ESAIM: COCV 25 (2019) 81
https://doi.org/10.1051/cocv/2017034

**ESAIM: Control, Optimisation and Calculus of Variations**
www.esaim-cocv.org

# A NOVEL ONLINE GAIT OPTIMIZATION APPROACH FOR BIPED ROBOTS WITH POINT-FEET

Majid Anjidani[1], M.R. Jahed Motlagh[1], M. Fathy[1] and M. Nili Ahmadabadi[2]

**Abstract.** Designing a stable walking gait for biped robots with point-feet is stated as a constrained nonlinear optimization problem which is normally solved by an offline numerical optimization method. On the result of an unknown modeling error or environment change, the designed gait may be ineffective and an online gait improvement is impossible after the optimization. In this paper, we apply Generalized Path Integral Stochastic Optimal Control to closed-loop model of planar biped robots with point-feet which leads to an online Reinforcement Learning algorithm to design the walking gait. We study the ability of the proposed method to adapt the controller of RABBIT, which is a planar biped robot with point-feet, for stable walking. The simulation results show that the method, starting a dynamically unstable initial gait, quickly compensates the modeling error and reaches to a walking with exponential stability and desired features in a new situation which was impossible by the past methods.

**Mathematics Subject Classification.** 49J15, 93E35, 68T40.

Received August 21, 2016. Revised March 14, 2017.

## 1. Introduction

There are many different sociological and commercial motivations such as, the appeal to replace humans in hazardous situations [1], rehabilitate the motion in the disabled [2,3], and operate in human-like ways [4] to research on walking biped robots.

Biped robots are examples of underactuated robots [5]. In general, underactuation introduces non-integrable constraints either at the kinematic or dynamic level [5]. For this reason, and because the number of generalized coordinates is larger than the available number of inputs, planning for an underactuated system is a challenging problem [5–8].

The control algorithms of the biped robots fall into two groups: time-dependent and time-invariant algorithms [9]. The most popular algorithms are time-dependent as shown in Figure 1a and involve the tracking of *precomputed* trajectories [9], *e.g.* the trajectories which are generated by Central Pattern Generators (CPGs) [10,11], or generated by a length-varying inverted pendulum [12,13]. They are usually characterized by a heavy reliance on heuristics or on principles such as the zero moment point (ZMP) criterion [14–16] that do not guarantee the stability in dynamic locomotion [9,17]; as a result, only slow motions may be achieved [9]. Dynamic motions, such as balancing, running or fast walking, are excluded with these approaches [18].

*Keywords and phrases.* Legged locomotion, gait optimization, orbital stability.

[1] Computer Department, Iran University of Science and Technology, Tehran, Iran. jahedmr@iust.ac.ir

[2] Electrical and Computer Department, University of Tehran, Tehran, Iran. mnili@ut.ac.ir

Article published by EDP Sciences

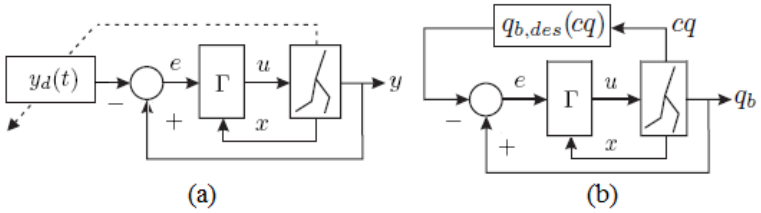


FIGURE 1. (a) Block diagram of a time-dependent controller: the controller $\Gamma$ forces the error $e = y - y_d$ to zero so that output $y$ tracks the desired trajectory $y_d(t)$. The dashed line indicates that the trajectories $y_d(t)$ may be modified on the basis of the robot's state. It is challenging to design the trajectories in such a way that the resulting nonlinear, time-varying, closed-loop system is stable [9]. (b) Block diagram of a time-invariant controller: the controller $\Gamma$ forces the signal $e = q_b - q_{\mathrm{b,des}}(cq)$ to zero so that the body coordinates $q_b$ tracks $q_{\mathrm{b,des}}(cq)$ where $c$ is a constant row vector[9].

In contrast, there have been only a few time-invariant control schemes proposed as shown in Figure 1b, in which generally the idea is imposing a set of constraints by feedback controller [19–30] instead of tracking of precomputed trajectories. Hybrid zero dynamics (HZD) is a low-dimensional submodel of the closed-loop hybrid robot model that leads to fast offline algorithms for designing the holonomic constraints to reach a stable dynamic locomotion [9]. Designing the holonomic constraints will be stated as a constrained nonlinear optimization problem which may be solved with many of the numerical optimization tools currently available [9, 28–30]. However, we show that finding an acceptable initial gait with respect to the environment for the optimization is rather hard. Also, there are unknown modeling errors which are ignored in the submodel during the optimization process. Therefore the designed gait for the real robot may be not as well as possible and the robot is incapable to learn for improvement after the optimization process. Regarding to the difficulty of finding an initial gait and the need for gait improvement after the optimization, we think of an alternative online approach, which is also applicable to the real robot, to simplify designing a stable walking gait with arbitrary features in a certain environment.

Reinforcement Learning (RL) methods are another alternative which can be used to adjust the walking gait parameters. Most of the RL methods suffer a problem of un-scalability to high-dimensional robots such as humanoid robots [31]. However, Path Integral Policy Improvement ($\mathrm{PI}^2$) [31–33] is an online model-free RL algorithm based on reward-weighted averaging methods [34], with interesting features like having an arbitrary state-dependent cost function part, exploration noise as the only open algorithmic tuning parameters, numerically robust performance in high-dimensional learning problems and better performance than gradient-based methods [31]. $\mathrm{PI}^2$ has been compared with the other efficient methods [31], *e.g.* REINFORCE [35, 36], GPOMDP [37], PG [38], ENAC [39] and PoWER [40] in multiple examples and the results show that $\mathrm{PI}^2$ surpasses all of them [31]. $\mathrm{PI}^2$ algorithm is resulted from applying Generalized Path Integral Stochastic Optimal Control (GPISOC) to Dynamic Movement Primitive (DMP) [41] equations to adjust trajectory parameters. However, $\mathrm{PI}^2$ with the *precomputed* joint trajectories, which are generated by DMPs, is not a good choice for the case of the stable walking with point-feet as mentioned above. In contrast, Hybrid Zero Dynamic (HZD) controller [9], by designing closed-loop joint trajectories via the virtual constraints, has a proof of stability which is an interesting feature for walking with point-feet [9]. In this paper, we apply GPISOC to the closed-loop model of the planar biped robots with point-feet to adjust the walking gait parameters. The result is a new online algorithm which we call it $\mathrm{PI}^2$-WG. The simulation results show that using $\mathrm{PI}^2$-WG, the robot is able to learn how to walk with desired speed/gesture and with exponential stability in a new environment *e.g.* a certain slope surface with desired friction in the presence of a certain external force and modeling error.

In Section 2, GPISOC is introduced. In Section 3, the walking gait and the time-invariant controller are described. In Section 4, the closed-loop model of the robot is approximated such that GPISOC can be applied

to it. In Section 5, the new algorithm is presented through applying GPISOC to the closed-loop model. Related works, simulation results and conclusions are respectively mentioned in Sections 6–8.

## 2. GPISOC and policy parameterization

For self-sufficiency and simplifying derivation of our learning method, we present this section with two parts. In the first part, GPISOC [31] is introduced. In the second part, it is shown that GPISOC can be also applied to the parameterized dynamics equations of the bipedal robots with point-feet.

### 2.1. Generalized path integral stochastic optimal control

GPISOC is a method for computing the optimal controls of a dynamical system of the following class:

$$\dot{x}_t = f(x_t, t) + G(x_t)(u_t + \varepsilon_t) = f_t + G_t(u_t + \varepsilon_t)$$

where $x_t \in \mathbb{R}^l$ is the time dependent system state and $l$ denotes the state dimension. $f_t \in \mathbb{R}^l$ denotes the passive dynamics and $G_t = G(x_t) \in \mathbb{R}^{l\times p}$ is the transition control matrix. $u_t \in \mathbb{R}^p$ is the control vector where $p$ is the number of actuated joint. It is assumed that $\varepsilon_t \in \mathbb{R}^p$ is a mean-zero Gaussian noise with variance $\Sigma_\varepsilon = \lambda R^{-1}$ and a positive semi-definite weight matrix $R$ which is usually chosen to be diagonal and invertible [31].

Essentially the stochastic system is partitioned into the controlled equations, in which $x_t^{(l_2)} \in \mathbb{R}^{l_2}$ is directly actuated part of the state, and the uncontrolled equations[3], in which $x_t^{(l_1)} \in \mathbb{R}^{l_1}$ is not directly actuated part of the state, as follows:

$$\begin{bmatrix} \dot{x}_t^{(l_1)} \\ \dot{x}_t^{(l_2)} \end{bmatrix} = \begin{bmatrix} f_t^{(l_1)} \\ f_t^{(l_2)} \end{bmatrix} + \begin{bmatrix} 0_{l_1\times p} \\ G_t^{(l_2)} \end{bmatrix} (u_t + \varepsilon_t) \tag{2.1}$$

where the passive dynamics term $f_t$ is partitioned as $f_t = [f_t^{(l_1)}; f_t^{(l_2)}]$ with $f_t^{(l_1)} \in \mathbb{R}^{l_1}$ and $f_t^{(l_2)} \in \mathbb{R}^{l_2}$[4]. The transition control matrix is partitioned as $G_t = [0_{l_1\times p}; G_t^{(l_2)}]$ with $G_t^{(l_2)} \in \mathbb{R}^{l_2\times p}$.

Trajectories of the dynamical system are sampled over a uniform subdivision of time $\{t_0, t_1, \ldots, t_N\}$ and $\delta t$ denotes the sampling time. Such a sampled trajectory is denoted by $\tau{=}(x_{t_i}, x_{t_{i+1}}, \ldots, x_{t_N})$. In GPISOC, the cost of the trajectory $\tau \in \chi_i$ is defined as [31]:

$$C(\tau) = \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j}\delta t + \frac{1}{2}\sum_{j=i}^{N-1} \left| \frac{x_{t_{j+1}}^{(l_2)} - x_{t_j}^{(l_2)}}{\delta t} - f_{t_j}^{(l_2)} \right|^2_{H_{t_j}^{-1}} \delta t + \frac{\lambda}{2}\sum_{j=i}^{N-1} \log\|H_{t_j}\|, \quad i \in \{0, \ldots, N-1\} \tag{2.2}$$

where $\chi_i$ denotes the space of all trajectories, starting at time $t_i$ in state $x_{t_i}$ and generated by the noisy controls $(u_{t_i} + \varepsilon_{t_i}, \ldots, u_{t_{N-1}} + \varepsilon_{t_{N-1}})$. $\phi_{t_N} = \phi(x_{t_N})$ is the final cost at time $t_N$. $Q_{t_j} = Q(x_{t_j}, t_j)$ is a state-dependent arbitrary cost function, which is defined by cost function designer, involving the most of informations used by optimization process. $H_{t_j} = G_{t_j}^{(l_2)} R^{-1} G_{t_j}^{(l_2)T}$ and $\|.\|$ denotes determinant. $\lambda$ is a positive coefficient[5]. For notational simplicity, we write $v^T A v$ as $|v|_A^2$ for a vector $v$ and a positive semi-definite weight matrix $A$.

The optimal controls of GPISOC in every time step are computed as [31]:

$$u_{t_i} = \int_{\tau\in\chi_i} P(\tau) u_L(\tau) \mathrm{d}\tau^{(l_2)}, \quad i \in \{0, \ldots, N-1\} \tag{2.3}$$

[3]The notations $.^{(l_1)}$ and $.^{(l_2)}$ respectively indicate belonging to the uncontrolled and controlled equations.

[4]In this paper, semicolons are used to form matrices in line as in MATLAB. In other words, a new row is started by a semicolon.

[5]$\lambda$ is a user design parameter, as $R$, $\phi_{t_N}$ and $Q_{t_j}$.

where $\mathrm{d}\tau^{(l_2)}=(dx_{t_i}^{(l_2)},\ldots,dx_{t_N}^{(l_2)})$. The trajectory weight $P(\tau)$ is defined as [31]:

$$P(\tau)=\frac{\exp(-C(\tau)/\lambda)}{\int_{\tau\in\chi_i}\exp(-C(\tau)/\lambda)\mathrm{d}\tau^{(l_2)}},\quad i\in\{0,\ldots,N-1\} \tag{2.4}$$

and the local control $u_L(\tau)$ is computed as [31]:

$$u_L(\tau)=R^{-1}G_{t_i}^{(l_2)T}\left(G_{t_i}^{(l_2)}R^{-1}G_{t_i}^{(l_2)T}\right)^{-1}\left(G_{t_i}^{(l_2)}\varepsilon_{t_i}-b_{t_i}\right),\quad i\in\{0,\ldots,N-1\} \tag{2.5}$$

where $b_{t_i}=\lambda H_{t_i}\Phi_{t_i}$ and $j$th component of the vector $\Phi_{t_i}$ is defined as follows [31]:

$$\Phi_{t_i,j}=\frac{1}{2}trace\left(H_{t_i}^{-1}\left(\frac{\partial H_{t_i}}{\partial x_{t_i,j}^{(l_2)}}\right)\right),\quad i\in\{0,\ldots,N-1\}.$$

Note that only the controlled equations play a role in the optimal controls of GPISOC. GPISOC can also used to compute the optimal parameter vectors in parameterized dynamics equations. In Section 2.2, the GPISOC formalism for computing the optimal parameter vectors of DMP equations is presented.

## 2.2. Policy parameterization and GPISOC

Here, the outline of applying GPISOC to DMP equations is introduced [31]. Consider the following robotic system with rigid body dynamics:

$$\ddot{q}=D(q)^{-1}\left(-W(q,\dot{q})-\mathcal{G}(q)\right)+D(q)^{-1}u \tag{2.6}$$

where $D\in\mathbb{R}^{n\times n}$ is inertia matrix and $n$ is the number of degrees of freedom of the system. Coriolis and centripetal forces are denoted by $W\in\mathbb{R}^{n\times n}$. Vector $\mathcal{G}\in\mathbb{R}^n$ denotes the gravity and elastic forces. Vector $[q;\dot{q}]$, including the joint angles and angular velocities, denotes the robot state.

One approach for policy parameterization is using the following control structure [31]:

$$u=K_P\left(q_d-q\right)+K_D\left(\dot{q}_d-\dot{q}\right) \tag{2.7}$$

$$\begin{bmatrix}\dot{\eta}_t\\ \ddot{q}_d\end{bmatrix}=\begin{bmatrix}-\alpha\eta_t\\ \alpha_z\left(\beta_z(goal-q_d)-\dot{q}_d\right)\end{bmatrix}+\begin{bmatrix}0_{1\times p}\\ \mathrm{g}_t^{(l_2)T}(\eta_t)\end{bmatrix}(\theta+\varepsilon_t)\,. \tag{2.8}$$

The desired trajectory, $(q_d,\dot{q}_d)$, is generated by DMP equations (2.8) regardless of the robot dynamics (2.6), and tracked by PD controller (2.7) with positive definite gain matrices $K_P$ and $K_D$. In fact, DMPs code a learnable point attractor for movement from $q_{t_0}$ to *goal*, where $\theta$ determines the shape of the attractor [31]. $\alpha_z$ and $\beta_z$ are positive constants. The basis functions of the transition vector $\mathrm{g}_t^{(l_2)}(\eta_t)$ are defined by a piecewise linear function approximator with Gaussian weighting kernels [31] in which $\eta_t$ monotonically converges to zero by the first-order linear dynamics $\dot{\eta}_t=-\alpha\eta_t$. $\eta_t=1$ would indicate the start of the time evolution and $\eta_t$ close to zero means that *goal* has essentially been achieved. To compute the optimal parameter vectors of GPISOC, equations (2.8) should be reformulated to the form (2.1) as follows:

$$\begin{bmatrix}\dot{x}_t^{(l_1)}\\ \dot{x}_t^{(l_2)}\end{bmatrix}=\begin{bmatrix}f_t^{(l_1)}\\ f_t^{(l_2)}\end{bmatrix}+\begin{bmatrix}0_{l_1\times p}\\ \mathrm{g}_t^{(l_2)T}\end{bmatrix}(\theta+\varepsilon_t) \tag{2.9}$$

where $\varepsilon_t$ is assumed a mean-zero Gaussian noise with variance $\Sigma_\varepsilon=\sigma^2 I$ in the evaluations of [31] and $\sigma^2$ is decreased from 0.1 to near zero[6]. Since $\theta$ is time-invariant along a sample trajectory, we use "perturbed/unperturbed equations" instead of "controlled/uncontrolled equations". The perturbed equations[7]

[6]Assuming $R=(\sigma^2/\lambda)I$ leads to $\Sigma_\varepsilon=\sigma^2 I$ according to $\Sigma_\varepsilon=\lambda R^{-1}$. $\lambda$ is eliminated from $\Sigma_\varepsilon$ by the assumption.

[7]If $\theta$ is noiseless, there is no perturbed equation.

of (2.9) are:

$$\dot{x}_t^{(l_2)} = f_t^{(l_2)} + \mathrm{g}_t^{(l_2)T}(\theta + \varepsilon_t). \tag{2.10}$$

**Remark 2.1.** Only the perturbed equations, which can be formulated in the form (2.10), play a role in the optimal parameter vectors of GPISOC.

The trajectory cost (2.2) for equations (2.10) is formulated as [31]:

$$C(\tau) = \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} (\theta + \varepsilon_{t_j})^T M_{t_j}^T R M_{t_j}(\theta + \varepsilon_{t_j}) \tag{2.11}$$

where $M_{t_j} = R^{-1}\mathrm{g}_{t_j}^{(l_2)}\mathrm{g}_{t_j}^{(l_2)T}/(\mathrm{g}_{t_j}^{(l_2)T}R^{-1}\mathrm{g}_{t_j}^{(l_2)})$. The optimal parameter vectors of GPISOC according to (2.3) are [31]:

$$\theta_{t_i} = \int_{\tau\in\chi_i} P(\tau)\theta_L(\tau)\mathrm{d}\tau^{(l_2)} \quad where \quad P(\tau) = \frac{\exp(-C(\tau)/\lambda)}{\int_{\tau\in\chi_i}\exp(-C(\tau)/\lambda)\mathrm{d}\tau^{(l_2)}} \tag{2.12}$$

where $\chi_i$ denotes the space of all trajectories started from state $x_{t_i}$ and generated by $(\theta + \varepsilon_{t_i}, \ldots, \theta + \varepsilon_{t_{N-1}})$. The local parameter vector $\theta_L(\tau)$ with an initial guess of $\theta$ is formulated as follows according to (2.5) [31]:

$$\theta_L(\tau) = M_{t_i}(\theta + \varepsilon_{t_i}). \tag{2.13}$$

Therefore, $\theta$ can be iteratively updated by computing a weighted averaging of $\theta_{t_i}$ at every time steps as follows [31, 43], that it is called PI$^2$ algorithm:

$$\theta^{(new)} = \theta^{(old)} + \delta\theta, \quad \delta\theta = \frac{\left[\sum_{i=0}^{N-1}(N-i)\delta\theta_{t_i}\right]}{\sum_{i=0}^{N-1}(N-i)}, \quad \delta\theta_{t_i} = \int_{\tau\in\chi_i} P(\tau)M_{t_i}\varepsilon_{t_i}\mathrm{d}\tau^{(l_2)}. \tag{2.14}$$

The integrations in (2.12) can be approximated by the following summations over $K$ sample trajectories $\tau_{k=1,\ldots,K}$ which are generated by applying different noises to $\theta$ [31]:

$$\theta_{t_i} \approx \sum_{k=1}^{K} P(\tau_k)\theta_L(\tau_k) \quad where \quad P(\tau_k) \approx \frac{\exp(-C(\tau_k)/\lambda)}{\sum_{j=1}^{K}\exp(-C(\tau_j)/\lambda)}$$

where $-C(\tau_k)/\lambda$ is implemented as follows [31]:

$$\frac{-C(\tau_k)}{\lambda} = -10\frac{C(\tau_k) - \min_{j=1,\ldots,K} C(\tau_j)}{\max_{j=1,\ldots,K} C(\tau_j) - \min_{j=1,\ldots,K} C(\tau_j)}$$

where this procedure eliminates $\lambda$ and leaves the variance $\Sigma_\varepsilon$ as the only open algorithmic parameter for PI$^2$ [31].

Another approach which is used in this paper, is directly parameterizing the dynamics equations (2.6) by the following control policy such that the perturbed equations takes the form of (2.10):

$$u = u(q, \dot{q}, \theta + \varepsilon_t). \tag{2.15}$$

In this case, if the control policy (2.15) is selected such that the parameter $\theta$ represents the trajectory parameters, GPISOC can similarly apply to the perturbed equations to adjust the trajectory. In the next section, it is shown that the control effort of feedback controller is in the form of (2.15); therefore the major problem is formulating the closed-loop model in the GPISOC consistent form.

## 3. Walking gait and feedback controller

The parameterized equations that generate the walking gait are introduced in this section. Consider the following hybrid dynamics model of a planar point-feet biped robot [9]:

$$\begin{cases} D(q)\ddot{q} + W(q,\dot{q}) + \mathcal{G}(q) = [u;0] - F_{fr}(q,\dot{q}) + J(q)^T F_{ext}, & t \notin T_{\text{impact}} \\ [q^+;\dot{q}^+] = \Delta([q^-;\dot{q}^-]), & t \in T_{\text{impact}} \end{cases} \tag{3.1}$$

where $F_{fr}(q,\dot{q})$ denotes viscous and Coulomb friction torques. Vector $F_{ext} = [F_x; F_y]$ denotes the external force which is exerted on the highest point of the robot during walking and $F_x$ and $F_y$ are its horizontal and vertical components, *e.g.* see Figure A.1a in Appendix A. $J(q) \in \mathbb{R}^{n\times 2}$ is Jacobian matrix of the highest point[8] and $n$ is the number of degrees of freedom of the robot. $q = [q_b; q_n]$ where $q_b = [q_1; \ldots; q_{n-1}]$ contains $n-1$ body coordinates and $q_n$ provides the orientation of the robot with respect to inertial frame. $q_n$ is a cyclic coordinate meaning that the matrix $D(q)$ is independent of $q_n$, *i.e.* $D(q) = D(q_b)$ [9]. $u = [u_1; \ldots; u_{n-1}] \in \mathbb{R}^{(n-1)}$. During a walking step, the role of the leg which is on the ground is stance and the role of the other is swing. $T_{\text{impact}}$ is a set of all impact times (*i.e.* the time in which the swing leg impacts the ground). The generalized coordinates before and after impact moment are denoted by $q^-$ and $q^+$. At this moment, the role of the two legs exchanges and the function $\Delta : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ maps the robot state just before impact $x^- = [q^-; \dot{q}^-]$ to just after impact $x^+ = [q^+; \dot{q}^+]$; therefore a new step is started. The state space form of (3.1) is:

$$\begin{cases} \dot{x} = f(x) + h(x)u, & t \notin T_{\text{impact}} \\ x^+ = \Delta(x^-), & t \in T_{\text{impact}} \end{cases} \quad \text{where } f = \begin{bmatrix} \dot{q} \\ D^{-1}(q)(-W(q,\dot{q}) - \mathcal{G}(q) - F_{fr}(q,\dot{q}) + J^T F_{ext}) \end{bmatrix} \text{ and } h = \begin{bmatrix} 0_{n\times(n-1)} \\ D^{-1}(q)B \end{bmatrix}$$

and $x = [q; \dot{q}]$ and $B = [I_{(n-1)\times(n-1)}; 0_{1\times(n-1)}]$. The theories of this paper are related to this class of biped robots. As a simulation test-bed, we model a point-feet biped robot with 5 degrees of freedom, named RABBIT which is described in Appendix A. Figure 2a shows the trajectory of the RABBIT's swing leg end during a step of walking. Figure 2b shows the attitude of the stance leg during the step. The walking gait, $q_{\text{b,des}}(z)$, contains a set of smooth Bezier functions [9] that can be formulated as:

$$q_{\text{b,des}}(z) = G(s(z))\Theta$$

where $z = cq$ is a scalar in which $c \in \mathbb{R}^{1\times n}$ is a constant row vector and should be determined such that $z$ has a monotonic change between $z^+ = cq^+$ and $z^- = cq^-$ along a step of normal walking [9]. $s(z) = (z - z_1)/(z_2 - z_1)$ where if $z_2 = z^-$ and $z_1 = z^+$ as in [9] then we have $0 \le s(z) \le 1$. Throughout this paper, $z_1 = -2.9835$ and $z_2 = -3.3556$, which denote the values of $z^+$ and $z^-$ in a successful normal walking on the flat surface[9]. In the remaining part of this paper, $s$ is used instead of $s(z)$ for notational compactness. $G(s(z))$ and $\Theta$ are defined as:

$$G(s) = \begin{bmatrix} g(s)^T & & 0 \\ & \ddots & \\ 0 & & g(s)^T \end{bmatrix}, \quad \Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n-1} \end{bmatrix}, \quad \theta_d = \begin{bmatrix} \theta_d^0 \\ \theta_d^1 \\ \vdots \\ \theta_d^L \end{bmatrix}$$

where Bezier basis vector $g(s) \in \mathbb{R}^{(L+1)}$ is equal to:

$$g(s) = \left[ (1-s)^L; \ldots; \frac{L! s^m (1-s)^{L-m}}{m!(L-m)!}; \ldots; s^L \right]$$

[8] $J(q)^T$ is a matrix that maps $F_{ext}$ to the joint torques [9].

[9] In Section 6, we describe why we use fixed $z_1$ and $z_2$ in all scenarios contrary to [9].

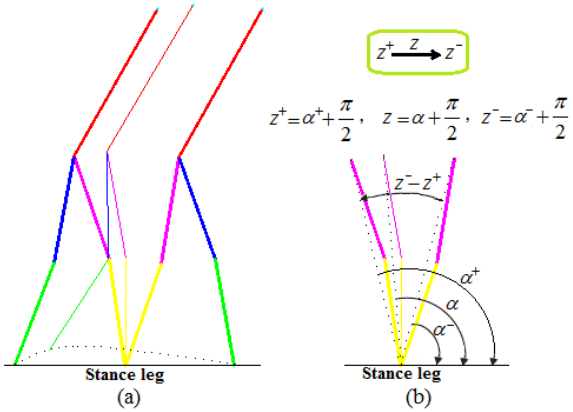


FIGURE 2. (a) A step of normal walking on the flat surface. (b) Stance leg location during the step. Note that a dotted line connects the hip joint to the foot of the stance leg and $\alpha$ denotes the angle between the dotted line and the surface in clockwise direction. We expect that $|z_2 - z_1| > \pi/3 \approx 1$ happens during fast running contrary to normal walking. Also, it can be realized that $z$ has a monotonic behavior during a step of normal walking.

and $\Theta$ includes the gait parameters that adjust the virtual constraints, *i.e.* $q_b - q_{\mathrm{b,des}}(z) = 0$, which should be imposed by feedback controller. In other words, the components of $\Theta$ are the parameter vectors of closed-loop joint trajectories ($\theta_d, \quad 1 \leq d \leq n-1$). Vector $\theta_d$ is the parameter of the $d$th joint trajectory with $L+1$ components ($\theta_d^m, \quad 0 \leq m \leq L$) where $L+1$ denote the number of Bezier basis in $g(s)$. Bezier function for the trajectory of $d$th joint is:

$$g(s)^T \theta_d = \sum_{m=0}^{L} \theta_d^m \frac{L!}{m!\,(L-m)!} s^m (1-s)^{L-m}.$$

The constant vector $c$ is generally selected such that $z = cq$ denotes the weighted sum of all joint angles excluding joint angles of swing leg, *e.g.* $c = [-1, 0, -0.5, 0, -1]$ in Figure 2 [9]. Among these joint angles only the angle of the joint between the torso and the femur of the stance leg observably changes along a step in normal walking. Hence, it is also clear that in a normal and smooth walking, $\ddot{z}$ is small while the swing leg joints can take larger angular accelerations. Two useful features of normal walking for our discussion are:

(i) Except in a short time interval after impact, $\ddot{z}$ has a small value.
(ii) $|z_2 - z_1| < 1$. The value of $(z^- - z^+)$ in a step of walking is illustrated in Figure 2.

In the next section, Theorem 4.3 shows the importance of $\ddot{z} \approx 0$; therefore in Section 7.1, we have designed the cost function such that a non-zero value for $\ddot{z}$ increases the cost. The second feature is used in the proof of Theorem 4.1.

**Remark 3.1.** The joint trajectories via virtual constraints in the feedback controller are closed-loop which leads to an *intelligent* change of the trajectories in the face of perturbations, *e.g.* faster/slower change in $z$ from $z^+$ to $z^-$ leads to faster/slower movement of the swing leg and consequently walking. It is clear that if an external force in opposite direction of movement leads to abnormal change in $z$ from $z^-$ to $z^+$, the robot starts walking backward [30].

The feedback linearization control rule is:

$$u = -(L_h L_f e)^{-1}(K_P e + K_D \dot{e} + L_f^2 e) \tag{3.2}$$

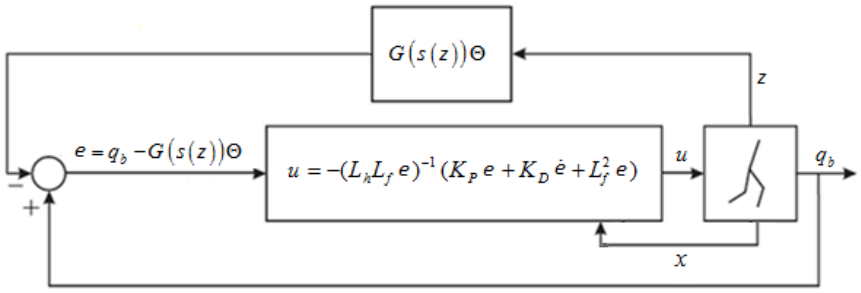


FIGURE 3. The closed-loop hybrid model of the point-feet biped robot: the controller is time-invariant.

which is calculated from the second derivative of $e$, *i.e.* $\ddot{e} = L_f^2 e + (L_h L_f e)u$ [9]. $e = q_b - q_{\mathrm{b,des}}$ and $\dot{e} = \dot{q}_b - \dot{q}_{\mathrm{b,des}}$ in which $\dot{q}_{\mathrm{b,des}}(z) = \dot{z}/(z_2 - z_1)\partial G(s)/\partial s\Theta$. $L_h$, $L_f$ and $L_f^2$ are Lie derivatives. $L_h L_f e$ is decoupling matrix which should be invertible along walking. The following closed-loop hybrid model of the robot, which is result of applying the control effort (3.2) to the model (3.1), is illustrated Figure 3:

$$\begin{cases} \dot{x} = f^{cl}(x, \Theta), & t \notin T_{\mathrm{impact}} \\ x^+ = \Delta(x^-), & t \in T_{\mathrm{impact}} \end{cases} \tag{3.3}$$

where $f^{cl}$ denotes the model of the bipedal robot in closed loop with the controller. There are some criteria to check the existence and exponential stability of a periodic walking motion using HZD, which is the reduced-order of the closed-loop hybrid model [9]. Since the criteria are satisfied by a range of values for $\Theta$, an offline optimization method is usually used to design a good $\Theta$ for HZD [9]. Experiments conducted on a bipedal test-bed named RABBIT[10] yielded stable[11] walking over a wide range of speeds by this method [9].

However, if we need a robot that is able to adapt to current situation itself, we should equip the robot to an online method which interacts with the full-ordered system. In this paper, we are going to propose an online method to design $\Theta$ during walking by the full-order closed-loop hybrid model on an arbitrary slope surface in presence of an arbitrary external force and modeling error. With the proposed method, the online optimization process can be continued on the real robot, with the designed $\Theta$ in the simulation.

## 4. PREPARING FOR APPLYING GPISOC

To prepare for applying GPISOC to the closed-loop system, we present this section with two part. In the first part, the closed-loop system is approximated by assumption $\ddot{z} \approx 0$. In the second part, it is shown that the state space form of the approximated system is consistent with GPISOC formalism.

### 4.1. The effects of $\ddot{z} \approx 0$

Here, we state two important results of the assumption $\ddot{z} \approx 0$. Theorem 4.1 formulates the closed-loop model into a form which is consistent with GPISOC formalism and Theorem 4.3 shows that $\ddot{z} \approx 0$ guarantees the invertibility of decoupling matrix.

[10]Note that we have considered the same assumptions about the robot and its environment as in [9].

[11]A combination of dynamical and mechanical stabilities results in the stability in bipedal walking. Dynamical stability means the existence of a stable periodic orbit for walking. Mechanical stability means the satisfaction of two force conditions in the contact point of the robot legs with the ground. The first is $F^N > 0$ which means the robot is exerting force to the ground ($F^N = 0$ means the robot may fly). The second is $|F^T| < \mu F^N$ which means the robot does not slip. $F^T$ and $F^N$ are tangential and normal forces in the contact point, respectively.

**Theorem 4.1** (Approximation of the closed-loop equations into GPISOC consistent form)**.** *Providing* $\ddot{z} \approx 0$*, the closed-loop model (3.3) takes the following form:*

$$\begin{aligned} \ddot{q}_b &= -K_P q_b - K_D \dot{q}_b + \tilde{G}(z,\dot{z})\,\Theta \\ \ddot{q}_n &= -D_{22}^{-1} D_{21} \ddot{q}_b - D_{22}^{-1} \Omega_2 \end{aligned} \tag{4.1}$$

*where* $\tilde{G}(z,\dot{z})$ *is independent of the model parameters as follows:*

$$\tilde{G}(z,\dot{z}) = \begin{bmatrix} \tilde{g}(z,\dot{z})^T & & 0 \\ & \ddots & \\ 0 & & \tilde{g}(z,\dot{z})^T \end{bmatrix}, \tag{4.2}$$

$$\tilde{g}(z,\dot{z}) = K_P g(s) + K_D \frac{\dot{z}}{z_2 - z_1} \frac{\partial g(s)}{\partial s} + \frac{\dot{z}^2}{(z_2 - z_1)^2} \frac{\partial^2 g(s)}{\partial s^2}.$$

*Proof.* The first row of the hybrid model (3.1) is expanded as:

$$\begin{cases} D_{11}\ddot{q}_b + D_{12}\ddot{q}_n + \Omega_1 = u \\ D_{21}\ddot{q}_b + D_{22}\ddot{q}_n + \Omega_2 = 0 \end{cases} \tag{4.3}$$

where $\Omega_1$ and $\Omega_2$ are the components of the partitioned matrix $\Omega(q,\dot{q}) = W(q,\dot{q}) + \mathcal{G}(q) + F_{fr}(q,\dot{q}) - J^T F_{ext}$. Matrices $D_{11}$, $D_{12}$, $D_{21}$ and $D_{22}$ are the components of the partitioned inertia matrix $D(q)$. Because the inertia matrix $D$ is positive definite, $D_{11}$ and $D_{22}$ are both positive definite and hence invertible [9]. Inserting (3.2) into (4.3) and substitution $\ddot{q}_n$ in the first row from the second row cause to the following closed-loop equations[12]:

$$\begin{aligned} \ddot{e} &= -K_P e - K_D \dot{e}, \\ \ddot{q}_n &= -D_{22}^{-1} D_{21} \ddot{q}_b - D_{22}^{-1} \Omega_2 \end{aligned} \tag{4.4}$$

where the first row in (4.4) is expanded as:

$$\ddot{q}_b - \left[ \frac{\dot{z}^2}{(z_2 - z_1)^2} \frac{\partial^2 G(s)}{\partial s^2} + \frac{\ddot{z}}{z_2 - z_1} \frac{\partial G(s)}{\partial s} \right] \Theta = -K_P (q_b - G(s)\Theta) - K_D \left( \dot{q}_b - \frac{\dot{z}}{z_2 - z_1} \frac{\partial G(s)}{\partial s} \Theta \right)$$

where $\dot{z}^2/(z_2 - z_1)^2 \partial^2 G(s)/\partial s^2 + \ddot{z}/(z_2 - z_1)\partial G(s)/\partial s$ is a diagonal matrix in which the diagonal components are the vector $\Pi = \Upsilon + \ddot{z}/(z_2 - z_1)\partial g(s)/\partial s$ where $\Upsilon = \dot{z}^2/(z_2 - z_1)^2 \partial^2 g(s)/\partial s^2$.

**Remark 4.2.** Providing $\ddot{z} \approx 0$, the term $\ddot{z}/(z_2 - z_1)\,\partial G(s)/\partial s$ can be ignored in the expression $\dot{z}^2/(z_2 - z_1)^2 \partial^2 G(s)/\partial s^2 + \ddot{z}/(z_2 - z_1)\partial G(s)/\partial s$ because of the following reasons:

(i) Since $|z_2 - z_1| < 1$ (see the previous section), $1/(z_2 - z_1)^2$ increases the contribution of $\Upsilon$ in $\Pi$.

(ii) The components of $1/(z_2 - z_1)^2 \partial^2 g(s)/\partial s^2$ and $1/(z_2 - z_1)\partial g(s)/\partial s$ are bounded in $s \in [0,1]$. The absolute values of a component of $1/(z_2 - z_1)^2 \partial^2 g(s)/\partial s^2$ are more than its corresponding component of $1/(z_2 - z_1)\partial g(s)/\partial s$ except in the vicinity of the zeros of $\partial^2 g(s)/\partial s^2$. It also increases the contribution of $\Upsilon$ in $\Pi$, except in the vicinity of the zeros (see Appendix B for more detail about derivatives and zeros of $g(s)$).

(iii) In the normal walking, $z$ monotonically changes from $z^+$ to $z^-$. It means that the sign of $\dot{z}$ does not change. Also, with regard to $\ddot{z} \approx 0$, $\dot{z}$ is slow varying *i.e.* almost constant. Note that the aim of this paper is stable and high-speed walking; therefore $\dot{z}$ cannot be near zero which leads to stop or very slow walking.

[12] $\ddot{e} = -K_P e - K_D \dot{e}$ is also resulted from inserting the control rule in $\ddot{e} = L_f^2 e + (L_h L_f e)u$.

With (i)–(iii) and $\ddot{z} \approx 0$, it is clear that:

$$\Pi \approx \begin{cases} 0, & \text{s is near to zeros of } \frac{\partial^2 g(s)}{\partial s^2} \\ \Upsilon, & \text{otherwise} \end{cases} \tag{4.5}$$

hence the curves of $\Pi$ and $\Upsilon$ are almost matched each other and the term $\ddot{z}/(z_2 - z_1)\partial G(s)/\partial s$ can be ignored. Ignoring term $\ddot{z}/(z_2 - z_1)\partial G(s)/\partial s$ according to Remark 4.2 and factorizing $\Theta$ in the remaining terms leads to (4.1). □

**Lemma 4.3** (Invertibility of $L_h L_f e$)**.** *$L_h L_f e$ is invertible provided that $\ddot{z} \approx 0$.*

*Proof.* See Appendix C. □

## 4.2. State space form of the approximated closed-loop system

Here, we show that if one or all of the joint parameters considered noisy, the approximated model (4.1) is consistent with GPISOC formalism. One noisy joint parameter is used for the case that each joint parameter adjusted independently (joint by joint optimization). All of the joint parameters are considered noisy for the case that all joint parameters adjusted simultaneously.

Reformulating (4.1) in the state space form can be done by considering $x = [x_1; x_2; \ldots; x_{2n}] = [q; \dot{q}]$ as follows:

$$\begin{aligned}
\dot{x}_1 &= x_{n+1} \\
&\vdots \\
\dot{x}_n &= x_{2n} \\
\dot{x}_{n+1} &= -K_P^1 x_1 - K_D^1 x_{n+1} + \tilde{g}(z, \dot{z})^T \theta_1 \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
\dot{x}_{n+d} &= -K_P^d x_d - K_D^d x_{n+d} + \tilde{g}(z, \dot{z})^T \theta_d \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
\dot{x}_{2n-1} &= -K_P^{n-1} x_{n-1} - K_D^{n-1} x_{2n-1} + \tilde{g}(z, \dot{z})^T \theta_{n-1} \\
\dot{x}_{2n} &= -D_{22}^{-1} D_{21} [\dot{x}_{n+1}; \dot{x}_{n+2}; \ldots; \dot{x}_{2n-1}] - D_{22}^{-1} \Omega_2
\end{aligned}$$

where $K_P^d$ and $K_D^d$ denote the $d$th diagonal component of $K_P$ and $K_D$ respectively. Therefore providing all component of $\Theta$ considered noisy, the perturbed equations are:

$$\begin{bmatrix} \dot{x}_{n+1} \\ \vdots \\ \dot{x}_{2n-1} \end{bmatrix} = \begin{bmatrix} -K_P^1 x_1 - K_D^1 x_{n+1} \\ \vdots \\ -K_P^{n-1} x_{n-1} - K_D^{n-1} x_{2n-1} \end{bmatrix} + \tilde{G}(z, \dot{z})(\Theta + \epsilon_t) \tag{4.6}$$

where $\epsilon_t = [\varepsilon_{1,t}; \varepsilon_{2,t}; \ldots; \varepsilon_{n-1,t}]$ and $\varepsilon_{d,t}$ denotes the exploration noise vector of $d$th joint parameter $\theta_d$ which is drawn from a mean-zero multivariate Gaussian distribution with variance $\Sigma_{d,\epsilon}$. Suppose all components of $\Theta$ excluding $\theta_d$ are considered noiseless. Then, the only perturbed equation is:

$$\dot{x}_{n+d} = -K_P^d x_d - K_D^d x_{n+d} + \tilde{g}(z, \dot{z})^T (\theta_d + \varepsilon_{d,t}). \tag{4.7}$$

GPISOC can be applied to (4.6) and (4.7) which are in form (2.10). In the next section, first GPISOC is applied to (4.7) for the case of one noisy parameter vector. The result is useful for applying GPISOC to (4.6), in which all parameter vectors are considered noisy. Applying GPISOC to (4.6) leads to a new algorithm which is able to improve the initial controller (which is parameterized by an initial $\Theta$) until became an optimal controller (parameterized by the designed $\Theta$).

## 5. Applying GPISOC to (4.6) and (4.7)

In this section, the new algorithm is presented through applying GPISOC to the approximated closed-loop model. First, we apply GPISOC to (4.7). In (4.7), $f_t^{(l_2)}$ and $(x_{t_{j+1}}^{(l_2)}-x_{t_j}^{(l_2)})/\delta t$ are:

$$f_t^{(l_2)} = -K_P^d x_d - K_D^d x_{n+d}, \quad \frac{x_{t_{j+1}}^{(l_2)} - x_{t_j}^{(l_2)}}{\delta t} = \dot{x}_{n+d}.$$

Above substitutions in the trajectory cost (2.2) result in:

$$C_d(\tau) = \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j}\delta t + \frac{1}{2}\sum_{j=i}^{N-1} \left|\tilde{g}_{t_j}^T(\theta_d+\varepsilon_{d,t_j})\right|^2_{H_{d,t_j}^{-1}} \delta t + \frac{\lambda}{2}\sum_{j=i}^{N-1} \log H_{d,t_j}$$

where $\tilde{g}_{t_j}$ denotes the value of $\tilde{g}(z,\dot{z})$ at time $t_j$. $H_{d,t_j}=\tilde{g}_{t_j}^T R_d^{-1}\tilde{g}_{t_j}$ is a scalar. The following theorem shows that the term $\lambda/2\sum_{j=i}^{N-1}\log H_{d,t_j}$ can be easily ignored by the cost function designer.

**Theorem 5.1** (Removing an ignorable term)**.** *Assuming* $R_d = (\sigma_d^2/\lambda)I_{(L+1)\times(L+1)}$, *the term* $\lambda/2\sum_{j=i}^{N-1}\log H_{d,t_j}$ *can be ignored with defining large* $Q_{t_j}$ *and small* $\lambda$ *by the cost function designer.*

*Proof.* See Appendix D. □

**Remark 5.2** (Validity of PI$^2$-like updating)**.** By ignoring $\lambda/2\sum_{j=i}^{N-1}\log H_{d,t_j}$ according to Theorem 5.1, the trajectory cost is:

$$C_d(\tau) \approx \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j}\delta t + \frac{1}{2}\sum_{j=i}^{N-1} \left|\tilde{g}_{t_j}^T(\theta_d+\varepsilon_{d,t_j})\right|^2_{H_{d,t_j}^{-1}} \delta t.$$

The time step $\delta t$ has a scaling role; hence it is removed without loss of generality [31]. Therefore we have:

$$\begin{aligned} C_d(\tau) &\propto \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \left|\tilde{g}_{t_j}^T(\theta_d+\varepsilon_{d,t_j})\right|^2_{H_{d,t_j}^{-1}} = \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} (\theta_d+\varepsilon_{d,t_j})^T \tilde{g}_{t_j} H_{d,t_j}^{-1}\tilde{g}_{t_j}^T(\theta_d+\varepsilon_{d,t_j}) \\ &= \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} (\theta_d+\varepsilon_{d,t_j})^T \frac{\tilde{g}_{t_j}\tilde{g}_{t_j}^T}{\tilde{g}_{t_j}^T R_d^{-1}\tilde{g}_{t_j}}(\theta_d+\varepsilon_{d,t_j}). \end{aligned}$$

With substituting $M_{d,t_j} = R_d^{-1}\tilde{g}_{t_j}\tilde{g}_{t_j}^T/(\tilde{g}_{t_j}^T R_d^{-1}\tilde{g}_{t_j})$, we have:

$$C_d(\tau) = \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} (\theta_d+\epsilon_{d,t_j})^T M_{d,t_j}^T R_d M_{d,t_j}(\theta_d+\epsilon_{d,t_j}) \tag{5.1}$$

where (5.1) is in the form of (2.11) which means that a PI$^2$-like updating can be used to design walking gait. The optimal parameter vectors are defined according to (2.12) as:

$$\theta_{d,t_i} = \int_{\tau\in\chi_i} P_d(\tau)\theta_{d,L}(\tau)\mathrm{d}\tau^{(l_2)} = \int_{\tau\in\chi_i} P_d(\tau) M_{d,t_i}(\theta_d+\epsilon_{d,t_i})\mathrm{d}\tau^{(l_2)}$$

where the weight $P_d(\tau)$ is also defined as (2.12). Since the trajectory cost (5.1) is similar to (2.11), $\theta_{d,L}(\tau)$ is computed as (2.13), *i.e.* $\theta_{d,L}(\tau) = M_{d,t_i}(\theta_d+\epsilon_{d,t_i})$, and $\theta_d^{(new)}$ can be iteratively computed as PI$^2$ algorithm in (2.14) for the case of one noisy joint parameter.

The main equations of the algorithm for updating $d$th parameter vector, which is derived from applying GPISOC to (4.7), can be summarized as follows:

$$
\begin{aligned}
C_d(\tau) &= \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} (\theta_d + M_{d,t_j}\varepsilon_{d,t_j})^T R_d(\theta_d + M_{d,t_j}\varepsilon_{d,t_j}),\\
P_d(\tau) &= \frac{\exp\left(-C_d(\tau)/\lambda\right)}{\int_{\tau\in\chi_i}\exp\left(-C_d(\tau)/\lambda\right)\mathrm{d}\tau^{(l_2)}},\\
\theta_d^{(new)} &= \theta_d^{(old)} + \delta\theta_d, \quad \delta\theta_d = \frac{\left[\sum_{i=0}^{N-1}(N-i)\delta\theta_{d,t_i}\right]}{\sum_{i=0}^{N-1}(N-i)}, \quad \delta\theta_{d,t_i} = \int_{\tau\in\chi_i} P_d(\tau) M_{d,t_i}\varepsilon_{d,t_i}\mathrm{d}\tau^{(l_2)}
\end{aligned}
\tag{5.2}
$$

where $M_{d,t_j}$ is only applied to $\varepsilon_{d,t_j}$ (not $\theta_d$) in $C_d(\tau)$ as described in [31]. Exploration noise $\varepsilon_{d,t}$ is the only open algorithmic parameter of the algorithm. The integrations in (5.2) can be approximated by the summations over $K$ sample trajectories $\tau_{k=1,\ldots,K}$ as mentioned in Section 2.

With the above algorithm, $n-1$ separate learning processes for adjusting $n-1$ joint parameter vectors should be done. Simultaneously applying the noise to all joint parameters can solve this problem that is studied in the next theorem.

**Theorem 5.3** (Simultaneous iterative updating possibility)**.** *Applying GPISOC to* (4.6) *leads to simultaneously iterative updating of each parameter vector as* (5.2) *but using the following shared trajectory cost:*

$$
C(\tau) = \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1}\sum_{d=1}^{n-1} (\theta_d + M_{d,t_j}\varepsilon_{d,t_j})^T R_d(\theta_d + M_{d,t_j}\varepsilon_{d,t_j}). \tag{5.3}
$$

*Proof.* We know that $\Sigma_{d,\varepsilon} = \lambda R_d^{-1}$ and $\Sigma_\epsilon = \lambda R^{-1}$; hence the matrices $\Sigma_\epsilon$, $R^{-1}$, and $H_{t_j}$ are:

$$
\Sigma_\epsilon = \begin{bmatrix} \Sigma_{1,\varepsilon} & & 0 \\ & \ddots & \\ 0 & & \Sigma_{n-1,\varepsilon} \end{bmatrix}, \quad R^{-1} = \begin{bmatrix} R_1^{-1} & & 0 \\ & \ddots & \\ 0 & & R_{n-1}^{-1} \end{bmatrix}
$$

$$
H_{t_j} = \tilde{G}_{t_j} R^{-1} \tilde{G}_{t_j}^T = \begin{bmatrix} \tilde{g}_{t_j}^T R_1^{-1} \tilde{g}_{t_j} & & 0 \\ & \ddots & \\ 0 & & \tilde{g}_{t_j}^T R_{n-1}^{-1} \tilde{g}_{t_j} \end{bmatrix} = \begin{bmatrix} H_{1,t_j} & & 0 \\ & \ddots & \\ 0 & & H_{n-1,t_j} \end{bmatrix}
$$

where $\tilde{G}_{t_j}$ denotes the value of $\tilde{G}(z,\dot{z})$ at time $t_j$. Hence, $C(\tau)$ for (4.6) is computed as:

$$
C(\tau) = \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j}\delta t + \frac{1}{2}\sum_{j=i}^{N-1} \left| \frac{x_{t_{j+1}}^{(l_2)} - x_{t_j}^{(l_2)}}{\delta t} - f_{t_j}^c \right|_{H_{t_j}^{-1}}^2 \delta t + \frac{\lambda}{2}\sum_{j=i}^{N-1} \log\|H_{t_j}\|
$$

where $\log\|H_{t_j}\| = \log\left(\prod_{d=1}^{n-1} H_{d,t_j}\right) = \sum_{d=1}^{n-1}\log H_{d,t_j}$. Hence $\lambda/2\sum_{j=i}^{N-1}\log\|H_{t_j}\| = \sum_{d=1}^{n-1}(\lambda/2\sum_{j=i}^{N-1}\log H_{d,t_j})$ which can be ignored according to Theorem 5.1 by the cost function designer. Therefore we have:

$$
C(\tau) \approx \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j}\delta t + \frac{1}{2}\sum_{j=i}^{N-1} \left| \frac{x_{t_{j+1}}^{(l_2)} - x_{t_j}^{(l_2)}}{\delta t} - f_{t_j}^c \right|_{H_{t_j}^{-1}}^2 \delta t
$$

where $\delta t$ has a scaling role and can be removed. Also, according to (4.6), $f_t^{(l_2)}$ and $(x_{t_{j+1}}^{(l_2)} - x_{t_j}^{(l_2)})/\delta t$ are:

$$f_t^{(l_2)} = \begin{bmatrix} -K_P^1 x_1 - K_D^1 x_{n+1} \\ \vdots \\ -K_P^{n-1} x_{n-1} - K_D^{n-1} x_{2n-1} \end{bmatrix}, \quad \frac{x_{t_{j+1}}^{(l_2)} - x_{t_j}^{(l_2)}}{\delta t} = \begin{bmatrix} \dot{x}_{n+1} \\ \vdots \\ \dot{x}_{2n-1} \end{bmatrix}.$$

Therefore we have:

$$\begin{aligned} C(\tau) \propto \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j} + \frac{1}{2} \sum_{j=i}^{N-1} \left| \tilde{G}_{t_j}(\Theta + \epsilon_{t_j}) \right|_{H_{t_j}^{-1}}^2 &= \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j} + \frac{1}{2} \sum_{j=i}^{N-1} (\Theta + \epsilon_{t_j})^T \tilde{G}_{t_j}^T H_{t_j}^{-1} \tilde{G}_{t_j} (\Theta + \epsilon_{t_j}) \\ &= \phi_{t_N} + \sum_{j=i}^{N-1} Q_{t_j} + \frac{1}{2} \sum_{j=i}^{N-1} (\Theta + \epsilon_{t_j})^T M_{t_j}^T R M_{t_j} (\Theta + \epsilon_{t_j}) \end{aligned}$$

where $M_{t_j}$ is defined as:

$$M_{t_j} = R^{-1} \tilde{G}_{t_j}^T H_{t_j}^{-1} \tilde{G}_{t_j} = \begin{bmatrix} M_{1,t_j} & & 0 \\ & \ddots & \\ 0 & & M_{n-1,t_j} \end{bmatrix}.$$

Therefore, expanding the third term in $C(\tau)$ results in:

$$\frac{1}{2} \sum_{j=i}^{N-1} (\Theta + \epsilon_{t_j})^T M_{t_j}^T R M_{t_j} (\Theta + \epsilon_{t_j}) = \frac{1}{2} \sum_{j=i}^{N-1} \sum_{d=1}^{n-1} (\theta_d + \varepsilon_{d,t_j})^T M_{d,t_j}^T R_d M_{d,t_j} (\theta_d + \varepsilon_{d,t_j}).$$

Note that $M_{d,t_j}$ should be only applied to $\varepsilon_{d,t_j}$ (not $\theta_d$) as described in [31]; therefore $C(\tau)$ takes the form (5.3). Also, since the trajectory cost $C(\tau)$ is similar to (2.11), the optimal parameter vectors according to (2.12) and (2.13) are:

$$\Theta_{t_i} = \int_{\tau \in \chi_i} P(\tau) M_{t_i} (\Theta + \varepsilon_{t_i}) \mathrm{d}\tau^{(l_2)}, \quad i = 0, \ldots, N-1. \tag{5.4}$$

Expanding (5.4) leads to:

$$\theta_{d,t_i} = \int_{\tau \in \chi_i} P(\tau) M_{d,t_i} (\theta_d + \varepsilon_{d,t_i}) \mathrm{d}\tau^{(l_2)}, \quad d = 1, \ldots, n-1. \qquad \square$$

The pseudo-code of the proposed algorithm which is called PI$^2$-WG is given in Table 1. During the learning with PI$^2$-WG, the robot uses the feedback controller to walk. PI$^2$-WG only needs to know the value of transition matrix $\tilde{g}(cq, c\dot{q})$ along all rollouts in which $q$ and $\dot{q}$ are specified by the robot sensors.

**Remark 5.4.** During the learning, when the cost of the current noiseless rollout is the lowest until now, updating the initial state $x_0$ with an intermediate state in the rollout can promote the results; hence we use this technique in our algorithm. Without the technique, the gait is designed with respect to a constant initial state; consequently considering a good initial state will be a problem.

**Remark 5.5.** The simulated robot may be incapable to walk even one successful step with initial $\Theta$ in the test phase. When the algorithm reaches to a stable walking gait, a continuous rollout capturing can be alternatively used instead of the capturing rollouts starting from $x_0$. In other words, a new rollout started after each impact which leads to overlapped rollouts. With this approach the algorithm is also applicable to the real robot and faster updating is attained.

TABLE 1. Pseudo-code of PI$^2$-WG for planar point-feet biped robot.

**Given:**
Transition matrix $\tilde{g}_t$ according to (4.2),
Arbitrary cost function part $Q_t$ and final cost $\phi_{t_N}$ (defined by cost function designer),
Variance matrices $\Sigma_{d,t}$ of mean-zero noises $\varepsilon_{d,t}$, $d = 1, \ldots, n-1$,
Initial values of the trajectory parameter vectors $\theta_d$, $d = 1, \ldots, n-1$,
Start state $x_0$ (*i.e.* Initial state of the robot),
**Repeat until convergence of noiseless trajectory cost**
Create $K$ rollout from current $x_0$ by stochastic $\theta_d + \varepsilon_{d,t}$
For $d = 1, \ldots, n-1$, $j = 0, \ldots, N-1$, and $k = 1, \ldots, K$ do:
| $H_{d,t_j,k} = \tilde{g}^T_{t_j,k} R_d^{-1} \tilde{g}_{t_j,k}$
| $M_{d,t_j,k} = (R_d^{-1} \tilde{g}_{t_j,k} \tilde{g}^T_{t_j,k}) / (\tilde{g}^T_{t_j,k} R_d^{-1} \tilde{g}_{t_j,k})$
For $i = 0, \ldots, N-1$ do:
| For $k = 1, \ldots, K$ do:
| | $C(\tau_k) = \phi_{t_N,k} + \sum_{j=i}^{N-1} Q_{t_j,k} + \frac{1}{2} \sum_{j=i}^{N-1} \sum_{d=1}^{n-1} (\theta_d + M_{d,t_j,k} \varepsilon_{d,t_j,k})^T R_d (\theta_d + M_{d,t_j,k} \varepsilon_{d,t_j,k})$
| | $P(\tau_k) = \exp(-C(\tau_k)/\lambda) / \sum_{j=1}^{K} \exp(-C(\tau_j)/\lambda)$
| For $d = 1, \ldots, n-1$ do:
| | $\delta\theta_{d,t_i} = \sum_{k=1}^{K} P(\tau_k) M_{d,t_i,k} \varepsilon_{d,t_i,k}$
For $d = 1, \ldots, n-1$ do:
| $\delta\theta_d = \left[ \sum_{i=0}^{N-1} (N-i) \delta\theta_{d,t_i} \right] / \sum_{i=0}^{N-1} (N-i)$
| $\theta_d = \theta_d + \delta\theta_d$
Create one noiseless rollout to check the convergence of noiseless trajectory cost.

## 6. RELATED WORKS

In this section, we discuss related work in the area of designing dynamically stable walking gait with desired features for biped robots with point-feet. In the area of planar biped robot with point-feet which is the robot type of this paper, Westervelt *et al.* [9, 28] show that if $\Theta$ in (3.3) is selected so that the 1-DOF hybrid zero dynamics admits an exponentially stable orbit, then an exponentially stable walking motion can be achieved. Minimization of the cost $J(\Theta)$ through the reduced-order HZD to reach a gait with desired features must be done fast, but it is rather hard from the following views:

(i) Walking with initial $\Theta$ should have *dynamical stability*, because the cost $J(\Theta)$ is computed over a single step of walking on the periodic orbit. Finding such initial gait is difficult especially when a robot with high DOF and large number of Bezier basis is considered, *i.e.* when $n$ and $L$ are large numbers.

(ii) A valid initial $\Theta$ for the optimization may be invalid by a little change in the slope of learning surface. Since the robot state just before impact defined as $q^- = [\theta_1^L; \ldots; \theta_{n-1}^L; q_n^-]$, $q_n^-$ is tuned so that the swing and stance legs are both in contact with the surface in state $q^-$ that is illustrated in Figure 4. Therefore $q_n^-$ (and consequently $z^-$) is determined with respect to the slope of the surface. Therefore the slope of the surface affects on the gait $G(s(z))\Theta$ in which $s(z) = (z - z^+)/(z^- - z^+)$. It means that for each $q$ and $\Theta$, $G(s(z))\Theta$ has different values on different slopes. Therefore we expect that a valid initial gait on a special slope is invalid for the other slopes. We usually should find a new initial $\Theta$ for optimization on a different slope which is hard according to (i).

(iii) Minimization of the torques requires the computation of the torques *via* the full-dynamic model [27]. Also, the computation of the reaction forces at the end of swing leg just after impact is done *via* the impact map of the full-dynamic model. In fact, we cannot claim that the optimization is purely done on the reduced-order HZD and it includes the computations which are costly in calculation time.

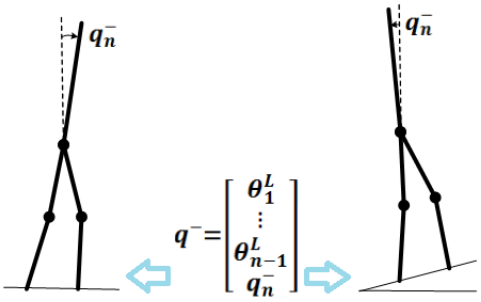


FIGURE 4. Tuning $q_n^-$ for different slopes with the same $[\theta_1^L; \ldots; \theta_{n-1}^L]$. It seems that $G(s(z))\Theta$ is an invalid gait for the slope in the right of the figure.

There is also an extension of Westervelt method for 3D biped robots with point-feet in which cost minimization can be similarly done while preserving dynamical stability [29].

In the recent year in the area of biped robots with point-feet, some research are done by Akbari Hamed *et al.* [44, 45] that their objective are only stabilization of periodic orbit[13] by enforcing eigenvalues of Poincare map Jacobian in the unit circle via solving a LMI/BMI optimization problem [44,45]. In other words, some extra inequality constraints are required to reach the desired features[14]. But, these constraints *cannot* be expressed in the format of BMIs and LMIs [44]. So it is assumed that these constraints are somehow satisfied [44]. Hence, the method cannot guarantee mechanical stability especially in a slope surface with high degrees.

In the case of *planar* biped robots with point feet, it is clear that checking exponential stability via the reduced-order HZD is easier and faster than Akbari Hamed method. In the next section, we show how the cost of PI$^2$-WG is increased by unsatisfaction of exponential stability conditions which are defined via the reduced-order HZD[15].

Note that we are going to design the cost function of PI$^2$-WG so that an initial walking gait *without dynamical stability*[16] can be enough for the learning. Also, we realize that the initial walking gait can be used in a lot of different scenarios by replacing $z^+$ and $z^-$ with fixed $z_1$ and $z_2$ in $s(z) = (z - z^+)/(z^- - z^+)$, which leads to easier gait design[17] for planar biped robot with point-feet. Without the replacement, slope change leads to change of $z^+$ and $z^-$ according to (ii) and consequently change in $s(z)$. Therefore the walking gesture, *i.e.* $G(s(z))\Theta$, is also changed with modification of $z^+$ and $z^-$ which usually leads to invalidity of $G(s(z))\Theta$ for the new slope. But using fixed $z_1$ and $z_2$ in $s(z)$, the slope change does not change the walking gesture and consequently there are more chances of the gait validity for the new slope.

## 7. RESULT

In this section, the performance of PI$^2$-WG is examined on the simulated model of RABBIT [9] and compared with the Westervelt method.

### 7.1. Cost function definition

PI$^2$-WG is evaluated in three different learning scenarios starting from initial $\Theta$ and with/without a modeling error (See Appendix A for more detail about initial $\Theta$ and modeling errors ME1-ME3). The first

[13] *I.e.* only dynamical stability is considered and mechanical stability is ignored to formulate in the optimization problem.

[14] *E.g.* feasibility of positions, velocities, torques and ground reaction forces (*i.e.* mechanical stability) in case of bipedal walking.

[15] In our future works on 3D biped robots with point-feet, we will use Akbari Hamed method to reward stability conditions.

[16] Several steps of walking without mechanical stability are enough for the proposed method. See Remark A.2 for more detail.

[17] With this approach, $[\theta_1^0; \ldots; \theta_{n-1}^0]$ and $[\theta_1^1; \ldots; \theta_{n-1}^1]$ should be also participated in optimization contrary to Westervelt method.

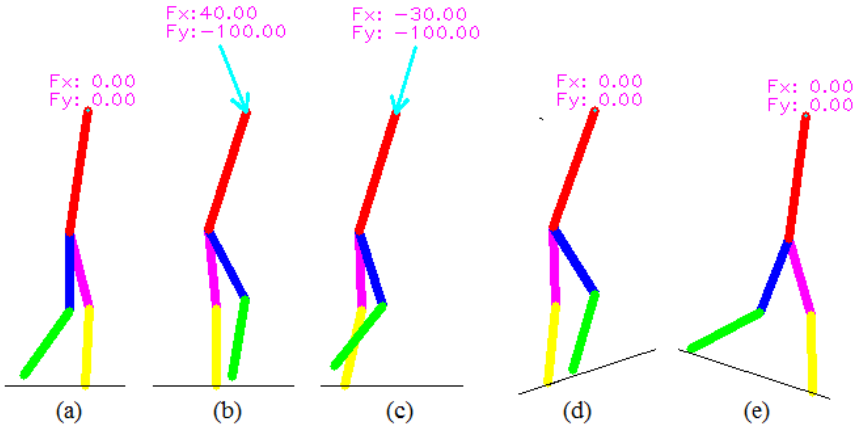


FIGURE 5. A snapshot of the walking gesture in the different scenarios: (a) on the flat (b) on the flat in the presence of the external force [40 N; –100 N] (c) on the flat in the presence of the external force [–30 N; –100 N] (d) on a slope of 18 degrees (e) on a slope of –18 degrees.

learning scenario is designing the walking gait for the flat regarding the following viewpoints of measuring performance:

(i) Mechanical stability.
(ii) Existence and exponential stability of a periodic orbit.
(iii) Normal walking conditions.
(iv) Desired walking speed.
(v) Acceptable amounts of applied torque.

The second scenario is the same as the first scenario, but an external force is exerted on the torso of the robot. The third learning scenario is designing the walking gait for a slope surface with the same points of view. Figure 5 incudes the walking snapshots of the different learning scenarios. The numbers above the torso of the robot denote the horizontal and vertical components of the external force which is exerted on the torso. The yellow and green links of the robot respectively denote the tibia in the stance and swing legs.

In the case of the point-feet biped robot, applying large noises to $\Theta$ may cause to fail walking because the decoupling matrix may be not invertible or mechanical instability occurs. Hence, only small noises are acceptable. The variance matrices of the exploration noises are assumed according to Theorem 5.1 as follows:

$$\Sigma_{d,\varepsilon} = \sigma_d^2 I_{(L+1)\times(L+1)}, \quad d = 1, \ldots, n-1 \tag{7.1}$$

where the initial $\sigma_d^2$ is determined based on the components of initial $\theta_d$, which is defined in (A.1), as:

$$\sigma_{d,\text{init}}^2 < 0.8 \left( \max_{0 \le m \le L} \theta_{d,\text{init}}^m - \min_{0 \le m \le L} \theta_{d,\text{init}}^m \right).$$

Therefore, we assume the following values for $\sigma_d^2$:

$$\sigma_3^2 = \rho, \quad \sigma_1^2 = 2\rho, \quad \sigma_2^2 = 3\rho, \quad \sigma_4^2 = 5\rho$$

where $\rho$ is initialized as mentioned in Table 2. 1500 updates are done for each scenario. The value of $\rho$ has been uniformly decreased from $\rho_{init}$ to 0.0001 along the early 1000 updates and after that $\rho = 0.0001$. Since constant exploration noise along a rollout (*i.e.* $\forall i, \varepsilon_{d,t_i,k} = \varepsilon_{d,t_0,k}$) leads to a faster convergence [43], we use it in this paper. We generate 10 rollouts per update ($K = 10$). The sampling time $\delta t = t_i - t_{i-1}$ is set to 0.01s. Each rollout captures 3 seconds of walking, *i.e.* $N = 3/\delta t = 300$.

TABLE 2. Initial value of $\rho$ and desired speed.

| Scenario | $\boldsymbol{\rho}_{\text{init}}$ | $\boldsymbol{v}_{\mathbf{des}}$ (m/s) |
|---|---|---|
| Flat (without exerting external force) | 0.002 | 0.8 |
| Force –30, –100 (on the flat) | 0.02 | 0.8 |
| Force 40, –100 (on the flat) | 0.005 | 1.4 |
| Slope –18 (without exerting external force) | 0.02 | 0.8 |
| Slope 18 (without exerting external force) | 0.005 | 0.5 |

**Remark 7.1.** We know that $\sigma_3^2 < \sigma_1^2 < \sigma_2^2 < \sigma_4^2$ with the following reasons:

(i) In $cq$, the coefficients of the joint angles of the swing leg, *i.e.* $q_2$ and $q_4$, are zero according to $c = [-1, 0, -0.5, 0, -1]$, *i.e.* $q_2$ and $q_4$ do not play a role in determining the gait $G(s(cq))$; therefore $q_2$ and $q_4$ can take more noise. It means $\sigma_3^2, \sigma_1^2 < \sigma_2^2, \sigma_4^2$.
(ii) Among the joint angles of the stance leg, only $q_1$ observably changes along a step in normal walking. Therefore the hip joint $q_1$ can take more noise than the knee joint $q_3$, *i.e.* $\sigma_3^2 < \sigma_1^2$.
(iii) The knee of the swing leg has the lowest limitation for movement during walking; therefore $q_4$ can take more noise, *i.e.* $\sigma_2^2 < \sigma_4^2$.

With above information, we choose $\sigma_3^2 = \rho, \quad \sigma_1^2 = 2\rho, \quad \sigma_2^2 = 3\rho, \quad \sigma_4^2 = 5\rho$ to simplify decreasing amount of the noise during learning by decreasing $\rho$.

Regarding the mentioned performance viewpoints 1–4, the arbitrary cost function part $Q_t$, $t \in \{t_0, \ldots, t_{N-1}\}$, and the final cost $\phi_{t_N}$ are defined as:

$$\begin{aligned} Q_t &= r_{t,\text{stance}} + r_{t,\text{impact}} + r_{t,\text{knee}} + r_{t,\text{torque}} + 2\ddot{z} \\ \phi_{t_N} &= A(\text{speed}_{\text{lastStep}} - v_{\text{des}})^2 \end{aligned}$$

where the final cost $\phi_{t_N}$ is proportional to square difference between the speed of the last step in a rollout and the desired speed $v_{\text{des}}$ which is set as mentioned in Table 2. The coefficient $A$ is set as:

$$A = \begin{cases} 5\,000\,000, & \text{speed}_{\text{lastStep}} < v_{\text{des}} \\ 50\,000, & \text{otherwise} \end{cases}$$

and $\text{speed}_{\text{lastStep}}$ in a rollout is computed as follows:

$$\text{speed}_{\text{lastStep}} = \frac{\text{last step length}}{\text{last step time}} = \frac{|p_2(q_{t_{\text{lastImpact}}})|}{t_{\text{lastImpact}} - t_{\text{lastButOneImpact}}}$$

where $p_2 = (p_2^h; p_2^v)$ denotes the position of the swing leg end in the coordinate frame which is placed in the stance leg end. Note that $p_2$ is a function of the robot configuration $q$. The last step in a rollout is the step which ends with the last impact. The last impact moment is $t_{\text{lastImpact}}$ and the last but one impact moment is $t_{\text{lastButOneImpact}}$. $|.|$ denotes the Euclidean distance from the origin.

**Note 7.2.** The coefficients in the cost function can be specialized with regard to the importance of the desired features. However, we use the same coefficient values in all the scenarios in order to simplify reporting.

$Q_t$ contains five parts which are defined by the cost function designer. The first part of $Q_t$, *i.e.* $r_{t,\text{stance}}$, measures the mechanical stability of the stance leg at time $t$ as follows:

$$r_{t,\text{stance}} = \begin{cases} 1500\,(\mu_1 - \mu_{\text{des}})\,, & \mu_1 > \mu_{\text{des}} \\ 0, & \text{otherwise} \end{cases}$$

where $\mu_1 = |F^T_{1,t}|/(F^N_{1,t} + \epsilon)$ in which $F^T_{1,t}$ and $F^N_{1,t}$ are the tangential and normal components of the forces exerted at the contact of the stance leg with the ground at time $t$ respectively and $\epsilon = 0.0000000001$. Therefore, $\mu_1$ denotes the minimum required friction coefficient of the ground contact point which insures the stance leg not to slip. $\mu_{\text{des}} = 0.7$ for learning on a slope surface and is set to 0.5 for the other scenarios. The second part of $Q_t$ is $r_{t,\text{impact}} = r^1_{t,\text{impact}} + r^2_{t,\text{impact}} + r^3_{t,\text{impact}}$ in which $r^1_{t,\text{impact}}$ measures the mechanical stability of the swing leg at impact as:

$$r^1_{t,\text{impact}} = \begin{cases} 15000(\mu_2 - \mu_{\text{des}}), & \mu_2 > \mu_{\text{des}} \text{ and } t \in T_{\text{impact}} \\ 0, & \text{otherwise} \end{cases}$$

where $\mu_2 = |F^T_{2,t}|/(F^N_{2,t} + \epsilon)$ in which $F^T_{2,t}$ and $F^N_{2,t}$ are the tangential and normal components of the forces exerted at the contact of the swing leg end with the ground at impact, respectively. In other words, $\mu_2$ denotes the minimum required friction coefficient of the ground at impact which insures the swing leg not to slip. Since during the learning, an unrealistic and very large fiction coefficient is considered for the surface, mechanical instability only happens when the normal force $F^N$ is zero. A very large cost (and consequently small updating weight) is a result of the occurrence of $F^N = 0$ in a rollout that it is due to the term $\mu = F^T/(F^N + \epsilon)$ in the cost function. It means that the proposed algorithm is able to quickly save the robot from the mechanical instability, provided that it is possible. Note that if $\mu > 200$, we consider $\mu = 200$ to reach a smoother learning curve.

The second part of $r_{t,\text{impact}}$, *i.e.* $r^2_{t,\text{impact}}$, increases the cost providing that the robot cannot lift its stance leg after impact without interaction:

$$r^2_{t,\text{impact}} = \begin{cases} 1500000|\dot{p}^N_1 - 0.01|, & \dot{p}^N_1 \leq 0 \text{ and } t \in T_{\text{impact}} \\ 0, & \text{otherwise} \end{cases}$$

where $\dot{p}^N_1$ denotes the normal component of the post-impact swing-leg velocity which should be positive, otherwise the robot cannot lift its stance leg without interaction [9].

$r^3_{t,\text{impact}}$ increases the cost if walking is not exponentially stable. Expressing hybrid zero dynamics of (3.3) in coordinate $(\xi_1; \xi_2)$ where $\xi_1 = z$, $\xi_2 = \frac{1}{2}\gamma^2$, and $\gamma = D(q)\dot{q}$, Poincare map (*i.e.* impact map $\Delta : (\xi^-_1; \xi^-_2) \to (\xi^+_1; \xi^+_2)$) of the hybrid zero dynamics is defined as [9, 28]:

$$\begin{aligned} \xi^+_1 &= z^+ \\ \xi^+_2 &= \delta^2_{\text{zero}} \xi^-_2 . \end{aligned}$$

Therefore, computing $\delta^2_{\text{zero}}$ will help us to check if the following criteria of existence and exponential stability of periodic orbit are satisfied or not (NBEC4-NBEC5 constraints in [28] or NEC4-NEC5 in [9]):

(NEC4) Existence criterion: $CR < 0$.

(NEC5) Exponential stability criterion: $0 < \delta^2_{\text{zero}} < 1$.

where $CR = \mathcal{V}_{\text{zero}}(z^-)\delta^2_{\text{zero}}/(1 - \delta^2_{\text{zero}}) + \mathcal{V}^{\text{MAX}}_{\text{zero}}$ in which $\mathcal{V}_{\text{zero}}(z)$ denotes the potential energy of the hybrid zero dynamics and $\mathcal{V}^{\text{MAX}}_{\text{zero}} = \max_{z^+ < z < z^-} \mathcal{V}_{\text{zero}}(z)$ [9,28]. Therefore $r^3_{t,\text{impact}} = r^3_{t,NEC4} + r^3_{t,NEC5} + r^3_{t,\text{FixedPoint}}$ where $r^3_{t,NEC4}$, $r^3_{t,NEC5}$, and $r^3_{t,\text{FixedPoint}}$ defined as:

$$r^3_{t,NEC4} = \begin{cases} 100CR, & CR > 0 \text{ and } t \in T_{\text{impact}} \\ 0, & \text{otherwise} \end{cases}, \quad r^3_{t,NEC5} = \begin{cases} 100\delta^2_{\text{zero}}, & \delta^2_{\text{zero}} > 1 \text{ and } t \in T_{\text{impact}} \\ 0, & \text{otherwise} \end{cases}$$

$$r^3_{t,\text{FixedPoint}} = 100|\mathcal{V}_{\text{zero}}(z^-) - \mathcal{V}^{\text{Previous}}_{\text{zero}}(z^-)|, \quad t \in T_{\text{impact}}$$

where $r^3_{t,\text{FixedPoint}}$ penalizes the initial state which is not already on a periodic orbit ($\mathcal{V}_{\text{zero}}(z^-)$ should be a constant on a periodic orbit). The third part of $Q_t$ is $r_{t,\text{knee}} = r_{t,\text{knee1}} + r_{t,\text{knee2}}$ in which $r_{t,knee1}$ and $r_{t,\text{knee2}}$ are defined as:

$$r_{t,\text{knee1}} = \begin{cases} -100000q_3, & q_3 < 0 \\ 0, & \text{otherwise} \end{cases}, \quad r_{t,\text{knee2}} = \begin{cases} -100000q_4, & q_4 < 0 \\ 0, & \text{otherwise.} \end{cases}$$

It means that we do not allow a gait in which the knee joints have negative angles. In other words, these two parts avoid abnormal walking.

The forth part of $Q_t$, *i.e.* $r_{t,\text{torque}}$, is considered to decrease the pick torques of motors. It insures that the torques will be in the valid range [-150 Nm, 150 Nm] and it is defined as:

$$r_{t,\text{torque}} = 100 \sum_{d=1}^{n-1} sat_d$$

where $sat_d$ is defined as follows:

$$sat_d = \begin{cases} |u_d| - pick_{\text{des}}, & |u_d| > pick_{\text{des}} \\ 0, & \text{otherwise} \end{cases}$$

where $u_d$ denotes the applied torque to the $d$th joint. Note that there is no torque limitation during the learning and saturation is activated after learning. We consider $pick_{\text{des}} = 60$ for the first scenario and $pick_{\text{des}} = 140$ for the other scenarios.

The fifth part of $Q_t$ is added to avoid generating the walking gaits in which the decoupling matrix is not invertible. We find that choosing a small coefficient for $\ddot{z}$ leads to a better performance, but $\ddot{z} \approx 0$ may not be satisfied. Therefore, we also think of the case that the robot fails walking, when the decoupling matrix is not invertible or the robot is going to fall down, which is discovered by checking the hip height of the robot, or walking backward. In these cases the remaining immediate costs in the current rollout fill with 1500000 which lead to a large cost and the rollout is terminated.

Since the noise is small, we have discovered that ignoring $1/2 \sum_{j=i}^{N-1} \sum_{d=1}^{n-1} (\theta_d + M_{d,t_j} \varepsilon_{d,t_j})^T R_d (\theta_d + M_{d,t_j} \varepsilon_{d,t_j})$ in $C(\tau)$ does not have a sensible effect on the results, *i.e.* it acts like a constant that is added to the cost of the rollouts which are participated in an update (see the sample run in Fig. 6). Therefore, we can ignore this part to reduce the complexity of PI$^2$-WG. Figure 6 also illustrates that the variation of the term $\sum_{d=1}^{n-1} \lambda \sum_{j=1}^{N-1} \log H_{d,t_j}$ is trivial in proportion to the value of the other terms in the trajectory cost. Therefore, this term can be ignored without a sensible effect on the performance that verifies Theorem 5.1.

**Remark 7.3.** With above simplification, PI$^2$-WG is similar to the simplified version of PI$^2$ in [43] (We also extend the simplified version of PI$^2$ to adjust the gait and observe that it is also applicable with about the same performance).

## 7.2. Evaluations and discussion

Figure 5 illustrates the snapshots of walking attitudes with the gaits which are designed by PI$^2$-WG on the different scenarios. The robot cannot walk even one successful step with the initial $\Theta$ according to Remark A.2 in Appendix A, but after learning, it is able to walk with exponential stability. Table 3 illustrates the features of the gaits which are designed by PI$^2$-WG on the different scenarios.

The statistics of the designed gaits, which are mentioned in Table 4, show that all of the designed gaits satisfy the criteria NEC4-NEC5. In other words, the results show that using PI$^2$-WG, the robot is able to learn how to walk with exponential stability in the new situations even in the presence of large modeling error in designing the feedback controller.

**Remark 7.4.** The ability of compensating large modeling errors is an interesting feature for continuing the learning with the real robot which was impossible by the offline optimization on the reduced-order system.

The rows 1–6 of Table 3 show that the robot can walk on a flat surface with the static friction 0.6 in the presence of the external force ([–30 N; –100 N] or [40 N; –100 N]) and modeling errors ME1-ME2, which are

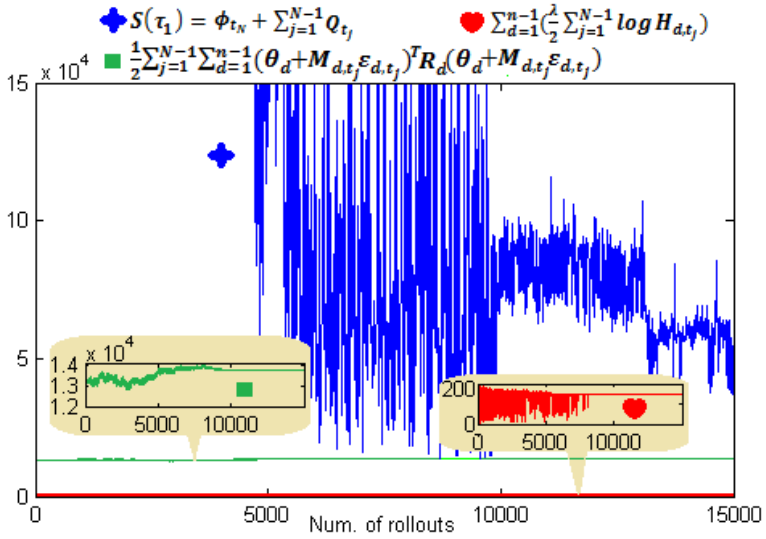


FIGURE 6. The value of $\sum_{d=1}^{n-1} \lambda \sum_{j=1}^{N-1} \log H_{d,t_j}$ during a sample run of PI$^2$-WG on a slope of –18 degrees is reported (with $\rho = 0.02$ and $\lambda = 0.1$).

TABLE 3. The features of some example designed gaits for the robot.

| Gait | Learned environment | $\mu_{\text{req}}$ [a] | $\lvert \text{speed}_{\text{lastStep}} - v_{\text{des}} \rvert$ (m/s) | $\sum_{i=0}^{N-1}\sum_{d=1}^{n-1} sat_d$ (Nm) |
|---|---|---|---|---|
| G1 | Flat | 0.4854 | 0.0003 | 0 |
| G2 | Flat with ME1 | 0.5892 | 0.0009 | 41 |
| G3 | Force [–30 N;-100 N] | 0.4021 | 0.0019 | 0 |
| G4 | Force [–30 N;-100 N] with ME2 | 0.4175 | 0.0142 | 0 |
| G5 | Force [40 N;-100 N] | 0.4993 | 0.0168 | 0 |
| G6 | Force [40 N;-100 N] with ME2 | 0.4139 | 0.0759 | 0 |
| G7 | Slope –18 | 0.6758 | 0.1619 | 0 |
| G8 | Slope –18 with ME3 | 0.6725 | 0.0174 | 0 |
| G9 | Slope 18 | 0.3946 | 0.0022 | 0 |
| G10 | Slope 18 with ME3 | 0.6456 | 0.0045 | 0 |

[a] $\mu_{\text{req}}$: Required friction coefficient of the ground, *i.e.* $\mu_{\text{req}} = \max(\max_{\text{during rollout}}(\mu_1), \max_{\text{during rollout}}(\mu_2))$.

defined in Table A.2 in Appendix A. The rows 7–10 of the table show that the robot can walk on a slope of –18 (or 18) degrees with the static friction 0.7 even in presence of the modeling error ME3[18].

Figure 7 includes the average learning curve which is extracted from averaging 10 runs of PI$^2$-WG for designing G3/G4, which are learned in presence of external force [–30 N; –100 N] with/without a modeling error. The curve is drawn versus the number of the updates which are iteratively done by PI$^2$-WG. The phase portrait in Figure 8 is drawn for 100 steps of walking with G3 and G4. The red circles in the figure denote the initial states of the robot. Figure 9 denotes the applied torques of motors during walking with G3 and G4. The figure shows that the motor torques are in the valid range [–150 Nm, 150 Nm].

Note that the figures of the average learning curves, phase portraits, and motor torques are only shown for G3 and G4 in this paper, but the reader can generate the figures for the other gaits by our executable program which can be downloaded from `www.RoboRL.ir`.

[18] We realize that finding an exponentially stable walking gait is possible even for a slope of 30 degrees but with a different initial gait.

TABLE 4. Example gait statistics for the robot. The statistics shows that all the designed gaits are exponentially stable.

| Gait | $\mathcal{V}_{\text{zero}}(z^-)$ $(\text{kgm}^2/\text{s})^2$ | $\mathcal{V}_{\text{zero}}^{MAX}$ $(\text{kgm}^2/\text{s})^2$ | $\delta^2_{\text{zero}}$ | CR | Pick torque (Nm) | $\bar{v}^a$ (m/s) |
|---|---|---|---|---|---|---|
| GW[b] | –142 | 182 | 0.741 | –224 | 64 | 0.800 |
| G1 | –78 | 71.9 | 0.82 | –283 | 59.3 | 0.80 |
| G2 | –79.3 | 69.5 | 0.815 | –280 | 60.99 | 0.80 |
| G3 | –117.4 | 2.20 | 0.89 | –948 | 122 | 0.80 |
| G4 | –141 | 2.6 | 0.86 | –864 | 119.9 | 0.81 |
| G5 | –381 | 122.2 | 0.41 | –143 | 133 | 1.42 |
| G6 | –3.7e+2 | 115.8 | 0.42 | –152 | 131 | 1.48 |
| G7 | –8.2e+2 | 1.0e+2 | 0.24 | –159 | 108.4 | 0.97 |
| G8 | –9.6e+2 | 1.6e+2 | 0.20 | –80 | 131 | 0.82 |
| G9 | –15.2 | 12.5 | 0.88 | –99 | 125.5 | 0.50 |
| G10 | –23 | 16 | 0.81 | –82 | 93 | 0.50 |

[a] $\bar{v}$: Average walking speed. [b] GW: the gait which is designed by Westervelt method for the flat and reported from [9].

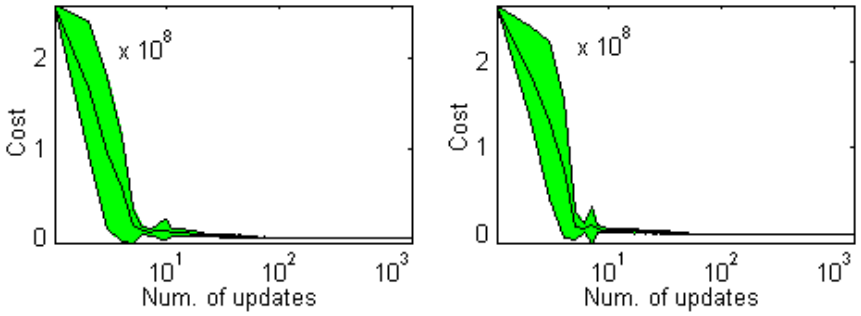


FIGURE 7. Average learning curve for learning left: G3, right: G4.

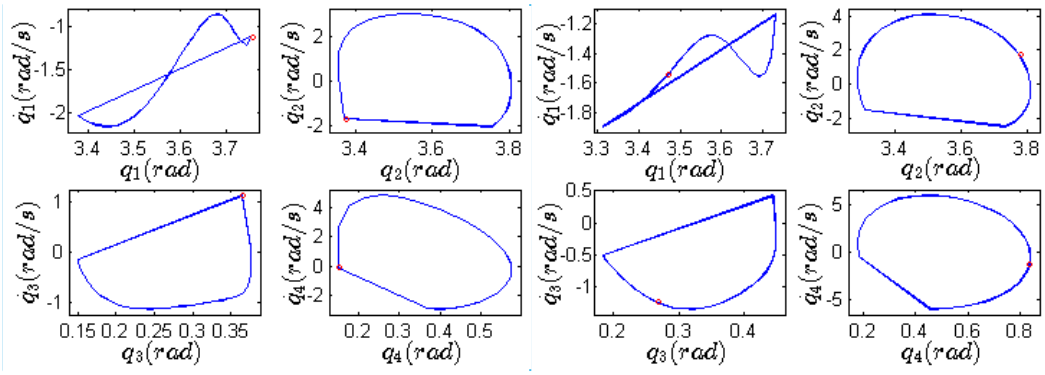


FIGURE 8. The phase portraits for 100 steps of walking by *left*: G3, *right*: G4.

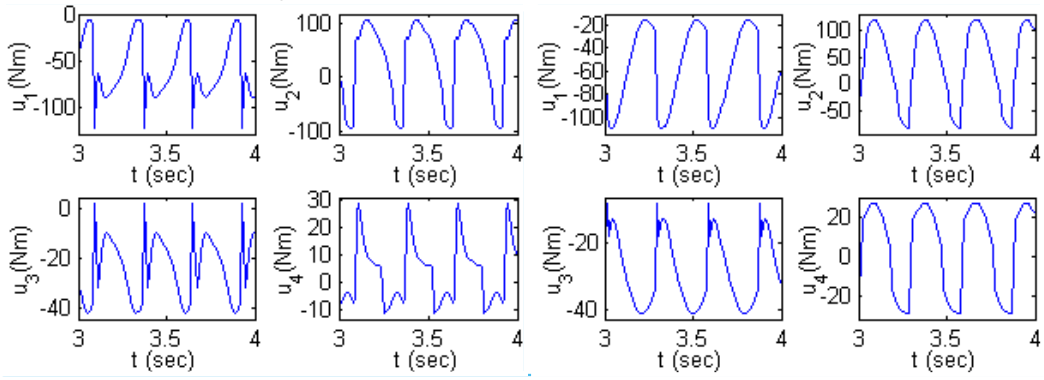


FIGURE 9. Applied torques during walking in the time interval [3s, 4s] by *left*: G3, *right*: G4.

Since Westervelt *et al.* [9] have reported the results of their method only for the flat, we only draw a comparison in the flat scenario. The first row of Table 4 includes the statistics of the designed gait for the flat by Westervelt method [9] in which 7 basis functions have been used for the walking gait, *i.e.* $L = 6$. The aim of optimization was decrement of the actuator torques while some constraints, *e.g.* $v_{\text{des}} = 0.8m/s$, are satisfied. After optimization, the pick torque of actuators was 64 Nm [9][19].

The second row of Table 4 includes the statistics of the designed gait by PI$^2$-WG on the flat scenario. The results show that the gait which is generated by PI$^2$-WG is as good as (or better than) the gait which is generated by Westervelt method. The point is that PI$^2$-WG in all the scenarios starts with the dynamically unstable initial gait (A.1), which is not valid for Westervelt method even in the flat scenario.

In this paper, we used a walking gait with 4 basis function for all the scenarios except the flat scenario. For the flat scenario, the initial gait (A.1) is converted to a gait with 7 basis functions.

The results stated in this paper can be regenerated by the executable program. A lot of other examples can be examined by adjusting the parameters of the algorithm and RABBIT. It can be simply done by editing the text files in the program folder.

## 8. CONCLUSION

PI$^2$ is a RL algorithm with interesting features for optimizing the trajectory parameters in DMP equations. In this paper, PI$^2$ is extended for optimizing the walking gait in the closed-loop dynamics equations of the robot, called PI$^2$-WG. PI$^2$-WG inherits the interesting features from PI$^2$. It has an arbitrary state-dependent cost function part, the exploration noises are the only open algorithmic tuning parameters, it has numerically robust performance in high-dimensional learning problems and it is a simple algorithm for implementation.

Using PI$^2$-WG, the robot is able to learn how to walk with desired speed and exponential stability in a certain environment *e.g.* a certain slope surface with desired friction in the presence of a certain external force and modeling error.

There are at least three motivations for using PI$^2$-WG. First, selecting an initial gait for PI$^2$-WG is easier than the past methods. Second, the selected initial gait is also valid for learning on many other situations contrary to the past methods. Third, the learning can be continued with the real robot to compensate unknown modeling errors. Using the *dynamically unstable* initial gait, the robot cannot walk even a step in the test phase and its walking cost in the learning phase is very large according to the cost function definition. After some iterative updates of the parameters which last only several seconds on an ordinary personal computer, the algorithm

[19]Note that in [28] a better result than [9] is reported. The reason is ignoring viscous and Coulomb joint friction in [28].

achieves a gait with much lower cost, which means arbitrary feasible walking features can be found by designing a proper cost function.

We discover that there exists a tradeoff between satisfaction of the desired features and amount of robustness. The focus of this paper is on satisfaction of the desired features in a new environment. In the future works, we are going to show that how PI$^2$-WG can be extended to design an exponential stable gait which is robust to modeling errors or perturbations, *e.g.* external forces or uneven terrains.

Also, we are going to extend PI$^2$-WG for 3D biped robots with point feet to design a gait with desired features in which the exponential convergence to a periodic orbit is checked by Akbari Hamed method instead of the mentioned HZD method.

## Appendix A. RABBIT robot and its parameters

RABBIT is a point-feet robot with planar motions and 5 degrees of freedom ($n = 5$) [9]. Figure A.1 illustrates the robot and its model parameters. The two legs have the same properties. Table A.1 includes the value of the model parameters for RABBIT. For more information about modeling RABBIT and computation of the ground reaction forces, $F_1^T$, $F_1^N$, $F_2^T$, and $F_2^N$ see [9]. MATLAB code for generating the equations of motion for RABBIT is available at *http://web.eecs.umich.edu/∼grizzle/biped_book_web/*.

RABBIT starts walking with the initial state $x_0 = [q_0; \dot{q}_0]$ in early rollouts of the learning, where $q_0$ and $\dot{q}_0$ are set as follows (the unit of the components of $q_0$ is radian and $\dot{q}_0$ is radian/s):

$$q_0 = [3.2741; 3.4364; 0.1701; 0.6643; -0.012],$$
$$\dot{q}_0 = [-1.5344; 2.0201; -0.2029; 0.0244; 0]$$

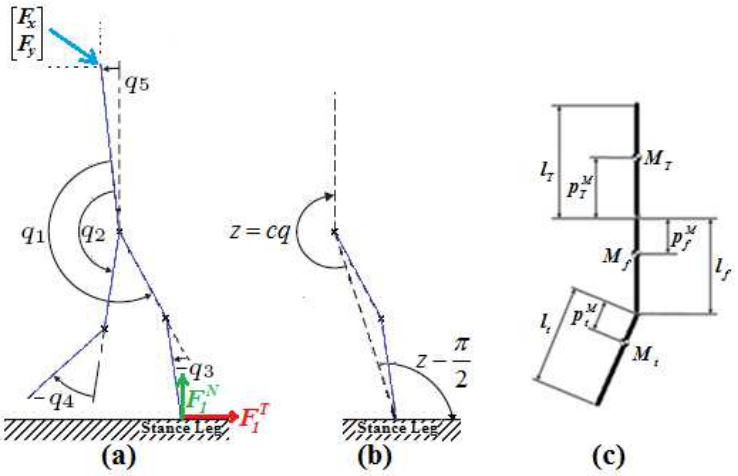


Figure A.1. (a) Generalized coordinates of RABBIT (b) $z$ with $c = [-1, 0, -0.5, 0, -1]$ (c) link length and center of mass position [9].

Table A.1. Parameters for RABBIT which is illustrated in Figure A.1.

| Model parameter | Value |
|---|---|
| Link mass (kg) | $M_T = 12$, $M_f = 6.8$, $M_t = 3.2$ |
| Link inertia (kg$m^2$) | $I_T = 1.33$, $I_f = 0.47$, $I_t = 0.2$ |
| Link length (m) | $l_T = 0.63$, $l_f = 0.4$, $l_t = 0.4$ |
| Mass center (m) | $p_T^M = 0.24$, $p_f^M = 0.11$, $p_t^M = 0.24$ |

TABLE A.2. Errors in model parameters for HZD controller.

| No. | $M_T$ | $I_T$ | $l_T$ | $p_T^M$ | $M_f$ | $I_f$ | $l_f$ | $p_f^M$ | $M_t$ | $I_t$ | $l_t$ | $p_t^M$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ME1 | 0.4 | 0.4 | 0.01 | –0.1 | –0.4 | 0.4 | 0.01 | 0.1 | –0.4 | 0.4 | 0.01 | –0.1 |
| ME2 | 0.4 | –0.4 | 0.01 | –0.1 | 0.4 | –0.4 | 0.01 | 0.1 | –0.4 | –0.4 | 0.01 | –0.1 |
| ME3 | –0.4 | 0.4 | –0.01 | 0.1 | –0.4 | 0.4 | –0.01 | 0.1 | 0.4 | –0.4 | 0.01 | 0.1 |

The row vector $c$ is set to $c = [-1; 0; -0.5; 0; -1]^T$. As it is mentioned, $q = [q_b; q_n]$ where $q_n$ denotes the angle between the torso of the robot and the vertical line and it is a cyclic coordinate. Fixed $z_2 = -3.556$ and $z_1 = -2.9835$ are considered during the learning and test phase. The initial $\Theta$ is defined as (the unit of the numbers is radian):

$$\Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n-1} \end{bmatrix}, \quad \theta_1 = \begin{bmatrix} 3.47 \\ 3.294 \\ 3.399 \\ 2.947 \end{bmatrix}, \quad \theta_2 = \begin{bmatrix} 2.934 \\ 3.606 \\ 3.62 \\ 3.445 \end{bmatrix}, \quad \theta_3 = \begin{bmatrix} 0.1962 \\ 0.1712 \\ 0.153 \\ 0.115 \end{bmatrix} \quad \theta_4 = \begin{bmatrix} 0.1258 \\ 1.147 \\ 0.3627 \\ 0.2013 \end{bmatrix}. \tag{A.1}$$

This initial $\Theta$ is used in all the scenarios except the flat scenario. In the flat scenario, each $\theta_i$ is converted to a vector with 7 components using function approximation.

**Remark A.1.** In this paper, all components of each $\theta_i$ participate in optimization contrary to [9]. The number of Bezier basis functions is 4 in all the scenarios except in the flat scenario which is 7 as in [9].

**Remark A.2.** In the test phase that the torque and friction limitation are considered, even one successful step is impossible by the initial gait (A.1). In the learning phase,

(i) the saturation of $\pm 150$ Nm [9] is deactivated;
(ii) an unrealistic and very large fiction coefficient is considered for the surface;
(iii) walking is not failed with happening $\dot{p}_1^N \leq 0$.

Therefore, the robot is usually able to walk at least one successful step with the initial $\Theta$ in the learning phase of all the learning scenarios. The PD gains of the feedback controller are set as follows:

$$K_P = 500 I_{(n-1)\times(n-1)}, \quad K_D = 50 I_{(n-1)\times(n-1)}.$$

The time step of applying the feedback controller to robot is $\delta t = 0.01$[20]. The joint friction is modeled by the viscous and Coulomb friction terms as in [9] and compensated by the controller:

$$F_{fr}(q, \dot{q}) = F_v \dot{q} + F_s sgn(\dot{q})$$

where $F_v = diag(16.5, 16.5, 5.48, 5.48, 0)$ and $F_s = diag(15, 15, 8.84, 8.84, 0)$. Three modeling errors are mentioned in Table A.2. They are going to be applied to the model parameters in the controller design to specify the effects of the modeling errors on the performance of the proposed algorithm. The numbers in the cells denote amounts of the error in the model parameters, *e.g.*, 0.4 and –0.4 mean a 40% increase and decrease in the nominal parameter value, respectively.

**Remark A.3.** The different modeling errors are considered to show that PI$^2$-WG can compensate a large modeling error.

[20]Note that we use a smaller time resolution at impact. At impact, we use $\delta t = 0.001$ in the learning phase and $\delta t = 0.0001$ in the test phase.

## Appendix B. Derivatives of g(s) and their zeros

The components of the first and second derivatives of $g(s)$ are:

$$\frac{\partial g}{\partial s} = \begin{bmatrix} -L(1-s)^{L-1} \\ L(1-s)^{L-2}(1-Ls) \\ \vdots \\ \frac{L!s^{m-1}(1-s)^{L-m-1}}{m!(L-m)!}(m-Ls) \\ \vdots \\ Ls^{L-2}(L-1-Ls) \\ L\ s^{L-1} \end{bmatrix}$$

$$\frac{\partial^2 g}{\partial s^2} = \begin{bmatrix} L(L-1)(1-s)^{L-2} \\ L(L-1)(1-s)^{L-3}(Ls-2) \\ \frac{L(L-1)}{2}(1-s)^{L-4}\left(L(L-1)s^2-4(L-1)s+2\right) \\ \vdots \\ \frac{L!s^{m-2}(1-s)^{L-m-2}}{m!(L-m)!}\left(L(L-1)s^2-2m(L-1)s+m(m-1)\right) \\ \vdots \\ \frac{L(L-1)}{2}s^{L-4}\left(L(L-1)(1-s)^2-4(L-1)(1-s)+2\right) \\ L(L-1)s^{L-3}\left(L(1-s)-2\right) \\ L(L-1)s^{L-2} \end{bmatrix}$$

we know that the components of $g(s)$ do not have any zero between 0 and 1. All the zeros are placed at 0 and 1 and the total order of the zeros is $L$.

However the $m$th component of $\partial g/\partial s$ have a zero which is positioned at $s = m/L$ ($m \in \{0, \ldots, L\}$) and the other zeros are placed at 0 and 1. The total order of the zeros in each component of $\partial g/\partial s$ is $L-1$.

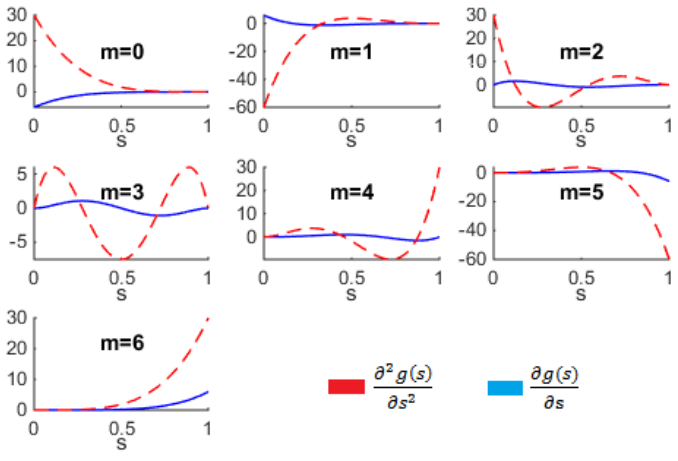


Figure B.1. Curves for components of $\partial g/\partial s$ and $\partial^2 g/\partial s^2$ with $L = 6$.

The $m$th component of $\partial^2 g/\partial s^2$ have two zeros as follows:

$$s = \frac{\left(m \pm \sqrt{\frac{m(L-m)}{(L-1)}}\right)}{L}$$

and the other zeros are placed at 0 and 1. The total order of the zeros in each component of $\partial^2 g/\partial s^2$ is $L-2$.

The components of $g(s)$, $\partial g/\partial s$, and $\partial^2 g/\partial s^2$ are bounded in $s \in [0,1]$. From the coefficients of the components, it is clear that a component of $\partial^2 g/\partial s^2$ has a wider range than its corresponding component of $\partial g/\partial s$. For an example, we plot the components of $\partial g/\partial s$ and $\partial^2 g/\partial s^2$ in Figure B.1 for $L = 6$ which is used in this paper.

## Appendix C. Proof of Theorem 4.3

The equations (4.3) can be reformulated as:

$$\begin{cases} \ddot{q}_b = -D_{11}^{-1} D_{12} \ddot{q}_n - D_{11}^{-1} \Omega_1 + D_{11}^{-1} u \\ \ddot{q}_n = -D_{22}^{-1} D_{21} \ddot{q}_b - D_{22}^{-1} \Omega_2. \end{cases}$$

Substitution $\ddot{q}_n$ in the first row by the second row leads to:

$$\ddot{q}_b = D_{11}^{-1} D_{12} (D_{22}^{-1} D_{21} \ddot{q}_b + D_{22}^{-1} \Omega_2) - D_{11}^{-1} \Omega_1 + D_{11}^{-1} u.$$

Moving all the terms containing $\ddot{q}_b$ to the left results in:

$$(I - D_{11}^{-1} D_{12} D_{22}^{-1} D_{21}) \ddot{q}_b = D_{11}^{-1} D_{12} D_{22}^{-1} \Omega_2 - D_{11}^{-1} \Omega_1 + D_{11}^{-1} u.$$

The above equation can be reformulated as:

$$\ddot{q}_b = (D_{11}\Psi)^{-1} (D_{12} D_{22}^{-1} \Omega_2 - \Omega_1) + (D_{11}\Psi)^{-1} u \tag{C.1}$$

where $\Psi = (I - D_{11}^{-1} D_{12} D_{22}^{-1} D_{21})$. The second derivative of $e = q_b - G(s(z))\Theta$ which is used for computing $L_h L_f e$ and $L_f^2 e$ is:

$$\ddot{e} = \ddot{q}_b - \left[ \frac{\dot{z}^2}{(z_2 - z_1)^2} \frac{\partial^2 G(s)}{\partial s^2} + \frac{\ddot{z}}{z_2 - z_1} \frac{\partial G(s)}{\partial s} \right] \Theta.$$

Providing $\ddot{z} \approx 0$, the term $\ddot{z}/(z_2 - z_1)\partial G(s)/\partial s$ can be ignored according to Remark 4.2, hence we have:

$$\ddot{e} = \ddot{q}_b - \frac{\dot{z}^2}{(z_2 - z_1)^2} \frac{\partial^2 G(s)}{\partial s^2} \Theta.$$

Substituting $\ddot{q}_b$ by (C.1) leads to:

$$\ddot{e} = -\frac{\dot{z}^2}{(z_2 - z_1)^2} \frac{\partial^2 G(s)}{\partial s^2} \Theta + (D_{11}\Psi)^{-1} (D_{12} D_{22}^{-1} \Omega_2 - \Omega_1) + (D_{11}\Psi)^{-1} u.$$

Consequently, $L_f^2 e$ and $L_h L_f e$ expressed as:

$$L_f^2 e = -\frac{\dot{z}^2}{(z_2 - z_1)^2} \frac{\partial^2 G(s)}{\partial s^2} \Theta + (D_{11}\Psi)^{-1} (D_{12} D_{22}^{-1} \Omega_2 - \Omega_1), \quad L_h L_f e = (D_{11}\Psi)^{-1}.$$

Hence $(L_h L_f e)^{-1}$ is $D_{11}\Psi$ and the controller effort $u$ is:

$$u = -(D_{11}\Psi) \left( K_P e + K_D \dot{e} + \frac{\dot{z}^2}{(z_2 - z_1)^2} \frac{\partial^2 G(s)}{\partial s^2} \Theta \right) - D_{12} D_{22}^{-1} \Omega_2 + \Omega_1.$$

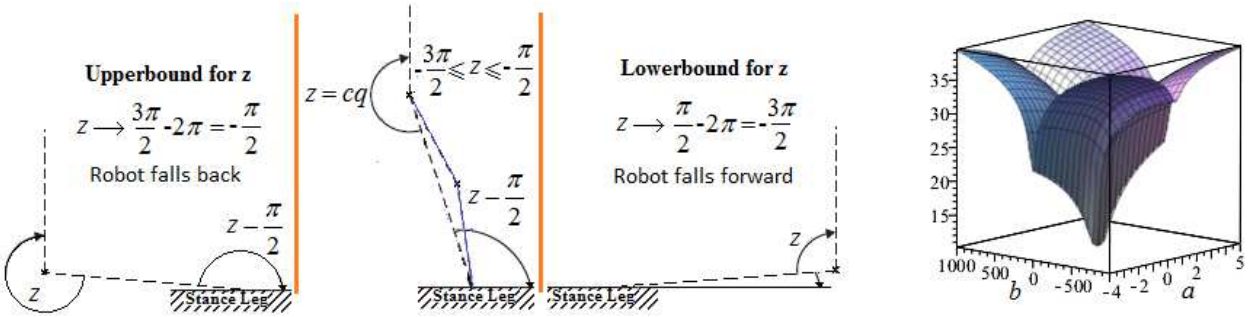


FIGURE D.1. *Left*: the figure illustrates that $-3\pi/2 \leq z \leq -\pi/2$. Note that we need a negative angle for $z$ regarding to the negative components of $c$, hence we subtract the angle from $-2\pi$. *Right:* 3D plot for $LGH(s_z, z_{dot})$ which is drawn by Maple 17. Very large and unrealistic interval $s_z \in [-4, 5]$ and $z_{dot} \in [-1000, 1000]$ are considered to show the logarithmic behavior of the function.

## APPENDIX D. PROOF OF THEOREM 5.1

First we must remember that:

$$\log H_{d,t_j} = \log(\tilde{g}_{t_j}^T R_d^{-1} \tilde{g}_{t_j})$$

where $\tilde{g}(z, \dot{z}) = K_P g(s(z)) + K_D \frac{\dot{z}}{z_2 - z_1} \frac{\partial g(s(z))}{\partial s(z)} + \frac{\dot{z}^2}{(z_2 - z_1)^2} \frac{\partial^2 g(s(z))}{\partial s(z)^2}$ and $R_d = (\sigma_d^2/\lambda) I_{(L+1)\times(L+1)}$. To find the upper-bound and lower-bound value of $\log H_{d,t_j}$, we assume that $s(z)$ and $\dot{z}$ are two independent variables (renamed to $s_z$ and $z_{dot}$) and then calculate $\max(\log H_{d,t_j})$ and $\min(\log H_{d,t_j})$ respectively. In other words, we define the following function:

$$LGH(s_z, z_{dot}) = \log(\vartheta(s_z, z_{dot})^T \vartheta(s_z, z_{dot}))$$

where $\vartheta(s_z, z_{dot}) = K_P g(s_z) + K_D \frac{z_{dot}}{z_2 - z_1} \frac{\partial g(s_z)}{\partial s_z} + \frac{z_{dot}^2}{(z_2 - z_1)^2} \frac{\partial^2 g(s_z)}{\partial s_z^2}$ [21]. Considering $z_1 = -2.9835$, $z_2 = -3.3556$, $s(z) = (z - z_1)/(z_2 - z_1)$ and Figure D.1(*Left*), we have $-3.7966 \leq s(z) \leq 4.6463$. Therefore, we consider a very large and unrealistic interval $s_z \in [-4, 5]$ and $z_{dot} \in [-1000, 1000]$ for $LGH(s_z, z_{dot})$ to compute the upper-bound and lower-bound of $\log H_{d,t_j}$ as:

$$LB + \log\left(\frac{\lambda}{\sigma_d^2}\right) \leq \log H_{d,t_j} \leq UB + \log\left(\frac{\lambda}{\sigma_d^2}\right)$$

where $UB = LGH(-4, -1000) = LGH(5, 1000)$ and $LB = 10.1786$. $LB$ is equal to $\min LGH(s_z, z_{dot})$ which is computed by Maple 17 using the following commands:

$$> \text{with(Optimization)}$$
$$> \text{Minimize}(LGH)$$

and $UB$ is equal to $\max LGH(s_z, z_{dot})$. Figure D.1(*Right*) illustrates the 3D plot of $LGH(s_z, z_{dot})$ which is drawn by Maple 17. The values of $LB$ and $UB$ do not have much difference and they are small numbers (It is the effect of log). The bounds for $\frac{\lambda}{2} \log H_{d,t_j}$ is determined as:

$$\frac{\lambda}{2} LB + \frac{\lambda}{2} \log \lambda - \frac{\lambda}{2} \log \sigma_d^2 \leq \frac{\lambda}{2} \log H_{d,t_j} \leq \frac{\lambda}{2} UB + \frac{\lambda}{2} \log \lambda - \frac{\lambda}{2} \log \sigma_d^2.$$

[21] $\log(\vartheta(s_z, z_{dot})^T R_d^{-1} \vartheta(s_z, z_{dot})) = \log((\lambda/\sigma_d^2)\vartheta(s_z, z_{dot})^T \vartheta(s_z, z_{dot})) = \log(\vartheta(s_z, z_{dot})^T \vartheta(s_z, z_{dot})) + \log(\lambda/\sigma_d^2)$. We do not ignore $\log(\lambda/\sigma_d^2)$ and use it in the remaining part of the proof.

Since according to L'Hôpital's rule $\lim_{\lambda\to 0^+}\lambda\log\lambda=\lim_{\lambda\to 0^+}\frac{\log(\lambda)}{1/\lambda}=\lim_{\lambda\to 0^+}-\frac{1/\lambda}{1/\lambda^2}=0$, with a trivial value for $\lambda$ we have:

$$\frac{\lambda}{2}LB\leq\frac{\lambda}{2}\log H_{d,t_j}\leq\frac{\lambda}{2}UB$$

and the bounds for $A=\sum_{d=1}^{n-1}(\lambda/2\sum_{j=i}^{N-1}\log H_{d,t_j})$, which is defined in Theorem 5.3, can be similarly computed as follows:

$$(n-1)(N-i)\frac{\lambda}{2}LB\leq A\leq(n-1)(N-i)\frac{\lambda}{2}UB.$$

The small values of $LB$ and $UB$ allows the cost function designer to easily ignore $A$ by defining large $Q_t$ and small $\lambda$ (in Fig. 6, we have compared the value of $A$ with the other parts of the cost function that confirms this issue).

## References


[1] Y. Baudoin and M.K. Habib, Using Robots in Hazardous Environments: Landmine Detection, De-Mining and Other Applications. Woodhead Publishing Limited (2011).

[2] P. Agarwal, Pei-Hsin Kuo, R.R. Neptune and A.D. Deshpande, A novel framework for virtual prototyping of rehabilitation exoskeletons. *IEEE International Conference on Rehabilitation Robotics* (2013).

[3] B. Dellon and Y. Matsuoka, Prosthetics, Exoskeletons, and Rehabilitation. *IEEE Robotics and Automation Magazine* (2007).

[4] Wu Guorong and W. Wei, Application of human-machine interaction in toy design. *Information Technology and Artificial Intelligence Conference* (ITAIC) (2011).

[5] M. Cefalo and G. Oriolo, Task-Constrained Motion Planning for Underactuated Robots. *IEEE International Conference on Robotics and Automation (ICRA)*. Washington (2015).

[6] B. d'Andra-Novel and S. Thorel, Control of non holonomic or under-actuated mechanical systems:The examples of the unicycle robot and the slider. *ESAIM: COCV* (2016).

[7] A. Mohammadi, M. Maggiore and L. Consolini, On the Lagrangian structure of reduced dynamics under virtual holonomic constraints. *ESAIM: COCV* (2016).

[8] R. Lecaros and L. Rosier, Control of underwater vehicles in inviscid fluids. *ESAIM: COCV* **20** (2014) 662–703.

[9] E.R. Westervelt, J.W. Grizzle, C. Chevallereau, Jun Ho Choi and B. Morris, Feedback Control of Dynamic Bipedal Robot Locomotion. CRC Press, Taylor and Francis Group (2007).

[10] N. Sugimoto and J. Morimoto, Phase-dependent Trajectory Optimization for CPG-based Biped Walking Using Path Integral Reinforcement Learning. *11th IEEE-RAS International Conference on Humanoid Robots, Bled, Slovenia* (2011).

[11] J. Yu, M. Tan, J. Chen and J. Zhang, A Survey on CPG-Inspired Control Models and System Implementation. *IEEE Transactions on Neural Networks and Learning Systems* **25** (2014) 441–455.

[12] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Yokoi and H. Hirukawa, A realtime pattern generator for biped walking, In *Proc. of the 2002 IEEE International Conference on Robotics and Automation.* Washington, D.C. (2002) 317.

[13] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation, In *Proc. of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems.* Maui, HI (2001) 23946.

[14] M. Vukobratovic, B. Borovac, D. Surla and D. Stokic, Biped Locomotion. Springer-Verlag, Berlin (1990).

[15] M. Vukobratovic and B. Borovac, Zero-moment point–thirty five years of its life. *International Journal of Humanoid Robotics* **1** (2004) 15773.

[16] K. Hirai, M. Hirose, Y. Haikawa and T. Takenake, The development of Honda humanoid robot, In *Proc. of the 1998 IEEE International Conference on Robotics and Automation.* Leuven, Belgium (1998) 132126.

[17] R.D. Gregg, Timothy Bretl and M.W. Spong, Asymptotically Stable Gait Primitives for Planning Dynamic Bipedal Locomotion in Three Dimensions, *2010 IEEE International Conference on Robotics and Automation, Anchorage, Alaska, USA* (2010).

[18] A. Goswami, Postural stability of biped robots and the foot-rotation indicator (FRI) point. *International Journal of Robotics Research* **18** (1999) 52333.

[19] Y. Hurmuzlu, Dynamics of bipedal gait Part 1: objective functions and the contact event of a planar five-link biped. *J. Appl. Mechan.* **60** (1993) 3316.

[20] Y. Hurmuzlu, Dynamics of bipedal gait Part 2: stability analysis of a planar five-link biped. *J. Applied Mechanics* **60** (1993) 33743.

[21] J.W. Grizzle, G. Abba and F. Plestan, Proving asymptotic stability of a walking cycle for a five DOF biped robot model. In *Proc. of the 1999. Int. Conf. on Climbing and Walking Robots* (1999) 69–81.

[22] J.W. Grizzle, G. Abba and F. Plestan, Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. *IEEE Transactions on Automatic Control* **46** (2001) 51–64.

[23] J.W. Grizzle, F. Plestan and G. Abba, Poincares method for systems with impulse effects: Application to mechanical biped locomotion, In *Proc. of the 1999 IEEE International Conference on Decision and Control, Phoenix, AZ* (1999).

[24] E.R. Westervelt, G. Buche and J.W. Grizzle, Experimental validation of a framework for the design of controllers that induce stable walking in planar bipeds. *Int. J. Robotics Res.* **23** (2004) 5592.

[25] E.R. Westervelt, G. Buche and J.W. Grizzle, Inducing dynamically stable walking in an underactuated prototype planar biped, In *Proc. of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, LA* (2004) 42349.

[26] E.R. Westervelt, J.W. Grizzle and C. Canudas, Switching and PI control of walking motions of planar biped walkers. *IEEE Trans. Automatic Control* **48** (2003) 30812.

[27] C. Chevallereau, E.R. Westervelt and J.W. Grizzle, Asymptotically stable running for a five-link, four-actuator, planar bipedal robot. *Int. J. Robotics Res.* **24** (2005) 431–464.

[28] E.R. Westervelt, J.W. Grizzle and D.E. Koditschek, Hybrid zero dynamics of planar biped walkers. *IEEE Trans. Automatic Control* **48** (2003) 42–56.

[29] C. Chevallereau, J. Grizzle and C.-L. Shih, Asymptotically stable walking of a five-link underactuated 3-D bipedal robot. *Robotics, IEEE Trans.* **25** (2009) 37–50.

[30] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E.R. Westervelt, C. Canudas-de-Wit and J.W. Grizzle, RABBIT: A Testbed for Advanced Control Theory, *IEEE Control Systems Magazine*, Paper number CSM-02-038, Revision June 8 (2003).

[31] E.A.Theodorou, J. Buchli and S. Schaal, A Generalized Path Integral Control Approach to Reinforcement Learning. *J. Machine Learning Res.* **11** (2010) 3137–3181.

[32] J. Buchli, F. Stulp, E. Theodorou and S. Schaal, Learning variable impedance control. *Int. J. Robot. Res.* **30** (2011) 820–833.

[33] F. Stulp, E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti and S. Schaal, Learning Motion Primitive Goals for Robust Manipulation. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011).

[34] F. Stulp and O. Sigaud, Policy improvement methods: Between blackbox optimization and episodic reinforcement learning, in Journees Francophones sur la Planification, la Decision et l'Apprentissage pour la conduite de systemes (JFPDA) (2012).

[35] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning (1992).

[36] Peters and S. Schaal, Reinforcement learning of motor skills with policy gradients. *Neural Networks* **21** (2008) 68297.

[37] J. Baxter and P.L. Bartlett, Infinite-horizon policy-gradient estimation. *J. Artificial Intell. Res. Arch.* **15** (2001) 3190–350.

[38] R.S. Sutton, D. McAllester, S. Singh and Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, In Vol. 12 of Advances in Neural Information Processing Systems. MIT Press (2000) 1057–1063.

[39] J. Peters and S. Schaal, Natural actor critic, Neurocomputing (2008b).

[40] J. Koeber and J. Peters, Policy search for motor primitives, In Vol. 21 of Advances in Neural Information Processing Systems. (NIPS 2008). Vancouver, BC, Cambridge, MA: MIT Press (2008) 297–304.

[41] A.J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor and S. Schaal, Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Comput.* **25** (2013) 328–373.

[42] R.F. Stengel, Optimal Control and Estimation, Dover books on advanced mathematics. Dover Publications, New York (1994).

[43] F. Stulp and O. Sigaud, Path Integral Policy Improvement with Covariance Matrix Adaptation, *29 th International Conference on Machine Learning, Edinburgh, Scotland, UK* (2012).

[44] K. Akbari Hamed, B.G. Buss and J.W. Grizzle, Exponentially stabilizing continuous-time controllers for periodic orbits of hybrid systems: Application to bipedal locomotion with ground height variations. To appear in: *Int. J. Robotics Res.* (2015). https://doi.org/10.1177/0278364915593400

[45] K. Akbari Hamed and J.W. Grizzle, Event-Based Stabilization of Periodic Orbits for Underactuated 3-D Bipedal Robots With Left-Right Symmetry. *IEEE Trans. Robotics* **30** (2014) 365–381.