Feedback Error Learning for Rhythmic Motor Primitives

Nakul Gopalan[1], Marc Peter Deisenroth[1], and Jan Peters[1,2]

Abstract— **Rhythmic motor primitives can be used to learn a variety of oscillatory behaviors from demonstrations or reward signals, e.g., hopping, walking, running and ball-bouncing. However, frequently, such rhythmic motor primitives lead to failures unless a stabilizing controller ensures their functionality, e.g., a balance controller for a walking gait. As an ideal oscillatory behavior requires the stabilizing controller only for exceptions, e.g., to prevent failures, we devise an online learning approach that reduces the dependence on the stabilizing controller. Inspired by related approaches in model learning, we employ the stabilizing controller's output as a feedback error learning signal for adapting the gait. We demonstrate the resulting approach in two scenarios: a rhythmic arm's movements and gait adaptation of an underactuated biped.**

I. Introduction

Learning rhythmic behavior is an essential motor skill acquisition problem as a large number of skills ranging from ball paddling [7], drumming [13], walking and running [10], [14] are generically rhythmic. As suggested by Ijspeert [5], parametrized dynamical systems can be used to data-efficiently model and generate trajectories using supervised learning methods. Subsequently, reinforcement learning techniques [7], [12] can be used for improving the behavior with respect to a cost function.

However, many behaviors consist of two components, an oscillatory movement in several parts of the body (e.g., the legs or arms) and a stabilizing controller in the remainder of the body (e.g., the torso and head during locomotion or the hands while carrying a tablet). In such tasks, a supervised learning approach for rhythmic motor primitives may yield low-performance solutions if it does not take the stabilizing controller into account. *Rhythmic Motor Primitives* [5] that minimize the stabilizing control effort are preferable as they allow for weaker motors, less electronics, and lower sampling rates, while keeping desired behavior attainable. Hence, it is essential to learn rhythmic behavior (e.g., a gait) such that the stabilizing controller (e.g., the torso balance controller) is only needed in unforeseen occasions, as the behavior otherwise will be already inherently stable. Nevertheless, most common approaches [5], [7] ignore this additional constraint of learning the stabilizing controller's behavior, despite the fact that important tasks, such as hopping, walking, and running of both robot or animals, all require a balance controller in addition to their rhythmic behavior.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement no. 270327.

[1]Dept. of Computer Science, Technische Universität Darmstadt, Germany.
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany.
{gopalan, marc, peters}@ias.tu-darmstadt.de

Despite that the problem of optimizing rhythmic motor primitives for stabilizing control effort reduction appears to have a reinforcement learning based solution, we can devise a supervised learning approach to this problem. To accomplish the goal of minimizing the control effort caused by the rhythmic motor primitives' output, we update their parameters using the control signal of the stabilizing controller as an error signal. As this control signal corresponds to the feedback error, our novel approach for improving gaits can be seen as a form of *Feedback Error Learning*.

Feedback error learning was introduced by Kawato [6] in the context of learning inverse dynamics models for model-based control of a fixed trajectory, see Fig. 1(a). In contrast, our approach adapts the trajectory generator (i.e., the rhythmic motor primitives) while assuming an existing control law, see Fig. 1(b).

Such feedback error learning of rhythmic motor primitives has substantial advantages over both imitation and reinforcement learning. Unlike classical imitation learning, our method does not require a well-demonstrated behavior where the balance controller already remains silent; instead, our method can be used in conjunction with a (potentially bad) demonstration to initialize the rhythmic motor primitives. Nevertheless, our learning method still results in a supervised learning problem formulation, and is hence a generically easier problem than reinforcement learning. Furthermore, learning can be achieved online and on-policy. As a result, our learning method may even be used to adapt a behavior to a non-stationary environment, e.g., adapt a walking gait to a novel terrain.

We evaluate the core insight of feedback error learning and the resulting architecture in two scenarios: a toy problem with a two- link robot arm, and a planar biped walking robot with a torso. In the first simulation, the robot arm moves the shoulder along a rhythmic trajectory while it is supposed to keep the second link upright. In the second scenario, a planar biped walking robot with a torso has to accomplish a gait while the torso has to remain at a pre-specified angle. Here, the gait is adapted to minimize the balance control law's efforts. Both evaluations demonstrate the applicability of this novel way of learning rhythmic behavior.

II. Feedback Error Learning of Rhythmic Motor Primitives

The goal of this paper is to learn rhythmic or oscillatory motions of a mechanical system such that the control effort of an additional stabilizing control law is minimized. To achieve this goal, we require several components. Firstly, we need a parametrized representation of the movement, which we describe in Section II-A. Secondly, we need to initialize these primitives from a demonstrated trajectory using imitation learning, see Section II-B. Thirdly, we need a control architecture that translates the output of the rhythmic motor primitives into motor commands, i.e., torques, sent to the motors of the robot, discussed in Section II-C. Fourthly, in Section II-D, we determine why and how the feedback motor command can be used as an error signal. Finally, we show how to improve the rhythmic motor primitives by gradient descent feedback error-learning in Section II-E.

A. Representation Behavior with Rhythmic Motor Primitives

Movement primitives [14] are trajectory generators that allow a trajectory to be described as a dynamical system with an attractor landscape. This landscape can be modified with learnable non-linear parameters to fit the desired trajectory. For a rhythmic action, this dynamical system is a limit cycle. The equation for the dynamical system can be given by

$$\tau^2 \ddot{\boldsymbol{q}} = \alpha_z(\beta_z(\boldsymbol{g} - \boldsymbol{q}) - \tau \dot{\boldsymbol{q}}) + \boldsymbol{\Psi}(\boldsymbol{x})\boldsymbol{\theta}, \tag{1}$$

where $\boldsymbol{\Psi}(\boldsymbol{x})$ are the non-linear features dependent on the state $\boldsymbol{x}$ of a canonical system given by, $\tau\dot{\boldsymbol{x}} = \boldsymbol{C}_{\mathrm{c}}$. This canonical system allows the system to be independent of time which allows us to scale the periodic properties of the system more easily [14]. The features used throughout this paper are von Mises basis functions. $\boldsymbol{\theta}$ denotes parameters that need to be learned, i.e., the weights of the basis function. Parameters α_z and β_z are timing constants. Time period of the rhythmic action is denoted by τ. Parameter $\boldsymbol{g}$ is the baseline of the rhythmic trajectory. In the trajectory generation phase, Eq. (1) is used. Assuming the weights $\boldsymbol{\theta}$ are given, the above equations are used to generate the second-order derivatives of the trajectories. The required trajectory is then obtained by numerical integration of these second-order derivatives [14]. In short, using rhythmic motor primitives we can represent rhythmic trajectories with an attractor landscape that can be modulated using non-linear features.

B. Initialize Rhythmic Motor Primitives by Imitation

During the learning phase using Eq. (1) the weights $\boldsymbol{\theta}$ are learned given a demonstrated trajectory consisting of joint angles $\boldsymbol{q}_d$, joint velocities $\dot{\boldsymbol{q}}_d$ and joint accelerations $\ddot{\boldsymbol{q}}_d$. To learn a trajectory we need to learn the weights for the features $\boldsymbol{\Psi}(\boldsymbol{x})$, such that the error between the learned and generated trajectory is minimum. Both the learning and generation of trajectories can be done using the model in Eq. (1) for rhythmic motor primitives. Motor primitive learning by imitation can be achieved by minimizing the squared error

$$E = \frac{1}{2}\sum\nolimits_{k=1}^{N} \left\| \tau^2 \ddot{\boldsymbol{q}}_{\mathrm{d}}^k - (\alpha_z(\beta_z(\boldsymbol{g} - \boldsymbol{q}_{\mathrm{d}}^k) - \tau\dot{\boldsymbol{q}}_{\mathrm{d}}^k)) - \boldsymbol{\Psi}(x^k)\boldsymbol{\theta} \right\|^2 \tag{2}$$

with respect to the parameters $\boldsymbol{\theta}$, where $\ddot{\boldsymbol{q}}_{\mathrm{d}}$, $\dot{\boldsymbol{q}}_{\mathrm{d}}$, $\boldsymbol{q}_{\mathrm{d}}$ are from the demonstrated trajectory. Both gradient descent and single step solutions exist [14]. The time period τ of the rhythmic motion needs to be extracted in advance using methods like Fourier analysis. The period of the basis functions needs to be the same as the one of the gait they are modeling. Choosing a dynamical system to model ensures stable, smooth and differentiable generated trajectories [14]. Due to their ease of modeling and stability properties, we chose rhythmic motor primitives to represent and learn the required trajectories.

C. Control Architecture for Feedback Error Learning

To turn the desired trajectories generated by the motor primitives into actual behaviors, we assume a nonlinear feedforward control architecture [2]. In this architecture,

$$\boldsymbol{u} = \boldsymbol{u}_{\mathrm{ff}} + \boldsymbol{u}_{\mathrm{fb}},$$

where $\boldsymbol{u}$ is the motor command. It consists of a feedforward motor command $\boldsymbol{u}_{\mathrm{ff}}$, which generates the torques needed for the desired behavior, and a feedback control law

$$\boldsymbol{u}_{\mathrm{fb}} = \boldsymbol{K}_{\mathrm{P}}(\boldsymbol{q}_{\mathrm{d}} - \boldsymbol{q}) + \boldsymbol{K}_{\mathrm{D}}(\dot{\boldsymbol{q}}_{\mathrm{d}} - \dot{\boldsymbol{q}}), \tag{3}$$

with positive gains $\boldsymbol{K}_{\mathrm{P}}$ and $\boldsymbol{K}_{\mathrm{D}}$ to ensure that the behavior is achieved in the presence of uncertainty and model errors. By assuming that our robot is a rigid body system, we can use an inverse dynamics model to generate the feedforward motor commands [2]

$$\boldsymbol{u}_{\mathrm{ff}} = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{g}(\boldsymbol{q}). \tag{4}$$

Here, $\boldsymbol{q}, \dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ denote positions, velocities and acceleration, respectively, $\mathbf{M}(\boldsymbol{q})$ is the inertia matrix, $\mathbf{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ denotes centripetal and Coriolis forces, and $\mathbf{g}(\boldsymbol{q})$, the gravity forces.

D. The Feedback-Error as Learning Signal

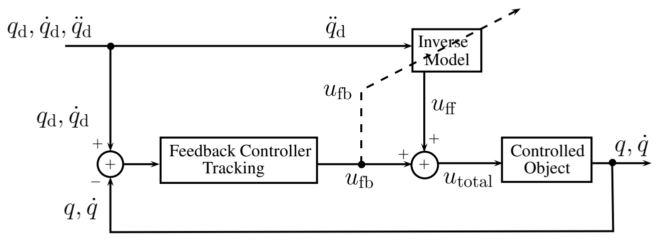
Classically, the idea of feedback error learning [6] has been derived from an inverse dynamics learning perspective as illustrated in Fig. 1(a). In this case, the model from Eq. (4) is assumed to be unknown and instead learned using a parametric representation $\boldsymbol{u}_{\mathrm{ff}} = \boldsymbol{\Phi}(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}})\boldsymbol{\theta}$, with basis functions $\boldsymbol{\Phi}(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}})$ defined on desired input trajectories and weights $\boldsymbol{\theta}$ that describe the inverse model based on the basis functions. Similar to motor primitive learning from demonstrations as described in Section II-B, inverse dynamics model learning is achieved by minimizing the sum of squared errors

$$E = \frac{1}{2}\sum\nolimits_{k=1}^{N} \left\| \boldsymbol{e}^k \right\|^2 = \frac{1}{2}\sum\nolimits_{k=1}^{N} \left\| \boldsymbol{u}^k - \boldsymbol{f}\left(\boldsymbol{q}_{\mathrm{d}}^k, \dot{\boldsymbol{q}}_{\mathrm{d}}^k, \ddot{\boldsymbol{q}}_{\mathrm{d}}^k, \boldsymbol{\theta}\right) \right\|^2 \tag{5}$$

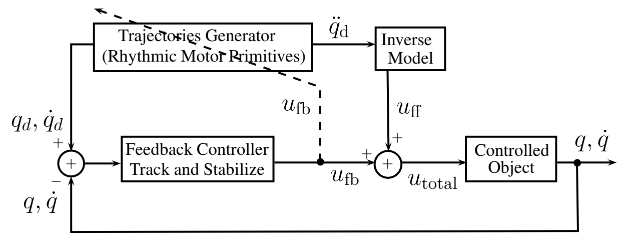
with respect to the parameters $\boldsymbol{\theta}$. If the inverse dynamics model is precise, the feedforward $\boldsymbol{u}_{\mathrm{ff}}$ will be error free, and hence dominant. This results in $\ddot{\boldsymbol{q}}_{\mathrm{d}} \approx \ddot{\boldsymbol{q}}$, forcing the controller $\boldsymbol{u}_{\mathrm{fb}}$ to remain dormant, i.e., $|\boldsymbol{u}_{\mathrm{fb}}| \approx 0$. Otherwise, the feedback controller will generate torques to make sure that the output of the system follows the control laws and gives desired outputs. Kawato [6] realized that the feedback of a linear control law can be used as an error signal

$$\boldsymbol{e} = \boldsymbol{u} - \boldsymbol{\Phi}(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}})\boldsymbol{\theta} = \boldsymbol{u} - \boldsymbol{u}_{\mathrm{ff}} = \boldsymbol{u}_{\mathrm{fb}} \tag{6}$$

for inverse dynamics learning where $\boldsymbol{f}(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}}, \boldsymbol{\theta}) = \boldsymbol{\Phi}(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}})\boldsymbol{\theta}$. Hence, in inverse dynamics learning, it is easy to see that the feedback signal serves as an error.


$q_\mathrm{d}, \dot{q}_\mathrm{d}, \ddot{q}_\mathrm{d}$
$\ddot{q}_\mathrm{d}$
Inverse Model
$q_\mathrm{d}, \dot{q}_\mathrm{d}$
u_fb
u_ff
Feedback Controller Tracking
Controlled Object
$q, \dot{q}$
u_total

(a) Original feedback error learning.



(b) Modified feedback error learning.

Fig. 1. (a) Original feedback error learning model where an inverse model is being learned using the feedback error and consists only of a tracking feedback, from [6]. (b) Modified feedback error learning of gaits. The feedback from the the balance controller's feedback output is used for learning the gait and not the inverse model, and also the feedback used for learning is not just a tracking feedback but more importantly a stabilizing feedback as well.

In the context of optimizing rhythmic motor primitives by feedback-error learning, the relationship is more contrived as

$$\boldsymbol{f}\left(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ddot{\boldsymbol{q}}_\mathrm{d}, \boldsymbol{\theta}\right) = \mathbf{M}(\boldsymbol{q}_\mathrm{d})\ddot{\boldsymbol{q}}_\mathrm{d} + \mathbf{c}(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}) + \mathbf{g}(\boldsymbol{q}_\mathrm{d}). \quad (7)$$

As illustrated in Fig. 1(b), here, we want to learn the motor primitive generating the desired trajectory. There exist two kinds of feedback controllers: The feedback controllers that track the rhythmic trajectory given by

$$\boldsymbol{u}_{\text{track},i} = \boldsymbol{u}_{\text{fb},i} = \boldsymbol{K}_{\mathrm{P},ii}\left(\boldsymbol{q}_{\mathrm{d},i} - \boldsymbol{q}_i\right) + \boldsymbol{K}_{\mathrm{D},ii}\left(\dot{\boldsymbol{q}}_{\mathrm{d},i} - \dot{\boldsymbol{q}}_i\right),$$

and, the feedback controllers that act as stabilizing controllers such as

$$\boldsymbol{u}_{\text{stabilize},j} = \boldsymbol{u}_{\text{fb},j} = \boldsymbol{K}_{\mathrm{P},jj}\left(\boldsymbol{q}_{\mathrm{d},j} - \boldsymbol{q}_j\right) + \boldsymbol{K}_{\mathrm{D},jj}\left(-\dot{\boldsymbol{q}}_j\right),$$

where $\boldsymbol{u}_{\text{track},i}$ and $\boldsymbol{u}_{\text{stabilize},j}$ represent the tracking feedback for the i^th and the stabilizing feedback for the j^th links in accordance with Eq. (3). The matrices $\boldsymbol{K}_\mathrm{P}$ and $\boldsymbol{K}_\mathrm{D}$ are the positive feedback gain matrices. It can be seen that for the stabilizing feedback the desired joint angle for stability is $\boldsymbol{q}_{\mathrm{d},j}$ and the desired joint velocity for stability is 0, such that the joint remains in the stable position throughout the rhythmic trajectory.

For learning the rhythmic motor primitive, in a fully actuated system we realize that $\boldsymbol{e}_j = \boldsymbol{u}_{\text{stabilize},j}$ can be used as an error signal for all stabilizing controllers and $\boldsymbol{e}_i = 0$ for all other dimensions. However, in underactuated systems the tracking and stabilizing feedbacks are summed to be given into a single actuator. Thus, to learn one input trajectory for underactuated systems we would need to minimize the sum of the stabilizing and tracking feedbacks. In our simulations, the robot arm is a fully actuated system and the planar biped is an underactuated system. In the following paragraphs, we detail how the basic idea behind feedback error learning can be transferred to learning of rhythmic motor primitives by gradient descent.

E. Gradient Descent Feedback Error Learning of Rhythmic Primitives

For optimizing the cost function in Eq. (5), a common approach minimizes the squared error with respect to the parameters $\boldsymbol{\theta}$ based on N measurements. The solution can be obtained by the least squares solution in a single step. However, if we want to continuously update the model, the parameters of this model can be optimized by stochastic gradient descent for an error function by

$$\begin{aligned}\boldsymbol{\theta}_{h+1} &= \boldsymbol{\theta}_h - \alpha_h \nabla_{\boldsymbol{\theta}} E \qquad (8)\\ &= \boldsymbol{\theta}_h + \alpha_h \sum\nolimits_{k=1}^{N} \boldsymbol{e}^k \frac{\partial \boldsymbol{f}(\boldsymbol{q}_\mathrm{d}^k, \dot{\boldsymbol{q}}_\mathrm{d}^k, \ddot{\boldsymbol{q}}_\mathrm{d}^k, \boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_h},\end{aligned}$$

where α_h is a learning rate or step size. Stochastic gradient descent for a convex error is guaranteed to converge to the optimal solution under mild conditions such as $\sum_{h=1}^{\infty} \alpha_h \to \infty$ and $\sum_{h=1}^{\infty} \alpha_h^2 < \infty$, see [1].

For inverse dynamics learning, such a gradient update yields the update step

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha_h \sum\nolimits_{k=1}^{N} \boldsymbol{u}_\mathrm{fb}^k \boldsymbol{\Phi}\left(\boldsymbol{q}_\mathrm{d}^k, \dot{\boldsymbol{q}}_\mathrm{d}^k, \ddot{\boldsymbol{q}}_\mathrm{d}^k\right)^T, \quad (9)$$

which is known as feedback error learning [6]. Learning and control using feedback error learning for inverse dynamics is stable under the conditions defined for the gains $\boldsymbol{K}_\mathrm{P}$ and $\boldsymbol{K}_\mathrm{D}$ as $\left\|\boldsymbol{K}_\mathrm{D}^2\right\| > \|\boldsymbol{K}_\mathrm{P}\|$ [11]. Nevertheless, these gains have to be chosen on trial and error basis initially.

Following the method of feedback error learning, the feedback error is used to correct the weights of the rhythmic motor primitives using stochastic gradient descent comparable to Eq. (9) given by

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha_h \sum\nolimits_{k=1}^{N} \boldsymbol{u}_\mathrm{fb}^k \frac{\partial \boldsymbol{f}(\boldsymbol{q}_\mathrm{d}^k, \dot{\boldsymbol{q}}_\mathrm{d}^k, \ddot{\boldsymbol{q}}_\mathrm{d}^k)}{\partial \boldsymbol{\theta}_h},$$

where the forward torque $\boldsymbol{u}_\mathrm{ff} = \boldsymbol{f}(\boldsymbol{q}_\mathrm{d}^k, \dot{\boldsymbol{q}}_\mathrm{d}^k, \ddot{\boldsymbol{q}}_\mathrm{d}^k)$and $\boldsymbol{\theta}$ are the weights of the rhythmic motor primitives. This leads by the application of the chain rule to the weight update rule

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha_h \sum\nolimits_{k=1}^{N} \boldsymbol{u}_\mathrm{fb}^k \frac{\partial \boldsymbol{f}(\boldsymbol{q}_\mathrm{d}^k, \dot{\boldsymbol{q}}_\mathrm{d}^k, \ddot{\boldsymbol{q}}_\mathrm{d}^k)}{\partial \ddot{\boldsymbol{q}}_\mathrm{d}^k} \frac{\partial \ddot{\boldsymbol{q}}_\mathrm{d}^k}{\partial \boldsymbol{\theta}_h}. \quad (10)$$

The derivative $\partial \boldsymbol{f} / \partial \ddot{\boldsymbol{q}}_\mathrm{d}^k$ of the forward torque are calculated analytically or using finite differences. We used finite difference methods to calculate the derivatives $\partial \ddot{\boldsymbol{q}}_\mathrm{d}^k / \partial \boldsymbol{\theta}_h$, because the rhythmic motor primitives in Eq. (1) have a temporal component that does not allow us to calculate precise analytical derivatives of $\partial \ddot{\boldsymbol{q}}_\mathrm{d}^k / \partial \boldsymbol{\theta}_h$ as needed in Eq. (10). The conditions for stability of both, the stochastic gradient descent, and that of the feedback error learning are the same as before. The error minimization condition in Eq. (10) changes the forward gait $\boldsymbol{q}_\mathrm{d}$ in the direction of minimizing the summed feedback of the control laws. The major modifications to the feedback error learning model is that we learn trajectories instead of inverse dynamical models as in the original paper [6], and stabilizing feedback error is more important for learning than the tracking feedback error. This causes a slightly different update equation to the weight vectors of the trajectories. In the next section we analyze two simulation results using our novel approach to feedback error learning using Eq. (10).

III. Results

In this section, we describe two simulations used to test our feedback error learning model. In the first simulation stable joint trajectory for a two-link robot arm's upper link is learned. In the second simulation a biped torso's balanced trajectory is learned, while executing the gait.

A. Evaluation on a Two-link Robot Arm

The two-link robot arm served as a toy example for feedback error learning for rhythmic motor primitives. This two-link robot arm can also be regarded as a double link pendulum. The arm had two links and two actuators as shown in Fig. 2. Each link of the arm weighed $1\,\mathrm{kg}$ and measured $1\,\mathrm{m}$ in length. The lower link of the arm oscillated between $(\pi/2 \pm \pi/6)\,\mathrm{rad}$, while inverted, i.e., with its mass is above its pivot. The upper link initially had a random, unknown feedforward trajectory that is unbalanced. The random and unstable nature of this initial feedforward trajectory could lead to the robot arm getting out of balance.

The period of oscillation for the lower link was $1\,\mathrm{s}$. The forward trajectory for the lower link was generated using rhythmic motor primitives [14] with 10 basis functions. The initial forward trajectory for the upper link was chosen by randomly sampling the weights for the rhythmic motor primitives. These rhythmic motor primitives also had 10 basis functions. The dynamics of the robot arm were taken from Yoshikawa [15], and the equations had the same form as Eq.(4). The sampling frequency for the simulator was $100\,\mathrm{Hz}$.

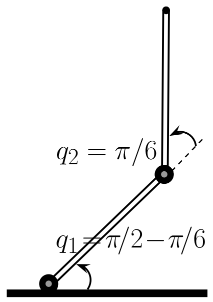
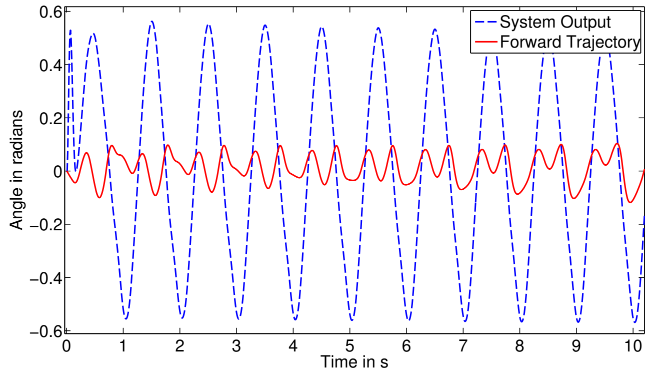


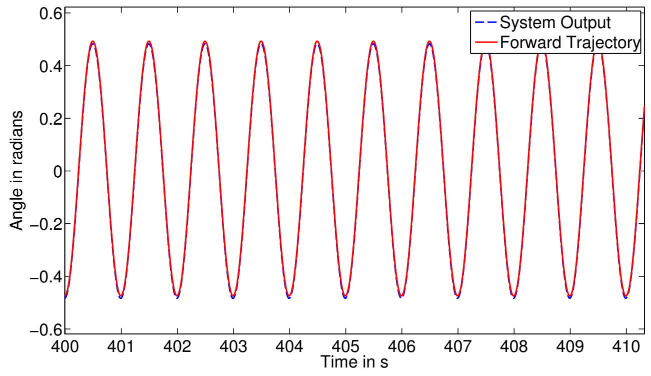
Fig. 2. Model for the two-link robot arm in the balanced state

The control signal for each link was the sum of the torques generated by the inverse model $\boldsymbol{f}$ from Eq. (7) and the feedback torque generated by the PD controller. The feedback torque for the lower link measured the PD error between the generated trajectory and the joint angles of the lower link. This lower link's feedback torque was the tracking feedback. The feedback torque for the upper link measured the PD error of the upper link of the robot arm from the vertical line using joint angles. This upper link's feedback was the stabilizing feedback torque $\boldsymbol{u}_{\text{stabilize}}$ that needed to be minimized by feedback error learning. The angle $\boldsymbol{q}_1$ was the system output joint angle for the lower link measured with respect to the horizontal axis and $\boldsymbol{q}_2$ was the system output joint angle for the upper link measured with respect to the lower link as seen in Fig. 2. The forward trajectory for the links are referred to as $\boldsymbol{q}_{1\mathrm{d}}$ and $\boldsymbol{q}_{2\mathrm{d}}$, respectively. For a balanced upper link the upper link's forward and system output angle needed to converge to $\pi/2$ radians with respect to the horizontal and its derivative to 0 radians per second, i.e., $q_{1\mathrm{d}}^k + q_{2\mathrm{d}}^k = \pi/2$ and $\dot{q}_{2\mathrm{d}}^k = 0$. We used the control laws as described in Eq. (3). Parameters $K_{\mathrm{P}1}$, $K_{\mathrm{D}1}$, $K_{\mathrm{P}2}$ and $K_{\mathrm{D}2}$ are PD gains for the both the links with $|K_{D2}^2| > |K_{P2}|$ for stable control and learning [11].

The upper link's initial trajectory was random and unbalanced. However, the control law followed the condition that

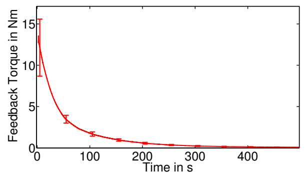


(a) Initial trajectories for upper link.

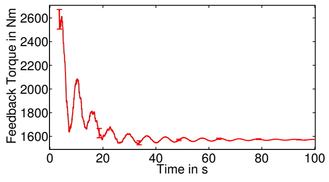


(b) Final forward trajectory for upper link.

Fig. 3. Two-link robot arm's simulation results (a) Initial trajectories of the upper link. The dashed line is the system output and the continuous line is the forward trajectory. (b) Final trajectories of the upper link. The two trajectories completely overlap showing that the forward gait was completely learned using feedbacks to converge to the control law's criterion.



(a) Two-link Arm feedback (Nm).



(b) Biped Feedback (Nm).

Fig. 4. (a) Moving average of the absolute feedback for the two-link robot arm averaged over 20 different initializations. As learning progressed, the feedback reduced, and became nil in the case of the two-link robot arm. (b) Moving average of the sum of feedbacks for the biped robot averaged over 20 different initializations. Here, the feedback reduced and reached a minimum when the learning stopped. The nonzero value of the feedback is because of the underactuation of the biped model.

the upper link of the robot arm should always be vertical. Therefore, initially the *stabilizing feedback* ensured that the system was never out of balance. These feedbacks trained the weight vectors of the *feedforward* rhythmic motor primitives using gradient descent, minimizing the absolute balancing feedback torque as per Eq. (10).

Using feedback error learning, the upper link's feedforward gait $\boldsymbol{q}_{2\mathrm{d}}$ was learned in a direction such that the stabilizing feedback was reduced. For this fully actuated system, only the stability feedback, i.e., the upper link's feedback, was minimized, which was sufficient for learning a stable gait.

As learning proceeded, the learned feedforward gait produced outputs that satisfied the control laws and kept the feedback at zero, when passed through the inverse model and the system. The convergence to a balanced gait using feedback of the balance controller is shown in Fig. 3. As a result, the balancing feedback torque reduced drastically and became nil after about 400 seconds. The two-link robot arm's average absolute feedback torque values over 20 random initializations are shown in Fig. 4(a).

As a step towards a biped simulator, we evaluated our approach on a two-link robot arm with a weight distribution and frequency closer to that of a walking human, with the lower link acting as legs and the upper link acting as torso. We set the upper link's weight to $40\,\mathrm{kg}$ and the lower link's weight to $15\,\mathrm{kg}$, the length of both to $1\,\mathrm{m}$ and the period to $1\,\mathrm{s}$. All these models converged well within 500 cycles of learning, and the feedback was reduced to nil for both links after this period.

B. Biped Model's simulation

In the following , we consider learning the torso's gait for a biped model that has the additional challenges of underactuation a moving base. Previously, there have been methods to learn biped gaits from demonstration [10], [9] or using reinforcement learning [8], [3] using rhythmic motor primitives. A biped's torso's gait is learned using feedback error learning without the use of complicated value functions or giving supervised inputs from demonstrations. The goal of this simulation was to minimize the feedback required by the torso to remain in balance when its initial trajectory with respect to the legs is random. As in the previous section a gait to follow was given. In particular the gait of the lower limbs, we want to learn the gait of the torso with respect to the legs according to a control law.

The biped robot model consisted of three links: two legs, a torso, and two actuators. The actuators were between the two leg-to-torso joints. There were three joint angles to be considered and the angles were measured in the uniform coordinate system from the vertical as shown in Fig. 5. The weight and size of the biped were chosen similar to that of humans. Thus, the torso, each of the legs and the hip weighed $40\,\mathrm{kg}$, $15\,\mathrm{kg}$ and $10\,\mathrm{kg}$ respectively. Each link was $1\,\mathrm{m}$ long. Since this simulation needed a biped with a torso, the biped dynamics model was chosen from Grizzle et al. [4]. This model had a torso unlike other simple biped models or compass walkers. The 3 links of the biped gave it 5 degrees of freedom, one for each link and two for the coordinates of the fixed support, i.e., the stance leg. The sampling rate of the simulator was $200\,\mathrm{Hz}$.

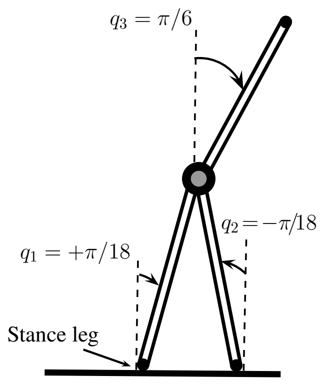
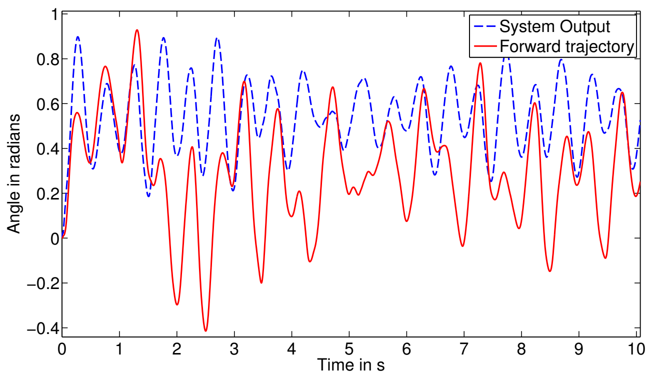


Fig. 5. Model for the biped robot in the balanced state.

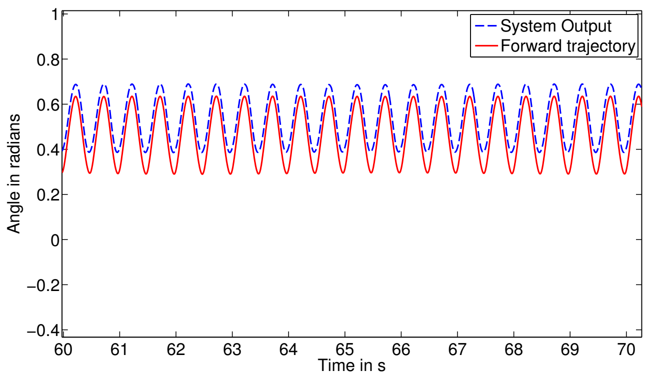
The objective of the simulation was to modify the gait of the torso of the biped while it continued to walk, such that the balance controller's feedback was minimized. The biped's dynamics equations were of the same form as Eq. (4). The three joint angle trajectories were modeled using rhythmic motor primitives. We used 16 basis functions per cycle for each of the rhythmic movement primitives. The walking rate was assumed to be two steps a second, i.e., a complete walk cycle in $1\,\mathrm{s}$. A walk cycle consists of pivoting the whole body on the stance leg while pushing the swing leg forward, and further switching the stance and the swing legs and repeating the previous action. The stage of switching the stance and swing legs is called the impact stage. The gait for this simulation was a dynamic gait, where the impact stage was instantaneous, i.e., the switching of the stance and swing legs was instantaneous.

The torques during the stance and the swing phase for a leg were completely different. The dynamical model of the biped used was defined in terms of the stance leg as a fixed support [4] and the torso and the swing leg being in motion attached to this fixed support. Thus, at each impact stage there needed to be a transformation of coordinates between the old stance leg to the new stance leg. In our simulation no special force model was assumed for the impact stage. There was a simple switch in trajectories and forces between the old stance leg and the new stance leg. Hence, if the before impact state was $X^- = [q_1, q_2, q_3, \dot{q}_1, \dot{q}_2, \dot{q}_3]$, the after impact state was $X^+ = [q_2, q_1, q_3, \dot{q}_2, \dot{q}_1, \dot{q}_3]$, as during the impact stage coordinates between the two legs were swapped. Along with these angles, their corresponding force equations were switched. This switching of forces was valid because the before and after impact joint velocities of the limbs were zero. The switching was only a change of coordinates for the corresponding limbs switching between swing and stance legs. For a detailed proof we refer [4].

The biped had one stabilizing feedback for the torso's link and two tracking feedbacks for the limbs. Minimizing only the stabilizing feedback, i.e., the torso's balance feedback, would have led to increasing the other two tracking feedbacks. As, there were two actuators and three feedbacks, the feedbacks were coupled and then provided to their corresponding actuations. Hence, decreasing stabilizing feedback alone led to an increase in the other two tracking feedbacks. This increase in other feedbacks could have led to a wrong gait followed or a wrong gait learned. Hence,



(a) Initial forward trajectories for the torso.



(b) Final forward trajectory for the torso.

Fig. 6. Underactuated biped. The dashed line is the system output, the continuous line is the forward trajectory. (a) Initial forward trajectories for the torso. (b) Final forward trajectory for the torso: Due to the underactuation, convergence was not perfect. The periodic pattern of the trajectories was learned perfectly, but the baseline of the trajectory differed by a small margin.

the cost function chosen was the sum of squares of the three feedbacks using Eq. (5) and it was minimized with respect to the weights of the rhythmic motor primitives for the torso's joint using the learning rule in Eq. (10). In this simulation, we forced the torso to be at $\pi/6$ radians to the vertical axis while starting from an unknown random trajectory. The lower limbs maintained a periodic sinusoidal gait between $\pm\pi/18$. The feedbacks hence were based on this stabilizing criterion for the torso's link and regular trajectory tracking condition for the other two links using the PD control in Eq. (3).

Fig. 6 shows that the torso's forward gait was learned from an initial unknown gait, to a final converged and balanced gait. The learned gait has the same periodic structure, but differs in the baseline by a small margin. As the summed feedback could never be reduced to zero, the upper torso did not perfectly converge to the required limit of $+\pi/6$, which can be seen in Fig. 6. The convergence in the two-link robot arm's case was perfect to the desired gait as the link to be stabilized was independently actuated. This was not the case for the biped's torso. Hence, a perfect output according to the control law is difficult to achieve in the biped, but our method ensured that all the tasks are fulfilled at best without any unstability in the system.

The plot of average absolute summed feedback torque over 20 runs is given in Fig. 4(b). The feedback was reduced considerably from the start time as the gait was learned. It can be seen that there was a considerable amount of learning in the first 40 s as the feedback reduced the most here and then reached a minimum. After learning, the sum of the absolute feedback torques was almost halved from the initial value, and the standard deviation is almost zero.

IV. Conclusion

We have presented an approach to learn gaits online based on the feedback error learning architecture. Feedback error learning of rhythmic motor primitives allows for lower feedback gains compared to stabilizing controllers without such a learning mechanism, while increasing the safety of the robots. Our approach uses linear control laws without the need of any supervised examples or value functions. We evaluated our approach on stabilizing a two-link arm and an underactuated biped robot model. Our results indicate that for the fully actuated two-link arm, the stabilizing feedback can be used as a learning signal to learn trajectories. However, in the case of the underactuated biped, the sum of all the feedback errors affecting the stabilization actuator must be used as the learning signal to learn forward trajectories. In both cases, the learned trajectories minimized the feedback.

Our method is an improvement compared to the other learning methods as it does not need supervised inputs or value functions, and provides a much simpler solution to the learning problem. However, our method can replace traditional methods only in scenarios where control laws can be formulated for the trajectories to be learned.

References

[1] L. Bottou. Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*. Cambridge University Press, 1998.
[2] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., 1989.
[3] M. P. Deisenroth, R. Calandra, A. Seyfarth, and J. Peters. Toward fast policy search for learning legged locomotion. In *IROS*, 2012.
[4] 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(1), 2001.
[5] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *NIPS*, 2003.
[6] M. Kawato. Feedback-error-learning neural network for supervised motor learning. *Advanced Neural Computers*, 1990.
[7] J. Kober and J. Peters. Learning motor primitives for robotics. In *ICRA*, 2009.
[8] Jun Morimoto, Garth Zeglin, and Christopher G. Atkeson. Minimax differential dynamic programming: An application to robust biped-walking. In *NIPS*, 2002.
[9] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. A framework for learning biped locomotion with dynamic movement primitives. In *Humanoids*, 2004.
[10] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. In *Robotics and Autonomous Systems*, 2004.
[11] J. Nakanishi and S. Schaal. Feedback error learning and nonlinar adapative control. In *Neural Networks*, 2004.
[12] J. Peters and S. Schaal. Policy gradient methods for robotics. In *IROS*, 2006.
[13] D. Pongas, A. Billard, and S. Schaal. Rapid synchronization and accurate phase-locking of rhythmic motor primitives. In *IROS*, 2005.
[14] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *ISRR*, 2004.
[15] T. Yoshikawa. *Foundations of Robotics*. MIT Press, 1990.