2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Force Adaptation with Recursive Regression Iterative Learning Controller

Bojan Nemec[1], Tadej Petrič[1], and Aleš Ude[1]

Abstract— **In this paper we exploit Iterative Learning Controllers (ILC) schemes in force adaptation tasks. We propose to encode the control signal with Radial Basis Functions (RBF), which enhances the robustness of the ILC scheme and allows to vary the execution speed of the learned motion. For that a novel control scheme is proposed, which updates the feed-forward compensation signals based on current iteration cycle signals in contrast to the standard ILC, which uses signals from the previous iteration cycle. This reduces the computational burden and enhances the adaptation speed. Stability of the proposed control law is analysed and discussed. The proposed approach is evaluated in simulation and on a Kuka LightWeight Robot Arm where the task was to perform force-based surface following with both discrete and periodic movements.**

I. Introduction

Iterative learning control (ILC) is often used in robotics due to its simplicity, effectiveness and robustness when dealing with repetitive operations. Many tasks can be found in industry as well as in home environments, that need to be executed repeatedly. In such cases, humans usually acquire and perfect the skill over several repetitions of the task. The same learning principle can also be adopted in machine motor control theory, where a system follows a similar trajectory repeatedly and updates necessary parameters in order to perfect the skill.

One possible method for improving the skill knowledge is iterative learning control. The main objective of ILC is to improve the behavior of the control system that operates repeatedly by iterative refinement of the feed-forward compensation signal [1]. The ILC is also closely related to the Repetitive Control (RC), where the desired task trajectory is a periodic function of time, and without resetting between periods [2]. In contrast to the RC framework, the ILC system is designed to return to the same initial condition before each new execution of the task. The ILC framework has been successfully applied to many practical tasks that arise in manufacturing and robotics [3], [4], humanoid robotics [5], path tracking [6], medical robotics [7], health care robotics [8], and visual servoing [9]. Furthermore, the basic ILC schemes can be extended with methods that enhance their robustness [10], [11] and the adaptability of the overall system [12], [13]. The problem of the minimum variance learning has also been addressed within the ILC framework [14].

[1]Humanoid and Cognitive Robotics Lab, Department of Automatics, Biocybernetics and Robotics, Jožef Stean Institute, Ljubljana, Slovenia, bojan.nemec@ijs.si, tadej.petric@ijs.si, ales.ude@ijs.si

In [15] an ILC scheme was applied for motion coordination of a bimanual robot task using force feedback. In this case the dynamic motion primitives (DMP) framework was used for the representation of motion trajectories. The feed-forward compensation signal was generated by the ILC. DMPs in conjunction with ILC were applied also in [16] to improve a force based assembly skill, where phase stopping algorithm was utilized to enhance the robustness of the overall framework. By replacing the time dependency in the ILC with the phase dependency, a standard ILC assumption of the equal duration of the trials could be removed. Using this substitution, ILC framework was successfully applied also for speed optimization of the demonstrated skills [17].

In this paper we propose a new learning controller that combines the ideas of standard ILC and on-line coaching [18], resulting in improved adaptation speed, robustness and ease of implementation. The structure of the proposed algorithm is closely related to the DMP framework. The proposed algorithm is general and can be used with both types of movements that can be represented by DMPs, i. e. discrete and periodic movements. The performance of the proposed algorithm was evaluated and compared with ILC schemes as proposed by our previous research [16], using both simulation and real robot experiments. Two experiments were performed, one showing the performance of the algorithm on a discrete task and the other on a periodic task, respectively.

The paper is organized as follows. In Section II we briefly outline the dynamic movement primitives for discrete and periodic tasks. In Section III, the classical ILC control scheme and how this scheme evolves into the newly proposed algorithm is presented. In Section IV the experimental results and the effectiveness of the proposed algorithm in comparison to the more standard ILC approaches is shown. Two different cases were considered 1) discrete tasks and 2) periodic tasks, both involving force interaction with the environment.

II. Encoding trajectories with DMPs

In this section we give a brief overview of the underlying representation used in our approach. It is based on parametric description of trajectories and signals with dynamic movement primitives [19]. Within this framework, trajectories, given either in joint or in task space, are described as an output of the second order differential equation modulated with radial basis functions to get the desired shape. Every

degree of freedom is described by its own dynamic system, but with a common phase which synchronise them together.

For point-to-point movements (also referred to as discrete movements) the trajectory for each degree of freedom y is described by the following system of nonlinear differential equations

$$\tau \dot{z} = \alpha_z(\beta_z(g-y)-z) + f(x), \quad (1)$$

$$\tau \dot{y} = z, \quad (2)$$

$$\tau \dot{x} = -\alpha_x x, \quad (3)$$

where x is the phase variable and z is an auxiliary variable. α_x, α_z, β_z and τ are defined such that the system converges to the unique equilibrium point $(z, y, x) = (0, g, 0)$. By setting $\alpha_z = 4\beta_z$, the linear dynamic system is critically damped.

The nonlinear term f contains free parameters that are used to modify the dynamics of the second-order differential equation system to approximate any smooth point-to-point trajectory from the initial position y_0 to the final configuration denoted as goal g. The nonlinear term f is given by

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \quad (4)$$

$$\Psi_i(x) = \exp\left(-h_i\,(x-c_i)^2\right), \quad (5)$$

where the given initial and final velocity are equal to zero. N radial basis functions with width $h_i > 0$ are equally distributed along the trajectory where c_i are the centers. Weights w_i are estimated with regression in such that the DMP encodes the desired trajectory.

For the periodic representation of the trajectories, the nonlinear differential equations take the identical form as for discrete signals given with (1) and (2), except for the phase, which is

$$\dot{x} = \Omega. \quad (6)$$

Here, $\Omega = 1/\tau$ denotes the frequency of an periodic signal. Note that in the case of the periodic representation the phase x is a monotonically increasing function. Also the non-linear function f has the same form as for discrete signals, except that the kernel functions ψ_j are Gaussian periodic functions defined as

$$\psi_j(x) = \exp\left(\frac{1}{2h_j^2}(\cos(x-c_j)-1\right). \quad (7)$$

Parameters c_j and h_j respectively define the center and width of the j-th periodic basis function, while w_i are adjustable weights used to define the desired shape of the trajectory.

One of the advantages of DMPs is that they can be modulated both spatially and temporally without changing the overall shape of motion. Ijspeert et al. [19] introduced a slow-down feedback, where the robot is automatically halted on excessive position error. In [16] this principle was used to slow down the trajectory execution on excessive forces and torques. Another benefit of the DMPs is the robustness against the sudden change of the goal variable g. Whenever a new goal is specified during the DMP evolution, the resulting trajectory is governed according to the response of the second order system.

III. Recursive Regression Iterative Learning Controller (RRILC)

The output y of the DMP enables us to follow the demonstrated trajectories. In many practical situation, it is necessary to change the output of the learned DMP to achieve the desired goal (see Section IV for some practical examples). In this paper we consider the situation where we add a control signal u to the output of the DMP y, i. e. the positional signal for robot control is defined as

$$y = y_{\mathrm{DMP}} + u. \quad (8)$$

To learn the optimal control signal u, the widely used iterative learning control approach can be applied [1]

$$u_{l+1}(k) = Q(u_l(k) + Le_l(k+d)), \quad (9)$$

where l is the learning iteration index, k denotes the time sample, d is constant time delay, $e_l(k)$ is the error signal that describes the difference between the desired and the actual task execution. See Section IV for some practical implementations of e. Q and L could be selected as discrete transfer functions that determine the ILC behaviour and the overall stability of the learning system. In many cases, including in our experiments, they are set to constant values. In the cases when the control system time delay is known in advance, a transient response of the system can be improved by setting parameter d to a proper value, otherwise it is often set to 1.

Standard ILC assumptions include: 1) Stable system dynamics, 2) System returns to the same initial conditions at the start of each trial, 3) Each trial has the same length. Stability analysis can be performed in a lifted time domain system or in frequency domain using z-transformation [1]. Parameters may be tuned also using heuristic approach [20] where the choice of Q is a tradeoff between the stability region and steady-state error. Q in the form of low pass filter rejects disturbances and enhances the robustness [20].

ILC in the form (9) 'waits' one cycle before it modifies the control signal. To override this limitation, a feedback control signal is usually added to the plant resulting in

$$\begin{aligned} u_{l+1}(k) &= Ce_{l+1}(k) + Q(u_l(k) + Le_l(k+d)) \\ &= Ce_{l+1}(k) + s_l(k). \end{aligned} \quad (10)$$

This scheme is often referred to as 'current iteration' ILC [1]. Please note that $C = C(q)$ is actually a discrete transfer function, where q^{-1} is the backward shift operator [21]. For the sake of simplicity we omitted q dependence. The ILC feed-forward signal $s_l(k)$ can be computed in advance after the completion of each learning phase l. Here we propose to replace the sequence $\{s_l(k)\}$ with $\sigma(\mathbf{w}_l, x)$, where $\sigma(\mathbf{w}_l, x)$ is the same signal encoded with radial basis functions, i.e.

$$\sigma(\mathbf{w}_l, x) = \frac{\sum_{i=1}^{N} w_{l,i} \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)}, \quad (11)$$

where Ψ is defined as in (5) in the case of discrete movements and as in (7) in the case of periodic movements. x is the phase (3) used to drive the DMP. In this case, (10) turns into

$$u_{l+1}(k) = \sigma(\mathbf{w}_l, x(k)) + Ce_{l+1}(k) \tag{12}$$

where $\sigma(\mathbf{w}_l, x(k)) \approx s_l(k)$ provides the phase dependent feed-forward compensation signal. The benefits of encoding the feed-forward signal using (11) are:

- Since the reference trajectory $y_{DMP}(k)$ is encoded with DMPs (1) – (3), the time dependency is removed form the ILC update and replaced with the phase dependency. Consequently, each learning trial can have different duration and assumption 3) changes to: Each trial has the same length in the phase domain. This enables us to modulate the execution speed according to the interaction with the environment, which enhances the robustness of assembly tasks [16].
- Signal encoded with RBF is close to the original signal passed through the low pass filter. Therefore, with RBF encoding of the ILC feed-forward compensation signal we inherit the robustness of the overall system.

We can estimate weights $\mathbf{w}_{l+1}$ for the feed-forward signal $\sigma(\mathbf{w}_{l+1}, x)$, which will be used in the next iteration $l+1$ also recursively with a recursive least-squares regression,

$$\begin{aligned}\mathbf{w}_{l+1}(k\text{–}d) &= \mathbf{w}_{l+1}(k\text{–}d\text{–}1) + \big[Q(u_l(k-d) + Le_l(k) + \\ &\quad -\mathbf{a}(k\text{–}d)^{\mathrm{T}}\mathbf{w}_{l+1}(k\text{–}d\text{–}1)\big]\,\mathbf{P}(k\text{–}d)\mathbf{a}(k\text{–}d), \end{aligned}\tag{13}$$

$$\begin{aligned}\mathbf{P}(k\text{–}d) &= \frac{1}{\lambda}\Bigg(\mathbf{P}(k\text{–}d\text{–}1) + \\ &\quad \frac{\mathbf{P}(k\text{–}d\text{–}1)\mathbf{a}(k\text{–}d)\mathbf{a}^T(k\text{–}d)\mathbf{P}(k\text{–}d\text{–}1)}{(\lambda + \mathbf{a}^T(k\text{–}d)\mathbf{P}(k\text{–}d\text{–}1)\mathbf{a}(k\text{–}d))}\Bigg),\end{aligned}$$

$$\mathbf{a}(k\text{–}d) = \frac{1}{\sum_{i=1}^{N}\Psi(x(k-d))}\begin{bmatrix}\Psi_1(x(k\text{–}d))\\ \vdots \\ \Psi_N(x(k\text{–}d))\end{bmatrix}, \tag{14}$$

where λ is the forgetting factor, usually set between 0.97 and 1. This mitigates the need for saving discrete samples $s_l(k)$ needed to compute (10). We need two sets of weights, $\mathbf{w}_l$, calculated recursively in the previous learning cycle and used in the current learning cycle and $\mathbf{w}_{l+1}$, calculated in the current cycle for the next learning cycle.

Let's consider the possibility to use $\mathbf{w}_{l+1}$ already in the current cycle. Doing so, we have to take into account that $\mathbf{w}$ changes in each k-th time instance and affects the control instance in the next time instance. The update equation becomes

$$\begin{aligned}u_l(k) &= Q\sigma(\mathbf{w}_{l+1}(k-1), x(k)) + Ce_l(k) \\ &\approx Qu_l(k-1) + Ce_l(k).\end{aligned}\tag{15}$$

We can see that this choice simply integrates control signal u across the samples k. Moreover, the signal σ depends only on the initial weights at $k=1$. Therefore, it does not change in the next cycle and the algorithm loses its learning capability.

Instead, we propose to replace (13) with

$$\mathbf{w}_l(k) = \mathbf{w}_l(k-1) + K_r e_l(k)\mathbf{P}(k)\mathbf{a}(k), \tag{16}$$

similar as for coaching in [18]. K_r is suitably chosen gain.

In continuation we will show that (16) and (12) actually form an ILC. First, it is essential to know the transfer function, which maps $K_r e(k) \rightarrow \mathbf{w}(k) \rightarrow \sigma(\mathbf{w}(k), x(k))$ using (16), (13) and (11). Since it is difficult to evaluate it analytically, we used identification instead. The results of the identification showed, that σ can be approximated with

$$\sigma(\mathbf{w}_l(k), x(k)) \approx \sum_{i=1}^{l} GK_xK_re_i(k), \tag{17}$$

where K_x is fixed gain and G is a transfer function of a second order low pass filter. Both the gain K_x and the parameters of the filter G depend on the widths h_i (5) and the number of the gaussian kernel functions. Fig. 1 shows an example, where σ was calculated using (16),(13) and (11) and approximated using identified system given by (17).

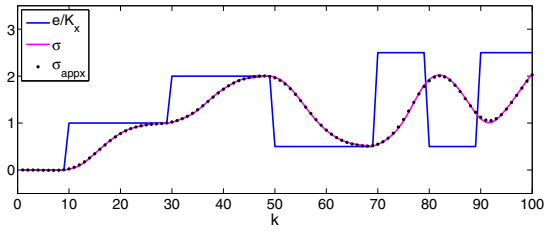

Fig. 1. Comparison of $\sigma(\mathbf{w}, x)$ and approximated σ using (17) at given error e/K_x, generated randomly. 22 gaussian kernel functions with width $h = 0.75$ were used. Identified system was $G = \frac{0.067z^{-2}+0.135z^{-1}+0.0675}{z^{-2}-1.143z^{-1}+0.413}$ and $K_x = 39.216$ respectively.

Let's rewrite (17) into the form

$$\begin{aligned}\sigma(\mathbf{w}_l(k), x(k)) &\approx \sum_{i=1}^{l-1} GK_xK_re_i(k) + GK_xK_re_l(k) \\ &= \sigma(\mathbf{w}_{l-1}, x(k)) + GK_xK_re_l(k)\end{aligned}\tag{18}$$

If we now include also the current iteration loop controller C, the control update equation becomes

$$\begin{aligned}u_l(k) &= \sigma(\mathbf{w}_l, x(k)) + Ce_l(k)), \\ &= \sigma(\mathbf{w}_{l-1}, x(k)) + (GK_xK_r + C)e_l(k),\end{aligned}\tag{19}$$

which proves, that parameter update using (16) results in an ILC like algorithm, where $L = 0$ (referred as 'current cycle learning' [22]). Note that this algorithm learns signal $\sigma(\mathbf{w}, x(k))$, which is not exactly the control input from the previous iteration. As we use recursive regression, we name this approach Recursive Regression Iterative Learning Controller (RRILC). For discrete tasks, the vector $\mathbf{a}(k)$ is formed using kernel functions according to (5). For repetitive tasks, kernel functions in vector $\mathbf{a}(k)$ in (14) are chosen according to (7). In the continuation of the paper we refer to the version of the algorithm with periodic kernels as Recursive Regression Repetitive Controller (RRRC). Note

that the smoothing of the previous control inputs with filter Q in (10) is inherited by (11) in (19). Note also that this kind of smoothing does not introduce a phase lag as with Q. If an additional attenuation of the control signal from the previous cycle is necessary, this can be done simply by multiplying learned weights $\mathbf{w}$ by a scalar value Q_0 after the completion of each learning cycle, $\mathbf{w}_l = Q_0 \mathbf{w}_{l-1}$. In contrast to the standard ILC, this formulation contains additional feedback term in the current iteration loop, which can be approximated with $GK_xK_re(k)$. With careful selection of parameters, this term can enhance the robustness and stability, as it will be shown latter in our experiments. Additionally, this algorithm allows to non-uniformly vary the speed of the given task during the iterations, which is not possible with standard ILC.

A. Stability analysis

The aim of the stability analyses is to determine how the iteration loop affects the stability of our ILC system. It is assumed, that the feedback system without the iteration loop is already stable. If we apply one sided z-transformation to (19), we obtain

$$\begin{aligned} U_l &= U_l^* + CE_l \\ &= U_{l-1}^* + DE_l, \end{aligned} \tag{20}$$

where U^* denotes the z-transformation of the signal σ (19) and $D = GK_xK_r + C$. Capital letters denote z-transformation of the corresponding time dependent signals denoted with lower case letters. Again, for sake of simplicity, we omitted the z dependence in transfer functions and signals. Further, we assume that S is known transfer function which describes the linearized robot dynamics. Z-transformation of the feedback error can be thus given in the form

$$E_l = Y_d - SU_l, \tag{21}$$

where Y_d denotes the z-transformation of the desired output value Y. Inserting (20) into (21) yields

$$\begin{aligned} E_l &= Y_d - S(U_{l-1}^* + DE_l) \\ &= Y_d - S(U_{l-1}^* + CE_{l-1} - CE_{l-1} + DE_l) \\ &= Y_d - S(U_{l-1} - CE_{l-1} + DE_l) \\ &= Y_d - Y_{l-1} + SCE_{l-1} - SDE_l \\ &= E_{l-1} + SCE_{l-1} - SDE_l, \\ \frac{E_l}{E_{l-1}} &= \frac{1+SC}{1+SD}. \end{aligned} \tag{22}$$

Asymptotic stability is assured iff $\frac{E_l}{E_{l-1}} < 1 \quad \forall l$. Therefore, the condition for the asymptotic stability becomes [21], [22]

$$\frac{1 + S(e^{j\omega})C(e^{j\omega})}{1 + S(e^{j\omega})D(e^{j\omega})} < 1, \quad \forall \omega, \tag{23}$$

where $\omega = [-\pi, \pi]$. ω denote a frequency normalized with the sampling time of a time discrete system. Stability condition also implies that the tracking error $E_l(e^{j\omega})$ tends to 0 for $l \to \infty$. The design of the ILC controller can be thus performed in subsequent steps; 1) check if (23) is fulfilled, by e.g. Bode or Nyquist plot, 2) Tune the parameters of the transfer function C, gain K_r, widths h and number N of the Gaussian Kernel Functions until (23) is fulfilled.

IV. Simulation and experimental evaluation

In this section we evaluate the performance of the proposed algorithm and compare it to the standard ILC as given by (10). As some features and benefits of the new algorithm are easier to show in simulation, we evaluate the methods both in simulation and in real experiments. Real experiments were performed with KUKA LWR robot equipped with Barrett hand. The robot was controlled from Matlab/Simulink environment using Fast Research Interface [23] at 100 Hz. Both in simulation and in real world experiments we used an identical setup. We selected low gains in the position controller of the KUKA robot, which results in compliant behaviour. The arm stiffness was set to $1000, 1000$ and 500 N/m for translational axes and to 100 Nm/rad for rotational axes, respectively.

A. Force based surface following

In many industrial applications the robot is required to maintain a contact with a previously unknown surface. Typical operations which involve this problem are polishing, grinding, cleaning, etc. This problem is relevant also for the future generation of home robots while performing task from everyday life such as cleaning the table, polishing furniture, etc [24]. Force control is needed to solve such problems. Clearly, if the robot is able to move around while precisely maintaining the contact force, the surface shape can be directly captured from the robot's motion. In our experiment, one robot was holding a paint roller in the hand. The task was to follow the previously unknown surface at constant speed of 0.15 m/s in X direction while maintaining the constant force of 5 N in Z direction. The shape of the object was a triangle as illustrated in Fig. 2. The search trajectory was defined as a straight line in X direction and encoded with DMPs using (1) and (2). We applied the admittance stiffness force control [25], combined with ILC, in discrete time form,

$$\begin{aligned} \mathbf{y}_{c,l+1}(k) &= \mathbf{y}_{DMP}(k) + \sum_{j=1}^{k} \mathbf{K}_{fi}\mathbf{e}_{l+1}(j) + \\ &\quad \mathbf{K}_{fp}\mathbf{e}_{l+1}(k) + \mathbf{s}_l(k), \\ \mathbf{e}_{l+1}(k) &= \mathbf{F}_d(k) - \mathbf{R}\mathbf{F}_{m,l+1}(k)), \end{aligned} \tag{24}$$

where $\mathbf{y}_c$ is the control position fed to the robot controller, $\mathbf{y}_{DMP}$ is the position as returned from the DMPs, $\mathbf{R}$ is the current robot rotation matrix, $\mathbf{F}_d$ are the desired forces, $\mathbf{F}_m$ are measured forces in the tool coordinate system and $\mathbf{K}_{fi}$ and $\mathbf{K}_{fp}$ are the force feedback matrices, respectively. ILC feed-forward signal $\mathbf{s}_l(k)$ was calculated as in (10), where error signal (24) was used for learning. d was set to 3.

In RRILC, we applied the same control law as in (24), except that time dependent feed-forward signal $\mathbf{s}_l(k)$ was replaced with the phase dependent RRILC generated signal $\sigma(\mathbf{w}_l, x(k))$, calculated for each robot coordinate according

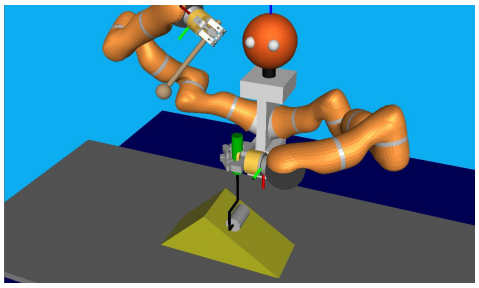

Fig. 2. Simulated environment

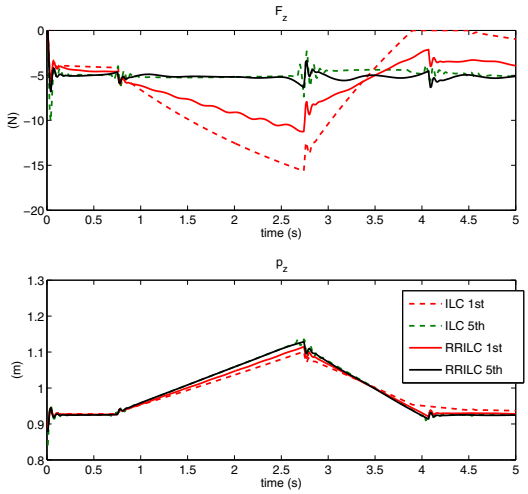

Fig. 3. Comparison of simulation results obtained with ILC and RRILC. Upper graph shows the force tracking in Z direction. Lower graph shows the learned Z coordinate of the surface

to (11) and (12). The corresponding weights $\mathbf{w}$ were calculated using (16) for each Cartesian coordinate, where the error signal $e_l(k)$ was the corresponding component of the force error (24).

Five learning cycles were performed with ILC and RRILC. The forgetting factor λ was set to 1, i.e., no forgetting was applied. After each learning cycle, we reset the covariance matrix to $10\mathbf{I}$. Parameters $\mathbf{K}_{fi}$, $\mathbf{K}_{fp}$, $\mathbf{L}$, $\mathbf{K}_r$, $\mathbf{Q}$, h and N were set to 0.0001 $\mathbf{I}$, 0.001 $\mathbf{I}$, 0.001 $\mathbf{I}$, 0.0005 $\mathbf{I}$ and 0.99 $\mathbf{I}$, 0.75, and 20, respectively. Fig. 3 shows compensation signals, forces, positions and orientations of the robot as obtained in simulation. For the sake of clarity, only the results of the first and the fifth (last) learning cycle are shown. It is evident that RRILC outmatches the ILC especially in the first cycle. As it generates the feed-forward compensation signal already in the first cycle, it converges considerably faster than the standard ILC. Note that the standard ILC lost the contact with the surface after 4 seconds. It can be easily verified that the controller (24) is not capable of tracking constant environment slope with zero force error in the first cycle, as the compensation term $s_1(k)$ is zero. The gains $\mathbf{K}_{fi}$ and $\mathbf{K}_{fp}$ were experimentally set in such a way that the closed loop system exhibited critically damped response on impact. We can notice also smoother response of RRILC due to the smoothing provided by regression.

The same task was repeated in real environment. The only difference was that the shape of the unknown object was selected as shown in Fig. 4. The object was made of thin aluminium plate, one part of the plate was firmly attached to the bottom, while the other was not. Therefore, during the surface following the environment stiffness was changing from very stiff at the beginning to very compliant at the end. Again, we compared the performance of the ILC and RRILC algorithm. The results are shown in Fig. 5. Also in the real experiment we can see that the newly proposed RRILC results in faster learning. Similar as in the simulated environment, also here the standard ILC has lost contact with the surface at the end of the first learning cycle. However, after five repetitions, the results were virtually identical with both algorithms, which is the expected behavior.

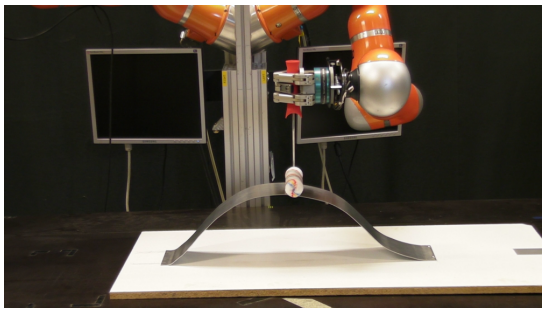

Fig. 4. Experimental setup

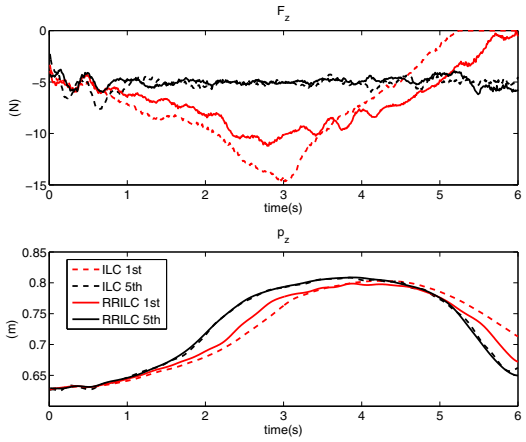

Fig. 5. Comparison of the real world experiments performed with ILC and RRILC. Upper graph shows the force tracking in Z direction. Lower graph shows the learned Z coordinate of the surface.

B. Stirring in a pot

One of the most common cooking activities is stirring. The purpose is to mix liquid ingredients by moving a spoon (or similar tool) around the pot in a circular motion. In many cases it is necessary to apply small forces to the pot wall while stirring. Here, we consider a scenario, where the stirring motion was obtained by the demonstration for another pot with smaller diameter and elliptical shape. The desired forces were captured from human demonstration. The center of the pot was displaced in X and Y directions for 2

cm and 3 cm with respect to the center of the demonstrated motion, respectively. The task of the robot was to adapt to the new situation as quickly as possible. For this, we applied RC and RRRC and compared the performance of both algorithm.

The control algorithm in the time discrete form was

$$\begin{aligned} \mathbf{y}_c(k) &= \mathbf{y}_{DMP}(x(k)) + (K_f e(k) + s(k-T))\ \mathbf{r}(x(k)), \\ e(k) &= \mathbf{r}(x)^{\mathrm{T}}(\mathbf{F}_d(x(k)) - \mathbf{R}\mathbf{F}_m(k)), \\ \mathbf{r}(x) &= [\cos(x), \sin(x), 0]^{\mathrm{T}}, \end{aligned} \tag{25}$$

where x denotes the phase angle calculated by (6), $\mathbf{y}_c$ is the commanded robot position, $\mathbf{y}_{DMP}$ the trained periodic DMP, $\mathbf{R}$ is the current robot rotation matrix, $\mathbf{F}_d$ are the desired forces obtained from user demonstration, $\mathbf{F}_m$ are the measured forces in the robot tool coordinate system, e is the radial force error, T is the stirring period, and K_f is the scalar force feedback gain. Transformation $\mathbf{r}(x)$ transforms the phase angle x into a position on the unit circle at height $z = 0$.

The RC feed-forward signal $s(k-T)$ corresponds to the radial displacement. It was estimated according to the RC implementation of the ILC update (10), where the signals from the previous learning phase were replaced with the signals delayed for one stirring period T. d was set to 3. Parameters K_f, L and Q were set to 0.001, 0.0005, and 0.99, respectively. For RRRC, the feed-forward compensation signal $s(k-T)$ was replaced with the phase dependent RRRC generated signal $\sigma(x(k))$, calculated according to (11) and (16). The forgetting factor λ, the gain K_r, h and N were experimentally set to 0.992, 0.0005, 200 and 80, respectively. The duration of the stirring cycle was 2 seconds and 8 stirring cycles were preformed. The results are shown in Fig. 6. It can be seen that the RRRC established full contact with the pot wall already in the second stirring cycle and proceeded with almost unchanged response for the following 6 cycles. On the other hand, RC took 6 stirring cycles to establish full contact with the pot wall. In the last cycle, learned compensation terms were almost identical for both algorithms. Fig. 7 shows the comparison of the learned path for both algorithms, measured at the wrist position. Due to the compliance of both orientational axes of the robot and hand fingers, this position slightly deviates from the end effector (spoon) position. Finally, we investigated how the motion can be adapted to a pot with non-smooth edges, e.g, to a square pot, as shown in Fig. 8. The reference trajectory and forces were the positions and forces learned in the last stirring cycle from the previous example. The stirring trajectory was slowed down by a factor of 4 by changing the τ variable in DMPs from 2 to 8. After two stirring cycles we increased the stirring speed by factor of 2 and after four cycles again by factor of 3. As this is possible only with RRRC learning algorithms, we could not compare the results with RC algorithm. Results are outlined in Fig. 9. Also in this case, RRRC exhibited fast adaptation to the new geometry and to the speed variations. Fig. 10 shows the learned path measured at the robot wrist. Although the spoon followed the square pot wall, the wrist trajectory slightly differs due to the high compliance of the robot wrist.

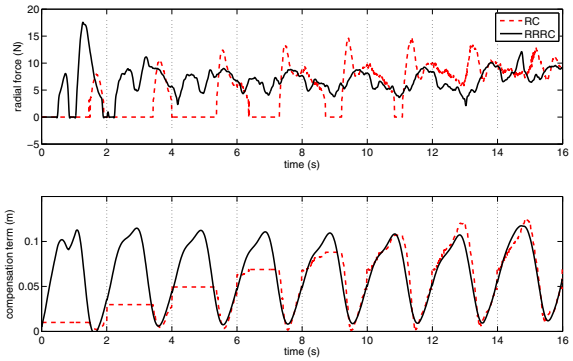

Fig. 6. Comparison of the experimental results for pot stirring obtained with RC and RRRC. Upper graph shows the force tracking in radial direction. Lower graph shows learned compensation term in radial direction. Dotted vertical lines denote stirring cycles.

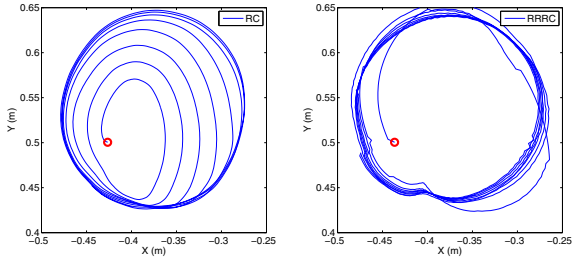

Fig. 7. Comparison of the learned trajectory for pot stirring obtained with RC (left) and RRRC (right).

V. Conclusions

We proposed a new controller based on ILC and RC paradigms, which applies recursive estimation of the feed-forward signal for the minimization of the control error. It enables to compensate pure time delays in the controlled plant and generates smooth control signals. The proposed learning controller was verified both in simulation and in real experiments involving force interaction with the environment. Simulation and experimental results have shown that the main benefit of the new controller is the significantly improved speed of learning. The proposed controller belongs to a class of causal ILC controller, for which it was proved that the same steady state error can be obtained with standard feedback controller applying high gains [26].

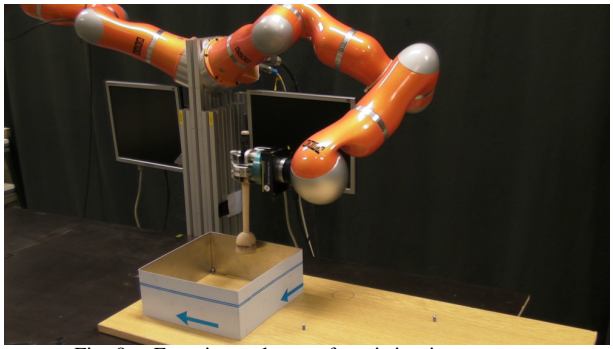

Fig. 8. Experimental setup for stirring in square pot.

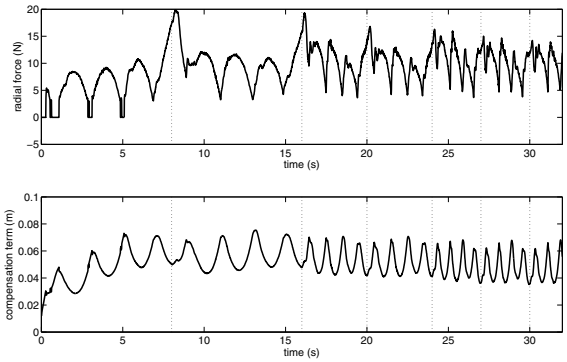

Fig. 9. Experimental results for stirring in square pot obtained with RRRC. Upper graph shows the force tracking in radial direction. Lower graph shows learned compensation term in radial direction. Dotted vertical lines denote stirring cycles.

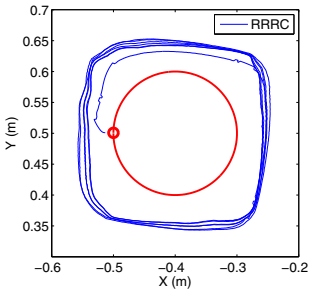

Fig. 10. Learned trajectory for stirring in square pot obtained with RRRC . The plotted positions correspond to the robot wrist positions. Red and blue line denote initial and learned trajectory, respectively

However, high gain controllers are not suitable for robots working in unstructured environment and interacting with humans, therefore we consider that ILC is still favourable. In this work our aim was to learn the force-based skills with highly compliant robot applying low control gains, which are beneficial for robots working in cooperation with humans and in unstructured environments. In the future we will investigate also the behaviour of the proposed algorithm in torque-based force control schemes.

REFERENCES

[1] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[2] Y. Wang, F. Gao, and F. J. Doyle, "Survey on iterative learning control, repetitive control, and run-to-run control," *Journal of Process Control*, vol. 19, no. 10, pp. 1589–1600, 2009.

[3] B. Bukkems, D. Kostic, B. de Jager, and M. Steinbuch, "Learning-based identification and iterative learning control of direct-drive robots," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 537–549, 2005.

[4] P. Cano Marchal, O. Sörnmo, B. Olofsson, A. Robertsson, J. Gómez Ortega, and R. Johansson, "Iterative learning control for machining with industrial robots," in *Preprints of the 19th IFAC Congress*, Cape Town, South Africa, 2014.

[5] P. A. Bhounsule and K. Yamane, "Iterative learning control for high-fidelity tracking of fast motions on entertainment humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Atlanta, Georgia, USA, 2013.

[6] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013.

[7] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, 2010, pp. 2074–2081.

[8] C. Freeman, E. Rogers, A. Hughes, J. Burridge, and K. Meadmore, "Iterative learning control in health care: Electrical stimulation and robotic-assisted upper-limb stroke rehabilitation," *IEEE Control Systems Magazine*, vol. 32, no. 1, pp. 18–43, 2012.

[9] P. Jiang, L. Bamforth, Z. Feng, J. E. F. Baruch, and Y.-Q. Chen, "Indirect iterative learning control for a discrete visual servo without a camera-robot model," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 4, pp. 863–876, Aug 2007.

[10] M. Norrlöf and S. Gunnarsson, "Disturbance aspects of iterative learning control," *Engineering Applications of Artificial Intelligence*, vol. 14, no. 1, pp. 87–94, 2001.

[11] S. Gunnarsson and M. Norrlöf, "On the disturbance properties of high order iterative learning control algorithms," *Automatica*, vol. 42, no. 11, pp. 2031–2034, 2006.

[12] M. Norrlöf, "An adaptive iterative learning control algorithm with experiments on an industrial robot," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 245–251, Apr 2002.

[13] A. Tayebi, "Adaptive iterative learning control for robot manipulators," *Automatica*, vol. 40, no. 7, pp. 1195–1203, 2004.

[14] R. Tousain, E. van der Meche, and O. Bosgra, "Design strategy for iterative learning control based on optimal control," in *40th IEEE Conference on Decision and Control*, Orlando, Florida, 2001, pp. 4463–4468.

[15] A. Gams, B. Nemec, A. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 816–830, 2014.

[16] B. Nemec, F. J. Abu-Dakka, B. Ridge, J. A. Jorgensen, T. R. Savarimuthu, J. Jouffroy, H. G. Petersen, N. Krüger, and A. Ude, "Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile," in *16th International Conference on Advanced Robotics (ICAR)*, Montevideo, Uruguay, 2013.

[17] B. Nemec, A. Gams, and A. Ude, "Velocity adaptation for self-improvement of skills learned from user demonstrations," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Atlanta, Georgia, USA, 2013, pp. 423–428.

[18] T. Petrič, A. Gams, L. Žlajpah, A. Ude, and J. Morimoto, "Online approach for altering robot behaviors based on human in the loop coaching gestures." in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014, pp. 4770–4776.

[19] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[20] M. Norrlöf and S. Gunnarsson, "Experimental comparison of some classical iterative learning control algorithms," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 636–641, 2002.

[21] K. L. Moore, Y. Chen, and H.-S. Ahn, "Iterative learning control: A tutorial and big picture view," in *Decision and Control, 2006 45th IEEE Conference on*, dec. 2006, pp. 2352 –2357.

[22] J.-X. Xu, S. K. Panda, and T. H. Lee, *Real-time Iterative Learning Control*. London: Springer-Verlag, 2009.

[23] G. Schreiber, A. Stemmer, and R. Bischoff, "The fast research interface for the kuka lightweight robot," in *ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications*, Anchorage, Alaska, 2010, pp. 15–21.

[24] C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments [grand challenges of robotics]," *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 20–29, 2007.

[25] L. Villani and J. De Schutter, "Force control," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer, 2008, pp. 161–185.

[26] P. B. Goldsmith, "Brief on the equivalence of causal lti iterative learning control and feedback control," *Automatica*, vol. 38, no. 4, pp. 703–708, Apr. 2002.