Autonomous Cycle Time Reduction of Robotic Tasks Using Iterative Learning Control*

Lorenz Halt[1], Michael Meindl[2], Victor Bayer[1], Werner Kraus[1] and Thomas Seel[3]

***Abstract*— When robots are used to automate repetitive production tasks, the productivity of the manufacturing system crucially depends on the robot's task execution speed. An out-of-the-box solution is typically slow, whereas achieving shorter cycle times typically requires large efforts with respect to controller design and tuning. This dilemma can be resolved by learning control algorithms that autonomously improve performance without requiring any system-specific tuning. In the present work, we propose a novel learning control scheme that autonomously reduces the execution times of robotic systems that perform repetitive manufacturing tasks. To this end, we combine an Iterative Learning Control (ILC) approach with a trial-varying reference adaptation. The reference trajectory is slowly adapted to ensure that the given task is performed successfully on every single iteration without constraint violations. Therefore, the learning process can be carried out during operation. We validate the practical applicability of the method by real-world experiments on a 6-axis robot that performs a linear motion and a contact-force task. Despite the fundamentally different characteristics of these two tasks, the proposed algorithm achieves a remarkable reduction of cycle times, namely, by a factor of 4 in the linear motion task and a factor of 10 in the contact-force task. These results provide an important step toward robotic manufacturing systems that autonomously optimize their own performance during operation.**

I. INTRODUCTION

Rooted in the desire to automate increasingly complex manufacturing tasks, the importance of robots in industrial production continues to rise. Compared to manual operation, robotic manufacturing systems can achieve significantly higher production speeds, which define the efficiency of any manufacturing system. However, the effort required to configure a robotic production system is significantly larger than for manual production. In fact, much of the effort required for programming an industrial use-case is typically spent optimizing execution times manually.

For illustration, consider the example of a *snap-fit insertion task*, see, e.g., [1]–[3], that consists of grasping a part, approaching a target piece, inserting the part, and releasing it. While it is comparatively easy to program appropriate robot paths using methods such as classical teaching or free drive, the significant portion of development effort is typically spent reducing the robot's execution time. Here, the tasks consists in finding a satisfying trade-off between short execution times and undesired dynamical effects such as oscillations or overshoots. Manual optimization may be further complicated if the production task requires contact-rich or complex processes.

In order to reduce the effort necessary for deploying robotic systems, recent research has, on one hand, brought

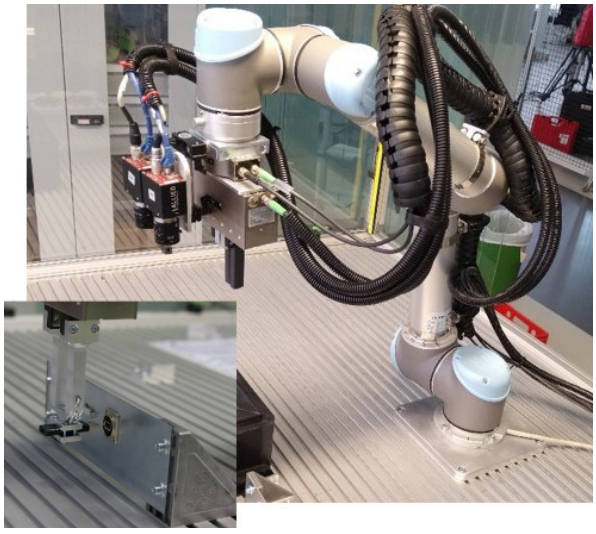

Fig. 1: (top) A 6 axis robotic arm that has to solve various tasks like a move-to and contact-force skill. (bottom) The proposed learning control scheme combines reference adaptation with Iterative Learning Control (ILC) to iteratively reduce the task's execution time while also complying with given constraints.

fourth concepts like *intuitive robot programming* [4]–[13]. On the other hand, one may also consider learning methods to automate the deployment process. For example, Iterative Learning Control (ILC) addresses reference tracking tasks in repetitive systems by iteratively updating a feedforward input trajectory based on the errors of previous trials [14]. Due to its robustness with respect to model uncertainties and unknown disturbances, ILC has found great appeal in practical applications. Experimental applications include gantry

robots, CNC machines, and industrial manipulators [15]–[20]. Methodological works have produced concepts such as *norm optimal ILC* [15], which employs model information to design effective learning laws, *reference-adapting ILC* to handle output constraints [21], or *point-to-point ILC* [22], in which the output trajectory only has to pass through a set of reference points. However, while all of these works make significant contributions to the methodology of ILC and its application to real-world problems, they only consider the problem of reference tracking and *not* optimization of execution times. Only in [23], a P2P-ILC problem is considered where a timing-based cost criterion is optimized in parallel. Similarly, [24] consider the problem of optimal time allocation in P2P-ILC. However, these contributions have in common that the main task still consists in tracking a set of reference points and that the optimization of timing is a secondary problem. Instead, we are interested in a learning method that primarily focuses on the iterative reduction of cycle times.

This paper addresses the problem of autonomously reaching the maximum performance of a robotic system without the need for manual tuning, see Figure 1. This is achieved by choosing a simple, poorly-tuned feedback controller that is only capable of tracking slow references and leveraging learning to iteratively decrease the execution time of the task. In particular, we propose a novel learning control method that combines Iterative Learning Control (ILC) with a reference adaptation scheme such that the execution time can be reduced iteratively. As the proposed scheme adapts the control just in minor increments, undesired dynamical effects such as oscillations or overshoots are avoided and optimization during operation becomes feasible. The method's practical applicability is demonstrated by experiments on a 6-axis robot that has to perform a *linear motion* and *contact-force* task. Because the proposed method can be seamlessly combined with state-of-the-art techniques such as *intuitive programming*, the proposed method contributes to automatically deploying robots in industrial production and reducing associated costs and efforts.

II. PROBLEM FORMULATION

Consider a robot that has to repeatedly solve a task like, e.g., grasping a part, approaching a target location, and inserting the part. We assume that the robot is capable of performing a slow movement that safely solves the task. Such a solution can, e.g., be programmed by an operator, learned via imitation learning, or obtained by the combination of a reference generator and feedback controller. However, the execution time of this initial motion significantly deviates from the optimal execution time. Hence, the cycle-time of the task must be reduced in order to maximize the robot's productivity.

This work addresses the problem of autonomously decreasing a production task's execution time during operation, i.e., the learning algorithm not only has to reduce the execution time, but also has to ensure that the production task is solved on each single iteration. The latter requires that undesired dynamical effects such as oscillations or excessive overshoots must be avoided during the learning process. Furthermore, the solution has to be applicable to real-world problems, and, hence, the reduction of cycle-times has to be carried out on the real-world production system and *not* in simulated environments.

Formally, we consider a discrete-time, repetitive system with input variable $u_j \in \mathbb{R}$ and output variable $y_j \in \mathbb{R}$, whereby $j \in \mathbb{N}$ is the trial index, see Figure 2. We assume that on each trial the output variable starts from the same initial value y_0. The control task consists in transitioning the output variable from the initial value y_0 to a desired set-point y^*. The input variable u_j consists of two components. First, there is the feedforward component u_j^{FF} that is determined by the learning algorithm. Second, there is the feedback component u_j^{FB} that is determined by the feedback controller such that the overall input is computed by

$$u_j = u_j^{\mathrm{FF}} + u_j^{\mathrm{FB}} \,. \tag{1}$$

The feedback controller computes u_j^{FB} based on the error signal e_j that is the difference between the trial-varying reference r_j and the output variable y_j, i.e.,

$$e_j = r_j - y_j \,. \tag{2}$$

To express the problem conveniently, the $N \in \mathbb{N}$ samples of each trial are collected in a trajectory. In particular, let $\mathbf{y}_j \in \mathbb{R}^N$ denote the output trajectory, let $\mathbf{u}_j^{\mathrm{FF}} \in \mathbb{R}^N$ denote the feedforward input trajectory, let $\mathbf{r}_j \in \mathbb{R}^N$ denote the trial-varying reference trajectory, and let $\mathbf{e}_j \in \mathbb{R}^N$ denote the error trajectory, i.e.,

$$\mathbf{y}_j := \begin{bmatrix} y_j(1+m) & \dots & y_j(N+m) \end{bmatrix} \tag{3}$$

$$\mathbf{u}_j^{\mathrm{FF}} := \begin{bmatrix} u_j^{\mathrm{FF}}(1) & \dots & u_j^{\mathrm{FF}}(N) \end{bmatrix} \tag{4}$$

$$\mathbf{r}_j := \begin{bmatrix} r_j(1) & \dots & r_j(N) \end{bmatrix} \tag{5}$$

$$\mathbf{e}_j := \mathbf{r}_j - \mathbf{y}_j \,, \tag{6}$$

whereby $m \in \mathbb{N}_{\geq 0}$ is the robot's relative degree. Now, the control task consists in having the output trajectory $\mathbf{y}_j$ converge to the desired set-point y^* as quickly as possible. The learning task consists in updating the feedforward trajectory $\mathbf{u}_j^{\mathrm{FF}}$ and reference trajectory $\mathbf{r}_j$ on each trial such that the output trajectory $\mathbf{y}_j$ converges to the desired set-point y^* in less time than on the previous trial. Furthermore, the learning algorithm has to ensure that the output reaches the desired set-point on each trial without undesired dynamical effects such as overshoots or oscillations.

III. PROPOSED ALGORITHM

The control problem of transitioning from an initial output value to a desired set-point requires, first of all, a suited reference trajectory. This reference trajectory can be chosen by either of the two following strategies. First, a comparatively slow reference trajectory can be chosen which comes at the advantage of the feedback controller being capable of precisely tracking the reference. However, the resulting motion is inevitably slow and does not fulfill the requirements regarding the robot's desired speed of operation. The

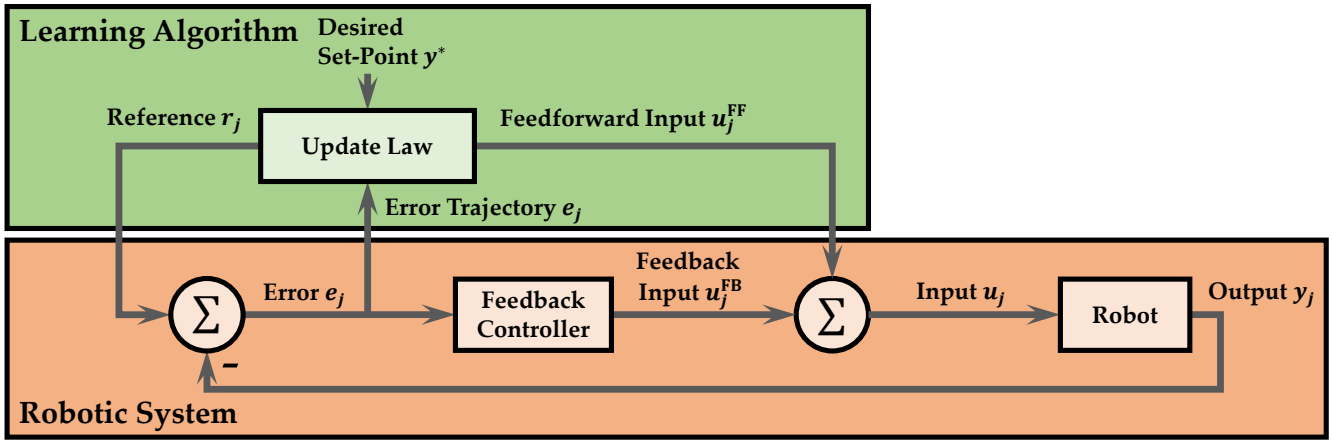

Fig. 2: Block diagram of the problem setup: The learning algorithm has to iteratively update the feedforward control input $\mathbf{u}_j^{\mathrm{FF}}$ and the trial-varying reference $\mathbf{r}_j$ such that the execution time of the production task is decreased from trial to trial while also ensuring that the output trajectory $\mathbf{y}_j$ converges to the desired set-point y^* on each iteration.

second strategy consists in choosing a comparatively fast reference trajectory that, if precisely executed, would satisfy the requirements regarding the robot's desired speed of operation. However, due to the comparatively large bandwidth of a fast motion, the feedback controller may not be capable of precisely tracking the reference trajectory and the task may fail. To improve tracking performance, the feedback controller could be combined with Iterative Learning Control (ILC). However, due to the large tracking errors on the initial trials, the ILC would aggressively adapt the control signal, which would result in undesired dynamical effects such as overshoots or oscillations, and, hence, the task may fail at least in some trials during the learning process meaning that learning can not be performed during operation.

To circumvent the issue of excessive changes in the output trajectory and subsequent failures of the production task, we propose a novel learning control scheme that combines ILC with an iterative reference adaptation, which results in a trial-varying ILC scheme, i.e.,

$$\forall j \in \mathbb{N}_{\geq 0}, \quad \mathbf{u}_{j+1}^{\mathrm{FF}} = \mathbf{Q}\left(\mathbf{u}_j^{\mathrm{FF}} + \mathbf{L}\left(\mathbf{r}_j - \mathbf{y}_j\right)\right), \tag{7}$$

where $\mathbf{Q} \in \mathbb{R}^{N\times N}$ is the Q-filter matrix, and $\mathbf{L} \in \mathbb{R}^{N\times N}$ is the learning gain matrix. The update law (7) is well-established in the ILC literature but typically employs a trial-invariant reference [25]. One possible method for designing the Q-filter matrix is to lift the transfer function of a low-pass filter to limit learning to a bandwidth that is relevant to the application and, thus, increase robustness of the learning system [14]. The learning gain matrix $\mathbf{L}$ can be chosen as the identity matrix multiplied by a scalar gain, where the latter is chosen to achieve a reasonable trade-off between fast but also safe learning [14].

The trial-varying reference in (7) is computed by a reference generator and depends on three parameters, namely, the initial output value y_0, the desired set-point y^*, and the desired transitioning time $t_j^* \in \mathbb{R}$. Based on these three parameters, the reference generator computes a trajectory that smoothly connects the initial and the desired output value within the time specified by t_j^*, where the latter allows us to dictate the speed of the desired motion. For the remainder of this paper, we employ cosine references, i.e., $\forall n \in [1, N]$,

$$\mathbf{r}_j = \mathbf{r}(y_0, y^*, t_j^*) \tag{8}$$

$$= \frac{y^* + y_0}{2} - \frac{y^* - y_0}{2} \cos\left(\frac{\pi}{t_j^*} Tn\right), \tag{9}$$

whereby $T \in \mathbb{R}$ is the sampling period. Note that the choice of cosine references is without loss of generality because the proposed method can be applied with any type of reference trajectory that smoothly connects the initial and desired output values.

To ensure that the control task is fulfilled on each trial without undesired dynamical effects, while also iteratively decreasing the execution time, the following learning control scheme is proposed: First, we choose a desired transitioning time t_0^* that is sufficiently large such that the feedback controller can precisely track the reference, i.e., the desired set-point is reached without undesired dynamical effects. Next, the ILC input $\mathbf{u}_0$ is initialized as the zero vector, applied to the plant, and the output trajectory $\mathbf{y}_0$ is recorded. From here on wards, like on any other trial, the norm of the error trajectory is computed, and if it falls below the error threshold $\bar{e} \in \mathbb{R}$, the desired transitioning is adjusted by the update law

$$\forall j \in \mathbb{N}_{\geq 0}, \quad t_{j+1}^* = \begin{cases} t_j^* & \|\mathbf{e}_j\|_2 \geq \bar{e} \\ \alpha t_j^* & \|\mathbf{e}_j\|_2 < \bar{e} \end{cases}, \tag{10}$$

whereby $\alpha \in (0, 1)$ is a time-scaling factor. The trial-wise adjustment of the desired transitioning time (10) ensures a comparatively small tracking error on each trial such that the feedback controller and ILC only make minor adjustment to the control signal, and, hence, undesired dynamical effects

or even failure of the control task is avoided. Based on the adjusted transitioning time t^*_{j+1}, the next-trial reference $\mathbf{r}_{j+1}$ is computed, and the ILC update (7) is performed. The latter yields the next-trial input trajectory $\mathbf{u}_{j+1}$, which is again applied to the plant, and the procedure is repeated.

IV. EXPERIMENTAL VALIDATION

In this section, we first compare the proposed learning scheme to two baseline methods in simulations of a simple example dynamics. Afterwards, the proposed learning scheme's capability of reducing the execution time of production tasks is demonstrated on a robotic testbed, which has to solve two different tasks.

A. Simulations

First, we use simulation to compare the proposed method's performance to i) only feedback control and ii) a standard ILC. For this purpose, we simulate a simple spring-mass-damper system, in which the output $y \in \mathbb{R}$ is given by the position of the mass and the input $u \in \mathbb{R}$ is an external force acting on the mass. The system is sampled at a frequency of $50\,\mathrm{Hz}$, the transfer function is given by

$$G(z) = \frac{0.0002z + 0.0002}{z^2 - 2z + 0.9998}\,, \tag{11}$$

and the input is additionally delayed by 10 samples.

The system has to solve a *move-to skill* starting from a position of $0\,\mathrm{m}$ and has to reach a position y^* of $1\,\mathrm{m}$. The task is solved if the output variable reaches and remains within a tolerance band of $\pm 0.05\ \mathrm{m}$ at the goal position. Ideally, the task is solved in the desired time of $2\,\mathrm{s}$, and the task must be solved in a maximum time of $8\,\mathrm{s}$.

To solve this problem, we consider the following three approaches:

- Feedback Control: First, we only employ a PD-type feedback that was manually tuned to achieve a minimal execution time but also be robust with respect to the input delays between 5-12 samples.
- Standard ILC: Second, we employ a standard ILC on top of the feedback control, where the learning gain matrix is designed via norm-optimal ILC and the weights were chosen to achieve rapid learning performance. To determine the reference, we employ the reference generator (8) with the desired execution time of $2\,\mathrm{s}$.
- Reference-Adapting ILC: Third, we employ the proposed reference-adapting ILC scheme with the same learning gain matrix as in the standard ILC approach. The error threshold is set to $\overline{e} = 2$, and the time scaling factor is set to $\alpha = 0.9$.

The results depicted in Figure 3 show that the feedback controller requires a comparatively long period of roughly $6.5\,\mathrm{s}$ to solve the task and does not improve performance from trial to trial. The standard ILC is capable of reducing the execution time to roughly $3\,\mathrm{s}$ on the tenth trial. However, due to undesired dynamical effects such as overshoots during the learning, the standard ILC does not complete the task on each trial and the execution times increase on some trials. In contrast, the proposed reference-adapating ILC manages to solve the task on each single trial. Furthermore, the execution time decreases monotonically from trial to trial and reaches the desired execution time of $2\,\mathrm{s}$ on the tenth trial. The results demonstrate that the proposed method can not only leverage the potential of learning to iteratively improve execution times, but can also ensure that the task is fulfilled on each single trial by adapting the reference trajectory on each trial.

B. Experimental Setup

The experimental studies were carried out using a Universal Robots UR5 equipped with an end-of-arm 6D force-torque-sensor KMS40 by Weiss robotics. The UR5 robot is a 6 axis, lightweight industrial robot with a payload of $5\,\mathrm{kg}$ and a reach of $850\,\mathrm{mm}$. The velocity of each of the robot's joint is controlled by an inner feedback loop. To solve different tasks, as, e.g., either reaching a desired contact force or Cartesian coordinate of the end-effector, an outer control loop is implemented using the *pitasc* framework [2], [7]. Depending on the task, the reference of pitasc, and thus the controller, is either the Cartesian end-effector position or the force-torque measurements. In either case, pitasc outputs corrective joint velocities with a rate of 125 Hz that are calculated by a proportional feedback controller with small gains. Note that the goal in feedback design consisted in spending little tuning effort but also in ensuring that arbitrary skills can be safely performed, which comes at the price of comparatively slow motions.

To solve arbitrary problems, the robot has to perform the following six elementary skills, with which complex tasks can be composed. First, there is the *Move* skill, in which the manipulator has to move at a desired velocity. Second, there is the *Move-To* skill, in which the manipulator has to move to a desired position. And third, there is the *Push* skill, in which the manipulator has to reach and maintain a desired contact force. The remaining three skills are the rotational counterparts to the described skills. By combining the elementary skills, the robot can solve a variety of assembly tasks [7].

To demonstrate the proposed method's capability of reducing the execution time of typical production tasks, we consider two different experimental tasks. First, there is the *linear motion task*, in which the manipulator has to transition from one position to another. Second, there is the *contact-force task*, in which the manipulator has to produce and maintain a desired contact force.

In each of the two experiments, a reference generator is implemented that computes a cosine reference trajectory based on the initial state, goal state, and desired transitioning time [2], [3]. On each respective initial trial, the desired transitioning time was chosen such the resulting reference trajectory can be precisely tracked by the feedback controller. The resulting output trajectories successfully transitions the output value to the desired set-point. However, the solution requires a comparatively long period of time. To decrease the execution time and, subsequently, increase the manufacturing system's productivity, the proposed learning method

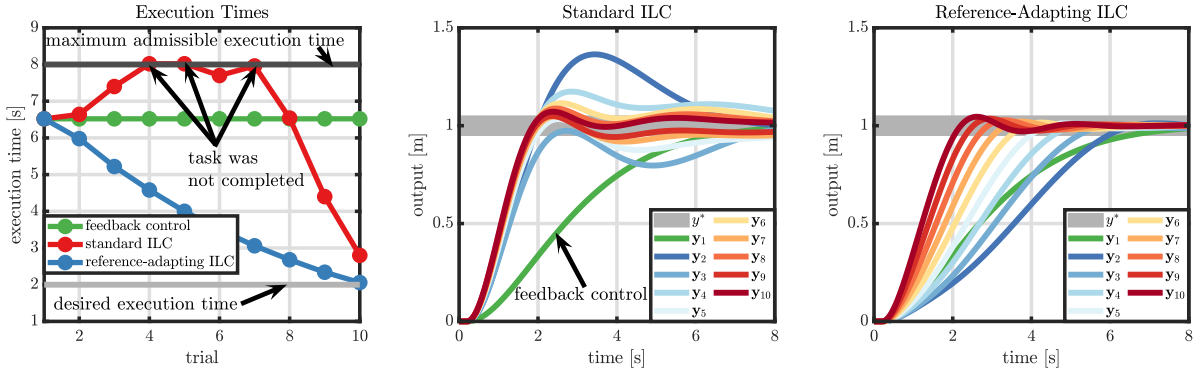

Fig. 3: Comparison of the proposed reference-adapating ILC to i) a PD-type feedback control and ii) a standard ILC. The feedback controller manages to solve the task in a comparatively long period of roughly 6 s, but can not improve its performance from trial to trial. The standard ILC is capable of reducing the execution time to roughly 3 s, but fails to complete the task on each single trial. In contrast, the proposed reference-adapting ILC not only solves the task on each single trial, but also manages to monotonically decrease the execution time to the desired time of 2 s withing ten trials.

TABLE I: symbolic representation of elementary robot capabilities (compare [2])

	Symbol	skill name	task space Coordinates
(a)		*Move*	$\chi_f = [\dot{x}, \dot{y}, \dot{z}]^T$
(b)		*Move (rotational)*	$\chi_f = [\dot{\alpha}, \dot{\beta}, \dot{\gamma}]^T$
(c)		*Move to*	$\chi_f = [x, y, z]^T$
(d)		*Move to (rotational)*	$\chi_f = [\alpha, \beta, \gamma]^T$
(e)		*Push*	$\chi_f = [f_x, f_y, f_z]^T$
(f)		*Push (rotational)*	$\chi_f = [m_x, m_y, m_z]^T$

is applied.

C. Linear Motion Task

In the first experiment, the robot has to solve a linear motion task that consists in the elementary *move-to skill*, i.e., the robot has to change its position along one Cartesian coordinate by a value of $0.05\,\mathrm{m}$. The manipulator's coordinate is defined as the system output y, the initial value is given by $y_j(0) = 0.05\,\mathrm{m}$, and the desired set-point is given by $y^* = 0\,\mathrm{m}$. Using the initial and desired output, the combination of reference generator and feedback controller yields an initial output trajectory $\mathbf{y}_0$ that principally solves the motion task, see Figure 4. However, execution requires a comparatively long period of $12\,\mathrm{s}$ because the feedback controller was configured to strictly avoid any kind of overshoot. Next, the proposed learning scheme is applied to decrease the execution time.

The ILC's Q-filter matrix $\mathbf{Q}$ and learning gain matrix $\mathbf{L}$ are chosen according to the procedure described in Section III, i.e., $\mathbf{Q}$ is chosen as the lifted form of a fifth order, forward-backward Butterworth, and the proportional learning $\mathbf{L} = 0.6\mathbf{I}$ is employed. The ILC uses a phase-lead of $m = 25$ with a sampling period of $T = 0.008$ ms. The reference adaptation (10) uses an error threshold of $\bar{e} = 0.03$, i.e., the desired execution-time t_j^* is adjusted if the norm of the error trajectory $\mathbf{e}_j$ falls below 0.03, and the time scaling factor $\alpha = 0.9$. The proposed learning scheme is applied for a total of 100 trials.

The resulting output trajectories depicted in Figure 4 show that throughout all trials the time until reaching the set-point is not only decreased, but the output trajectories also remain in the close proximity of the desired set-point. We, hence, conclude that the task is solved on each iteration, i.e., learning can take place during operation. The progression of the execution time over trials, see Figure 4, shows that the learning methods manages to reduce the execution time from an initial value of $12\,\mathrm{s}$ to a value as small as $3\,\mathrm{s}$ after 100 iterations.

D. Contact-Force Task

In the second experiment, the robot has to solve a contact force task that consist in the elementary *force skill*, i.e., the robot has to reach and maintain a contact force of $10\,\mathrm{N}$, which is measured by an end-of-arm 6D force-torque-sensor. The contact force is defined as the system output y, the initial value is given by $y_j(0) = 0\,\mathrm{N}$, and the desired set-point is given by $y^* = 10\,\mathrm{N}$. The initial trajectory resulting from the reference generator and feedback controller solves the task but requires an execution time of $50\,\mathrm{s}$ because the feedback controller has to ensure that the task is solved despite the distance and the pliability of the surface are unknown. To again optimize performance, the proposed learning method is applied.

The ILC update again uses the lifted form of a fifth order, forward-backward Butterworth as Q-filter matrix $\mathbf{Q}$, the proportional learning $\mathbf{L} = 5 \times 10^{-5}\mathbf{I}$, and a phase-lead of $m = 5$ with a sampling period of $T = 0.008$ ms. The

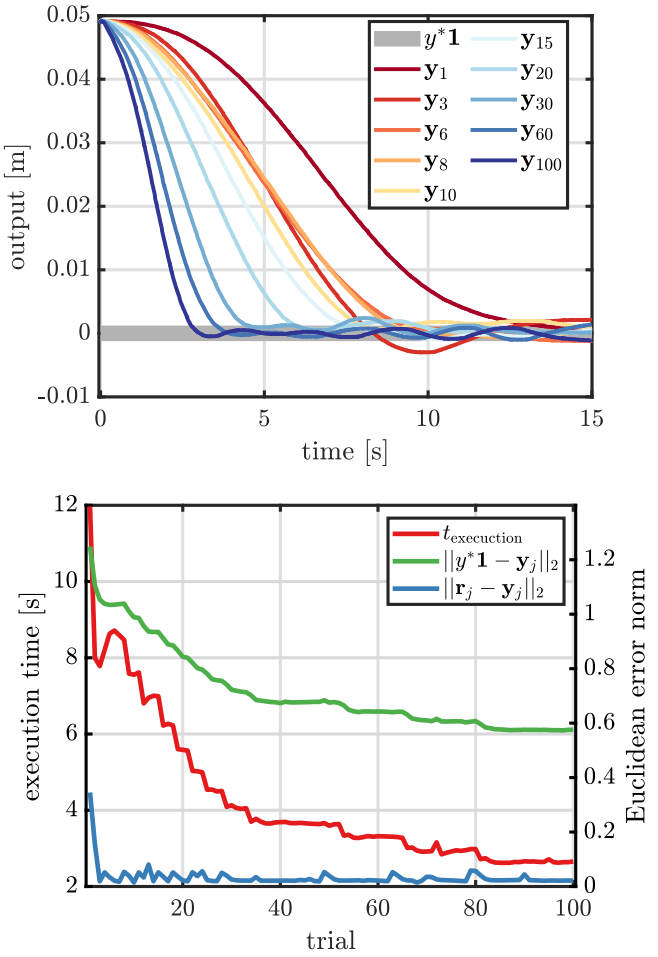

Fig. 4: (top) Selected output trajectories of the *linear motion* experiment. (bottom) Progression of the execution time and error norms throughout trials.

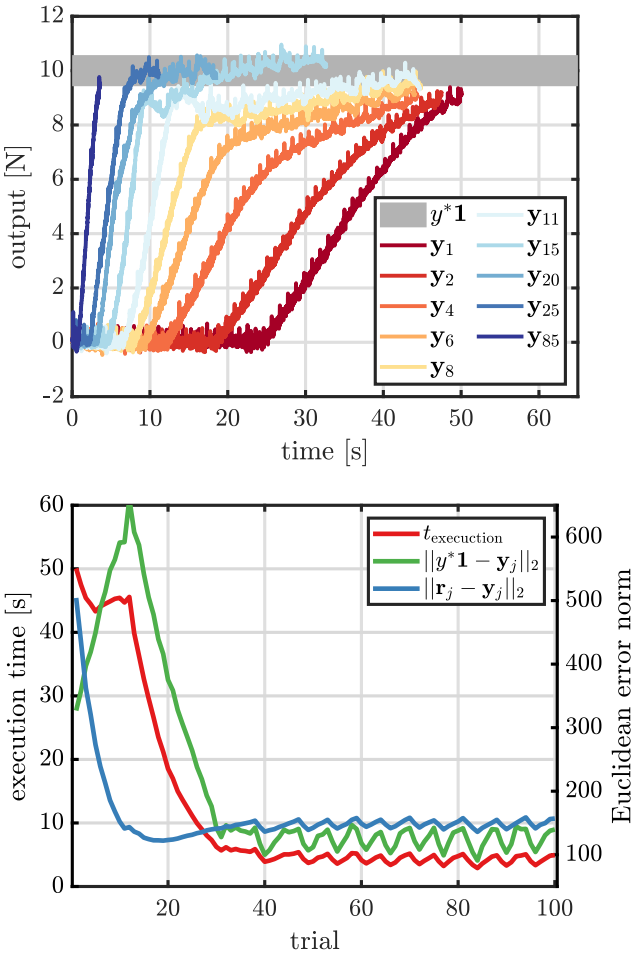

Fig. 5: (top) Selected output trajectories of the *contact force* experiment. (bottom) Progression of the execution time and error norms throughout trials.

reference adaptation (10) uses an error threshold of $\bar{e} = 150$, i.e., the desired transitioning-time t_j^* is adapted if the norm of the error trajectory $\mathbf{e}_j$ falls below 150, and the time scaling factor $\alpha = 0.9$. The proposed learning scheme is applied for a total of 100 trials.

The output trajectories depicted in Figure 5 again show that the proposed learning method not only decreases execution time but also solves the task on each trial without overshoots or other undesired dynamical effects. The progression of the execution time demonstrates that the proposed method reduces the execution time from an initial value of 50 s to a final value of roughly 5 s after 100 trials. Note that in comparison to the first experiment, the dynamics involved in the force task are fundamentally different from a linear motion task. Hence, the experiments indicate that the proposed learning scheme generalizes to different tasks in robotics.

V. CONCLUSION

In this work, the problem of autonomously reaching the maximum performance of a robotic system without the need for manual tuning was considered. This was achieved by choosing simple, poorly-tuned feedback control that is only capable of tracking slow references and leveraging learning to iteratively decrease the execution time of the task. In particular, have proposed a learning algorithm that combines Iterative Learning Control (ILC) with a novel reference adaptation scheme to iteratively reduce the execution times of robotic tasks. Due to the method's incremental approach, undesired dynamical effects such as overshoots or oscillations are avoided during learning such that the production task is solved on each iteration. Hence, the proposed learning scheme can be applied during operation.

The proposed method was validated in real-world experiments on a 6 axis robot that has to perform a *linear motion* and a *contact-force* task. The proposed method was capable of reducing the initial execution time by a factor of 4 in the first experiment and a factor of 10 in the second experiment. The different characteristics of the two tasks indicate that the proposed method is generic, i.e., it can be applied to a variety of typical application tasks in robotics. We believe that the proposed method can inspire research that aims at employing

learning control methods to solve problems that are relevant to robotic application such as, e.g., autonomously and safely optimizing the execution time of robots during operation.

The presented results represent a first step towards autonomously optimizing complex robotic systems but are limited because only single-input/single-output systems and single-phase tasks were considered. Future research is going to extend the method to multi-input/multi-output systems and is going to consider complex multi-phase tasks that consist of multiple skills such as moving, rotating, and pushing. Furthermore, the parameters of the reference adaptation have to be tuned to ensure compliance with the constraints. Hence, the proposed method can be improved by explicitly incorporating the constraints into the reference adaptation scheme.

REFERENCES

[1] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, and A. Wortmann, "A new skill based robot programming language using UML/P Statecharts," in *2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*. Piscataway, NJ: IEEE Computer Society, 2013, pp. 461–466.

[2] L. Halt, P. Tenbrock, F. Nägele, and A. Pott, "On the implementation of transferable assembly applications for industrial robots," in *2018 IEEE . Symposium on Robotics (ISR)*. VDE Verlag, 2018.

[3] F. Nägele, L. Halt, P. Tenbrock, and A. Pott, "A prototype-based skill model for specifying robotic assembly tasks," in *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2018.

[4] T. Hasegawa, T. Suehiro, and K. Takase, "A model-based manipulation system with skill-based execution." *IEEE Transactions on Robotics and Automation, Vol. 8, S. 535-54*, 1992.

[5] R. Bischoff, A. Kazi, and M. Seyfarth, "The MORPHA style guide for icon-based programming," in *2002 IEEE Int. Workshop on Robot and Human Interactive Communication (ROMAN)*. Piscataway, NJ: IEEE Computer Society, 2002, pp. 482–487.

[6] U. Thomas and F. M. Wahl, "Assembly Planning and Task Planning — Two Prerequisites for Automated Robot Programming," in *Robotic Systems for Handling and Assembly*, D. Schütz and F. M. Wahl, Eds. Berlin, Heidelberg: Springer, 2011, pp. 333–354.

[7] L. Halt, F. Nägele, P. Tenbrock, and A. Pott, "Intuitive constraint-based robot programming for robotic assembly tasks," in *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2018.

[8] F. Nägele, L. Halt, P. Tenbrock, and A. Pott, "Composition and Incremental Refinement of Skill Models for Robotic Assembly Tasks," in *2019 IEEE Int. Conf. on Robotic Computing (IRC)*. Los Alamitos: Conference Publishing Services, IEEE Computer Society, 2019, pp. 177–182.

[9] N. Somani, A. Gaschler, M. Rickert, A. Perzylo, and A. Knoll, "Constraint-based task programming with CAD semantics: From intuitive specification to real-time control," in *2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Piscataway, NJ: IEEE Computer Society, 2015, pp. 2854–2859.

[10] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, "Intuitive instruction of industrial robots: Semantic process descriptions for small lot production," in *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Piscataway, NJ: IEEE Computer Society, 2016, pp. 2293–2300.

[11] Pedersen, M. Rath, N. Lazaros, S. R. Andersen, S. Casper, B. Simon, and K. Volker, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing, Vol. 37, S. 282-21*, 2016.

[12] S. M. und Haage Mathias und Ekin Anna Topp, "Simplified programming of re-usable skills on a safe in- dustrial robot." *2017 ACMIEEE Int. Conf. on Human-Robot Interacti- on (HRI), S. 463–472*, 2017.

[13] A. Nakamura, K. N. nd Kensuke Harada, and N. Yamanobe, "Using simplified geometric models in skill-based manipulation for objects used in daily life," *Artificial Intelligence Research, Vol. 6*, 2017.

[14] "A survey of iterative learning control," *IEEE Control Systems*, vol. 26, no. 3, pp. 96–114, 2006.

[15] J. D. Ratcliffe, P. L. Lewin, E. Rogers, J. J. Hatonen, and D. H. Owens, "Norm-Optimal Iterative Learning Control Applied to Gantry Robots for Automation Applications," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1303–1307, 2006.

[16] C. T. Freeman, P. L. Lewin, E. Rogers, and J. D. Ratcliffe, "Iterative learning control applied to a gantry robot and conveyor system," *Transactions of the Institute of Measurement and Control*, vol. 32, no. 3, pp. 251–264, 2010.

[17] D.-I. Kim and S. Kim, "An iterative learning control method with application for CNC machine tools," *IEEE Transactions on Industry Applications*, vol. 32, no. 1, pp. 66–72, 1996.

[18] C. Wang, M. Zheng, Z. Wang, C. Peng, and M. Tomizuka, "Robust Iterative Learning Control for Vibration Suppression of Industrial Robot Manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 140, no. 1, 2018.

[19] C. Wang, M. Zheng, Z. Wang, and M. Tomizuka, "Robust two-degree-of-freedom iterative learning control for flexibility compensation of industrial robot manipulators," in *2016 IEEE Int. Conf. on Robotics and Automation (ICRA)*. Piscataway, NJ: IEEE Computer Society, 2016, pp. 2381–2386.

[20] T. Hsiao, "Iterative Learning Control for Trajectory Tracking of Robot Manipulators," *International Journal of Automation and Smart Technology*, vol. 7, no. 3, pp. 133–139, 2017.

[21] M. Meindl, F. Molinari, J. Raisch, and T. Seel, "Overcoming output constraints in iterative learning control systems by reference adaptation," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1480–1486, 2020.

[22] C. T. Freeman and Y. Tan, "Iterative Learning Control With Mixed Constraints for Point-to-Point Tracking," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 604–616, 2013.

[23] M. J. Wu, M. Cobb, C. Vermillion, and K. Barton, "A flexible-time iterative learning control framework for linear, time-based performance objectives," in *2020 American Control Conference (ACC)*, 2020, pp. 4792–4797.

[24] Y. Chen, B. Chu, and C. T. Freeman, "Point-to-point iterative learning control with optimal tracking time allocation," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 5, pp. 1685–1698, 2018.

[25] H.-S. Ahn, Y. Chen, and K. L. Moore, "Iterative learning control: Brief survey and categorization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1099–1121, 2007.