# Learning Approximation of Feedforward Control Dependence on the Task Parameters with Application to Direct-Drive Manipulator Tracking

Dimitry Gorinevsky, *Member, IEEE*, Dirk E. Torfs, *Associate Member, IEEE*, and A. A. Goldenberg, *Fellow, IEEE*

***Abstract*—This paper presents a new paradigm for model-free design of a trajectory tracking controller and its experimental implementation in control of a direct-drive manipulator. In accordance with the paradigm, a nonlinear approximation for the feedforward control is used. The input to the approximation scheme are task parameters that define the trajectory to be tracked. The initial data for the approximation is obtained by performing learning control iterations for a number of selected tasks. The paper develops and implements practical approaches to both the approximation and learning control. As the initial feedforward data needs to be obtained for many different tasks, it is important to have fast and robust convergence of the learning control iterations. To satisfy this requirement, we propose a new learning control algorithm based on the on-line Levenberg–Marquardt minimization of a regularized tracking error index. The paper demonstrates an experimental application of the paradigm to trajectory tracking control of fast (1.25 s) motions of a direct-drive industrial robot AdeptOne. In our experiments, the learning control converges in five to six iterations for a given set of the task parameters. Radial Basis Function approximation based on the learning results for 45 task parameter vectors brings an average improvement of four times in the tracking accuracy for all motions in the robot workspace. The high performance of the designed approximation-based controller is achieved despite nonlinearity of the system dynamics and large Coulomb friction. The results obtained open an avenue for industrial applications of the proposed approach in robotics and elsewhere.**

## I. Introduction

THIS PAPER considers a learning control approach to output tracking in a nonlinear system. The term *learning control* appears in the title of many papers and denotes one of a few different approaches applicable in the absence of a system dynamics model, where the control or the system is 'learned' on the basis of the past operational data for the system. Early work in the learning control systems developed into the modern adaptive control theory, e.g., see [43]. Recently, many adaptive control approaches employing iterative estimation of the system dynamics in the neural network of fuzzy system context have been called learning control.

In this paper, we particularly refer to the learning control approach introduced in the works by Arimoto and others (e.g., see [3], [4]), mostly for robotics applications. The referenced and many other related papers consider one motion of a nonlinear system (manipulator) that is repeatedly executed with updated feedforward input until a desired tracking performance is achieved. The main advantage of such approach is that it does not require an accurate model of the system dynamics. The major practical drawback is that the feedforward control is obtained only for a *single* given task. Should the trajectory change, even slightly, the learning process has to be repeated anew. We remove this barrier by designing an efficient learning-based feedforward controller that works for a *range* of the task parameters. Such task parameters comprise the initial and the final setpoints of the system and define the trajectory to be tracked. Our approach is based on a paradigm of a nonlinear approximation of the feedforward control dependence on these task parameters. The initial data for the approximation is obtained by performing learning control iterations for a set of selected task parameters within a given range.

The paradigm and techniques for obtaining the approximation of the feedforward control are the first and main contribution of this paper. Motivation and application examples for the concept of approximating the dependency of the feedforward control on the task parameters can be found in [16], [18], [20], and [23]. In this paper, we use a radial basis function (RBF) network approximation [35], [36]. RBF approximation has a number of very attractive properties such as excellent accuracy, algorithmic simplicity, and efficient handling of vector-valued functions. It has recently become a much used tool in control engineering applications, where it is often used in the neural network or fuzzy system context.

We would like to note that Arimoto's work, as well as many subsequent papers present human motor skill learning as a

Manuscript received October 31, 1994; revised September 29, 1996. The work of D. E. Torfs was supported by the Prof. R. Snoeys Foundation, University of Toronto. The material in this paper was presented in part by the 1995 American Control Conference, San Francisco, CA, June 1995. This paper was recommended for publication by Associate Editors A. De Luca and Y. Nakamura and Editor A. J. Koivo upon evaluation of the reviewers' comments.

D. Gorinevsky was with the Robotics and Automation Laboratory, University of Toronto, Toronto, Ont., Canada M5S 1A4. He is now with Honeywell Measurex, North Vancouver, B.C., Canada V7J 3S4.

D. E. Torfs is with Trasys Space, Horizon Center, B-1930 Zaventem, Belgium.

A. A. Goldenberg is with the Robotics and Automation Laboratory, Department of Mechanical Engineering, University of Toronto, Toronto, Ont., Canada M5S 1A4.

Publisher Item Identifier S 1042-296X(97)05908-9.

motivation for the development of learning control approach. Elaborating on this idea, one can add that human motor skills are usually acquired for a set of tasks, rather than for a single task. Thus, this paper further extends the motivating idea of learning control. More discussion on learning of the task parameter approximation of feedforward control with regard to human motor control modeling can be found in [20].

A number of papers develop approaches using neural networks or other approximation techniques to learn (identify) nonlinear dynamics model of a system from experimental data, for example see the survey [26]. Although these papers might use similar approximation techniques the approaches are very different from what we propose in this paper. A fundamental advantage of the proposed approach is that its complexity depends on the dimension of the task parameter vector, while complexity of the system dynamics approximation is defined by the state vector dimension, which is usually much larger. Since the approximation complexity grows exponentially with the dimension, our approach is much more attractive for many practical applications.

This paper emphasizes *practical* approaches to both the approximation and initial feedforward data learning. The initial feedforward data for the approximation needs to be obtained for many different tasks, hence, there is a need for fast and robust convergence of the learning process. To satisfy this need, we develop an efficient learning algorithm which has several novel features. This algorithm is the second major contribution of the paper.

Let us describe how the proposed learning algorithm relates to the prior work in the area. Let $v^{(k)}(t)$ be the feedforward input applied to the system at learning iteration $k$ and $e^{(k)}(t)$ be the tracking error at the same iteration; the time $t$ is reset to zero in the beginning of each iteration. A typical learning control algorithm updates the feedforward by using the tracking error information from the *previous* iteration as follows

$$v^{(k)}(t) = v^{(k-1)}(t) + \Gamma e^{(k-1)}(t) \tag{1}$$

where $\Gamma$ is a linear operator.

In the original work of Arimoto, as well as in many subsequent papers, $\Gamma$ describes a standard PID controller, or its special case (P, D, PI, etc.). Some related papers, e.g., [8], [28], [34], consider controllers $\Gamma$ with more sophisticated transfer functions. These papers use conventional design techniques of *causal* controllers. Yet, at the iteration $k$ of the learning control, the entire history of the previous iteration error $e^{(k-1)}(t)$ is available, including the error forward in time.

In order to provide fast convergence of the learning control process, many authors attempted to use various types of the system inverses as the learning gain. Some of them assume that the state vector of the system and its derivative are accessible and use the inverse dynamics of the nonlinear system to compute the feedforward update. With this approach, some authors assume that the system dynamics are known exactly (e.g., [2], [5]), or with a bounded error (e.g., [29]). In other papers, an approximation of the dynamics used in the learning control update is recurrently estimated on-line using neural networks or other schemes [31], [37]. A drawback of these methods is that they are not applicable for output tracking of higher order plants. In this paper, we use an input-output approach and do not assume that the entire state vector is available.

Learning control approaches related to our paper are considered in [11], [24], and [25]. In these papers, continuous-time feedforward is computed as an expansion with respect to a set of certain shape functions. In [24] and [25], the weights of such expansions are updated by integrating the error $e^{(k-1)}(t)$ with time-varying multipliers resulting in learning controllers which are proved to be stable, but are not optimized for speed of convergence.

A consistent way to achieve a high performance of the learning control is to design a controller $\Gamma$ by posing an appropriate *optimization* problem. In this paper, similarly to [11], [16], and [24], we compute the feedforward as a finite expansion with respect to certain time-functions, and design the controller by optimizing a quadratic performance index. Operation of such controller can be considered as an on-line iterative *optimization* process, which includes task execution experiments in the iteration loop. Such optimization-based design of the learning controller was proposed in [17] and [18]. A similar technique is considered in the recent papers [10], [41] published after the completion of this work. Related concepts of the optimization-based learning have been considered in the task-level learning control framework [1]. Note that for a quadratic tracking error index, a step of the Newton–Gauss optimization can be presented in the form (1) with the learning gain $\Gamma$ computed through an inverse of the system linearization.

Our learning control update is based on the on-line *Levenberg–Marquardt* optimization of a quadratic performance index. Compared to the Newton–Gauss method used in [10], [11], [17], [18], and [41], the Levenberg–Marquardt attenuates the feedback learning gain, which improves the performance in uncertain conditions. In the experiments described below, such uncertainty is created by the Coulomb friction. Note that one of the learning algorithms of [27] proposed for the case of uncertainty in the system dynamics is mathematically close to the Levenberg–Marquardt algorithm, though it is not called so.

A key reason, why our learning control approach works well in practice is that it uses a *regularized* performance index with a penalty for the control effort. The experimental results and theoretical analysis presented in the paper demonstrate that the regularization is critical for the convergence, robustness, and acceptable static error sensitivity.

Another important feature of our approach is that the B-splines are used as shape functions in the expansion of feedforward control function. The system gain (gradient of the input-output mapping) defines the direction of the optimization update step and is defined by the responses to variations of these shape function. For a linear time-invariant system, responses to B-splines inputs differ only by shift in time and the gain matrix has a special, block Toeplitz-structure. This matrix is fully defined by a few column vectors. As our experiments show, despite significant nonlinearity of the experimental system, assuming a block-Toeplitz gain matrix structure actually improves the convergence. Pulse response

description of the controlled system is also used as a basis of the learning algorithms of [27], [41].

In a typical tracking problem, learned feedforward input is used together with an inner-loop feedback, which linearizes the system. The importance of the linearizing feedback in a classical learning control is discussed in [7].

The implemented algorithm uses an *adaptive* technique by updating the estimates of the pulse responses in the course of the learning control iterations. Thus, the learning update gain of our algorithm is tuned on line - the algorithm is *adaptive*. Despite the usage of the term "adaptive" in the text and titles, most of the papers surveyed above use a *constant gain* learning control update. That is, the operator $\Gamma$ in (1) does not change from iteration to iteration.

As a result of the described design features, the proposed learning algorithm achieves remarkably fast convergence. In the reported experiments, accurate tracking of fast manipulator motions is achieved in five to six iterations, compared to 20–50 iterations usually needed for the original Arimoto's method. This fast convergence of the learning iterations enabled us to acquire an approximation of the feedforward on the task parameters in experiments of a reasonable duration. Of course, the presented algorithms are applicable to many other real-life control problems as well.

The paper layout is as follows. Section II formulates the problems to be considered in the paper. In Section III, we present the Levenberg–Marquardt algorithm for the learning control and theoretically analyze its convergence and robustness. Section IV considers estimation of the system gain matrix which is needed to compute the learning control gain. Section V describes the experimental setup and discusses the results obtained with the implemented learning controller. Finally, Section VI considers task-parameter approximation of the feedforward using Radial Basis Function approximation method and presents the experimental results in testing the approximation accuracy.

## II. Problem Statement

This section presents a general problem of the task-level feedforward controller design. A nonstandard feature of the problem is that we do not assume a dynamical model of the system to be available. We formulate the learning control approach further used to design the feedforward controller in the absence of a dynamical model.

### A. Control System

Let us consider a control system shown in Fig. 1. For example, such control systems can be typically encountered in robotic devices. The system includes a nonlinear time-invariant (NLTI) multiple-input multiple output (MIMO) controlled **Plant** with the input (control) vector $w(t) \in \Re^n$ and the output (observation) vector $y(t) \in \Re^n$. The system further includes a **Feedback Controller**, which is used to track the desired plant output $y_d(t) \in \Re^n$. The **Feedback Controller** uses the tracking error $e(t) = y(t) - y_d(t), e(t) \in \Re^n$ to compute the feedback control signal $u_{fb}(t)$. In robotic devices, typically, and in the experiments described below,

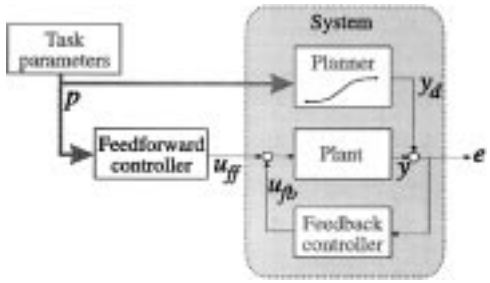


Fig. 1. General scheme of the control system.

in particular, vector $w(t)$ defines the control voltages of the drives, vector $y(t)$ gives the position sensor outputs, and the feedback controller is a standard PID controller.

The desired plant output $y_d$ is computed by a **Planner**. The **Planner** computes the output $y_d$ for the given task parameters. Typically, the components of the task parameter vector $p$ are the initial and final setpoints for the system outputs. In robotic applications, these setpoints define the drive positions at the beginning and at the end of the motion. Thus, the task parameter vector can be expressed as

$$p = \begin{bmatrix} y_d(0) \\ y_d(T) \end{bmatrix} \in \Re^{2n} \tag{2}$$

where we assume that the planned move commences at $t = 0$ and should be completed by $t = T$. The presented control system configuration presumes the execution of a series of tasks. In this paper, we assume that the time $t$ is reset to zero at the beginning of each task.

We assume that the motion time $T$ is a fixed parameter and is the same for all tasks. We further assume that the system is initially in a steady state so that the system output has the desired value, $y(0) = y_d(0)$. In practice, the steady state of the system can be achieved by waiting till the transients die out. In general, it is convenient to assume that the planned trajectory is also from a steady state to a steady state, i.e., $\dot{y}_d(0) = \dot{y}_d(T) = 0$.

In Fig. 1, as well as in subsequent figures, there are two sorts of arrows describing the information flow in the control system. Thin black arrows show continuous-time signals, such as control input to the system, $w(t)$. Fat grey arrows symbolize variables that are defined once per control task, such as the task parameter vector $p$. Later on, we will be considering algorithms working for a *sequence* of control tasks, and it is important to note the mentioned distinction.

The plant control input $w(t)$ is a sum of the feedback control $u_{fb}(t)$ and the feedforward control $v(t)$. The feedforward $v(t)$ is computed by the **Feedforward Controller**. Since the computed feedforward depends on the control task as defined by the task parameter vector $p$, the vector $p$ is shown as an input to the **Feedforward Controller**.

We further consider the **Plant**, the **Feedback Controller**, and the **Planner** as given parts of the system. The topic of this paper is the design of the **Feedforward Controller**. The problem that we are going to solve is to find the feedforward $v(t)$ defined on the time interval $[0, T]$ that allows to achieve a small tracking error $e(t)$. A more formal problem statement is given below.

*Control Problem:* Let the plant, the feedback controller, the path planner, and the motion time $T$, be given and fixed. The closed-loop dynamics of the system are assumed to be unknown. By performing a number of the input-output experiments with the plant, for any value of the task parameter vector $p$ (2) in the given compact domain $p \in \mathcal{P} \subset \Re^{2n}$, find the feedforward input $v(t)$ so that the output tracking error $e(t) = y(t) - y_d(t)$ is small on the interval $[0, T]$, and stays small after the motion time $T$.

The next subsection describes the approach we take for the solution of the formulated problem.

### B. Approximation of Feedforward

In this paper, we develop a nonstandard approach to the nonlinear feedforward controller design. The approach is based on the approximate computation of the optimal feedforward input. If the approximation error is sufficiently small, an acceptably small tracking error can be achieved. As we will demonstrate, one advantage of this approach is that it can be computationally effective. Another advantage is that using this approach we are able to design a nonlinear feedforward controller without relying on an analytical model of the system dynamics. Instead, as the next subsection discusses, a learning control algorithm can be used to obtain the initial design data *directly* from the experiments with the system.

In order to simplify the approximation problem, let us describe the feedforward $v(t) \in \Re^n$ as a time function by using a finite number of parameters. The controller to be designed should compute these parameters depending on the task parameter vector $p$. Let us divide the interval $[0, T]$ into $(N_u + 1)$ subintervals by introducing a sequence of sampling times $t_j = j\tau_u; j = 1, \cdots, N_u$, where $\tau_u = T/(N_u + 1)$. We consider the feedforward input $v(t)$ that is a linear combination of the B-spline functions $B_{\tau_u}(t - t_j)$ centered at the points $t_j$ and can be written in the form

$$v(t) = \sum_{j=1}^{N_u} u^j B_{\tau_u}(t - t_j) \tag{3}$$

where $u^j = [u_1^j \cdots u_n^j]^T$ are the parameter vectors defining the shape of the feedforward control input.

As discussed in the Introduction, it is very important that the functions $B_{\tau_u}$ in the expansion (3) differ only by translations with the base $\tau_u$. This fact is used in Section IV.

It is possible to use B-splines of a different order in (3). In particular, zero-order B-splines will give a piecewise constant feedforward (3), while cubic B-splines will result in a twice continuously differentiable feedforward. In our experiments we used first order B-spline function $B_{\tau_u}(t)$ that has the support width $2\tau_u$ and can be written in the form

$$B_{\tau_u}(t) = \begin{cases} 0, & \text{if } |t| > \tau_u \\ 1 - |t|/\tau_u, & \text{if } |t| \le \tau_u. \end{cases} \tag{4}$$

A component $i$ of the feedforward control vector (3) and (4) is a piecewise-linear function of the form illustrated in Fig. 2 by a thick line. This function has values $u_i^j$ at the sampling instants $t_j = j\tau_u$. If the order $N_u$ of the expansion (3) is sufficiently large, it is possible to represent any continuous function $v(t)$ in the form (3) with a desired accuracy. Thus, by constraining the discussion to the feedforward of the form (3), no generality is lost.

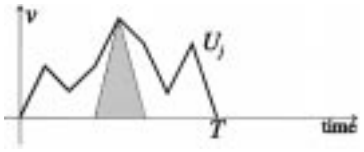


Fig. 2. Shape of the feedforward control.

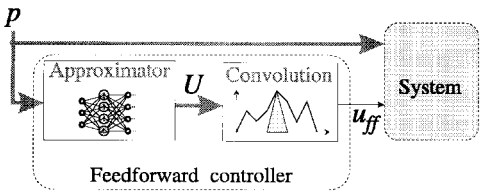


Fig. 3. Conceptual design of the feedforward controller.

Note that the control (3) can be considered as a convolution of the train of impulses with amplitudes $u^j$ applied at the times $t_j$ with the shape function (4). This presents a certain similarity with the *input shaping* method of [39].

The feedforward (3) is defined by the weights $u_j^i$, which we will collect in the *input shape vector*

$$U = [u_1^1 \cdots u_1^{N_u} \cdots u_i^j \cdots u_n^1 \cdots u_n^{N_u}]^T \in \Re^{nN_u}. \tag{5}$$

The input shape vector $U$ should be chosen to minimize the trajectory tracking error. The optimization problem for determining the vector $U$ is formulated in the next section. This optimization problem and its solution depend on the task parameter vector $p$. Let us denote this solution by $U = U_*(p)$. The feedforward controller we are designing should be able to compute vector $U_*(p)$ for an arbitrary task parameter vector in the given domain, $p \in \mathcal{P}$. Given that a dynamic model of the controlled system is unavailable, we assume the following approach to computing $U_*(p)$. First, we take a set of task parameter vectors $p_j \in \mathcal{P}, (j = 1, \cdots, N_a)$, and determine optimal shape vectors $U_*(p_j)$ for these tasks. This is done with the help of a learning control procedure described in the next section. Next, we use the set of the input/output pairs $\{p_j, U_*(p_j)\}$ to build an approximation of the mapping $U_*(p)$ for any value of the argument $p \in \mathcal{P}$. Section VI briefly describes a Radial Basis Function approximation method we used in the controller design.

Fig. 3 illustrates a conceptual design of the feedforward controller we are going to develop. Here **System** includes **Plant**, **Feedback Controller**, and **Planner** as outlined in Fig. 1. The major part of the following text is devoted to the problem of obtaining, through learning control iterations, a single optimal shape vector $U_*$ for a fixed task parameter vector $p$. We return to the approximation problem in Section VI.

### C. Input/Output Mapping and Learning Control Problem

A computationally implementable learning algorithm needs to sample the tracking error signal $e(t)$. Let us sample the error $e(t)$ with an interval $\tau_y$. The input sampling interval $\tau_u$

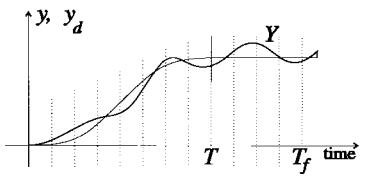


Fig. 4. Sampling of the output error.

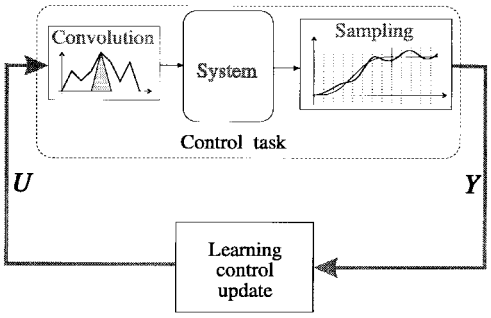


Fig. 5. Learning control concept.

(4) is assumed to be an integer multiple of the tracking error sampling interval, $\tau_y$. The reason for this will be explained later on.

In order to monitor the error just after the desired time $T$ of the task completion, we consider an observation interval $[0, T_f]$, where $T_f = \tau_y N_y \geq T$. We assume that for $T \leq t \leq T_f$ the system setpoints are kept constant, that is, $y_d(t) \equiv y_d(T)$. A vector of the sampled output values has the form

$$Y = [e_1(\theta_1) \cdots e_1(\theta_{N_y}) \cdots e_i(\theta_j) \cdots e_n(\theta_1) \cdots e_n(\theta_{N_y})]^T \in \Re^{nN_y} \tag{6}$$

where $N_y \geq N_u$ and $e_i(\theta_j)$ is the $i$-th component of the tracking error $e(t) = y(t) - y_d(t)$ at time $\theta_j = j\tau_y$. The sampling of the tracking error $e(t)$ is shown in Fig. 4. If the sampling period $\tau_y$ is sufficiently small, no information will be lost if we use the vector $Y$ instead of the continuous signal $e(t)$.

The feedforward shape vector $U$ (5), and the tracking error output vector $Y$ (6), are related by a nonlinear mapping, which depends on the task parameter vector $p$ and has the form

$$Y = S(U, p), \quad S: [\Re^{nN_u}, \Re^{N_p}] \mapsto \Re^{nN_y}. \tag{7}$$

As the controlled system is nonlinear, the mapping (7) is also nonlinear. Further, since we do not assume any knowledge of the system dynamic model, the mapping (7) is unknown. However, we are able to obtain input/output values of this mapping by performing experiments with the system for any task parameter vector $p$ (5): applying the input shape vector $U$ (4), and obtaining the sampled output vector $Y$ (7).

For a given value of the vector $p$, the optimal feedforward shape $U_*$ can be obtained without knowledge of the mapping (7) as a result of an iterative learning process. At each learning iteration, a new feedforward input shape vector $U$ is computed in order to reduce the error $Y$ obtained in the previous experiment with the system. A conceptual scheme for the learning control update is shown in Fig. 5.

## III. Learning Control for a Single Task

This and the next section are devoted to the first stage of the controller design—learning a single optimal shape vector $U_*$. We assume that the task parameter $p$ is fixed and not write dependencies on the vector $p$ explicitly.

### A. Learning Control as On-Line Optimization

For a sufficiently small output sampling interval $\tau_y$, a (weighted) norm of the sampled tracking error $Y$ gives us an accurate indication of the tracking accuracy. In order to achieve accurate tracking, we seek a feedforward shape vector $U_*$ that minimizes the following performance index

$$J(U, Y) = \rho U^T U + Y^T Y \to \min \tag{8}$$

where $\rho$ is a scalar parameter, $0 \leq \rho \ll 1$. The first term in the performance index (8) is added in accordance with the *regularization* technique [40]. The regularized optimization problem (8) is better posed than one for $\rho = 0$. The nonzero penalty on the vector $U$ in (8) prevents obtaining oscillatory inputs with high amplitude. At the same time, for a sufficiently small $\rho$, minimization of (8) provides small tracking error. Importance of introducing the regularization term in the performance index (8) will be made clear from our experimental results presented below in Section IV. A theoretical justification for the choice of the regularization parameter $\rho$ is given in the next two subsections. Some additional theoretical results on the tracking problem solution by means of the regularized performance index minimization can be found in [19].

We are going to solve the following control problem.

*Learning Control Problem:* Let us assume that $p$ in (7) is given and fixed. We assume that the mapping (7) is unknown, but we can repeatedly perform experiments with the system by applying an input vector $U$ as shown in Fig. 5 and observing the output vector $Y$. The problem is to design a learning control algorithm that iteratively updates the input vector $U$ in order to minimize the performance index (8).

The formulated problem can be considered as a standard nonlinear least-square numerical optimization problem. The only conceptual difference with the usual numerical optimization problems is that in our case each evaluation of the function (7) includes an *experiment* with a real system.

We determine an optimal feedforward control vector $U_*$ by applying the Levenberg–Marquardt algorithm to the numerical minimization of the performance index (8). This algorithm is known to work very well for nonlinear least-square problems. The Levenberg–Marquardt algorithm can be derived using an affine model for the system (7) of the form $Y = GU + Y_0$, where $G = \partial S / \partial U$ is the gain matrix of the system [gradient of the mapping (7)], and the vector $Y_0$ gives the system output for $U = 0$. The matrix $G$ plays an important role in what follows. For the nonlinear system (7), $G$ actually depends on the argument $U$ (we do not discuss dependence on $p$ in this subsection).

Let $U^{(k)}$ and $Y^{(k)}$ be input and output vectors at the learning iteration $k$; let $G^{(k)}$ be the gradient matrix at the same iteration. Let us consider the affine model of the system at step $k$, of

the form

$$Y = Y^{(k)} + G^{(k)}(U - U^{(k)}). \tag{9}$$

The Levenberg–Marquardt update can be obtained by substituting the affine model (9) into the problem (8) and imposing a constraint on the update step length: $||U^{(k+1)} - U^{(k)}||^2 \leq \delta^2$ (see [12] for more detail). This update has the form

$$U^{(k+1)} = U^{(k)} - [(\rho + r)I + G^{(k)^T} G^{(k)}]^{-1} \cdot (\rho U^{(k)} + G^{(k)^T} Y^{(k)}) \tag{10}$$

where $r \geq 0$ is a Lagrange multiplier defined by the update step bound $\delta$. Effectively, the parameter $r$ reduces the feedback gain of the learning control update (10). In the commercially available software implementations of the Levenberg–Marquardt method, the step bounding parameter $r$ in (10) is chosen at each step. This, however, requires several evaluations of the minimized function. In the learning control problems, each evaluation includes an experiment with the controlled system and, thus, has an especially large cost. Therefore, we are using the update (10) with a constant preselected parameter $r$.

Note that for $r = 0$ the update (10) becomes the Newton–Gauss update. If $N_u = N_y$ and the system (8) is invertible (and well conditioned), it is possible to set $\rho = 0$ in (10). For $r = \rho = 0$, (10) becomes the learning control update with the system inverse as the learning gain. Such update is considered in a number of papers, for instance [10].

The update rule (10) presumes that an estimate $G^{(k)}$ of the gradient (system gain) matrix $G$ is known at each step. As this matrix is, in fact, unknown, the standard practice in numerical optimization is to estimate it with a finite difference method. Such estimation is the subject of Section IV.

### *B. Robust Convergence of the Learning Control Algorithm*

Let us consider conditions for the learning control algorithm (10) to converge. We will following an established approach to the convergence analysis of the nonlinear algorithms (e.g., see [12]) and assume that the system (7) is affine in $U$ in the vicinity of the optimum. The affine model has the form

$$Y = GU + Y_0. \tag{11}$$

The Levenberg–Marquardt algorithm (10) convergence can be proven for any positive values of the parameters $\rho$ and $r$, and it is robust with respect to the error of estimating the matrix $G$**.** The sufficient condition for the convergence is given by the following theorem.

*Theorem 1:* Let us consider the update (10) of the system (11) input. The algorithm asymptotically converges for any initial condition $U^{(1)}$, if some $k_0 \geq 1$ exists such that for any $k \geq k_0$ the maximal singular value of the gradient estimation error satisfies the following inequality

$$\overline{\sigma}(G^{(k)} - G) < \frac{2\rho}{\sqrt{\rho + r}}. \tag{12}$$

*Remark 1:* Contrary to intuitive expectation, the robustness bound estimate (12) deteriorates for larger $r$, i.e., for smaller learning control gain in (10). The explanation is that a smaller learning feedback gain results in poorer disturbance rejection properties of the algorithm.

*Remark 2:* From a practical point of view, choosing $r > 0$ still brings an improvement. The reason is that in the absence of the convergence for poorly observable/controllable modes, smaller learning feedback gain will lead to slower instability development. In that case, on-line estimation of the matrix $G$ might still have a chance to bring down the error $\overline{\sigma}(G^{(k)} - G)$ and eventually achieve convergence.

*Proof:* From (11), we have $Y^{(k)} = GU^{(k)} + Y_0$. By substituting the last equality into (10) and doing some transformations, we obtain

$$U^{(k+1)} = D^{(k)}[rI + G^{(k)^T}(G^{(k)} - G)]U^{(k)} - D^{(k)} G^{(k)^T} Y_0 \tag{13}$$

$$D^{(k)} = [(\rho + r)I + G^{(k)^T} G^{(k)}]^{-1}. \tag{14}$$

One can observe from (13) that the iterations converge if

$$\overline{\sigma}(D^{(k)}[rI + G^{(k)^T}(G^{(k)} - G)]) < 1. \tag{15}$$

The sufficient condition for (15) to hold is

$$\overline{\sigma}(D^{(k)} rI) + \overline{\sigma}[D^{(k)} G^{(k)^T}(G^{(k)} - G)] < 1. \tag{16}$$

By noting that $\overline{\sigma}(D^{(k)} rI) \leq r(\rho + r)^{-1}$, we obtain the following condition of convergence

$$\overline{\sigma}(D^{(k)} G^{(k)^T}) \overline{\sigma}(G^{(k)} - G) < \frac{\rho}{\rho + r}. \tag{17}$$

Let us consider a singular value decomposition of the gradient matrix $G^{(k)} = A\Gamma B$, where $\Gamma$ is a diagonal matrix of the form $\Gamma = \operatorname{diag}\{\gamma_i\}_{i=1}^{N_u}$. Since $A$ and $B$ are orthogonal matrices, we obtain

$$[(\rho + r)I + G^{(k)^T} G^{(k)}]^{-1} G^{(k)^T} = B^T \Lambda A^T$$
$$\Lambda = \operatorname{diag}\{\lambda_i\}_{i=1}^{N_u}, \quad \lambda_i = \frac{\gamma_i}{\rho + r + \gamma_i^2}. \tag{18}$$

Since in (18) $\lambda_i$ are the singular values of the matrix in the left-hand side, (17) can be presented in the form

$$\overline{\sigma}(G^{(k)} - G)(\max \lambda_i) < \rho/(\rho + r).$$

By noting further that the inequality $x/(\rho + r + x^2) \leq 1/(2\sqrt{\rho + r})$ is valid for any real $x$, we obtain that $(\max \lambda_i) < 1/(2\sqrt{\rho + r})$, and that (17) holds if

$$\frac{1}{2\sqrt{\rho + r}} \overline{\sigma}(G^{(k)} - G) < \frac{\rho}{\rho + r}. \tag{19}$$

The inequality (19) proves (12). □

Theorem 1 shows that the convergence robustness improves for larger values of the regularization parameter $\rho$ and is absent if no regularization is performed. At the same time, increasing $\rho$ increases the steady-state tracking error $||Y||$ for the convergence achieved. The next subsection studies how the convergence point of (10) is influenced by the value of $\rho$ and imprecise knowledge of the system gain matrix $G$.

### C. Static Error

In this subsection, we keep assuming that the mapping (7) has the form (11).

First, let us assume that the matrix $G$ is known precisely and $G^{(k)} = G$ in (10). By substituting (11) into the extremum condition $dJ/dU = 0$ (8), we obtain that the optimal input $U_*$ satisfies

$$\rho U_* + G^T G U_* + G^T Y_0 = 0. \tag{20}$$

If we multiply (20) by $G$ from the left and use (11), we obtain the system output $Y_*$ at the converged state $U_*$ of the learning algorithm

$$Y_* = (I + \rho^{-1} G G^T)^{-1} Y_0. \tag{21}$$

For $\rho > 0$, the output $Y_*$ (21) represents a static tracking error of the learning control algorithm. Similarly with the previous subsection, let us consider the singular value decomposition $G = A\Gamma B, \Gamma = \operatorname{diag}\{\gamma_i\}_{i=1}^{N_u}$. The static error (21) can be written in the form

$$\begin{aligned} Y_* &= APF; \quad P = \operatorname{diag}\left\{\frac{1}{1+\gamma_i^2/\rho}\right\}_{i=1}^{N_u}, \\ F &= A^T Y_0 = \operatorname{col}\{f_i\}_{i=1}^{N_u}. \end{aligned} \tag{22}$$

Since $A$ is an orthogonal matrix, we obtain from (22) that the tracking error after the convergence of the learning algorithm can be presented in the form

$$||Y_*||^2 = \sum_{i=1}^{N_u} \frac{f_i^2}{(1+\gamma_i^2/\rho)^2}. \tag{23}$$

Note that $GG^T = A\Gamma^2 A^T$ can be considered as a "controllability Grammian" of the system. Thus, $f_i$ in (23) are projections of the tracking error in the absence of the feedforward $Y_0$ onto the controllability Grammian eigenvectors—columns of the matrix $A$. In the expression (23) for $||Y_*||$, terms with $\gamma_i > \sqrt{\rho}$ add little to the sum, since the respective weights at $f_i$ are small. For $\gamma_i < \sqrt{\rho}$, the weights tend to 1. For diminishing $\rho$, the tracking error $||Y_*||$ in (23) diminishes, since fewer eigenvalues $\gamma_i^2$ of the controllability Grammian contribute to the error.

Now let us see how the stationary solution to iterative procedure (10) deviates from $U_*$ if the estimates $G^{(k)}$ of the system gain matrix $G$ are not exact. We assume that $G^{(k)} \equiv \hat{G}$ does not change with $k$. The stationary solution to (10) satisfies the equation

$$\rho U_s + \hat{G}^T Y_s = 0. \tag{24}$$

By substituting (11) into (24), subtracting (20) from the result, and using (21), we obtain

$$U_s - U_* = (\rho I + \hat{G}^T \hat{G})^{-1} (G - \hat{G})(I + \rho^{-1} \hat{G} \hat{G}^T)^{-1} Y_0. \tag{25}$$

If the first braces in (25) contain an invertible matrix, we can use the following estimate

$$||U_s - U_*|| \le \frac{\overline{\sigma}(G - \hat{G})}{\underline{\sigma}(\rho I + \hat{G}^T \hat{G} + \hat{G}^T (G - \hat{G}))} ||Y_0||. \tag{26}$$

Let us now assume that the estimation error $G - \hat{G}$ is small and $\hat{G}^T G \approx \hat{G}^T \hat{G}$. Then, the deviation (26) of the steady state input $U_s$ from its optimal value can be estimated as

$$||U_s - U_*|| \le \rho^{-1} \overline{\sigma}(G - \hat{G}) ||Y_0||. \tag{27}$$

The estimate (27) is reasonable, since usually the matrix $\hat{G}^T \hat{G}$ has a number of small or zero eigenvalues. Due to the large multiplier $\rho^{-1}$, the steady-state control input error (27) can be significant even for a small estimation error $\hat{G} - G$. This error can be reduced by increasing the regularization parameter $\rho$.

As indicated in Section II-B, we are going to use the learned optimal inputs $U_*$ for building an approximation to the dependence $U_*(p)$. At the same time, the learning algorithm will converge to the stationary solution $U_s$, instead of $U_*$. The error in determining values of $U_*$ can be very undesirable, since it causes deterioration of the approximation accuracy.

## IV. Estimation of the System Gain Matrix $G$

As theoretically demonstrated in Section III, an accurate knowledge of the system gain matrix $G$ (gradient of the mapping (8)) is instrumental for the learning algorithm convergence. This section considers how the matrix $G$ can be estimated by a finite-difference method.

### A. Preliminaries

Let us assume that the nonlinear control system in Fig. 1 gives rise to a twice continuously differentiable mapping (7). We can write a first-order Taylor expansion for the mapping (7)

$$Y(U + \Delta U) = S(U, p) + \left.\frac{\partial S}{\partial U}\right|_U \Delta U + O(||\Delta U||^2). \tag{28}$$

By applying the two inputs, $U$ and $U + \Delta U$, to the system, where $\Delta U$ is a small input variation, denoting $\Delta Y = Y(U + \Delta U) - Y(U)$, and neglecting the second order terms in (28), we can write

$$\Delta Y = G|_U \Delta U \tag{29}$$

where the gradient matrix $G = \partial S / \partial U$ defines the input/output sensitivity of the tracking error $Y$ to a variation of the feedforward input shape $U$, in other words, the system gain. The columns of the matrix $G$ in (29) can be estimated by making $1 + nN_u$ experiments with the manipulator, varying in turn each input shape vector component—each shape parameter $u_j^i$ in (3) and (5). Estimations of such type are routinely done in the numerical minimization algorithms and are called *secant* or *finite-difference* gradient estimation methods. The direct secant estimation of $G$ requires $1 + nN_u$ experiments; this might be an unacceptably large number. Fortunately, many fewer experiments are required for a linear system, namely only $1 + n$. Let us assume that the system in Fig. 1 is linear (both the plant and feedback controller are linear). By varying the feedforward input by $\Delta v(t)$ for the same control task, we obtain a variation of the tracking error $\Delta e(t)$, which can be written in the form

$$\Delta e(t) = \mathcal{G} \Delta v(t) = \int_0^t g(t - \tau) v(\tau)\, d\tau \tag{30}$$

where $\mathcal{G}: \mathcal{L}_2(\Re^n; \Re_+) \mapsto \mathcal{L}_2(\Re^n; \Re_+)$ is the linear operator of the closed-loop system. The operator $\mathcal{G}$ can be described by the respective transfer function or the closed-loop pulse response $g(t): \Re_+ \mapsto \Re^n$.

Let us now consider a variation of the feedforward corresponding to the variation of one of the vectors $u^i$ in (3). By substituting $\Delta v(t) = \Delta u^i B_{\tau_u}(t - t_i)$ into (30), we obtain the variation of the tracking error at the sampling instant $\theta_j$, which gives components of the output vector $Y$ (6) in the form

$$\Delta e(\theta_j) = \begin{cases} \left( \displaystyle\int_{t_i - \tau_u}^{\theta_j} g(\theta_j - \tau) B_{\tau_u}(\tau - t_i)\, d\tau \right) \Delta u^i \\ \qquad 0, \quad \text{for } \theta_j \geq t_i - \tau_u \\ \qquad 0, \quad \text{for } \theta_j < t_i - \tau_u. \end{cases} \tag{31}$$

The matrix $G$ relates variations $\Delta u^i$ of the respective components of vector $U$ and the variation $\Delta e(\theta_j)$ of the components of the vector $Y$ in accordance with (29). It can be seen that the integral in (31) is defined by the difference $\theta_j - t_i$. Thus, if the input sampling interval $\tau_u$ is an integer multiple of the output sampling interval $\tau_y$, then the matrix $G$ is a block-Toeplitz matrix. Each Toeplitz block of this matrix is completely defined by its first column.

As mentioned earlier, for a general nonlinear system, the finite-difference estimation of the matrix $G$ in the learning algorithm (10) might demand excessive resources. The analysis presented in Section III-B shows that the iterative learning algorithm has a certain degree of robustness to the error in estimating $G$. In many tracking problems system dynamics are linearized with a high-gain feedback. In such cases, the system nonlinearity can be neglected when estimating $G$, and $G$ can be assumed to have a block-Toeplitz form. In our experiments described in Section IV, we found that for $G$ estimated as a block-Toeplitz matrix, the learning algorithm converges well, despite the approximation error caused by the system nonlinearity.

### *B. Fitting a Toeplitz Matrix as the Gain Matrix G*

In this subsection, we suppose that the dynamics of the system are linear. We also assume that the system input and output are sampled with the same rate, so that in (3) and (6) $\tau_u = \tau_y$ and $t_j = \theta_j$ for $j = 1, \cdots, N_u$. In order to describe the structure of the matrix $G$, let us partition vectors $\Delta Y, \Delta U$, and matrix $G$ in (29) into the blocks corresponding to different inputs and outputs of the system as follows:

$$\underbrace{\Delta Y}_{nN_y \times 1} = \begin{bmatrix} \Delta Y^1 \\ \vdots \\ \Delta Y^n \end{bmatrix}, \quad \underbrace{G}_{nN_y \times nN_u} = \begin{bmatrix} G^{1,1} & \cdots & G^{n,1} \\ \vdots & \vdots & \vdots \\ G^{1,n} & \cdots & G^{n,n} \end{bmatrix}$$
$$\underbrace{\Delta U}_{nN_u \times 1} = \begin{bmatrix} \Delta U^1 \\ \vdots \\ U^n \end{bmatrix} \tag{32}$$

where $G^{j,i}$ is a $N_y \times N_u$ matrix which describes the sensitivity of the $j$th output, $\Delta Y^j$, to the $i$th input $\Delta U^i$. For a linear system, $G^{j,i}$ is a Toeplitz matrix. Let us consider a structure of the block $G^{j,i}$ in more detail. By using (31) and noting that $t_j = \theta_j$, we can write the block $G^{j,i}$ in the following Toeplitz form

$$\Delta Y^i = G^{j,i} \Delta U^j \tag{33}$$
$$G^{j,i} = \underbrace{\begin{bmatrix} g_1^{j,i} & 0 & \cdots & 0 \\ g_2^{j,i} & g_1^{j,i} & & 0 \\ \vdots & & \ddots & \vdots \\ g_{N_y}^{j,i} & g_{N_y-1}^{j,i} & \cdots & g_{N_y-N_u+1}^{j,i} \end{bmatrix}}_{(N_y \times N_u)}. \tag{34}$$

We will denote the first column of the Toeplitz matrix $G^{j,i}$ by $g^{j,i}$. The pulse response vector $g^{j,i}$ completely defines the matrix block $G^{j,i}$ and is defined by the integral in (31). The form of the matrix $G^{j,i}$ will be somewhat different for a different ratio of the input and output sampling periods.

In (31) components of vector $\Delta Y^i$ are related linearly to the components of the vector $g^{j,i}$, therefore we can rewrite (31) as

$$\Delta Y^i = X(\Delta U^j) g^{j,i} \tag{35}$$

where $X(\Delta U^j)$ turns out to be a Toeplitz matrix of the form

$$\underbrace{X(\Delta U^j)}_{N_y \times N_y} = \begin{bmatrix} \Delta u_1^j & 0 & \cdots & 0 & 0 \\ \Delta u_2^j & \Delta u_1^j & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & & \cdots & \Delta u_1^j & 0 \\ 0 & 0 & & \cdots & \Delta u_2^j & \Delta u_1^j \end{bmatrix}. \tag{36}$$

By putting together relations of the form (35) for all components of the tracking error output, we obtain

$$\Delta Y = Z(\Delta U^j) g^j, \tag{37}$$
$$\underbrace{\Delta Y}_{nN_y \times 1} = \begin{bmatrix} \Delta Y^1 \\ \vdots \\ \Delta Y^n \end{bmatrix},$$
$$\underbrace{Z(\Delta U^j)}_{nN_y \times nN_y} = \begin{bmatrix} X(\Delta U^j) & & 0 \\ & \ddots & \\ 0 & & X(\Delta U^j) \end{bmatrix},$$
$$\underbrace{g^j}_{nN_y \times 1} = \begin{bmatrix} g^{j,1} \\ \vdots \\ g^{j,n} \end{bmatrix}. \tag{38}$$

Now, by counting influences of all the input components, we obtain

$$\Delta Y = \sum_{l=1}^{n} Z(\Delta U^l) g^l = W(\Delta U) g. \tag{39}$$

Matrix $W$ and vector $g$ in (39) have the form

$$\underbrace{W(\Delta U)}_{nN_y \times n^2 N_y} = [Z(\Delta U^1) \cdots Z(\Delta U^n)]$$
$$\underbrace{g}_{n^2 N_y \times 1} = [g^{1^T} \cdots g^{n^T}]^T \tag{40}$$

where the blocks $Z(\Delta U^j)$ are described by (38), and the vector $g$ contains the responses of all outputs to a unit variation of the first shape functions for all input components. We will further use representation (39) to derive update for the estimate

of the vector $g$, which defines the block-Toeplitz matrix $G$ from the input/output data $\Delta U$ and $\Delta Y$. We further call $g$ the *basis vector* of the block-Toeplitz matrix $G$.

### C. Recurrent Update of the Toeplitz Sensitivity Matrix

Instead of $nN_u + 1$ experiments needed to obtain the gain matrix $G$ for a nonlinear system, the basis vector $g$ (40) defining a block-Toeplitz gain matrix can be computed after only $n+1$ experiments, which is a much smaller number. For instance, these $n+1$ experiments can be done by applying in turn variations of the vectors $U^j$ in (32) and using (37) to determine the vectors $g^j$ in (39). Note that, according to (38) and (35), the matrix $Z(\Delta U^j)$ in (37) is always invertible provided that the first component of the vector $U^j$ is nonzero, $u_1^j \neq 0$.

We would like, however, to update an estimate of the block-Toeplitz matrix $G$ recurrently, on-line, while performing the learning control iterations. One update algorithm to do this was proposed in [18], and it performs a secant update of the estimates of the basis vector $g$ of the block-Toeplitz model for the matrix $G$ by a projection method. An efficient recurrent update of the vector $g$ (40) would allow to achieve the best possible fit of the Toeplitz matrix to the experimental data. Such update algorithm, which is a modification of the algorithm of [18], is explained in more detail below.

Let us note that (39) is a linear regression equation with the parameter vector $g$ and the regressor matrix $W(\Delta U)$. The system (39) is under-determined, since the vector $g$ has the size $n^2N_y$, while the vector $\Delta Y$ is of the size $nN_y$. In the course of the learning control iterations (10), the input vector $U^{(k)}$ and the output vector $Y^{(k)}$ are changing at each step. An estimate $g^{(k)}$ of the vector (40) should relate the variations of the input and output vectors at step $k$ in accordance with (39), so that

$$Y^{(k)} - Y^{(k-1)} = W(U^{(k)} - U^{(k-1)})g^{(k)}. \tag{41}$$

We apply one of the most often used methods for update of linear regression parameter estimates, the *projection update* [15]. The projection update gives a solution to (41) with a least norm step from $g^{(k-1)}$ to $g^{(k)}$. At each step of the learning procedure, the variation $U^{(k)} - U^{(k-1)}$ of feedforward input (10) is used to build matrix $W^{(k)} = W(U^{(k)} - U^{(k-1)})$ in accordance with (36), (38), and (40). An estimate $g^{(k)}$ of the basis vector (40) is updated as follows

$$g^{(k+1)} = g^{(k)} - \frac{W^{(k)^T}(W^{(k)}g^{(k)} - Y^{(k)} + Y^{(k-1)})}{\alpha + ||W^{(k)}||_F^2} \tag{42}$$

where $||\cdot||_F$ denotes the Frobenius norm of the matrix, and $\alpha$ is a parameter used to avoid division by zero and to make the update more robust for a small norm $||W^{(k)}||_F = ||W(U^{(k)} - U^{(k-1)})||_F$ which corresponds to a small variation $U^{(k)} - U^{(k-1)}$ of the input.

The algorithm (42) has the following property

*Theorem 2:* Let the mapping (7) be defined by a linear system (30), that is, for some $g = \overline{g}$ (39) is valid for any variations of the input and the output. Then, for the recurrent estimator (42) the error $||\overline{g} - g^{(k)}||$ is a monotone nonincreasing sequence.

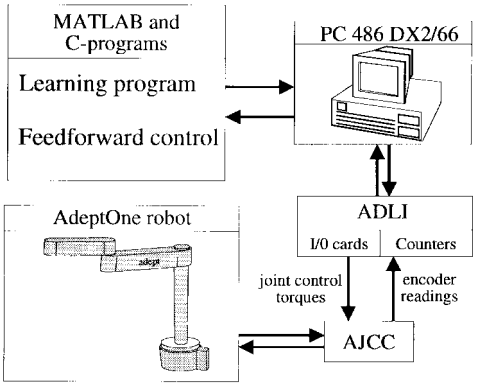


Fig. 6. Overview of the experimental setup.

*Remark 3:* To achieve the asymptotic convergence of the estimates $g^{(k)}$ to $\overline{g}$, the matrices $W^{(k)^T}$ should span the entire space $\Re^{n^2N_y}$ (persistensy of the excitation condition should hold). Following the standard practice of adaptive control, this can be achieved by adding a small self-excitation signal to the input sequence $U^{(k)}$.

*Remark 4:* The learning control algorithm (10) together with the update of the system gain estimate (42) give an *adaptive* algorithm of learning control. Since this algorithm uses a linear model of a nonlinear system, in general, only *local* convergence of the algorithm can be established. Such convergence results are provided by Theorems 1 and 2 that prove convergence for the linearization if the initial error of estimating $\overline{g}$ is not too large.

*Proof:* The proof of Theorem 2 follows from the straightforward application of the standard results of [15] and is as follows. By assumption, in (42) $Y^{(k)} - Y^{(k-1)} = W^{(k)}\overline{g}$. By subtracting $\overline{g}$ from both sides of (42), we obtain

$$g^{(k+1)} - \overline{g} = \left[I - \frac{W^{(k)^T}W^{(k)}}{\alpha + ||W^{(k)}||_F^2}\right](g^{(k)} - \overline{g}). \tag{43}$$

It can be checked that the singular values of the matrix in the square braces in (43) are less or equal to unity, which proves the result. □

## V. Experiments in Learning Control

The described approach was experimentally applied to the feedforward control of trajectory tracking in fast motions of a direct-drive manipulator. Since the controlled motions were fast, the influence of the dynamics nonlinearity was strong. A detailed account of the experiments as well as the simulation results can be found in the technical report [42].

### A. Experimental Setup

An overview of the experimental system is presented in Fig. 6. The control system structure can be very well described by Fig. 1. Let us start from the description of the controlled *Plant*.

The experiments were performed on a direct-drive AdeptOne robot. The robot has four degrees of freedom and a SCARA type arm configuration. In our experiments, we used only Joints 1 and 2 that move the arm in the horizontal plane.

These joints are powered by large Megatorque three-phase variable reluctance motors. With the use of direct drives and proprietary motor control components, the AdeptOne robot combines a high speed with considerable working envelope (a cirle with the 0.8 m radius) and load carrying capacity (6 kg).

Both joints are equipped with incremental joint encoders that have resolutions of 460 800 and 115 200 counts per revolution. The control system of the AdeptOne robot was retrofitted in the Robotics and Automation Laboratory, University of Toronto, as described in [14], to allow for an accurate torque control of the robot motors. In our experiments, the AdeptOne robot is controlled with a custom made AdeptOne Data Link Interface (ADLI) card and a PC-486/DX2-66. A C routine is used to read the joint status and position information from a Dual-Port RAM on the ADLI card. This card reads from all AdeptOne Joint Control Cards and updates the contents of the Dual-Port RAM every millisecond. Another C routine writes joint control data to Dual-Port RAM on ADLI card, and the ADLI processor stores the new data in its internal memory. This data is transferred to all Joint Control Cards every millisecond.

In our experiments, all low-level control functions needed for trajectory tracking were programmed in C. The learning control algorithms and the RBF approximation computations were programmed in MATLAB. To move the manipulator, the MATLAB program passes the planned motion trajectory and the parameters of the feedforward input shape (vector $U$) to a C program, which controls robot motion in real-time. After the completion of the motion, the C program passes the sampled tracking error data (vector $Y$) back to the MATLAB program.

As the **Feedback Controller**, we used a standard PID controller with the proportional (position) gains $K_{p,1} = 45$ N/m and $K_{d,1} = 17$ Nms/rad; derivative (velocity) gains $K_{p,2} = 60$ Nm/rad and $K_{d,2} = 25$ Nms/rad; and the integral gains $K_{i,1} = 10$ Nms$^2$/rad and $K_{i,2} = 5$ Nms$^2$/rad.

In the experiments, the desired motion time was $T = 1.25$ s and the full observation time horizon, $T_f = 2.125$ s. The **Planner** computes the desired trajectory as a third order polynomial in time, along a straight line in the joint angle space. For defining the feedforward shape and output vectors, the input and output sampling intervals are both $\tau_u = \tau_y = 78$ ms. Thus, $N_u = 15$ input sampling instants $t_j$ are considered in (3), and the vector $U$ has dimension 30. The tracking error is sampled at $N_y = 27$ instants in the observation interval $[0, T_f]$, so the vector $Y$ has dimension 54. The feedforward input shape vector $U$ (5) defines feedforward torque in Newton meters, and the output vector $Y$ (6) gives a tracking error in radians.

### B. Results for Learning a Single Motion

A detailed account of our experiments with the described manipulator system can be found in the technical report [42]. In this paper, we necessarily briefly discuss the main experimental results and the gained insights.

The experiments show that the Coulomb friction has a profound influence on the performance of the elbow joint. In fact, the friction reaches up to 30% of the maximal torque. The nonlinearity caused by the Coulomb friction can be especially bad for the proposed approach, since it causes the mapping (7) to be nonsmooth, which makes determining the gain matrix $G$ a problem. Yet, in our experiments, manipulator joints move in one direction till they reach the desired final position. During this motion, the friction torques do not change with small variations of the feedforward input and have no negative impact on the estimation of the system gain $G$. Unlike that, the transient motion on the interval $[T, T_f]$ creates some problems for the algorithm since the joint velocities reduce to zero or change their sign on this interval.

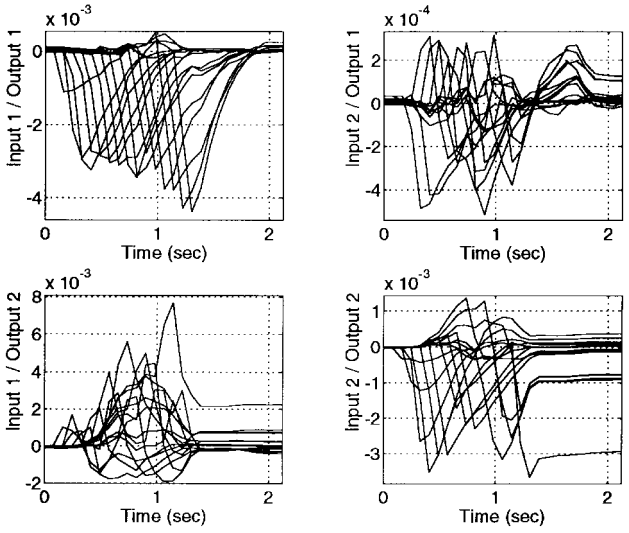


Fig. 7. Experimentally determined pulse responses representing columns of the system gain matrix $G$.

The representative experimental results have been obtained for the manipulator moving from the initial configuration with the joint angles $\phi_{1,i} = -1.65$ rad and $\phi_{2,i} = -1.7$ rad to the final configuration with angles $\phi_{1,f} = 0.05$ rad and $\phi_{2,f} = 0.1$ rad. Zero joint angles correspond to the maximally extended manipulator. We tried different values of the regularization parameter $\rho$ in (8), (10). In all presented results, the step bounding parameter $r$ in (10) was empirically chosen to be $r = 3\rho$.

First, the gain matrix $G$ has been directly estimated from (29) by varying in turn each of 30 components of the vector $U$. Fig. 7 shows the pulse responses corresponding to the columns of $G$. The responses show a strong influence of the Coulomb friction. Thus, $G$ can only very roughly be considered as a block-Toeplitz matrix. Many of the pulse responses comprising columns of $G$ are not quite smooth and regularly shaped. The iterative learning algorithm (10) was first tried using the obtained matrix $G$.

The results for the regularization parameter $\rho = 4 \cdot 10^{-5}$ are shown in Fig. 8. The two top plots in Fig. 8 show the learned feedforward inputs after two (dashed line), four (dashed-dotted), and six (solid line) steps. The two bottom plots show the respective tracking errors for Joint 1 (shoulder) and Joint 2 (elbow). The dotted lines show the tracking errors at step 0, for zero feedforward. About a tenfold improvement of the tracking accuracy is achieved at the iteration 6. Note that the learned feedforward inputs in Fig. 8 exhibit oscillations which are related to the irregularity of the pulse responses comprising $G$. By increasing the parameter $\rho$, it is possible to put a higher

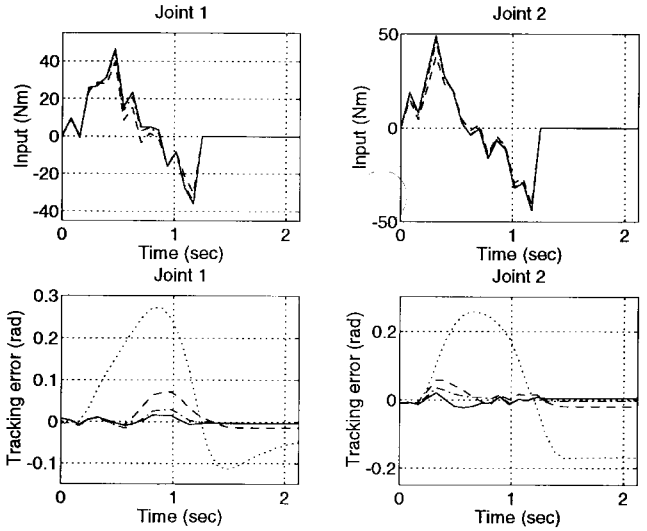


Fig. 8. Experimental results for learning control with the directly measured gain matrix $G$. Iterations 0, 2, 4, and 6 are marked by dotted, dashed, dashed-dotted, and solid lines, respectively.

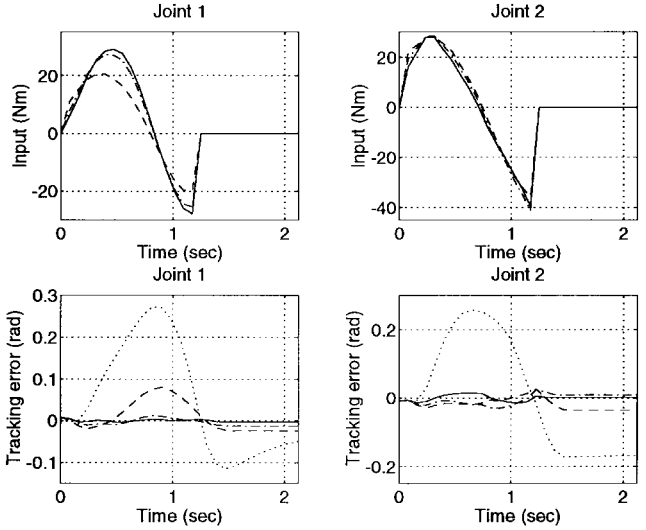


Fig. 9. Experimental results for learning control with a block-Toeplitz approximation for the gain matrix $G$. Iterations 0, 2, 4, and 6 are marked by dotted, dashed, dashed-dotted, and solid lines, respectively.

penalty on the oscillations and obtain a smoother input, but this will also increase the tracking error.

Next, a block Toeplitz model (32), (33), (34) for $G$ was estimated by applying two variations of the input vector components $\Delta U^1$ and $\Delta U^2$, i.e., by varying feedforward first for Joint 1, then for Joint 2. The components $g^1$ and $g^2$ of the basis vector $g$ (40) of the matrix $G$ were estimated by using (37). Thus estimated pulse responses are much smoother than the directly measured ones, probably because their estimation includes certain averaging of the data. The results of the learning control experiments using the obtained block-Toeplitz estimate of $G$ are presented in Fig. 9; the format is the same as in Fig. 8. No update of matrix $G$ was performed in these experiments, and the regularization parameter was $\rho = 4 \cdot 10^{-5}$. A sixteenfold improvement of the tracking error is achieved in Fig. 9 compared to the tenfold improvement in Fig. 8. Though the block-Toeplitz approximation of matrix $G$ has an inherent error for the experimental nonlinear system, the performance improvement could be explained by the fact that some of the friction-caused error in estimation of $G$ was filtered out.

Unlike Fig. 8, feedforward in Fig. 9 does not exhibit any oscillations. Diminishing the regularization parameter $\rho$ to $4 \cdot 10^{-6}$ for the same block-Toeplitz matrix $G$ slightly deteriorates tracking, while $\rho = 4 \cdot 10^{-7}$ results in the feedforward oscillations similar to those in Fig. 8 and two to three times increase of the tracking error. Therefore, we used $\rho = 4 \cdot 10^{-5}$ in the subsequent experiments.

Further improvement in the algorithm convergence and the tracking accuracy can be achieved by adaptively updating the estimate of the system gain matrix $G$ as described in Section IV-C. The adaptive update allows us to obtain a twentyfold improvement of the tracking error in six iterations. If one takes into account the measurement errors, the adverse effect of the Coulomb friction, and the input torqued resolution of the system (0.5 Nm), the obtained tracking accuracy is close to the achievable limit. Visually, the results look similar to Fig. 9.

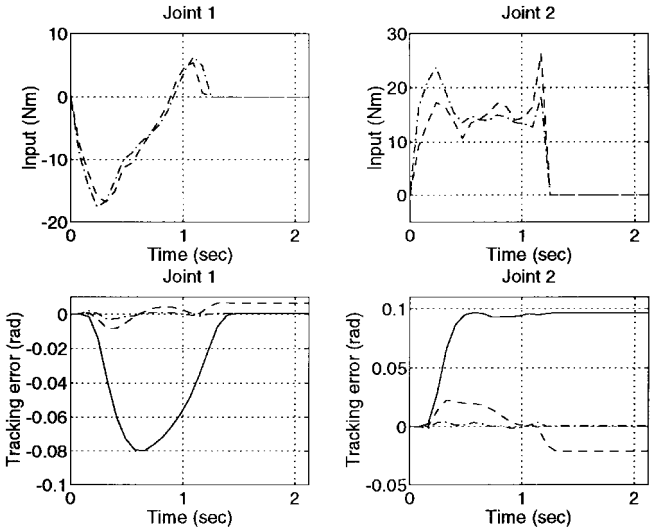


Fig. 10. Experimental results for learning control with adaptive update of the matrix $G$ demonstrating compensation of the friction caused error. Iterations 0, 2, and 6 are marked by solid, dashed, and dashed-dotted lines.

The experiments show that the feedforward input learned with an adaptive update of the matrix $G$ is able to cope with the static error caused by the Coulomb friction. This is illustrated by Fig. 10, which shows the results for another trajectory than Figs. 8 and 9. In Fig. 10, the initial joint angles are $\phi_{1,i} = 0$ rad and $\phi_{2,i} = -0.65$ rad, and final angles are $\phi_{1,f} = -1.5$ rad and $\phi_{2,f} = -0.2$ rad. In the absence of the feedforward, the error for the Joint 2 (solid line) exhibits a strong friction influence. This error is successfully compensated in the course of the learning.

## VI. Design and Experimental Verification of the Feedforward Controller

### A. Radial Basis Function Approximation

As described in the previous subsection, we can learn optimal feedforward input shape vectors $U_{*,k} = U_\star(Q_k)$ for

certain (discrete) values $Q_k$ of the task parameter vectors $p$. Given $K$ pairs of vectors $Q_k \in \mathcal{P}$ and $U_{*,k}$ as $\{Q_k, U_{*,k} = f(Q_k)\}_{k=1}^K$, it is possible to find an approximation for the smooth mapping $U = f(p)$ over the domain $\mathcal{P}$ of the task parameter vector $p$.

In this paper, we follow the approach used in [23] and [22] and approximate the nonlinear mapping $U_*(p)$ with the help of a RBF network. RBF approximation provides high accuracy, fast training (identification), and is computationally and algorithmically simple. In many applications, the RBF nework approximation has superior accuracy and training time compared to Mulilayered Perceptron Networks used in many ANN-based systems, e.g., see [6], [9], and [21]. Radial Basis Function networks were recently acknowledged [35], [36], [38] to possess a few advantageous theoretical properties, such as low pass spatial filtering of the approximated function. For more information and references on the RBF approximation see [13], [35], [36].

Standard RBF approximation results (e.g., [13], [35], [36]) are formulated for scalar-valued functions. In this work, we used an RBF-network approximation of the *vector-valued* function $U_\star(p)$. For any value of the task parameter vector $p$, RBF approximation of this vector-valued function can be written in the following form:

$$\hat{U}_\star(p) = \sum_{j=1}^{K} V_j h(||p - Q_j||) \tag{44}$$

where $V_j$ are the vector weights of the RBF network, $h(\cdot)$ is a given function of the radius $||p - Q_j||$ (hence, the name *radial function*), and $Q_j$ are the RBF centers. Several forms of the radial functions are commonly used. In the experiments, we used the Gaussian radial function $h(r) = \exp(-r^2/D^2)$. A standard choice of the Gaussian width $D$ is 1.6 times the distance between the RBF centres $Q_j$.

We compute the vector weights $V_j$ of the RBF network (44) from the *exact interpolation conditions*, which means that the approximation error is zero for each pair $\{Q_j, U_{*,j}\}$ in the training set. By substituting $p = Q_j$ into the right-hand side of (44) and writing the conditions $\hat{U}_\star(Q_i) = U_{*,i}$ for $i = 1, \cdots, K$ in the matrix form, we obtain the exact interpolation conditions in the form:

$$\begin{aligned} &\boldsymbol{U} = H\boldsymbol{V}; \quad H = \{h(||Q_i - Q_j||)\}_{i,j=1}^K \\ &\boldsymbol{U} = [U_{*,1}, \cdots, U_{*,K}], \quad \boldsymbol{V} = [V_1, \cdots, V_K]. \end{aligned} \tag{45}$$

The results obtained in [32] prove that the interpolation matrix $H$ in (45) is invertible for the commonly used radial functions (e.g., for the Gaussian function) and distinct node centres $Q_j$. Therefore, the weights $V_j$ of the network (44) can be computed as $V_j = \Sigma_{k=1}^K w_{jk} U_{*,k}$, where $w_{jk}$ are the entries of the matrix $H^{-1}$.

### *B. Experimental Results in Approximating Feedforward for Arbitrary Motions*

The feedforward control of the manipulator depends only on the change of the first joint angle, not on its absolute position.

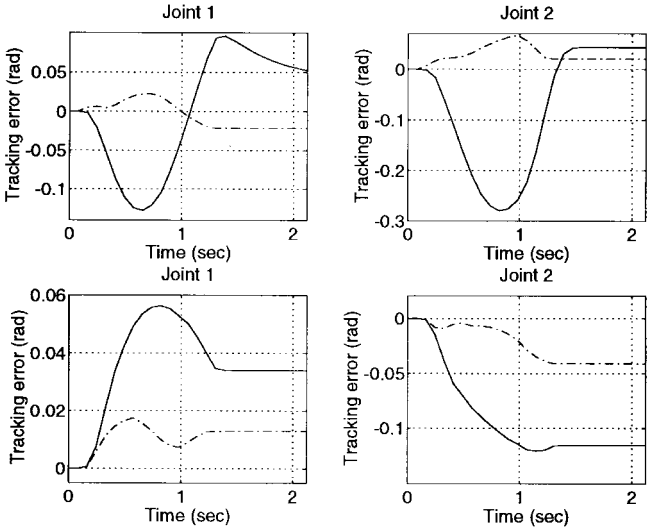


Fig. 11. Tracking errors for two test tasks. Without feedforward (dashed line) and with feedforward computed by the RBF-network (solid line).

Therefore, the task parameter vector $p$ has three components

$$p = [\phi_{1,f} - \phi_{1,i} \quad \phi_{2,i} \quad \phi_{2,f}]^T \tag{46}$$

where the subscripts $i$ and $f$ denote the 'initial' and 'final' configurations, and the first subscript gives the number of the joint.

We consider the following task parameter domain, which covers most of the robot workspace

$$\begin{aligned} \mathcal{P} = \{p \in \mathcal{P}: &-1.3 \le \phi_{1,f} - \phi_{1,i} \le 1.3 \\ &-1.5 \le \phi_{2,i}, \phi_{2,f} \le -0.2\}. \end{aligned} \tag{47}$$

The task parameter vector $p$ (46) has dimension three—much less than the dimension of the input vector $U$, which is 30. Therefore, the use of RBF network for approximation of mapping $p \mapsto U_*$ is especially attractive, since its computational complexity is only *linear* in dimension $nN_u$ of the vector $U$ [21].

The domain (47) is a 3-D cube in the task parameter space. We obtained training pairs $\{U_{*,k}, Q_k\}$ by applying the learning procedure as described in Section V, for the trajectories corresponding to the parameter vectors $p = Q_k$. The points $Q_k$ were placed on a uniform grid $5 \times 3 \times 3$ in the cube (47). We made six learning iterations for obtaining each vector $U_{*,k}$. Altogether, obtaining all 45 vectors $U_{*,k}$ took 315 tracking motions of the manipulator, which were completed in about two hours of continuously running the system.

The learned training pairs $\{U_{*,k}, Q_k\}$ were used to build an RBF network approximation of the dependence $U_*(p)$ as described in the previous subsection. The accuracy of tracking was checked for a number of test tasks within the set $\mathcal{P}$ (47) in order to validate the performance of the designed RBF network based feedforward controller. For the validation, we used the task parameter vectors maximally remote from all neighboring RBF centres $Q_k$. These test values of the parameter vector make a grid $2 \times 2 \times 4$ which is shifted halfway from the grid of RBF centres in the cube (47). Fig. 11 illustrates the experimental results obtained with the

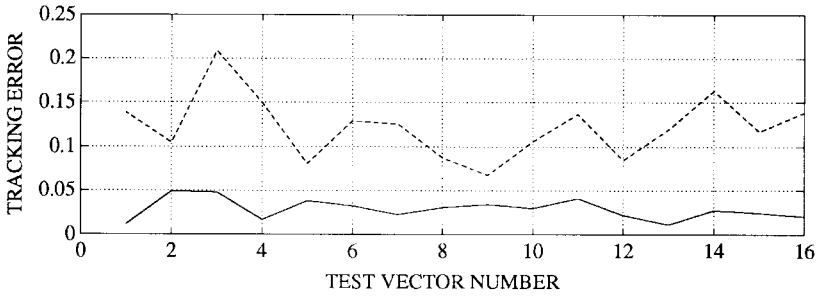


Fig. 12. Improvement of mean square tracking errors for the test tasks. Without feedforward—solid, with feedforward computed by the RBF network—dashed.

approximated feedforward for two test values $p_1$ and $p_2$ of the task parameter vector (46). These test vectors are as follows:

$$\begin{aligned} p_1\colon \phi_{1,f} - \phi_{1,i} &= -0.975 \text{ rad} \\ \phi_{2,i} &= -0.525 \text{ rad}, \quad \phi_{2,f} = -1.175 \text{ rad}, \\ p_2\colon \phi_{1,f} - \phi_{1,i} &= -0.325 \text{ rad}, \\ \phi_{2,i} &= -0.525 \text{ rad}, \quad \phi_{2,f} = -1.175 \text{ rad}. \end{aligned} \tag{48}$$

Fig. 12 summarizes the mean square tracking error results for the chosen test motions without and with the feedforward computed by the Radial Basis Function network. The task vectors (48), for which detailed tracking results are reported, correspond to test vector numbers 3 and 6 on the argument axis in Fig. 12. At least a fourfold average improvement is achieved. The improvement is less significant for motions with small amplitude that have smaller initial error. For such motions, compensation of the Coulomb friction can account for up to 80% of the feedforward drive torque.

Unlike the iterative learning control used in Section V-B, the RBF approximation-based controller tested in this section computes the feedforward for any task parameters without a need of completing several repetition of the same motion beforehand. The tracking accuracy achieved for each of the test motions using the RBF approximation of the feedforward control can be further improved by the factor of 4 to 6 by making one or two learning control iterations for this particular motion. As discussed in Section V-B, by repeating learning control iterations for a given motion it is possible obtain tracking error close to the achievable limit. At the same time, tracking for the approximated control is subject to the approximation error, which should diminish for finer grid of the approximation nodes. Yet, it might be undesirable to have too many approximation nodes, because of the more training required. From this point of view, the presented results show that even for a relatively coarse approximation grid, tracking accuracy improvement is significant.

The RBF network was trained on the feedforward vectors obtained with the regularization parameter $\rho = 4 \cdot 10^{-5}$. The report [42] also presents results obtained for $\rho = 4 \cdot 10^{-6}$. Though the tracking accuracy achieved in the learning iterations for each of the RBF centres $Q_j$ was almost the same in both cases, the error for the test motions is up to four times larger with $\rho = 4 \cdot 10^{-6}$. The reason, as the estimate (27) shows, is that a smaller $\rho$ generally leads to a larger learning control error in obtaining $U_*$. These errors act like disturbances in the training data $\{U_{*,k}, Q_k\}$ spoiling the accuracy of the RBF approximation.

For the AdeptOne robot used in experiments, the Coulomb friction is up to 30% of the maximal drive torque. The friction has adverse effect on the RBF approximation, as the input/output mapping of the system with friction is not smooth. With a smaller friction, the proposed approach would yield even more significant improvement of the tracking accuracy.

## VII. Conclusions

In this paper, we have presented and experimentally demonstrated feasibility of a novel model-free approach to the design of a nonlinear feedforward controller for a trajectory tracking task. The design is based on the nonlinear approximation of the feedforward dependence on the control task parameters. These task parameters comprise initial and final setpoints of the system and define the trajectory to be tracked.

The proposed design has serious advantages compared to the approaches based on a model-free (e.g., neural network) approximation of the controlled system dynamics. The most important advantage is that for our approach, the approximation complexity is defined by the dimension of the task parameter vector, which is usually much smaller than the dimension of the state-space covered by the system dynamics approximation. The pay for better performance is that the task parameter approximation is somewhat less general, since it works only for a certain family of tasks. This limitation is quite tolerable for many practical applications, such as one we have considered.

To make the task parameter approximation practical, we have presented an efficient learning control algorithm for obtaining initial data for the approximation. The learning algorithm uses the on-line Levenberg–Marquardt minimization of the regularized quadratic performance index to provide fast and robust convergence of the optimal feedforward input shape. The algorithm convergence is enhanced by the presented novel technique for the adaptive estimation of the system gain through the estimation of the system pulse response. We have experimentally shown that the learning algorithm converges in few iterations, faster than many other algorithms.

We have experimentally demonstrated the high performance of the developed algorithms in the difficult problem of tracking fast direct-drive manipulator motions. Both the learning algorithm used for obtaining training examples and the RBF network approximation worked very well in the experiment. Not only the algorithms cope well with the strongly nonlinear

system dynamics, they also compenste for the large Coulomb friction plaguing the experimental manipulator.

Experimental implementation of the approach has been possible with moderate computational resources and training (system identification) time. Therefore, the results of this paper demonstrate that the approach is very well suited and ready for industrial applications in Robotics. The presented results also open an avenue for other applications of the approach, such as automotive, process control, etc.

## REFERENCES


[1] C. G. Aboaf, C. G. Atkeson, and D. J. Reikensmeyer, "Task-level robot learning," in *IEEE Int. Conf. Robot. Automat.*, Philadelphia, PA, Apr. 1988, pp. 1311–1312.

[2] C. H. An, C. G. Atkeson, and J. M. Hollerbach, "Model-based control of a direct arm. Part II: Control," in *IEEE Int. Conf. Robot. Automation*, Philadelphia, PA, Apr. 1988, pp. 1386–1391.

[3] S. Arimoto, "Learning control theory for robotic motion," *Int. J. Adaptive Contr. Signal Processing*, vol. 4, pp. 453–564, 1990.

[4] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *J. Robot. Syst.*, vol. 1, pp. 123–140, 1984.

[5] C. G. Atkeson and J. McIntyre, "Robot trajectory learning through practice," in *IEEE Int. Conf. Robot. Automat.*, San Francisco, CA, Apr. 1986, pp. 1737–1742.

[6] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, no. 2, pp. 321–355, 1988.

[7] P. Bondi, G. Casalino, and L. Gambardella, "On the iterative learning control of robot manipulators," *IEEE Trans. Robot. Automat.*, vol. 5, 1987.

[8] J. J. Craig, "Adaptive control of manipulators through repeated trials," in *Amer. Contr. Conf.*, San Diego, CA, June 1984, pp. 1566–1573.

[9] S. Chen and S. A. Billings, "Neural networks for nonlinear dynamic system modeling and identifications," *Int. J. Contr.*, vol. 56, no. 2, pp. 319–346, 1992.

[10] W. Cheng and J. T. Wen, "Output trajectory tracking based on feedforward learning," in *Amer. Contr. Conf.*, Baltimore, MD, June 1994, pp. 1747–1751.

[11] W. Cheng, J. T. Wen, D. Hughes, "Experimental results of a learning controller applied to a tip tracking of a flexible beam," in *Amer. Contr. Conf.*, San Francisco, CA, June 1993, pp. 987–991.

[12] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[13] N. Dyn, "Interpolation of scattered data by radial functions," L. L. Schumaker, C. K. Chui, and F. I. Utreras Eds., *Topics in Multivariable Approximation*. Boston, MA: Academic, 1987. pp. 47–61.

[14] A. A. Goldenberg, I. Laniado, P. Kuzan, and C. Zhou, "Control of switched reluctance motor torque for force control applications," *IEEE Trans. Ind. Electron.*, vol. 41, pp. 461–466, 1994.

[15] G. C. Goodwin and K. S. Sin, "*Adaptive Filtering, Prediction and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[16] D. M. Gorinevsky, "Learning and approximation in database for feedforward control of flexible-joint manipulator," in *ICAR'91: 5th Int. Conf. Adv. Robot.*, Pisa, Italy, June 1991, pp. 688–692.

[17] ———, "Direct learning of feedforward control for manipulator path tracking," in *Proc. 1992 IEEE Intern. Symp. Intell. Contr.*, Glasgow, UK, Aug. 1992, pp. 42–47.

[18] ———, "Experiments in direct learning of feedforward control for manipulator path tracking," *Robotersysteme*, vol. 8, pp. 139–147, 1992.

[19] ———, "On the approximate inversion of linear system and quadratic-optimal control," *J. Comput. Syst. Sci. Int.*., vol. 30, no. 6, pp. 16–23, 1992.

[20] ———, "Modeling of direct motor program learning in fast human arm motions," *Biol. Cybern.*, vol. 69, pp. 219–228, 1993.

[21] D. M. Gorinevsky and T. H. Connolly, "Comparison of some neural network and scattered data approximations: The inverse manipulator kinematics example," *Neural Comput.*, vol. 6, no. 3, pp. 519–540.

[22] D. Gorinevsky, A. Kapitanovsky, and A. A. Goldenberg, "Radial basis function network architecture for nonholonomic motion planning and control of free-flying manipulators," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 491–496, 1996.

[23] ———, "Neural network architecture for trajectory generation and control of automated car parking," *IEEE Trans. Contr. Syst. Technol.*, vol. 4, pp. 50–56, 1996.

[24] K. Gugliemo and N. Sadeh, "Experimental evaluation of a new robot learning controller," in *Proc. 1991 IEEE Int. Conf. Robot. Automat.*, Sacramento, CA, Apr. 1991, pp. 734–739.

[25] R. Horowitz, W. Messner, and J. B. Moore, "Exponential convergence of a learning controller for robot manipulator," *IEEE Trans. Automat. Contr.*, vol. 36, pp. 890–894, 1991.

[26] K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthorp, "Neural networks for control systems—A survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[27] T. Ishihara, K. Abe, and H. Takeda, "A discrete-time design of robust iterative learning algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 74–84, 1992.

[28] T. Kavli, "Frequency domain synthesis of trajectory learning controllers for robot manipulators," *J. Robot. Syst.*, vol. 9, pp. 663–680, 1992.

[29] S. Kawamura, F. Miyasaki, and S. Arimoto, "Realization of robot motion based on a learning model," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 126–134, 1988.

[30] F. L. Lewis, *Applied Optimal Control and Estimation*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[31] W. Messner *et al*. "A new adaptive learning rule," *IEEE Trans. Automat. Contr.*, vol. 36, pp. 188–197, 1991.

[32] C. A. Micchelli, "Interpolation of scattered data: Distance matrices and conditionally positive definite functions," *Const. Approx.*, vol. 2, pp. 11–22, 1986.

[33] S. R. Oh, Z. Bien, and I. H. Suh, "An iterative learning control method with application for the robot manipulator," *IEEE J. Robot. Automat.*, vol. 4, pp. 508–514, 1988.

[34] F. Padiew and R. Su, "An $H_\infty$ approach to learning control systems," *Int. J. Adaptive Contr. Signal Processing*, vol. 4, pp. 465–474, 1990.

[35] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 7, pp. 1481–1497, Sept. 1990.

[36] M. J. D. Powell, "The theory of radial basis function approximation in 1990," in *Advances in Numerical Analysis*, W. Light, Ed., vol. 2. Oxford, England: Clarendon, 1992, pp. 102–205.

[37] T. D. Sanger, "Neural network learning control of robot manipulators using gradually increasing task diffifculty," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 323–333, 1992.

[38] R. M. Sanner and J.-J. E. Slotine, "Gaussian networks for direct adaptive control," *IEEE Trans. Neural Networks*, vol. 3, pp. 837–863, 1992.

[39] N. C. Singer and W. Seering, "Preshaping command inputs to reduce system vibration," *Trans. ASME. J. Dyn. Syst. Measure. Contr.*, vol. 112, no. 1, pp. 76–82, 1990.

[40] A. N. Tikhonov and V. Ya. Arsenin, *Methods for Solution of Ill-Posed Problems*. Moscow: Nauka, 1979, 2nd ed. (in Russian).

[41] K. M. Tao, R. L. Kosut, and G. Aral, "Learning feedforward control," in *Amer. Contr. Conf.*, Baltimore, MD, 1994, pp. 2575–2579.

[42] D. Torfs and D. M. Gorinevsky, *Learning and Task Parameter Approximation of Input Shaping Control for Fast Direct-Drive Manipulator Motions*, Joint Tech. Rep., Robot. Automat. Lab., Univ. Toronto, Canada, and Dep. Mech. Eng., Kath. Univ. Leuven, Belgium, Apr. 1994.

[43] Ya. Z. Tsypkin, *Foundations of the Theory of Learning Systems*. New York: Academic, 1973.



**Dimitry Gorinevsky** (M'91) received M.S. degree in 1982 from the Moscow Institute of Physics and Technology and the Ph.D. degree in 1986 from Moscow State University.

He was with the Russian Academy of Sciences, Moscow, from 1983 to 1990, and then with the Munich University of Technology during 1991–1992. He was with the University of Toronto from 1992 to 1994, and consulted for for Canadian Space Agency and industrial companies. He is currently a Senior Control Engineer with Honeywell-Measurex, North Vancouver, B.C., where he is developing advanced process control applications for paper industry. He is also an Adjunct Professor with the Department of Electrical and Computer Engineering, University of British Columbia. He worked in process control, robotics, automotive, satellite control, and biomechanics. He has authored and coauthored more than 70 technical papers, a book, and has six patents received or pending.

Dr. Gorinevsky was awarded the Alexander von Humboldt International Research Fellowship in 1991, the Canada Research Fellowship in 1992, and is listed in 1993–1994 Who's Who in the World. He is a member of several professional organizations and a registered Professional Engineer in Ontario.

**Dirk E. Torfs** (S'92–A'95) received the degree in mechanical engineering from the Katholieke Universiteit Leuven in 1989, and the Ph.D. degree in mechanical engineering in 1995.

From 1989 to 1994, he was a Research Assistant wotj the Katholieke Universiteit Leuven, working in the field of automation and robotics with emphasis on fast and accurate tracking control of motion controlled systems in the presence of flexibilities. During this period, he has been involved in several industrial projects in a national and international context and projects for the European Space Agency. In 1995, he jointed Trasys Space as a senior project engineer. Since 1996, he has been Head of the Robotics and Telescience Department. His main activities are system engineering, product assurance analysis and project management including control of subcontractors. He is the author and co-author of several papers in proceedings and journals.

**A. A. Goldenberg** (S'73–M'76–SM'88–F'96) received the B.A.Sc. and M.A.Sc. degrees from Technion-Israel Institute of Technology, Haifa, Israel, in 1969 and 1972, respectively, and the Ph.D. degree at the University of Toronto, Toronto, Ontario, in 1976, all in electrical engineering.

From 1975 to 1981, he was with Spar Aerospace Ltd., Toronto, where he worked mainly on control, analysis, design of the space shuttle remote manipulator system and satellite controls. During 1981–1982 he was an Assistant Professor of electrical engineering and from 1982 to 1987 he was an Associate Professor of mechanical engineering at the University of Toronto. Since 1987 he has been a Professor of Mechanical Engineering at the University of Toronto. He holds cross appointments with the Department of Electrical Engineering and the Institute of Biomedical Engineering. He has founded the Robotics and Automation Laboratory in 1982 and the Mechatronics Laboratory in 1996 at the University of Toronto. His current research interests are in the field of robotics and industrial automation, kinematics, control and dynamics of robots and dexterous end effectors. He is the founder and President of Engineering Services Inc., a high-technology company involved in the development of prototype robotic-based automation, and products. He is a consultant to IBM of Canada Ltd., Department of National Defence, Ontario Ministry of Agriculture and Food, RCMP, Consumers Gas, Vulcan-Vulcap Inc., DCIEM, etc.

Dr. Goldenberg is an Editor of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION. He is a Member of the American Society of Mechanical Engineers, and the Professional Engineers of Ontario.