Optimizing Robot Striking Movement Primitives with Iterative Learning Control

Okan Koç[1], Guilherme Maeda[2], Gerhard Neumann[2], Jan Peters[1,2]
{okan.koc, jan.peters}@tuebingen.mpg.de

***Abstract*— Highly dynamic tasks that require large accelerations and precise tracking usually rely on precise models and/or high gain feedback. While movement primitives allow for efficient representation of such tasks from demonstrations, the optimization of the required motor commands for systems with inaccurate dynamic models remains an open problem. To achieve accurate tracking for such tasks, we investigate two related Iterative Learning Control update laws and present a variant suited for optimizing hitting movement primitives. The resulting algorithm generalizes well to different initial conditions and naturally addresses striking movements where reaching specific velocities at certain positions is crucial. We evaluate the performance of our approach in a simulated putting task as well as in robotic table tennis, where we show how the striking performance of a seven degree of freedom anthropomorphic arm can be optimized. Our final implemented algorithm compares favorably with two state-of-the-art approaches.**

I. INTRODUCTION

Most reaching tasks in control and robotics can be phrased as *tracking* problems, where the dynamical system needs to follow a certain predefined trajectory in order to reach a goal state. Robotic table tennis in particular [22] consists of planning, generating and executing a series of such (episodic) single stroke trajectories. In order to reach the hitting state precisely, these high-speed trajectories need to be followed with appropriate motor commands. Computing the right motor commands is a nontrivial task when using cable-driven arms such as the Barrett WAM arm shown in Figure 1.

There have been many attempts in the reinforcement learning (RL) [32] and control literature to learn robotic tasks directly. Value function based methods take advantage of duality to solve the Bellman's equation but suffer from the initial bias or representation in estimating the value function and do not scale well to high dimensions. Policy search based RL methods (e.g., [17], [26], [34], [7]) directly solve the Bellman's equation in a parameterized policy space and can be more effective in practice.

Dynamic Movement Primitives (DMP) are a policy representation that leverages the dynamical systems approach to modify the spring dynamics with a forcing term that enables it to mimic a smooth trajectory. They include an internal phase or clock that ensures the convergence of the movement primitive to a goal state or a limit cycle [15]. DMPs do not suffer from the curse of dimensionality as the number of parameters (weights) grow linearly with dimension. They can be easily modulated in time and space to allow for variance in motion. However, approximation and control errors in robotic platforms make the application of DMPs less useful in practice.

Fig. 1: Robotic table tennis setup with the ball-launcher throwing balls to the robot. In order to hit the ball to a desired position on the opponent's court, we give the robot reference trajectories that facilitate the right striking motion. We can teach the robot such trajectories via kinesthetic teach-in.

Motor primitives can be modulated in different ways to adapt to unforeseen events or to ensure the optimal execution of required tasks. They work particularly well with episodic policy search methods that modify the weights of the forcing term based on the rewards received in every episode. By adapting the DMP that was initialized with imitation learning e.g., with kinesthetic teach-in [22], RL approaches are able to achieve complex robotic tasks, such as ball-in-a-cup [17]. However model-free methods (e.g., [17], [26]) require many iterations to converge, whereas more data-efficient approaches such as [7] suffer from computational runtime difficulties and cannot be implemented in real-time robotics tasks.

Inspired by the successes (and failures) of these previous approaches, the research question that we tackle in this article consists of the following:

- How can we learn to hit optimally, while efficiently exploiting an inaccurate dynamics model in a stable way?
- More specifically, when we have modelling inaccuracies and different initial conditions, how should we learn to track a DMP $s(t)$ such that the robot executes a desired hitting motion, either in table tennis or a similar reaching task e.g., putting in golf?

[1]Max Planck Institute for Intelligent Systems, Spemannstr. 38, 72076 Tuebingen, Germany
[2]Technische Universitaet Darmstadt, FG Intelligente Autonome Systeme Hochschulstr. 10, 64289 Darmstadt, Germany

We believe that learning in robotics tasks can be performed much more efficiently by taking advantage of existing imperfect models and demonstrated reference trajectories.

Iterative Learning Control (ILC) is a fundamental approach in control theory developed to track (time-varying) reference trajectories. It has been used successfully to follow trajectories under unknown repeating disturbances and model mismatch [5]. In ILC, usually the feedforward control inputs are adjusted after each trial based on the resulting deviations from the reference trajectory. The goal is to drive such deviations to zero. ILC can easily incorporate available dynamics models (e.g. [1], [2]).

In this article, we combine Iterative Learning Control with movement primitives by using ILC as a learning method to track rhythmic DMPs. This way we ensure a safe, reliant and robust way to track reference trajectories and more importantly, to execute hitting and striking motions optimally. We show that the incorporation of DMPs help to make our approach robust to initialization error, i.e. our algorithm generalizes ILC to different initial conditions. As opposed to model-free policy search approaches, we make full use of the nominal rigid body dynamics and inverse dynamics models, which helps us to quickly achieve the desired performance requirements. We validate the performance of the approach in two hitting tasks, and compare with existing methods.

Our contributions can be summarized as follows: we form a link between the ILC literature and the movement primitives by systematically formulating an iterative update for tracking the DMPs. A new and concise formulation of striking primitives with rhythmic DMPs is presented. We form a coherent learning framework and present a new update law for reaching the goal positions and velocities. These are studied in detail and the implemented *goal-based ILC*, or *gILC* in short, is shown to perform better than the state of the art ILC approaches.

In Section I-A we mention related work, especially the state of the art approaches in hitting and reaching tasks. In Section I-B we state the problem and introduce our new update law in Section II, relating it to the existing optimization-based ILC approaches. We formulate this update law, called *gILC*, in algorithmic form in Section II-D and show how it can be used with rhythmic DMPs. In Section III we evaluate *gILC* in robotic table tennis. We show that the method outperforms the state-of-the-art ILC approaches. Finally in Section IV we discuss the strengths and weaknesses of our method and conclude with brief mentions of promising future research directions.

A. Related Work

Dynamic movement primitives belong to the class of movement primitives [9]. Movement primitives are a kinematics-based approach to keep the learning tasks in high-dimensional tasks, such as in humanoid robotics, tractable. Like the options framework in Markov Decision Processes [31], they aim at reducing the curse of dimensionality in complex human-like robotics tasks. The first dynamical systems based formulation of movement primitives appeared in [14]. A good review with alternative and variant formulations is given in [15]. Discrete DMPs have been extended in [16], [22] for striking movements, where the goal state is also evolved in time to allow for high velocities around settling time.

The work of Arimoto et al. [3] was one of the first to define Iterative Learning Control with the D-type update law. See [5] and [1] for reviews. Theoretically, most ILC algorithms can be studied as a *linear repetitive process* using 2D-systems analysis [27]. Monotonic convergence and stability guarantees are of central importance for the practical usefulness of ILC algorithms. They are shown for example in [5], [23], [20]. In practice, some of the assumptions made in the ILC literature may often be violated. Robustness to varying initial conditions are considered e.g., in [13], [25], [8]. ILC should be used along with a robust feedback controller to reject nonrepeating disturbances, see e.g., [6],[20].

One of the first papers introducing an optimization based ILC approach incorporating a model of the dynamics is [2]. These methods are closely related to stable plant-inversion approaches [11]. As a more recent example of model-based ILC, Schoellig et al. [29] applied a Kalman-filter based convex optimization rule that avoids direct inversion and showed its performance in quadrocopter flight. An EM-based update law was given in [35] where an impressive application of ILC to a robotic surgical task was presented. ILC has also been combined with robust observers for controlling a heavy-duty hydraulic arm during excavation tasks [21].

While the large body of ILC literature has almost exclusively addressed invariant reference trajectories, we introduce the use of DMPs as a parameterized trajectory representation, allowing for variations on the desired trajectory and adding more flexibility to controllers based on ILC. DMPs have been also combined in a bimanual robotics task with ILC [10] where force feedback is used to enable compliant interaction with objects in an unknown environment. ILC is here used to learn a coupling term between the two arm trajectories. In our case we do not change the DMPs but try to track rhythmic DMPs that can succesfully return the ball to the opponent's court.

B. Problem Statement

The goal in trajectory tracking is to track a given reference $\mathbf{r}(t),\ 0 \leq t \leq T$, in state space $\mathbf{y} \in S \subset \mathbb{R}^{2n}$ by applying the control inputs $\mathbf{u}(t)$. The reference trajectory in our case enables the *optimal* execution of hitting and striking motions, e.g., forehand and backhand strikes in (table) tennis. We assume that such optimal trajectories are available to us, either through kinesthetic teach-in or given a desired Cartesian trajectory, computed recursively using inverse kinematics.

Consider the nonlinear robot dynamics of the form

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}), \\ \ddot{\mathbf{q}} &= \mathbf{M}^{-1}(\mathbf{q})\{\boldsymbol{\tau}(\mathbf{u}) - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})\} + \boldsymbol{\epsilon}(\mathbf{q}, \dot{\mathbf{q}}), \end{aligned} \tag{1}$$

where on the right hand side are the terms due to the rigid body dynamics model and $\boldsymbol{\epsilon}(\mathbf{q}, \dot{\mathbf{q}})$ are the (unmodeled) dis-

turbances that act on the robot, due to parameter mismatch, viscous friction, stiction, etc. This system can be linearized around a given joint space trajectory $\mathbf{r}(t),\ 0 \leq t \leq T$ with nominal inputs $\mathbf{u}_{\text{IDM}}(t)$ calculated using the inverse dynamics model [30]. We then obtain the following linear time varying (LTV) representation

$$\dot{\mathbf{e}} = \mathbf{A}(t)\mathbf{e}(t) + \mathbf{B}(t)\tilde{\mathbf{u}}(t) + \mathbf{d}(t, \mathbf{u}), \tag{2}$$

where the state error is denoted as $\mathbf{e}(t) = \mathbf{y}(t) - \mathbf{r}(t)$, the joint angles and velocities are $\mathbf{y} = [\mathbf{q}^\intercal, \dot{\mathbf{q}}^\intercal]^{\text{T}}$, $\tilde{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_{\text{IDM}}(t)$ and the continuous time varying matrices are

$$\begin{aligned} \mathbf{A}(t) &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{(\mathbf{r}(t), \mathbf{u}_{\text{IDM}}(t))}, \\ \mathbf{B}(t) &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{(\mathbf{r}(t), \mathbf{u}_{\text{IDM}}(t))}. \end{aligned} \tag{3}$$

In the error dynamics (2) the additional term $\mathbf{d}(t, \mathbf{u})$ accounts for the disturbances and the effects of the linearization. We can discretize (2-3) with step size δ, $N = T/\delta$ and step index $j = 1, \ldots, N$ to get the following discrete linear system

$$\mathbf{e}_{j+1} = \mathbf{A}_j \mathbf{e}_j + \mathbf{B}_j \tilde{\mathbf{u}}_j + \mathbf{d}_j(\mathbf{u}_1, \ldots, \mathbf{u}_j), \tag{4}$$

where the matrices $\mathbf{A}_j, \mathbf{B}_j$ are the discretizations of (3). Conventional ILC algorithms learn to compensate for the errors by iterating the control inputs $\tilde{\mathbf{u}}_j$ with an update law.

II. Iterative Learning Control with Movement Primitives

In a highly dynamic and complex task such as robot table tennis, one often needs to consider an extension of the standard trajectory tracking task. Based on the varying initial positions and velocities of the robot arm and the trajectory of the incoming ball, in each table tennis stroke the robot arm needs to track different trajectories that start from different initial conditions and end with different goal states of the arm. Moreover these trajectories need to be scaled in time to intercept the ball. Dynamic Movement Primitives (DMP) are especially useful for representing such a variety of movement patterns.

Based on these considerations, in this paper we focus on learning to track DMPs $\mathbf{s}(t) = [\mathbf{q}_{\text{des}}(t), \dot{\mathbf{q}}_{\text{des}}(t)]^{\text{T}}$. An initial DMP might be constructed out of a given demonstration or an optimal reference trajectory $\mathbf{r}(t)$ using regression techniques [15]. Representing a reference trajectory with movement primitives has some benefits: nonsmooth parts of the trajectory can be filtered, the evolution of desired states can be coupled with errors to ensure safety, time and scaling invariance of the differential equations can be used to adapt the trajectory to task changes in time as well as in space.

A. Movement Primitive Formulation

Striking movement primitives suited to table tennis have been proposed in [16] and [22] as an extension of discrete DMPs. Unlike the original formulation [14], these extensions allow for an arbitrary velocity profile to be attached to the primitives around hitting time. However, these *ad hoc* approaches are unnecessarily cluttered and involve additional tuning parameters. Arguably they are also against the philosophy behind the discrete primitive: the motion converges necessarily to a goal state with zero velocity. This can not only lead to potentially unsafe states at the completion of the primitive execution, but also requires the addition design of a returning trajectory.

We propose here instead to use rhythmic DMPs that allow for a limit cycle attractor, which is inevitable if we want to maintain the striking motion through goal state. After the striking is completed the DMP can be used to return back to initial state or it can be terminated by setting the forcing terms to zero. An example is shown in Figure 2.

The rhythmic movement primitive differential equations [17] can be rewritten as

$$\begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\tau^2 \alpha_g \beta_g & -\tau \alpha_g \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \tau^2 \begin{bmatrix} 0 & 0 \\ \alpha_g \beta_g & f(\phi) \end{bmatrix} \begin{bmatrix} g \\ A \end{bmatrix}, \tag{5}$$

where the phase ϕ evolves as $\dot{\phi} = \tau$. The constant τ determines the period of the limit cycle and the forcing term f enforces the limit cycle with M weighted Von-Mises basis functions Ψ_i

$$\begin{aligned} f(\phi) &= \frac{\sum_{i=1}^{M} \Psi_i w_i}{\sum_{i=1}^{M} \Psi_i}, \\ \Psi_i &= \exp(h_i(\cos(\phi - c_i) - 1)). \end{aligned}$$

Here A, h_i, c_i determine the amplitude of the motion, the width and the centers of the basis functions respectively.

The dynamical system (5) describes the motion of each of the desired joint states along a particular periodic path in joint space. The forcing term shapes this path by warping

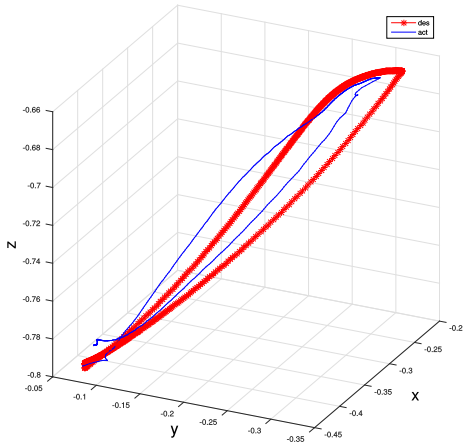


Fig. 2: An example of a striking movement primitive is shown in red. The rhythmic DMP creates a limiting cycle in the workspace of the robot. Executing this movement well will lead to a good hit. Control errors in tracking lead to a poor hitting performance, shown in blue. The tracking errors can be decreased efficiently by applying model-based iterative learning updates.

the motion of the spring dynamics. The weights w are obtained with regression using demonstration data. The spring constants α_g and β_g ensure that starting from any initial position and velocity s_0 the DMP converges to the limit cycle with center g and are usually chosen such that the dynamical system is critically damped.

Unlike the discrete DMP, the rhythmic DMP is a linear time-varying system and adaptions of target positions and velocities can be easily performed. Using linear systems theory one can show the feasibility of such adaptations and construct phase-dependent modifications of the free parameters g, A.

B. Derivation of ILC Updates

Most ILC update laws can be put in the following form

$$\mathbf{u}_{k+1} = \mathbf{\Gamma}(\mathbf{u}_k - \mathbf{L}\mathbf{e}_k). \tag{6}$$

Model based ILC can be cast in this form by stacking the model matrices in (4) together to get the following lifted-vector representation [5], [29]

$$\mathbf{e}_L = \mathbf{F}\mathbf{u}_L + \mathbf{d}_L, \tag{7}$$

where the submatrices of $\mathbf{F}$ are

$$\mathbf{F}_{(i,j)} = \begin{cases} \mathbf{A}_{i-1}\dots\mathbf{A}_j\mathbf{B}_{j-1}, & j < i, \\ \mathbf{B}_{j-1}, & j = i, \\ \mathbf{0}, & j > i. \end{cases} \tag{8}$$

Using this *input-to-output matrix* $\mathbf{F}$ we can analyze the effects of the feedforward inputs $\mathbf{u}_L = \mathrm{vec}(\tilde{\mathbf{u}})$ on the errors $\mathbf{e}_L = \mathrm{vec}(\mathbf{e})$ and compensate for the disturbances $\mathbf{d}_L$ with ILC.

1) Lagrange form: The quadratic cost functional as the optimality criterion

$$J(\tilde{\mathbf{u}}) = \int_0^T \mathbf{e}^{\mathrm{T}}\mathbf{Q}\mathbf{e} + \tilde{\mathbf{u}}^{\mathrm{T}}\mathbf{R}\tilde{\mathbf{u}}\ \mathrm{d}t + \mathbf{e}_T^{\mathrm{T}}\mathbf{Q}_T\mathbf{e}_T,$$

can be equally discretized and stacked in lifted vector form

$$J_L = \mathbf{e}_L^{\mathrm{T}}\mathbf{Q}_L\mathbf{e}_L + \mathbf{u}_L^{\mathrm{T}}\mathbf{R}_L\mathbf{u}_L,$$

where the symmetric positive definite matrix $\mathbf{Q}_L \in \mathbb{R}^{2Nn\times 2Nn}$ ($\mathbf{R}_L$ is defined analogously) is of the following form

$$\mathbf{Q}_L = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Q}_N \end{bmatrix}. \tag{9}$$

Using Newton's method we can optimize iteratively for $\mathbf{u}_L$

$$\begin{aligned} \mathbf{u}_{k+1} &= \mathbf{u}_k - \left(\frac{\partial^2 J_L}{\partial \mathbf{u}_L^2}\right)^{-1} \left.\frac{\partial J_L}{\partial \mathbf{u}_L}\right|_{\mathbf{u}_k}, \\ \frac{1}{2}\frac{\partial^2 J_L}{\partial \mathbf{u}_L^2} &= \frac{\partial}{\partial \mathbf{u}_L}\{\mathbf{F}^{\mathrm{T}}\mathbf{Q}_L\mathbf{e}_L + \mathbf{R}_L\mathbf{u}_L\} = \mathbf{F}^{\mathrm{T}}\mathbf{Q}_L\mathbf{F} + \mathbf{R}_L, \\ \mathbf{u}_{k+1} &= \mathbf{u}_k - (\mathbf{F}^{\mathrm{T}}\mathbf{Q}_L\mathbf{F} + \mathbf{R}_L)^{-1}(\mathbf{F}^{\mathrm{T}}\mathbf{Q}_L\mathbf{e}_k + \mathbf{R}_L\mathbf{u}_k). \end{aligned} \tag{10}$$

The update (10) will be referred to as the Newton-based update, or *N-ILC*. Organizing (10) and comparing to (6) we see that the filtering matrix $\mathbf{\Gamma}$ and the learning matrix $\mathbf{L}$ can be written as:

$$\begin{aligned} \mathbf{\Gamma} &= (\mathbf{F}^{\mathrm{T}}\mathbf{Q}_L\mathbf{F} + \mathbf{R}_L)^{-1}(\mathbf{F}^{\mathrm{T}}\mathbf{Q}_L\mathbf{F}), \\ \mathbf{L} &= (\mathbf{F}^{\mathrm{T}}\mathbf{Q}_L\mathbf{F})^{-1}(\mathbf{F}^{\mathrm{T}}\mathbf{Q}_L). \end{aligned}$$

These matrices modulate the dependency between the inputs $\mathbf{u}_k$ and the errors $\mathbf{e}_k$. The ILC update law (10) is noncausal (i.e. predictive) since $\mathbf{L}$ is not block lower-diagonal. See [2], [12] for the case where input derivative penalties are also considered. Note the connection of (10) to plant inversion methods [11]: taking $\mathbf{Q}_L = \mathbf{I}, \mathbf{R}_L = \mathbf{0}$, (10) becomes

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \mathbf{F}^{\dagger}\mathbf{e}_k. \tag{11}$$

A more intuitive way to understand ILC is to consider (11) as least squares regression on the disturbances estimated using the previous trial [29]

$$\begin{aligned} \mathbf{F}\mathbf{u}_{k+1} &\approx -\mathbf{d}_k, \\ \mathbf{u}_{k+1} &= \mathbf{F}^{\dagger}(\mathbf{F}\mathbf{u}_k - \mathbf{e}_k), \end{aligned}$$

which is equivalent to (11) if $\mathbf{F}$ is of full column rank. We perform least-squares regression by taking advantage of the linearized model in (4) to form the right correlations between the errors and the feedforward compensations. Compared with the *credit-assignment* issues of RL algorithms, we see that this brand of ILC, equipped with our linearized models for prediction, offers us a more principled way to assign errors to the control inputs.

2) Mayer form: In some applications, the trajectory is only an intermediary and does not need to be precisely tracked. Hitting primitives and strokes in table tennis are examples of such a scenario, where the task performance depends only on reaching the desired position and velocity at the right time. In optimal control literature optimization criteria that consider only the final state cost are said to be in Mayer form [19], as opposed to the usual Lagrange form that considers the intermediate steps as well.

Taking the quadratic final cost, we can relate it to the feedforward corrections $\mathbf{u}_L$ using our linearized model (7)

$$\begin{aligned} J(\mathbf{u}_L) &= \mathbf{e}_N^{\mathrm{T}}\mathbf{Q}_N\mathbf{e}_N, \\ &= (\mathbf{F}_N\mathbf{u}_L + \mathbf{d}_N)^{\mathrm{T}}\mathbf{Q}_N(\mathbf{F}_N\mathbf{u}_L + \mathbf{d}_N). \end{aligned} \tag{12}$$

where $\mathbf{F}_N$ is the block row entry of $\mathbf{F}$ in (8) corresponding to the hitting time $t = N$

$$\mathbf{F}_N = \begin{bmatrix} \mathbf{A}_{N-1}\dots\mathbf{A}_2\mathbf{B}_1 & \cdots & \mathbf{B}_N \end{bmatrix}.$$

Using Newton's method on (12) we reach the reweighted form of (11)

$$\mathbf{u}_{k+1} = \mathbf{u}_k - (\mathbf{F}^{\mathrm{T}}\mathbf{M}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{M}\mathbf{e}_k, \tag{13}$$

here $\mathbf{M}$ is defined as the matrix (9) where all the diagonal entries $\mathbf{Q}_t$ are set to zero except for the last block entry $\mathbf{Q}_N$ corresponding to hitting time.

C. Iterative Learning Control for Movement Primitives

Execution errors in tracking the rhythmic DMP prevents us from initializing the robot at each iteration to the same state. Starting from different initial conditions $\mathbf{y}_0 = [\mathbf{q}_0^\intercal, \dot{\mathbf{q}}_0^\intercal]^{\mathrm{T}}$ we can consider a movement primitive as in (5) to give a smooth and feasible interpolating trajectory that is similar to the demonstrations. For such movement primitives we consider the Mayer form ILC update (13) to be the natural candidate since in each trial the DMPs generate different trajectories. Trying to track these varying trajectories with an ILC update law as in (10) can make learning unstable.

To compensate for the varying initial conditions we recompute the reference control input $\mathbf{u}_{\mathrm{IDM}}$ based on the nominal inverse dynamics model. Results that prove increased robustness with similar compensation methods are given in [25],[8]. With this correction the total feedforward control commands $\mathbf{u}_{\mathrm{ff}}$ at iteration $k+1$ are computed as follows

$$\begin{aligned} \mathbf{u}_{\mathrm{ff}} &= \mathbf{u}_{\mathrm{IDM}} + \mathbf{u}_{k+1}, \\ \mathbf{u}_{k+1} &= \mathbf{u}_k - (\mathbf{F}^{\mathrm{T}}\mathbf{M}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{M}\mathbf{e}_k. \end{aligned} \tag{14}$$

Equation (14) is referred to as the goal based ILC or *gILC*.

D. Algorithm & Implementation

We use the update law derived in (14) in our algorithm, given in Algorithm 1. After the strike is completed, the rhythmic DMP can be further evolved to bring it close to $\mathbf{y}_0$. We consider only the errors on this striking part. This update law enables us to take advantage of the superlinear order of convergence property of Newton's method based descent methods while ensuring robustness on the variations in initial conditions $\boldsymbol{\delta}_k$.

Depending on the task and models available, different weighting matrices $\mathbf{M}$ and $\mathbf{R}$ can be incorporated as desired. $\mathbf{M}$ can be designed to consider not just the hitting state, but a hitting segment of the rhythmic DMP. Iteration dependent $\mathbf{R}_k$ will ensure that the learning performance does not degrade with additional regularization, and one can reduce the weighting as the errors get smaller. The practitioner can additionally ensure safety by applying a learning rate β. For simplicity we only show the learning rate in Algorithm 1.

Notice that methods based on differential dynamic programming, such as iLQR, show a stronger dependency on model accuracy since they update both the reference trajectory $\mathbf{r}(t)$ and the nominal inputs $\mathbf{u}_{\mathrm{IDM}}$, often relying on re-computation in short horizons and on efficient simulators [33] to account for model inaccuracies.

Under high mismatch cases where the input-to-output matrix $\mathbf{F}$ differs from the actual dynamics of the system substantially[1], the robust convergence criteria [5] will not be satisfied and exploration in joint space will be necessary to improve the model and adapt $\mathbf{F}$. We leave the extension of ILC to such adaptive settings for future work.

[1]Dynamics mismatch can be quantified with the maximum singular value of the difference.

Algorithm 1 *gILC*

Input: $\epsilon > 0$, $\mathbf{Q}_N, \mathbf{M} \succeq 0$, $0 < \beta \leq 1$, $\mathbf{s}$
Form $\mathbf{F}$ using $\mathbf{A}$, $\mathbf{B}$ matrices
Initialize $k = 1$, $\mathbf{u}_{\mathrm{ILC}} = \mathbf{0}$
repeat
  Rollout $\mathbf{s}$ from $\mathbf{y}_0 + \boldsymbol{\delta}_k$
  Compute $\mathbf{u}_{\mathrm{IDM}}$ with inverse dynamics
  Strike with controls $\mathbf{u}_{\mathrm{ff}} = \mathbf{u}_{\mathrm{IDM}} + \mathbf{u}_{\mathrm{ILC}}$
  Observe $\mathbf{e}_k = \mathbf{y}_k - \mathbf{s}$ from $\ddot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u})$
  Compute $J_k = \mathbf{e}_N^{\mathrm{T}}\mathbf{Q}_N\mathbf{e}_N$
  Update $\mathbf{u}_{\mathrm{ILC}} \leftarrow \mathbf{u}_{\mathrm{ILC}} - \beta(\mathbf{F}^{\mathrm{T}}\mathbf{M}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{M}\mathbf{e}_k$
  Follow $\mathbf{s}$ to $\mathbf{y}_0$
until $J_k < \epsilon$

III. Experiments

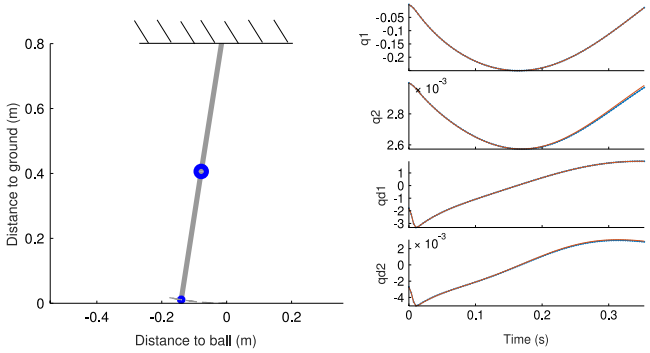
In this section, we demonstrate the effectiveness of the ILC algorithms presented in Section II for striking motions. In ILC one normally assumes that the reference trajectories are *feasible*, i.e. that it is possible to follow them arbitrarily well. Even in cases where this assumption does not hold, we can improve the hitting performance with the Mayer form ILC since only the hitting state, but not deviations during the tracking of a trajectory, are punished.

We consider two hitting tasks: putting in golf and table tennis strokes. In each task the trajectories and the extracted movement primitives are assigned in joint space, one for each joint of the robot. A low-gain feedback law is calculated in the second case using LQR with the linearized dynamics (4) which stabilizes the open-loop system $\mathbf{f}$. A conventional PD controller with high gains on the shoulder was also tested on the robotic table tennis setup and was found to be unstable in the iteration domain, see [4] for the more general case. The robustness that comes with the LQR feedback law and the DMP framework is an asset that strengthens the applicability of our approach outside of the tested platforms. At the cost of larger initial error, we suggest increasing the input penalties on the LQR to ensure stability of ILC in robotics applications.

A. Verification and Comparisons in Simulated Putting

Putting is a simple and natural domain for testing ILC algorithms, in particular we consider it as a scalable interface to more complex hitting tasks with high DoF robots. We illustrate in Figure 3 a simplified simulation of a two-link planar arm with two revolute joints hitting a golf ball on the ground. We can imagine that a golf stick is attached to the end-effector at the second joint. We assume that we have available a circular reference trajectory that intercepts the ball with a desired velocity of 1.8 m/s. This lets us account for the masses of the arm and the ball as well as a simulated kinetic friction of $\mu = 0.6$ between the golf ball and the ground. Good putting trajectories can be shown easily by expert humans or recorded via kinesthetic teach-in.

In order to simulate learning with ILC, we assume that there is a model mismatch due to the motor inertia of both



(a) Initial attempt (b) Final trajectory

Fig. 3: Robot arm must follow the assigned reference trajectory precisely in order to hit the ball with a desired velocity at the desired time. The reference trajectory in Cartesian space is shown as a blue dashed curve. We can see in (a) that the initial attempt falls short of the reference trajectory. ILC then modifies the control inputs to compensate for the modeling errors. In the last attempt shown in (b) the reference trajectory is executed almost perfectly. The ball will then approach the hole with approximately zero velocity.

joints. We give the motor inertia a Gaussian distributed disturbance with zero mean and a variance of 0.05, while the actual values of the motor inertia are 0.15 and 0.12 respectively. With these random disturbances we can construct error bars for different ILC algorithms and also test for their robustness.

The first and last iterations of ILC are also shown in Figure 3. Learning takes place in the joint space of the robot, since the matrix $\mathbf{F}$ is constructed using the nominal dynamics model (1) in joint space. The full dynamics $\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u})$ involves the nonlinear effects from both links. We can see how initially the inertial disturbances of the motors prevent the end-effector from following the trajectory precisely. The iterations show how ILC compensates for such an effect and in the last iteration the ball is given the desired velocity to reach the hole.

We compare the approach with two other ILC approaches by plotting the root-mean-squared (RMS) errors of each algorithm in Figure 4. Kalman-filter based convex optimization approach [29] using no regularization is labeled as *KF-ILC* and the finely-tuned *Arimoto-style* [3] model-free ILC using a basic PD-type update is labeled as *PD-ILC*. These are compared with the *N-ILC* in (10) where the control inputs are not penalized, that is $\mathbf{R} = \mathbf{0}$. The error bars indicate one standard deviation within 10 trials with different inertial mismatches. Notice that the other two approaches take much longer to converge.

The results of these experiments motivated us to try *gILC*, an extension of *N-ILC* for tracking movement primitives starting from different initial conditions, a case commonly encountered in highly dynamic games such as table tennis.

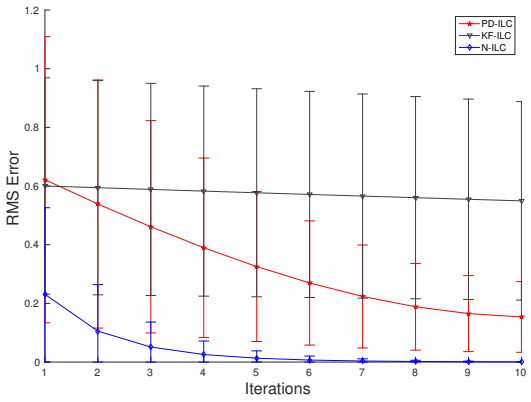


Fig. 4: Convergence of model based ILC to the reference putting trajectory is shown in blue. Convergence is quadratic for the simulated scenario where we have unknown inertial disturbances acting on the motors. Note the slower performance of the other two ILC approaches.

B. Application in Table Tennis

As a second and more complex task, we consider table tennis where we are interested in generating and executing accurate striking motions. For the robotic table tennis task we are using a seven degree of freedom (DoF) torque-controlled custom made Barrett WAM arm capable of high speeds and accelerations. A standard size racket (16 cm diameter) is mounted on the end-effector of the arm as can be seen in Figure 1. A vision system consisting of four cameras hanging from the ceiling around each corner of the table is used for tracking the ball [18]. The orange ball is tracked visually with a sampling rate of 60 Hz and filtered with an extended Kalman filter that accounts for some of the bouncing behavior of the ball and air drag effects. The table and the tennis balls are in accordance with the International Table Tennis Federation (ITTF) rules.

A ball launcher (see Figure 1) is available to throw balls accurately to a fixed position in Cartesian space to the forehand of the robot. The incoming ball arrives with low-variability in desired positions and higher-variability in ball velocities. The whole area to be covered amounts to about 1 m^2 circular region surrounding the initial forehand posture of the robot. This allows us to avoid the singularities of the robot. Any ball that appears outside of this circular *feasible* region will not be hit.

After the visual system predicts a ball trajectory that coincides with the feasible region in Cartesian space, the motion planning system has to come up with a trajectory that specifies how, where and when to intercept the incoming ball. Desired Cartesian position, velocity and orientations of the racket translate in joint space to a specification of 14 parameters: 7 joint angles and 7 joint velocities of the robot arm. Along with the desired hitting time (or the time until impact), these 15 parameters are used to train 7 joint space DMPs that correspond to the desired reference trajectory in Cartesian space. These movement primitives

are synchronized with the same phase and extracted from kinesthetic teach-in data.

Demonstrations from kinesthetic teach-in allow us to generate successful hitting motions, i.e. those that are guaranteed to return the ball to the opponent's court if executed well. In runtime, in order to generate feasible reference trajectories that account for the variations in incoming ball position and velocities, we adapt the goal states of the extracted movement primitives based on the predicted ball trajectory. We start these DMPs from the initial posture of the robot provided by the sensors.

Using successful demonstrations we trained a probabilistic model $p(\mathbf{w}|\mathbf{y}_b)$ given ball positions $\mathbf{y}_b$ using weighted regression. Weighting can be put in various ways and we opted for a weighting which put a higher reward on the fast strikes that landed on the edges. The generated DMPs of the model are guaranteed to be safe because these movements lie within the convex combination of demonstrations.

In order to show the applicability of our approach, we first compare it with two other approaches in Figure 5. Results in the figure are based on the realistic simulation environment SL [28], which also acts as a real-time interface to Barrett WAM in our experiments. To construct the error bars, we sample from the model $p(\mathbf{w}|\mathbf{y}_b)$ the weights of our movement primitives 5 times. Comparisons to two baselines illustrate the benefits of our approach, especially the faster convergence and increased accuracy of the proposed method. Notice that the update (10) shown in black considers the full cost over a static reference trajectory and hence can become easily unstable in our extended setting. In practice the model matrix (8) amplifies the effects of nonrepetitive starting positions[2] and nonzero velocities, violating the initial condition assumption typical of ILC updates.

The second baseline shown in red is the *current-iteration* ILC (*CI-ILC*) where we add the feedback from the previous iteration to the feedforward commands in the next iteration

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \mathbf{K}_{LQR}\mathbf{e}_k. \tag{15}$$

In practice we find that this additional adjustment of feedforward inputs makes learning more robust. We also observe robustness with respect to nonrepetive initial condition errors. We add this compensation also in the two other approaches in Figure 5. The best result is obtained by *gILC* which, in addition to performing better, shows much lower variance compared to the other approaches. See the attached video for a trial run in our robotic table tennis setup. Some examples of the generated trajectories are shown in Figure 6. *N-ILC* was found to be unstable for this trajectory. For *CI-ILC*, the final cost over 30 iterations is stable around 0.50, while the final cost for *gILC* decreases monotonically to 0.20. Repeating the experiment for another 10 iterations, we see that the final cost drops down to 0.13 for *gILC*. As a result of the Mayer formulation it can be seen in Figure 6 that *gILC* does not converge to the reference trajectory but generates very smooth trajectories that converge to the goal point.

[2]The initial positioning of the robot is given by a PD controller with high gains on the shoulder joints.

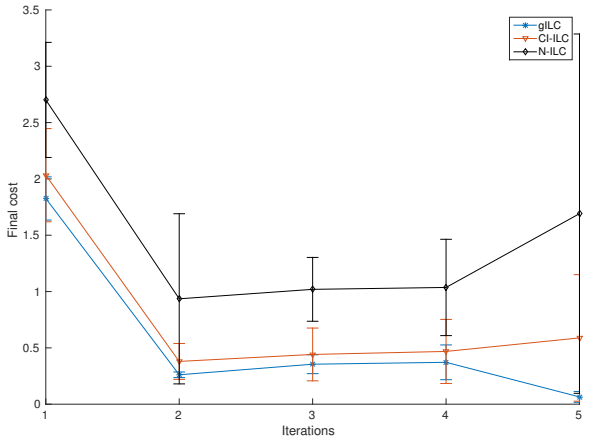


Fig. 5: Simulation results for our new goal-based ILC approach, which we call *gILC*, is shown in blue. Current-iteration ILC, which applies only the update in (15) is shown in red. Note the unstable performance of *N-ILC* which considers the full cost. This is because the initial starting conditions of the movement primitive are varying every iteration.

IV. CONCLUSION AND FUTURE WORK

In this paper we presented a novel Iterative Learning Control algorithm that learns to track the hitting states of dynamic movement primitives (DMPs). DMPs are generated in the joint space of the robot and enable the robot to execute optimal striking motions. Control inputs are updated after each trial by using a model based update rule that considers the deviations from the hitting state of the DMPs. We show two experiments where we evaluate the performance of the approach: putting in golf and striking in robotic table tennis.

Movement primitives are used in our approach as a way to end up at a fixed hitting goal state within the desired time period. DMPs can also be extended easily to new goal positions, while retaining the intended shape of the striking motion. We are currently evaluating new ways to extend our current framework where we can generalize the learning performance of ILC between movement primitives that reach different hitting states. Finally by extending our framework to include the Probabilistic Movement Primitives [24] we hope to increase the generalization ability of ILC and learn to track a whole *distribution* of movement primitives.

V. ACKNOWLEDGMENTS

Part of the research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreement 610878 (3rdHand).

REFERENCES

[1] Hyo-Sung Ahn, Yang Quan Chen, and K.L. Moore. Iterative learning control: Brief survey and categorization. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1099–1121, Nov 2007.

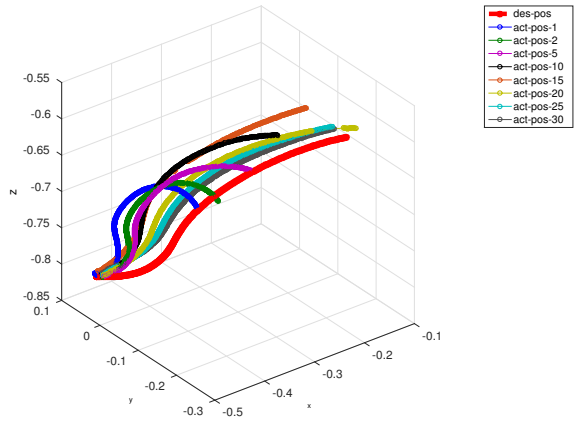


Fig. 6: Robot experiment results for ILC. The desired trajectory, implemented as a DMP, shown in red. See the attached video for the trial run. Note that here convergence is slower for this trajectory, where we exercise caution by applying a learning rate, $\beta = 0.2$. Final cost goes down to 0.20 in the last iteration.

[2] Notker Amann, David H. Owens, and Eric Rogers. Iterative learning control for discrete time systems with exponential rate of convergence, 1995.

[3] Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. Bettering operation of robots by learning. *Journal of Robotic Systems*, 1(2):123–140, 1984.

[4] J. Bolder, T. Oomen, and M. Steinbuch. Aspects in inferential iterative learning control: A 2d systems analysis. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 3584–3589, Dec 2014.

[5] D.A. Bristow, M. Tharayil, and A.G. Alleyne. A survey of iterative learning control. *Control Systems, IEEE*, 26(3):96 – 114, june 2006.

[6] Insik Chin, S.Joe Qin, Kwang S. Lee, and Moonki Cho. A two-stage iterative learning control technique combined with real-time feedback for independent disturbance rejection. *Automatica*, 40(11):1913–1922, November 2004.

[7] M.P. Deisenroth and C.E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML 2011)*, 2011.

[8] Yong Fang and T.W.S. Chow. 2-d analysis for iterative learning controller for discrete-time systems with variable initial conditions. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 50(5):722–727, May 2003.

[9] Tamar Flash and Neville Hogans. The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of neuroscience*, 5:1688–1703, 1985.

[10] A. Gams, B. Nemec, L. Zlajpah, M. Wachter, A. Ijspeert, T. Asfour, and A. Ude. Modulation of motor primitives using force feedback: Interaction with the environment and bimanual tasks. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5629–5635, Nov 2013.

[11] J. Ghosh and B. Paden. Pseudo-inverse based iterative learning control for nonlinear plants with disturbances. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 5, pages 5206–5212 vol.5, 1999.

[12] Svante Gunnarsson and Mikael NorrlöF. Brief on the design of ilc algorithms using optimization. *Automatica*, 37(12):2011–2016, December 2001.

[13] Stefan Hillenbrand and Madhukar Pandit. An iterative learning controller with reduced sampling rate for plants with variations of initial states. *International Journal of Control*, 73(10):882–889, 2000.

[14] A.J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 1398–1403, 2002.

[15] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Comput.*, 25(2):328–373, February 2013.

[16] J. Kober, K. Muelling, O. Kroemer, C.H. Lampert, B. Schoelkopf, and J. Peters. Movement templates for learning of hitting and batting. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[17] J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems 22 (NIPS 2008), Cambridge, MA: MIT Press*, 2009.

[18] Christoph H. Lampert and Jan Peters. Real-time detection of colored objects in multiple camera streams with off-the-shelf hardware components. *J. Real-Time Image Process.*, 7(1):31–41, March 2012.

[19] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, Princeton, NJ, USA, 2011.

[20] Richard W. Longman. Iterative learning control and repetitive control for engineering practice. *International Journal of Control*, 73(10):930–954, 2000.

[21] Guilherme J. Maeda, Ian Manchester, and David Rye. In press: Combined ILC and disturbance observer for the rejection of near-repetitive disturbances, with application to excavation. *IEEE Transactions on Control Systems Technology*, March 2015.

[22] K. Muelling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *International Journal of Robotics Research*, (3):263–279, 2013.

[23] Mikael Norrlöf and Svante Gunnarsson. Time and frequency domain convergence properties in iterative learning control, 2002.

[24] A. Paraschos, C. Daniel, J. Peters, and G Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems (NIPS), Cambridge, MA: MIT Press.*, 2013.

[25] Kwang-Hyun Park and Zeungnam Bien. A generalized iterative learning controller against initial state error. *International Journal of Control*, 73(10):871–881, 2000.

[26] Jan Peters, Katharina Mülling, and Yasemin Altün. Relative entropy policy search. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, pages 1607–1612. AAAI Press, 2010.

[27] E. Rogers, K. Galkowski, and David H. Owens. *Control Systems Theory and Applications for Linear Repetitive Processes*. Springer-Verlag, 2007.

[28] S. Schaal. The SL simulation and real-time control software package. Technical report, 2006.

[29] Angela P. Schoellig, Fabian L. Mueller, and Raffaello D'Andrea. Optimization-based iterative learning for precise quadrocopter trajectory tracking. *Autonomous Robots*, 33:103–127, 2012.

[30] Mark W. Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons, Hoboken (N.J.), 2006.

[31] Richard Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[32] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[33] Yuval Tassa, Tom Erez, and William D Smart. Receding horizon differential dynamic programming. In *Advances in neural information processing systems (NIPS)*, pages 1465–1472, 2008.

[34] E. A. Theodorou, J. Buchli, and S. Schaal. Learning policy improvements with path integrals. In *International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, 2010.

[35] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, Xiao-Yu Fu, K. Goldberg, and P. Abbeel. Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2074–2081, May 2010.