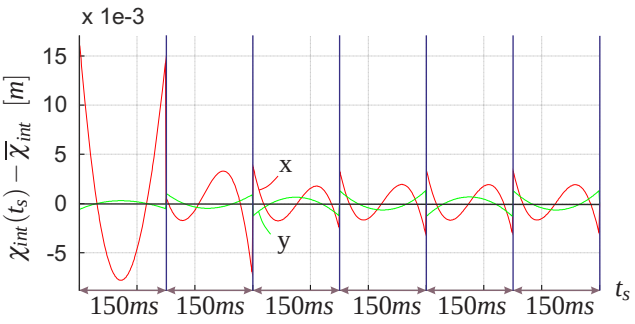

Fig. 12: Simulation of point-mass robot starting from $0\,\frac{m}{s}$ and then running at $2\,\frac{m}{s}$ (in x-direction). Plot shows deviation of force intersection point $\chi_{int}(t_s)$ from mean intersection point $\overline{\chi}_{int}$ (equivalent to stance foot position) in case that (in contrast to all other shown point-mass simulations) the *force projection* (29) is *not activated*. Along the time-axis, the stance phases are pieced together. Stance time was 150ms.

VI. Conclusion and Future Research Topics

In this paper, we proposed a new biologically inspired algorithm for force-based bipedal running in 3D. The proposed controller has dead-beat properties, i.e. in the nominal case it reaches the desired boundary conditions after just one stance phase. The controller facilitates agile and versatile running and is very robust against external perturbations. In a next step, we will embed the proposed controller into a QP-based whole-body controller (similar to [20]) to achieve running with more complex robot models. Additionally, we want to extend the algorithm to running on stepping stones, jumping over and onto obstacles and running on uneven terrain. Also, we want to control the planar physically compliant robot that our group is currently designing.

Acknowledgements

The authors want to thank the staff from Autonomous Systems Lab (ETH Zürich) for organizing the summer school "Dynamic Walking and Running with Robots" in 2011. This summer school gave the inspiration for this work.

Also, we cordially thank Roy Müller and Christian Rode from Department of Motion Science (Jena, Germany) for providing the human experimental data.

This research is partly supported by the Initiative and Networking Fund of Helmholtz Association through a Helmholtz Young Investigators Group (Grant no. VH-NG-808).

References

[1] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.

[2] M. Vukobratovic and Y. Stepanenko, "On the stability of anthropomorphic systems," *Mathematical Biosciences*, vol. 15, pp. 1–37, 1972.

[3] 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. on Robotics and Automation*, 2003, pp. 1620–1626.

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

[5] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, and K. Yokoi, "Biped walking stabilization based on linear inverted pendulum tracking," in *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2010, pp. 4489–4496.

[6] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot, 1st report: Walking gait pattern generation," in *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2009.

[7] M. Morisawa, S. Kajita, F. Kanehiro, K. Kaneko, K. Miura, and K. Yokoi, "Balance control based on capture point error compensation for biped walking on uneven terrain," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2012, pp. 734–740.

[8] T. Koolen, T. D. Boer, J. Rebula, A. Goswami, and J. E. Pratt, "Capturability-based analysis and control of legged locomotion. part 1: Theory and application to three simple gait models," *Int. J. of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[9] Y. Zhao and L. Sentis, "A three dimensional foot placement planner for locomotion in very rough terrains," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2012, pp. 726–733.

[10] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control using divergent component of motion," in *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2013.

[11] K. Nagasaka, Y. Kuroki, S. Suzuki, Y. Itoh, and J. Yamaguchi, "Integrated motion control for walking, jumping and running on a small bipedal entertainment robot," in *IEEE Int. Conf. on Robotics and Automation*, 2004, pp. 3189–3194.

[12] T. Takenaka, T. Matsumoto, T. Yoshiike, T. Hasegawa, S. Shirokura, H. Kaneko, and A. Orita, "Real time motion generation and control for biped robot -4th report: Integrated balance control-," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009.

[13] S. Cotton, I. M. C. Olaru, M. Bellman, T. van der Ven, J. Godowski, and J. Pratt, "Fastrunner: A fast, efficient and robust bipedal robot. concept and planar simulation," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2358–2364.

[14] D. Lakatos, C. Rode, A. Seyfarth, and A. Albu-Schäffer, "Design and control of compliantly actuated bipedal running robots: Concepts to exploit natural system dynamics," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014.

[15] A. Seyfarth, H. Geyer, M. Günther, and R. Blickhan, "A movement criterion for running," *Journal of biomechanics*, vol. 35, no. 5, pp. 649–655, 2002.

[16] H. Geyer, A. Seyfarth, and R. Blickhan, "Compliant leg behaviour explains basic dynamics of walking and running," *Proceedings of the Royal Society B: Biological Sciences*, pp. 2861–2867, 2006.

[17] B. Dadashzadeh, H. Vejdani, and J. Hurst, "From template to anchor: A novel control strategy for spring-mass running of bipedal robots," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, 2014.

[18] H. R. Vejdani, Y. Blum, M. A. Daley, and J. W. Hurst, "Bio-inspired swing leg control for spring-mass robots running on ground with unexpected height disturbance," *Bioinspiration and Biomimetics*, vol. 8, no. 4, p. 046006, 2013.

[19] D. Koepl and J. Hurst, "Impulse control for planar spring-mass running," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3-4, pp. 589–603, 2014.

[20] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3d-slip model," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 5134–5140.

[21] M. B. Popovic, A. Goswami, and H. Herr, "Ground reference points in legged locomotion: Definitions, biological trajectories and control implications," *Int. J. of Robotics Research*, vol. 24, no. 12, 2005.

[22] J. Rummel and A. Seyfarth, "Stable running with segmented legs," *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 919–935, 2008.

[23] H. Yu, M. Li, and H. Cai, "Analysis on the performance of the slip runner with nonlinear spring leg," *Chinese Journal of Mechanical Engineering*, vol. 26, no. 5, pp. 892–899, 2013.