An Approach to Parametric Nonlinear Least Square Optimization and Application to Task-Level Learning Control

D. Gorinevsky, *Member, IEEE*

***Abstract*— This paper considers a parametric nonlinear least square (NLS) optimization problem. Unlike a classical NLS problem statement, we assume that a nonlinear optimized system depends on two arguments: an input vector and a parameter vector. The input vector can be modified to optimize the system, while the parameter vector changes from one optimization iteration to another and is not controlled. The optimization process goal is to find a dependence of the optimal input vector on the parameter vector, where the optimal input vector minimizes a quadratic performance index. The paper proposes an extension of the Levenberg–Marquardt algorithm for a numerical solution of the formulated problem. The proposed algorithm approximates the nonlinear system in a vicinity of the optimum by expanding it into a series of parameter vector functions, affine in the input vector. In particular, a radial basis function network expansion is considered. The convergence proof for the algorithm is presented.**

The proposed approach is applied to task-level learning control of a two-link flexible arm. Each evaluation of the system in the optimization process means completing a controlled motion of the arm. In the simulation example, the controlled motions take only about 1.5 periods of the lowest eigenfrequency oscillations. The algorithm controls this strongly nonlinear oscillatory system very efficiently. Without any prior knowledge of the system dynamics, it achieves a satisfactory control of arbitrary arm motions after only 500 learning (optimization) iterations.

***Index Terms*—Approximation, convergence, flexible manipulator, learning control, Levenberg–Marquardt optimization, parametric nonlinear least squares, parametric programming, radial basis function network.**

I. Introduction

NONLINEAR optimization problems arise in numerous applications, many of them related to control, and a huge body of literature is devoted to numerical methods for their solution; e.g., see [9] and references thereof. In particular, learning (adaptive) systems can be considered as performing on-line optimization of the controlled process [43]. Such systems optimize the performance index on-line by means of input–output experiments with the controlled plant, which is not known precisely.

This paper presents a novel optimization problem statement and an algorithm for its solution. The algorithm is then applied to a learning control problem. To explain the scope of this work, let us start from the following standard optimization problem: we consider a continuously differentiable multivariate vector-valued nonlinear mapping

$$Y = f(U), \qquad Y \in \Re^{N_Y}, \qquad U \in \Re^{N_U}. \tag{1}$$

The problem of finding an argument vector $U = U_*$ that provides $||Y||^2 \to \min$ is called a *nonlinear least square* (NLS) minimization problem. In many applications, the NLS problem is either ill-conditioned, or $N_Y < N_U$ and the solution is not uniquely defined. In those cases, a common approach is to consider a *regularized* problem [41]

$$J(Y, U) = ||Y||^2 + \rho||U||^2 \to \min \tag{2}$$

where ρ is a small positive weight. In this paper, we consider (2) and assume that $\rho \geq 0$ can be zero, if not otherwise stated. For many applications related to control, the goal of making $||Y||$ small does not define the argument (control input) U uniquely. This redundancy can be exploited to achieve a better use of the control resources, as defined by (2).

The NLS problem (1), (2) is classical. This paper is devoted to a *parametric* NLS optimization problem, which is an extension of (1), (2). To formulate this problem, let us consider a two-argument nonlinear mapping

$$Y = f(U, p), \qquad Y \in \Re^{N_Y}, \qquad U \in \Re^{N_U}, \qquad p \in \mathcal{P} \subset \Re^{N_p} \tag{3}$$

which is twice continuously differentiable in both arguments U and p, and where $\mathcal{P}$ is a bounded domain. Throughout this paper, we will use the control terminology to address the mapping (3). We will call p a *parameter vector*, U, an *input vector*, and Y, an *output vector*. We assume that the parameter vector p can take different values at different steps of the optimization process. In other words, the parameter vector p can be regarded as an external (disturbance) input to (3). We assume that p is exactly known but cannot be controlled. The problem considered in this paper can be formulated as follows.

On-Line Parametric NLS Optimization Problem: Let $U_*(p)$ be an optimal solution of the least square problem (2), (3) for a given parameter vector p. The form of (3) is assumed to be unknown, but its values can be obtained for given arguments. Given a sequence of the parameter vectors $p^{(k)} \in \mathcal{P}$, compute and apply an input vector sequence $U^{(k)}$ to find the mapping $U_*(p)$ for $p \in \mathcal{P}$.

Manuscript received June 10, 1994; revised November 16, 1995 and November 1, 1996. Recommended by Associate Editor, K. Passino.

The author was with the Faculty of Applied Science and Engineering, University of Toronto. He is now with Honeywell-Measurex, North Vancouver, B.C., V7J 3S4 Canada (e-mail: dgorinev@nvdevron1.measurex.com).

Publisher Item Identifier S 0018-9286(97)05067-8.

The stated optimization problem can be interpreted as a control problem, where the goal is to find control U that keeps the output Y (3) close to zero despite the changing disturbance p. The designed controller should be adaptive, as (3) is initially unknown. In what follows, we make some additional assumptions about the mapping (3) and the parameter vector sequence $p^{(k)}$. We further design and study a procedure for computing an approximation for the optimizing mapping $U_*(p)$.

The general problem of minimizing a nonlinear function, which depends on additional parameters, is called *a nonlinear parametric programming problem*. Some aspects of this problem were studied in applied mathematics and operations research since the 1950's. A comprehensive survey of the early work in the field can be found in [28] and that of more recent research in [24]. Most of the parametric programming studies consider local properties of the optimizing mapping $U_*(p)$, its bifurcations, and qualitative partitioning properties for a general type of optimized function and input constraints. A few papers more relevant to our study regard *local* approximations for the optimizing mapping in the vicinity of the given parameter vector p. However, to the best of the author's knowledge, no computational algorithm has been developed for determining an approximation for the mapping $U_*(p)$ in an *on-line* regime, i.e., when the change of the parameter p is not controlled by the computational algorithm. The reason is a high-computational complexity of the problem which has made its treatment unfeasible until recently. In this paper, we approximate the mapping $U_*(p)$ in the entire domain $\mathcal{P}$, rather than considering a local approximation. We propose a numerical algorithm for finding such an approximation and demonstrate its feasibility and effectiveness in an example problem. The algorithm works very well for an important class of problems, where dimensions of U and Y can be high while dimension of p is moderate. This is the case for many control-related problems where U and Y contain the history of system input and output, respectively.

Some related work in parametric optimization with a scalar parameter p (time) is surveyed in [34]. This work is motivated by control applications but differs significantly from the approach of this paper since the time (parameter variable) is always monotonically growing. Thus, approximation of the optimal solution is only needed forward in time. Unlike that, this paper considers a *vector* of parameters, which can arbitrarily change on-line. To the best of the author's knowledge, the results of this paper present the first attempt to formulate a computational algorithm and provide a theoretical background for on-line numerical multivariate parametric NLS optimization.

The general scope of the algorithms proposed is related to on-line approximation control approaches that have been recently studied in intelligent control literature in context of learning systems, neural networks, and fuzzy systems; e.g., see [10], [23], [30], [32], [35], and [39]. Some recently published papers consider applications of on-line nonlinear optimization to generalized predictive control [26] and to nonlinear observer design [27]. These papers do not consider parametric optimization problems; however, the spirit and some parts of approaches studied there are related to this paper topic.

A very successful numerical method for the classical NLS problem of the form (1), (2) is the *Levenberg–Marquardt* algorithm, which is derived using a local affine model for the mapping (1). This paper proposes an extension of the Levenberg–Marquardt algorithm for the parametric NLS optimization problem (2), (3). The proposed extension is also based on a local affine model and is derived using tools of linear control and estimation theory.

The most important envisioned applications of the proposed algorithm are for on-line optimization in control and signal processing. In these applications, the parameter vector can describe the process parameters, such as setpoints or state variables, which change in accordance with some factors external to the optimization procedure. This paper demonstrates an application of the parametric NLS optimization to a problem of feedforward control learning. The problem studied differs significantly from the standard learning control formulation introduced in [2], [3], [22], and [29] and is related to the task-level learning paradigm as considered in [1], [5], [14], and [16]. A few more applied control problems related to parametric NLS optimization of the feedforward control are considered in [19] and [21].

The outline of the paper is as follows.

Section II considers a standard (parameter-independent) NLS optimization problem and formulates results on the convergence of a version of the Levenberg–Marquardt method in the presence of noise. These results are a departure point for further derivation and convergence analysis of the proposed algorithm. Section III presents an informal derivation of the parametric NLS optimization algorithm we propose. In Section IV, we formally study convergence of the proposed algorithm. Sections III and IV assume that certain nonlinear mappings appearing in the optimization process can be approximated by a functional series expansion. Section V introduces a specific type of such an expansion—a radial basis function (RBF) network approximation—and considers a numerical example of the parametric NLS optimization. Finally, Section VI demonstrates an application of the parametric NLS optimization to learning control, dependent on task parameters. We present simulation results for the learning control of fast point-to-point motions of a planar flexible arm.

II. NLS Minimization in the Presence of Noise

This section considers a standard NLS minimization problem (1), (2) and an iterative algorithm for finding its solution numerically. We introduce a version of the standard Levenberg–Marquardt minimization scheme using a one-step projection update of the gradient matrix estimate. We analyze convergence of the algorithm in the presence of the measurement noise. The main goal of this section is to prepare a background for derivation and subsequent study of the parametric NLS algorithm in the next sections. The version of the Levenberg–Marquardt minimization algorithm and the convergence proof presented in this section can be conveniently generalized for the algorithm we propose later.

At the first reading, only Sections II-A and II-B are needed to understand the algorithm proposed further in Section III. Section II-C contains technical results on the convergence which are used later in the convergence proofs of Section IV and can be omitted at the first reading.

A. Levenberg–Marquardt Algorithm

Let us first assume that the gradient matrix $G = \partial f/\partial U$ of the mapping (1) can be obtained for a given argument U. Let $U^{(n)}$ and $Y^{(n)} = f(U^{(n)})$ be input and output vectors obtained at iteration n. According to the Levenberg–Marquardt algorithm, the minimizing input is updated as

$$U^{(n+1)} = U^{(n)} - (I_{N_u}(\rho + \mu_n) + {G^{(n)}}^T G^{(n)})^{-1} \times (\rho U^{(n)} + {G^{(n)}}^T Y^{(n)}) \tag{4}$$

where $\mu_n \geq 0$ is a step length parameter, I_{N_U} is a $N_U \times N_U$ unity matrix, and $G^{(n)} = G(U^{(n)})$.

The motivation for (4) is as follows. Let us consider an affine model of the mapping (1) of the form

$$\hat{Y} = Y^{(n)} + G^{(n)}(U - U^{(n)}). \tag{5}$$

Let us also demand that the minimization step length does not exceed a value $d_n > 0$

$$U^{(n+1)} = U^{(n)} + s^{(n)}, \qquad ||s^{(n)}|| \leq d_n. \tag{6}$$

By solving (2) and (5) with respect to $U^{(n+1)}$, and using the Lagrange multiplier method to satisfy the constraint (6), we arrive at (4). The Lagrange multiplier μ_n is nonnegative and can be computed once the gradient G and the allowed step length d_n are given. With the increase of μ_n in (4), all eigenvalues of the inverted matrix in (4) increase, hence the step length $||s^{(n)}||$ decreases. Therefore, the dependence of μ_n on d_n is decreasing. In practice, instead of computing μ_n from d_n, usually μ_n itself is made a parameter of choice. More details on the method can be found in [9].

A proof for local convergence of the method in the vicinity of the optimal solution can be obtained by reformulating [9, Th. 10.2.6] for the mapping $U \mapsto [f(U)^T \ \sqrt{\rho}U^T]^T$. In what follows, we will consider a modification of the Levenberg–Marquardt method and prove its convergence under more stringent conditions than in [9]. The advantage of the proposed modifications of the method and of our proof is that they can be conveniently generalized for the parametric NLS optimization problem considered in Section IV.

B. Finite Difference Update of the Gradient

Let us proceed with a much more common situation, when the gradient G is not available and we can only estimate the function (1) itself. In that case, a natural approach is to consider an affine model (5) of the mapping (1) and update estimates of parameters of this model from the available input–output measurements.

The most common practically used method for estimating the gradient is the *Broyden secant update*. Let $G^{(n)}$ be an estimate of the gradient at the step n. Denote by $s^{(n)}$ the variation of input and by $w^{(n)}$ the corresponding variation of the output at the previous minimization step. For a small step length $||s^{(n)}||$, the gradient should satisfy

$$\begin{aligned} Gs^{(n)} &= w^{(n)}, \qquad s^{(n)} = U^{(n)} - U^{(n-1)} \\ w^{(n)} &= Y^{(n)} - Y^{(n-1)}. \end{aligned} \tag{7}$$

The Broyden update rule can be considered as an application to (7) of the projection estimation algorithm [13], which is very popular in adaptive control and signal processing applications. The Broyden update is used in conjunction with the input update (4) and has the form

$$\hat{G}^{(n+1)} = \hat{G}^{(n)} + (w^{(n)} - \hat{G}^{(n)}s^{(n)}){s^{(n)}}^T/(c^2 + ||s^{(n)}||^2) \tag{8}$$

where $c > 0$ is a scalar parameter used to avoid division by zero.

Local convergence of the Levenberg–Marquardt algorithm with the Broyden secant update can be proved using the *bounded deterioration* technique as considered in [9]. The idea of such a proof is that in the vicinity of the optimum and for a sufficiently small initial error of approximating the gradient G, the algorithm will converge before the gradient approximation error will have time to grow due to the system nonlinearity.

Let us now consider another method for estimating the gradient, which is more appropriate if the measurements are corrupted with a noise. In the parametric NLS optimization problems of the next section, the approximation error can be considered as such a noise. Let us write the affine model for the mapping (1) in the form of a linear regression

$$\hat{Y} = GU + Z = \theta W, \qquad \theta = [Z/c \quad G], \qquad W = \begin{bmatrix} c \\ U \end{bmatrix} \tag{9}$$

where c is a positive scaling constant, $\theta \in \Re^{N_Y, N_U+1}$ is a regression parameter matrix, and W is a regressor vector. The Broyden gradient update (8) is a two-step estimation algorithm for the regression (9) that first sets $Z = GU^{(n-1)} - Y^{(n-1)}$ in (9) and then updates an estimate for G with the projection method. A more natural, one-step projection estimation algorithm for the model (9) has the form

$$\hat{\theta}^{(n+1)} = \hat{\theta}^{(n)} + a^{(n)}(Y^{(n)} - \hat{\theta}^{(n)}W^{(n)}){W^{(n)}}^T/||W^{(n)}||^2 \tag{10}$$

where $\hat{\theta}^{(n)} = [Z^{(n)}/c \ G^{(n)}]$, $W^{(k)} = [c \ {U^{(k)}}^T]^T$, and $a^{(k)} \in \{0, 1\}$ is a scalar deadzone parameter which we will define exactly later on. Unlike the secant update (7), (8), which uses function values obtained on *two* consecutive steps, the update (10) uses only *one* function value. This makes it possible to generalize the update (10) for the parametric optimization case, as shown in the next section.

Let us write a step of the Levenberg–Marquardt algorithm for the affine model in the form (9). By solving (2) and (9) with respect to $U^{(n+1)}$ and using the Lagrange multiplier method to satisfy constraint (6), we obtain

$$U^{(n+1)} = U^{(n)} - (I(\rho + \mu_n) + {G^{(n)}}^T G^{(n)})^{-1} \times (\rho U^{(n)} + {G^{(n)}}^T \hat{Y}^{(n)}) \tag{11}$$

where $\hat{Y}^{(n)} = G^{(n)}U^{(n)} + Z^{(n)}$. We have $\hat{Y}^{(n)} = Y^{(n)}$ for the projection update (10), as long as $a^{(n)} = 1$. Therefore, (11) coincides with the Levenberg–Marquardt step (4).

C. Convergence in the Presence of Noise

We will concentrate on the convergence of the algorithm (10), (11), since this algorithm is the basis for the parametric NLS algorithm of the next section.

Let us review the standing assumptions about the function (1) to be used in the convergence proof. First, we assume that the mapping (1) is γ-Lipschitz in U, that is

$$\|G(U_1) - G(U_2)\| \le \gamma \|U_1 - U_2\|, \qquad G = \frac{\partial f}{\partial U} \tag{12}$$

where γ defines the degree of the mapping nonlinearity. For $\gamma = 0$, the mapping is affine in U. Property (12) infers the following useful inequality which can be found in [9, Lemma 4.2.1]:

$$f(U_1) - f(U_2) - G(U_1)(U_1 - U_2) \le \gamma \|U_1 - U_2\|^2/2. \tag{13}$$

Second, we assume that the function values $Y^{(n)}$ used in (10) are obtained with an error as

$$Y^{(n)} = f(U^{(n)}) + \eta^{(n)}, \qquad \|\eta^{(n)}\| \le \eta_0 \tag{14}$$

where η_0 is a bound on the output perturbation. In the parametric NLS problem of the next section, such a perturbation typically results from imperfect approximation of parametric mappings in the algorithm.

Further, we assume that $\mu_n = \mu$ for (10), (11). We also assume that a bounded excitation signal

$$\xi^{(n)} \in \Re^{N_U}, \qquad \|\xi^{(n)}\| \le \xi_0 \tag{15}$$

is added to the right-hand side of (11). The signal $\xi^{(n)}$ can include measurement and numerical errors, such as an approximation error considered in the next sections, and a self-excitation signal used to enhance convergence of the affine model estimates.

In order to provide for the convergence of the estimation algorithm (10) in the presence of the noise and modeling error (nonlinearity), we will follow the standard approach of adaptive control and parameter estimation theory [13, Sec. 3.6, pp. 88–91] and introduce a deadzone:

$$a^{(n)} = \begin{cases} 0, & \text{if } \|Y^{(n)} - \hat{\theta}^{(n)} W^{(n)}\| \le \Delta \\ 1, & \text{otherwise.} \end{cases} \tag{16}$$

Let us recall that U_* is the optimum input to (1) minimizing (2), and denote $Y_* = f(U_*)$ and $G_* = G(U_*)$. Let $\theta_* = [Z_*/c \; G_*]$ define an affine model (5) corresponding to the linearization of (1) at the optimum $U = U_*$. The following relations hold at the optimum (zero gradient of the performance index and fit of the affine model):

$$\rho U_* + G_*^T Y_* = 0, \qquad Z_* = Y_* - G_* U_*. \tag{17}$$

We will use the following notation for variations of the variables from their values at the optimum:

$$\begin{aligned} \tilde{U}^{(n)} &= U^{(n)} - U_*, \qquad \tilde{Y}^{(n)} = Y^{(n)} - Y_* \\ \tilde{\theta}^{(n)} &= \hat{\theta}^{(n)} - \theta_* \end{aligned} \tag{18}$$

where $\tilde{\theta}^{(n)} = [\tilde{Z}^{(n)}/c \; \tilde{G}^{(n)}]$, $\tilde{G}^{(n)} = G^{(n)} - G_*$, and $\tilde{Z}^{(n)} = Z^{(n)} - Z_*$.

For the assumptions made, it is difficult to obtain a clean result on the algorithm convergence. First, for a general nonlinear optimization algorithm, only *local* convergence results can be obtained and are available in the literature. Second, for bounded perturbations of the measurements, the convergence can only be guaranteed into a *deadzone* of the estimator. Therefore, it is only possible to show that the local convergence domain is much larger than the deadzone domain. By assuming that the input and output disturbance bounds ξ_0 and η_0 are small enough, we can formulate the following result.

Theorem 1: Consider algorithm (10), (11) in the conditions defined by (12) and (14)–(16). Assume that the regressor vector sequence $W^{(k)} = [c \; U^{(k)^T}]^T$ is strongly persistently exciting as defined in [13, Sec. 3.4, p. 73]. If the deadzone in (16) is chosen as $\Delta = \eta_0 + \delta_0$, where $\delta_0 > 0$ is a parameter, then the errors (18) of the algorithm converge into the domain defined by

$$\|\tilde{U}^{(n)}\| \le (\beta_0(\eta_0 + \delta_0) + \xi_0)(1 + \rho/\mu) \tag{19}$$

$$\|\tilde{\theta}^{(n)}\| \le (\eta_0 + \delta_0) \tag{20}$$

provided that the initial errors are in the domain

$$\|\tilde{U}^{(1)}\| \le \sqrt{2\delta_0/\gamma} \tag{21}$$

$$\|\tilde{\theta}^{(1)}\| \le (\sqrt{2\delta_0/\gamma} - \xi_0)\beta_0^{-1}(1 + \rho/\mu)^{-1} \tag{22}$$

where $\beta_0 > 0$ is a constant estimated in the Theorem proof.

Remark 1: As one can observe from (19)–(22), if the perturbation amplitudes ξ_0 and η_0 are small and the deadzone parameter δ_0 is of the same order of magnitude, then the size of the local convergence domain (19) and (20) exceeds the size of the deadzone domain (19)–(22) by a large factor of $\sqrt{2/(\gamma\delta_0)}\beta_0^{-1}(1+\rho/\mu)^{-1}$. As can be expected, the initial condition domain for which Theorem 1 guarantees convergence is large for small nonlinearity parameter γ.

Remark 2: Theorem 1 accommodates for $G_*^T G_*$ singular or ill-conditioned, which is the case for many control problems. This is provided thanks to the use of the regularized performance index (2) with $\rho > 0$. For the regularized problem, the modified Hessian matrix of the optimization problem is $\rho I + G_*^T G_*$ and has minimal singular value not less than ρ.

Remark 3: Signal (15) can be used to ensure the persistency of excitation needed for the convergence of the estimator. Note that as a particular case of such an exciting signal ξ, we can make small increments of the input vector U along each of the coordinate axes in turn, which would correspond to obtaining a new finite-difference estimate of the gradient G each N_U iterations—a common practice in numerical optimization.

In the absence of the measurement noise, a natural way of achieving convergence to the exact optimum is to set $\|\xi^{(n)}\| \le \xi_1/n$. Note that, in general, the convergence rate of the Levenberg–Marquardt method for ill-conditioned problems is not better than $1/n$ [34].

Proof of Theorem 1: We give a constructive proof of this result, which includes concrete estimates of the convergence parameters.

By substituting (18) into (11), using (17), and adding noise (15), we obtain after some transformations

$$\tilde{U}^{(n+1)} = (I_{N_U}(\rho+\mu) + G^{(n)^T}G^{(n)})^{-1} (\mu\tilde{U}^{(n)} - G^{(n)^T}(\tilde{Z}^{(n)} + \tilde{G}^{(n)}U_*) + \tilde{G}^{(n)}Y_*) + \xi^{(n)}. \quad (23)$$

An estimate for the right-hand side (23) can be obtained with help of the following useful fact. Consider any rectangular matrix G, scalar $r > 0$, and a unity matrix I of appropriate size. Denote by $\overline{\sigma}(\cdot)$ a maximal singular value of a matrix and let g_j be the singular values of G. Then

$$\overline{\sigma}((Ir + G^TG)^{-1}G^T) \le \max \frac{g_j}{g_j^2 + r} \le \frac{1}{\min(g_j + r/g_j)} \le \frac{1}{\min_{x>0}(x + r/x)} = \frac{1}{2\sqrt{r}}. \quad (24)$$

By using (24) and recalling that $\tilde{\theta}^{(n)} = [\tilde{Z}^{(n)}/c \;\; \tilde{G}^{(n)}]$, the following estimates can be obtained:

$$\|(I_{N_U}(\rho+\mu) + G^{(n)^T}G^{(n)})^{-1}G^{(n)^T}(\tilde{Z}^{(n)} + \tilde{G}^{(n)}U_*)\| \le \frac{\|\tilde{\theta}^{(n)}\|\sqrt{\|U_*\|^2 + c^2}}{2\sqrt{\rho+\mu}} \equiv \beta_1\|\tilde{\theta}^{(n)}\| \quad (25)$$

$$\|(I_{N_U}(\rho+\mu) + G^{(n)^T}G^{(n)})^{-1}\tilde{G}^{(n)}Y_*\| \le \|\tilde{\theta}^{(n)}\|\,\|Y_*\|/(\rho+\mu) \equiv \beta_2\|\tilde{\theta}^{(n)}\|. \quad (26)$$

By using (15), (25), and (26), we can obtain the following inequality from (23):

$$\|\tilde{U}^{(n+1)}\| \le \frac{\mu}{\rho+\mu}\|\tilde{U}^{(n)}\| + \beta_0\|\tilde{\theta}^{(n)}\| + \xi_0 \quad (27)$$

where $\beta_0 = \beta_1 + \beta_2$. According to [34, Lemma 1, ch. 2], (27) means that $\|\tilde{U}^{(n)}\|$ converges into the region

$$\|\tilde{U}^{(n)}\| \le \beta_0(\sup\|\tilde{\theta}^{(n)}\| + \xi_0)(1 + \rho/\mu). \quad (28)$$

Let us now return to the estimation algorithm (10), which is based on the affine model (9). By using (17), we can write

$$Y - \theta_*W = Y - G_*U - Z_* = Y - Y_* - G_*(U - U_*). \quad (29)$$

From (13) and (29), we obtain $\|Y - \theta_*W\| \le \gamma\|U - U_*\|^2/2$. By using (14), we get

$$\|Y^{(n)} - \theta_*W^{(n)}\| \le \eta_0 + \gamma\|\tilde{U}^{(n)}\|^2/2. \quad (30)$$

In accordance with the standard results [13, Lemma 3.3.2, Sec. 3.3], the error $\|\tilde{\theta}^{(n)}\|$ of the projection estimation algorithm (10) is monotone nonincreasing, provided that the nonlinearity influence in (30) is contained within the deadzone δ_0. By using (28), we obtain that the errors $\|\tilde{U}^{(n)}\|$ and $\|\tilde{\theta}^{(n)}\|$ never leave the initial condition domain (21) and (22). Provided that the control input is persistently exciting, the estimate $\|\tilde{\theta}^{(n)}\|$ will converge to the deadzone, hence, we obtain (21) from (27) similarly to (28). Since $\|\tilde{U}^{(n)}\| \le \|\tilde{U}^{(1)}\| < \sqrt{2\delta_0/\gamma}$ always, the nonlinearity influence in (30) continues to be bounded inside the estimation algorithm deadzone δ_0. Thus, the estimation algorithm converges in accordance with standard projection algorithm properties [13], provided the control input is persistently exciting. The persistent excitation can be provided by including a self-excitation in the signal $\xi^{(n)}$. □

III. Basic Algorithm for Parametric NLS Minimization

This section, as well as the next one, contains the main contribution of this paper. This section proposes an algorithm for on-line parametric NLS minimization. It presents a motivating discussion and an informal derivation of the proposed algorithm. In Section IV, we analyze convergence of the proposed algorithm in a more formal way.

A. Smoothness of the Parametric Optimum

Let us consider a parametric family of minimization problems of the form (2), (3) defined for the argument set

$$\mathcal{D}: \{(U, p) \in \mathcal{D}: p \in \mathcal{P} \subset \Re^{N_p}, U \in \mathcal{D}_U(p) \subset \Re^{N_p}\} \quad (31)$$

where $\mathcal{P}$ is a bounded open set and $\mathcal{D}_U(p)$ is a bounded open set for each $p \in \mathcal{P}$. We assume that for any $p \in \mathcal{P}$, a unique solution $U_*(p) \in \mathcal{D}_U(p)$ to (2), (3) exists and that for some $\delta > 0$ and any p, a δ-neighborhood of $U_*(p)$ belongs to $\mathcal{D}_U(p)$.

In what follows, we approximate the function $U_*(p)$ by means of a functional series expansion. In order for such an approximation to have a small error, the function $U_*(p)$ should have bounded derivatives. By abuse of the previous section notation, we will denote by $G = \partial f(p, U)/\partial U$ the gradient of the mapping (3) with respect to the input U. We assume that the second derivatives of (3) are uniformly bounded on $\mathcal{D}$ and that

$$\|G(p, U_1) - G(p, U_2)\| \le \gamma\|U_1 - U_2\|, \qquad \text{for } (U, p) \in \mathcal{D} \quad (32)$$

where, similarly to (12), γ describes a degree of the system nonlinearity.

In the available literature on parametric optimization (parametric programming), much attention was paid to optimizing function properties. One can find a survey and some general results in [24] and [36]. Below, we present a simple smoothness condition which can be practically convenient for the parametric NLS problem we study.

The optimality condition (2), (3) can be obtained by differentiation and has the form

$$G(U_*(p), p)^T f(U_*(p), p) + \rho U_*(p) = 0. \quad (33)$$

By computing the full derivative of (33) with respect to p, we obtain

$$\mathtt{A}\frac{\partial U_*(p)}{\partial p} = -\frac{\partial G}{\partial p}Y_* - G^T\frac{\partial f(U_*, p)}{\partial p}$$

$$\mathtt{A} = \rho I_{N_U} + G_*^TG_* + \frac{\partial G}{\partial U}Y_* \quad (34)$$

where $G_* = G(U_*(p), p)$, and $Y_* = f(U_*(p), p)$. By the Implicit Function Theorem, $U_*(p)$ will have a bounded derivative, provided that the lowest singular value of $\mathtt{A}$ is uniformly

bounded from zero. This, for instance, can be ensured, provided the last term in the expression for A is small. By using (32), we obtain an estimate

$$\left\| \frac{\partial G}{\partial U} Y_* \right\| \leq \gamma \|Y_*\|. \tag{35}$$

Thus, the derivative of $U_*(p)$ is bounded, provided the nonlinearity is not too severe (γ is sufficiently small).

B. Approximation by Functional Series

Let us introduce a set of N_a-shape functions $\{h_j(p)\colon \mathcal{P} \mapsto \Re\}$ and approximate the mapping $U_*(p)$ with an expansion of the form

$$\begin{aligned} U_*(p) &= \sum_{j=1}^{N_a} U_{*j} h_j(p) + \delta U(p) \\ \|\delta U(p)\| &\leq \delta_U, \qquad p \in \mathcal{P} \end{aligned} \tag{36}$$

where $U_{*j} \in \Re^{N_U}$ are the expansion weights and $\delta U(p) \in \Re^{N_U}$ is the approximation error. In the formulation of the proposed parametric optimization algorithm in this section and its analysis in the next section, we assume that the shape functions $h_j(p)$ are given piecewise continuous functions and the approximation error δ_U is small.

If the mapping $U_*(\cdot)$ has a uniformly bounded derivative on $\mathcal{P}$, as discussed in the previous section, the approximation error bound δ_U can be made arbitrarily small, provided the expansion order N_a is sufficiently large and appropriate shape functions are chosen. For a polynomial expansion, this is ensured by a multidimensional version of the Weierstrass theorem. In Section V, we consider another type of expansion for which this is valid—an RBF expansion. Other types of universal approximators linear in parameters can be also used, e.g., B-splines. Further discussion of the approximation approaches and their theoretical analysis is outside of the scope of this paper, and we refer the interested reader to standard applied mathematics and neural network literature. Let us introduce functions related to the parametric optimum

$$\begin{aligned} G_*(p) &= \frac{\partial f}{\partial p}(U_*(p), p) \\ Z_*(p) &= f(U_*(p), p) - G_*(p) U_*(p). \end{aligned} \tag{37}$$

Similarly to (36), we can write the expansions for approximation of the functions $Z_*(p)$, and $G_*(p)$

$$\begin{aligned} Z_*(p) &= \sum_{j=1}^{N_a} Z_{*j} h_j(p) + \delta Z(p) \\ \|\delta Z(p)\| &\leq \delta_Z, \qquad p \in \mathcal{P} \end{aligned} \tag{38}$$

$$\begin{aligned} G_*(p) &= \sum_{j=1}^{N_a} G_{*j} h_j(p) + \delta G(p) \\ \|\delta G(p)\| &\leq \delta_G, \qquad p \in \mathcal{P} \end{aligned} \tag{39}$$

where $Z_{*j} \in \Re^{N_Y}$ and $G_{*j} \in \Re^{N_Y, N_U}$ are the expansion weights. Derivatives of the functions $Z_*(\cdot)$ and $G_*(\cdot)$ are uniformly bounded on $\mathcal{P}$, provided that the derivatives of $U_*(\cdot)$ are bounded. Hence, δ_Z and δ_G can be made small for high-order N_a of the expansions (38), (39).

The algorithm that we are going to derive neglects the approximation errors δ_U, δ_Z, and δ_G. For $\delta_U = 0$, (37) can be represented in the linear regression form

$$\begin{aligned} U_*(p) &= K_* \Phi(p) \\ K_* &= [K_{*1} \ \cdots \ K_{*N_a}] \in \Re^{N_U, N_a} \qquad (40) \\ \Phi(p) &= [h_1(p) \ \cdots \ h_{N_a}(p)]^T \in \Re^{N_a}. \qquad (41) \end{aligned}$$

The Levenberg–Marquardt algorithm derivation in Section II is based on an affine model (9) of the mapping (1). Similarly to this, the proposed algorithm assumes an affine in U model of the mapping (3) for each p. The dependence of the affine model on p is assumed to have the form defined by the expansions (38) and (39)

$$\hat{Y} = \hat{G}(p) U + \hat{Z}(p) \tag{42}$$

$$\hat{Z}(p) = \sum_{j=1}^{N_a} Z_j h_j(p)$$

$$\hat{G}(p) = \sum_{j=1}^{N_a} G_j h_j(p). \tag{43}$$

The affine parametric model (42), (43) is the central part of the approach. This model can be compactly written in the form of a regression

$$\begin{aligned} \hat{Y} &= \Theta \overline{\Phi}(p, U) \qquad (44) \\ \Theta &\in \Re^{N_Y, N_a(N_U+1)}, \qquad \overline{\Phi}(p, U) \in \Re^{N_a(N_U+1)} \\ \overline{\Phi}(p, U) &= \Phi(p) \otimes W, \qquad W = [c \quad U^T]^T \qquad (45) \end{aligned}$$

where $\otimes$ denotes the Kronecker (direct) product of matrices and the parameter matrix has the form $\Theta = [Z_1/c \ G_1 \ \cdots \ Z_{N_a}/c \ G_{N_a}]$. The affine model (42)–(44) gives the linearization (37)–(39) of the mapping (3) at the parametric optimum for $\Theta = \Theta_*$, where

$$\Theta_* = [Z_{*1}/c \quad G_{*1} \quad \cdots \quad Z_{*N_a}/c \quad G_{*N_a}]. \tag{46}$$

C. Basic Algorithm

Our goal is to find an approximation of the form (40) to the optimal input mapping $U_*(p)$. We assume that a sequence of the parameter vectors $\{p^{(k)}\}_{k=1}^{\infty}$ is given. Let $K^{(k)}$ be the value of the input parameter matrix in (40) at the step k. Then, in accordance with (40), the input vector at step k is $U^{(k)} = K^{(k)} \Phi(p^{(k)})$. Let us denote by

$$U^{(k+1|k)} = K^{(k)} \Phi(p^{(k+1)}) \tag{47}$$

the input U, which would be obtained at step $k+1$ if the matrix $K^{(k)}$ is not updated.

Let us derive an iterative algorithm that updates approximation (40) to the parametric optimum by using the affine model (42), similarly to the Levenberg–Marquardt algorithm. Let us demand that, similar to (6), the minimization step should be bounded as

$$U^{(k+1)} = U^{(k+1|k)} + s^{(k)}, \qquad \|s^{(k)}\| \leq d_k. \tag{48}$$

By substituting (45) and (48) into (44), we obtain the affine model output at the step $k+1$ as

$$\begin{aligned}\hat{Y}^{(k+1)} &= \Theta\overline{\Phi}(p^{(k+1)}, U^{(k+1)}) \\ &\equiv \hat{Y}^{(k+1|k)} + \hat{G}(p^{(k+1)})s^{(k)} \\ \hat{Y}^{(k+1|k)} &= \Theta\overline{\Phi}(p^{(k+1)}, U^{(k+1|k)})\end{aligned} \tag{49}$$

where $\hat{G}(p)$ is the gradient estimate defined by the affine model (42), (44). By substituting $\hat{Y}^{(k+1)}$ (49) as Y and $U^{(k+1)}$ (47) as U into (2), and finding a constraint optimum for the affine model (49) with respect to the minimization step $s^{(k)}$, we obtain an analog of (11)

$$\begin{aligned}s^{(k)} =& -(I_{N_U}(\rho+\mu_k) + \hat{G}(p^{(k+1)})^T\hat{G}(p^{(k+1)}))^{-1} \\ &\times (\hat{G}(p^{(k+1)})\hat{Y}^{(k+1|k)} + \rho U^{(k+1|k)}).\end{aligned} \tag{50}$$

Now let us note that according to (40) and (48)

$$s^{(k)} = (K^{(k+1)} - K^{(k)})\Phi(p^{(k+1)}). \tag{51}$$

By finding a least square solution to (51) for the $K^{(k+1)} - K^{(k)}$ and substituting (50) for $s^{(k)}$, we obtain a step of the proposed basic parametric NLS optimization method

$$\begin{aligned}K^{(k+1)} =& K^{(k)} - (I_{N_U}(\rho+\mu_k) + \hat{G}(p^{(k+1)})^T\hat{G}(p^{(k+1)}))^{-1} \\ &\times (\hat{G}(p^{(k+1)})\hat{Y}^{(k+1|k)} + \rho U^{(k+1|k)}) \\ &\times \Phi(p^{(k+1)})^T/\|\Phi(p^{(k+1)})\|^2\end{aligned} \tag{52}$$

$$U^{(k+1)} = K^{(k+1)}\Phi(p^{(k+1)})^T + \xi^{(k)} \tag{53}$$

where the gradient estimate $\hat{G}(p)$ is defined by (43); $\hat{U}^{(k+1|k)}$ is the optimal step $k+1$ control input computed before the update in accordance with (41), (47); and $\hat{Y}^{(k+1|k)} = \Theta\overline{\Phi}(p^{(k+1)}, U^{(k+1|k)})$ is the output of the affine model (44) for the input $U^{(k+1|k)}$. The sequence $\xi^{(k)} \in \Re^{N_U}$ is a self-excitation signal added to the computed control, and it will be discussed in the next section. The parametric optimization update (52) has the form reminiscent of the Levenberg–Marquardt update (11).

The update (52), (53) is based on the affine model (42) of the mapping (3). This model needs to be estimated from the information about the mapping (3), obtained in the course of the minimization. At each step of our parametric NLS algorithm, we can apply a projection update similar to (9) to the estimation of the parameter matrix Θ in (42). The algorithm is similar to (10) and has the form

$$\begin{aligned}\hat{\Theta}^{(k+1)} &= \hat{\Theta}^{(k)} + a^{(k)}(Y^{(k)} - \hat{\Theta}^{(k)}\overline{\Phi}^{(k)})\overline{\Phi}^{(k)T}/\|\overline{\Phi}^{(k)}\|^2 \\ \overline{\Phi}^{(k)} &= \overline{\Phi}(p^{(k)}, U^{(k)})\end{aligned} \tag{54}$$

where, as in (10), $a^{(k)} \in \{0, 1\}$ is a scalar deadzone parameter

$$a^{(k)} = \begin{cases} 0, & \text{if } \|Y^{(k)} - \hat{\Theta}^{(k)}\overline{\Phi}^{(k)}\| \le \Delta \\ 1, & \text{otherwise.}\end{cases} \tag{55}$$

We will discuss the choice of the deadzone Δ in the next section, in conjunction with a proof of the algorithm convergence.

Equations (41), (45), (52), (54), and (55) constitute the proposed basic algorithm for on-line parametric NLS optimization. The affine model update (54), (55) in this algorithm uses only one, last data point for the update. This is similar to the projection update (10) and differs from the secant Broyden update (8) which uses two last data points. The affine parametric model (44) can be updated using the data obtained for different values of the parameter vector p at each step.

The proposed algorithm includes a single inversion of the $N_U \times N_U$ matrix in (52). As an alternative to the proposed algorithm, the matrix K_* in the approximation (40) of the parametric optimum could be directly optimized by using a standard optimization algorithm, such as the Levenberg–Marquardt algorithm. In that case, however, the optimization algorithm will work with $N_U N_a$ optimized parameters (entries of the matrix K_*). A Hessian matrix inverted by such a standard algorithm will have size $N_U N_a \times N_U N_a$, which shows that such a standard algorithm would be much more difficult to implement compared to the proposed one.

It will be further demonstrated that the proposed parametric optimization algorithm works fairly well in applications. A theoretical justification of the algorithm convergence is given in the next section.

IV. Convergence of the Parametric NLS Algorithm

In proving convergence of the proposed algorithm (41), (45), (52), (54), (55) for parametric NLS optimization, we will closely follow the scheme used in Section II-C.

A. Convergence Result

In order to present a formal result on the convergence, let us briefly review the assumptions we make. These assumptions are both technical and on the algorithm specifics.

- A1) We assume that (3) is γ-Lipschitz and (32) holds.
- A2) We assume that the approximation errors δ_U, δ_G, and δ_Z in the expansions (36), (38), and (39) can be considered small.
- A3) We assume that the step constraint in (52) is constant, $\mu_k = \mu$.
- A4) Similarly to (14), we assume that in (3) the output vector $Y^{(k)}$ at step k is related to the input vector $U^{(k)}$ as

$$Y^{(k)} = f(U^{(k)}, p^{(k)}) + \zeta^{(n)}, \qquad \|\zeta^{(k)}\| \le \zeta_0 \tag{56}$$

  where ζ_0 is a bound on the output perturbation. In the convergence analysis we will consider signals $\xi^{(n)}$ in (53) and $\zeta^{(n)}$ in (56) to be small.
- A5) We assume that the sequences $p^{(k)}$ and $U^{(k)}$ are such that the regressor vector sequences $\Phi(p^{(k)})$ (41) and $\overline{\Phi}(p^{(k)}, U^{(k)})$ (45), are persistently exciting (as discussed in Lemma 1 below).

By using (38), (39), and inequality (13) [which follows from (32)], we obtain the following estimate for the affine model accuracy:

$$\begin{aligned}&\|Y^{(k)} - \Theta_*\overline{\Phi}(p^{(k)}, U^{(k)})\| \\ &\qquad \le \gamma\|U^{(k)} - U_*(p^{(k)})\|^2/2 + \delta_G\|U^{(k)}\| + \zeta_0.\end{aligned} \tag{57}$$

We will denote the error of evaluating the affine model weight matrix at step k by $\tilde{\Theta}^{(k)} = \Theta^{(k)} - \Theta_*$ and the error

of evaluating the parametric optimum approximation weights K_* (40) by

$$\tilde{K}^{(k)} = K^{(k)} - K_*(p^{(k)}) \in \Re^{N_U, N_a}. \tag{58}$$

The convergence result can be formulated in the form similar to Theorem 1.

Theorem 2—Convergence of the Proposed Parametric Optimization Method: Let us consider algorithm (41), (45), (52)–(55) under Assumptions A1)–A5). The convergence condition can be formulated in terms of the following positive parameters, depending on the system (3):

- nonlinearity parameter: $\gamma_1 = 2\gamma \sup_{p\in\mathcal{P}} ||\Phi(p)||^2$;
- parameter of the input–output mapping approximation error: $\eta_1 = \zeta_0 + \delta_Z + \delta_G \sup_n U^{(n)} + \gamma\delta_U^2$;
- optimal control approximation error parameter: $\xi_1 = \xi_0 + \delta_U$;
- persistency of excitation parameters: ϵ and l ($0 < \epsilon < 1$, $l \geq 1$). These parameters are defined by the properties of the sequences $p^{(k)}$ and $U^{(k)}$, as discussed in Lemma 1 below.

If the deadzone in (55) is chosen as $\Delta = \eta_1 + \delta_1$, where $\delta_1 > 0$ is a parameter of the same order as η_1, then the algorithm errors converge into the domain defined by

$$||\tilde{K}^{(n)}|| \leq \beta(\eta_1 + \delta_1)l/\epsilon + l\xi_1/\epsilon \tag{59}$$

$$||\tilde{\Theta}^{(n)}|| \leq (\eta_1 + \delta_1) \tag{60}$$

provided that the initial errors are in the domain

$$||\tilde{K}^{(1)}|| \leq \sqrt{\delta_1/\gamma_1} \tag{61}$$

$$||\tilde{\Theta}^{(1)}|| \leq \sqrt{\delta_1/\gamma_1}\epsilon/(l\beta) - \xi_1/\beta \tag{62}$$

where $\beta > 0$ is a parameter estimated in the Theorem proof along with ϵ and l.

If the approximation errors η_1 and ξ_1 are small and $\delta_1 \approx \eta_1$, the initial condition domain (61), (62) is bigger than the convergence domain (59), (60) by a large factor $\epsilon/(l\beta\sqrt{\delta_1\gamma_1})$.

B. Proof of Theorem 2

Let us first consider evolution of the parametric optimum error weights $\tilde{K}^{(k)}$ (58). We can present (52) in the form

$$\begin{aligned}\tilde{K}^{(k+1)} = \tilde{K}^{(k)} &- (I_{N_U}(\rho+\mu) + \hat{G}^{(k+1)^T}\hat{G}^{(k+1)})^{-1} \\ &\times [\hat{G}^{(k+1)}\hat{Y}^{(k+1|k)} + \rho U^{(k+1|k)}] \\ &\times \Phi^{(k+1)^T}/||\Phi^{(k+1)}||^2 + \eta^{(k)}\end{aligned} \tag{63}$$

where $\Phi^{(k+1)} = \Phi(p^{(k+1)})$ and $\hat{G}^{(k+1)} = \hat{G}(p^{(k+1)})$.

By using (36), (40), (47), and (58), we can write

$$U^{(k+1|k)} = U_*(p^{(k+1)}) + \psi^{(k)} + \tilde{K}^{(k)}\Phi^{(k+1)} \tag{64}$$

$$\psi^{(k)} = \delta U(p^{(k+1)}) + \xi^{(k)}$$

$$||\psi^{(k)}|| \leq \delta_U + \xi_0 \equiv \xi_1 \tag{65}$$

where $\psi^{(k)}$ is a bounded signal similar to (15). The sequence $\psi^{(k)}$ includes the approximation error (36) and the self-excitation signal (53). The self-excitation signal $\xi^{(k)}$ can be chosen to provide persistent excitation required for the estimator convergence.

Note that (33) and (37) give combined

$$(G_*(p)^T G_*(p) + \rho I_{N_U})U_*(p) + G_*^T Z_* = 0_{N_U}. \tag{66}$$

Let us now consider the expression in the square braces in (63). By substituting $\hat{Y}^{(k+1|k)} = \Theta^{(k)}\overline{\Phi}(p^{(k)}, U^{(k+1|k)})$ and using (64), (66), we obtain after some transformations

$$\begin{aligned}&\hat{G}^{(k+1)}\hat{Y}^{(k+1|k)} + \rho U^{(k+1|k)} \\ &\quad = (\hat{G}^{(k+1)^T}\hat{G}^{(k+1)} + \rho I_{N_U})(\psi^{(k+1)} + \tilde{K}^{(k)}\Phi^{(k+1)}) \\ &\qquad + \hat{G}^{(k+1)^T}(\tilde{G}^{(k+1)}U_*(p^{(k+1)}) + \tilde{Z}^{(k+1)}) + \tilde{G}^{(k+1)}Y_*\end{aligned} \tag{67}$$

where $\tilde{G}^{(k+1)} = \hat{G}(p^{(k+1)}) - G_*(p^{(k+1)})$, and $\tilde{Z}^{(k+1)} = \hat{Z}(p^{(k+1)}) - Z_*(p^{(k+1)})$ are related to the parameter matrix error $\tilde{\Theta}^{(k+1)} = \hat{\Theta}^{(k+1)} - \Theta_*$ in accordance with (42) and (43). By substituting (67) into (63), we come to

$$\begin{aligned}\tilde{K}^{(k+1)} &= \tilde{K}^{(k)}F^{(k)} + (D^{(k)} + \mu I_{N_U})^{-1}\mu\tilde{K}^{(k)}(I_{N_a} - F^{(k)}) \\ &\quad - (D^{(k)} + \mu I_{N_U})^{-1}D^{(k)}\psi^{(k+1)}(I_{N_a} - F^{(k)}) \\ &\quad - (D^{(k)} + \mu I_{N_U})^{-1}[\hat{G}^{(k+1)^T}\tilde{\Theta}^{(k+1)}\overline{\Phi}(U_*(p^{(k+1)}) \\ &\quad p^{(k+1)}) + \tilde{G}^{(k+1)^T}Y_*(p^{(k+1)})] \\ D^{(k)} &= \rho I_{N_U} + \hat{G}^{(k+1)^T}\hat{G}^{(k+1)}\end{aligned} \tag{68}$$

where $F^{(k)} = (I_{N_a} - \Phi^{(k+1)}\Phi^{(k+1)^T})/||\Phi^{(k+1)}||^2$ is an idempotent (projection) matrix. The matrix $F^{(k)}$ has the following properties:

$$\begin{aligned}(F^{(k)})^2 &= F^{(k)} \\ &= F^{(k)^T} \\ F^{(k)}(I_{N_a} - F^{(k)}) &= 0_{N_a}.\end{aligned} \tag{69}$$

Equation (68) can be represented in the form

$$\tilde{K}^{(k+1)} = \tilde{K}^{(k)}F^{(k)} + A^{(k)}\tilde{K}^{(k)}(I_{N_a} - F^{(k)}) + e^{(k)} \tag{70}$$

$$A^{(k)} = \mu(D^{(k)} + \mu I_{N_U})^{-1}, \qquad ||A^{(k)}|| \leq \mu/(\mu+\rho) \tag{71}$$

where matrix $A^{(k)}$ is symmetrical and positive definite. The error matrix $e^{(k)} \in \Re^{N_U, N_a}$ in (70) includes the last two terms in (68) that are not proportional to $\tilde{K}^{(k)}$. The following estimates can be used:

$$||e^{(k)}|| \leq \beta||\tilde{\Theta}^{(k)}|| + ||\psi^{(k)}|| \tag{72}$$

where similarly to (27) we can obtain an estimate

$$\begin{aligned}\beta = &\frac{1}{2\sqrt{\rho+\mu}} \sup_{p\in\mathcal{P}}(||c^2 + U_*(p)^2||^{1/2}||\Phi(p)||) \\ &+ \frac{1}{\rho+\mu} \sup_{p\in\mathcal{P}} ||Y_*(p)||^2.\end{aligned} \tag{73}$$

The remaining part of the proof will require the following auxiliary result.

Lemma 1—Estimation with Persistent Excitation: Let us consider system (70), (71), where the regressor vector sequence is persistently exciting so that for some $l > 0$, some $\delta > 0$, and any k

$$\underline{\sigma}\left(\sum_{j=k+1}^{j=k+l} \frac{\Phi^{(k)}\Phi^{(k)^T}}{||\Phi^{(k)}||^2}\right) \geq \delta. \tag{74}$$

Then, for certain $\epsilon > 0$ the following inequality holds:

$$||\tilde{K}^{(k+l)}||_F \leq (1-\epsilon)||\tilde{K}^{(k)}||_F + \sum_{j=k}^{k+l} ||e^{(j)}||_F \tag{75}$$

where $||\cdot||_F$ is the Frobenius norm of a matrix.

The proof of Lemma 1 is given in the Appendix. Lemma 1 together with the estimate (68) describes the evolution of the parametric optimum error weight matrix $\tilde{K}^{(k)}$ (58). By applying Lemma 1 and using (65) and (72), we obtain

$$||\tilde{K}^{(k+l)}||_F \leq (1-\epsilon)||\tilde{K}^{(k)}||_F + l\beta\overline{\Theta} + l\xi_1. \tag{76}$$

To complete the proof, let us now consider evolution of the affine model error in the projection estimation algorithm (54), (55). This error $||\tilde{\Theta}^{(k)}||$ diminishes monotonically as long as the perturbation [modeling error plus measurement noise $\eta^{(k)}$] is within the deadzone Δ [13].

The projection algorithm perturbation is evaluated in (57). In accordance with (36) and (58), we obtain

$$\begin{aligned} &||U^{(k)} - U_*(p^{(k)})||^2 \\ &\quad \leq [||\tilde{K}^{(k)}|| \sup_{p\in\mathcal{P}} \Phi(p) + ||\delta U(p^{(k)})||]^2 \\ &\quad \leq 2||\tilde{K}^{(k)}||^2(\sup_{p\in\mathcal{P}} \Phi(p))^2 + 2||\delta U(p^{(k)})||^2. \end{aligned} \tag{77}$$

Therefore, (57) gives

$$||Y^{(k)} - \Theta_*\overline{\Phi}(p^{(k)}, U^{(k)})|| \leq \gamma_1||\tilde{K}^{(k)}||^2 + \eta_1 \tag{78}$$

where $\gamma_1 = 2\gamma(\sup_{p\in\mathcal{P}} \Phi(p))^2$, and $\eta_1 = \zeta_0 + \delta_Z + \delta_G \sup_n ||U^{(n)}|| + \gamma\delta_U^2$.

The estimates (78) and (76) are analogous to the estimates (30) and (27), respectively, used in the proof of Theorem 1. Therefore, we obtain the Theorem 2 convergence result similarly to how the Theorem 1 result is proved in Section II-C. □

Convergence of both the estimation algorithm (54) and the optimization algorithm (52) requires the sequences of the nonlinear regressor vectors $\Phi^{(k)} = \Phi(p^{(k)})$ and $\overline{\Phi}^{(k)} = \overline{\Phi}(p^{(k)}, U^{(k)})$ to be persistently exciting (PE). The PE properties depend on the set of the nonlinear functions $h_j(p)$ in (36), (38), and (39). In the next section, we consider one particular set of such functions.

V. Radial Basis Function Approximation

In this section, we briefly survey an auxiliary problem of approximating a smooth mapping using an expansion with respect to certain shape functions. This problem is important for justifying the use of the approximations (36), (38), (39) in the proposed algorithm. We further apply the algorithm to a simple illustrative problem.

A. Background

Let us consider a nonlinear mapping $\Re^{N_Y} \mapsto \Re^{N_p}$

$$Y = f(p), \qquad Y \in \Re^{N_Y}, \qquad p \in \mathcal{P} \subset \Re^{N_p} \tag{79}$$

where p is an argument (input) vector, $\mathcal{P}$ is a bounded domain, and Y is an output vector. We assume that the mapping (79) has the necessary number of continuous derivatives.

Let us consider an approximation of (79) with the expansion of the form

$$\hat{Y} = \sum_{j=1}^{N_a} Z^{(j)} h(p - Q^{(j)}) \tag{80}$$

where function $h(r)$ depends on the radius $||r||$ and $Q^{(j)} \in \Re^{N_p}$ are given vectors. Such expansions are known under the name of RBF approximation. RBF approximation has been extensively used in computer graphics and experimental data processing applications (e.g., geophysical data) for more than a decade and has been demonstrated to provide a high-quality approximation. One can find further details and references in [11], [33], [37], and [38]. Some of the most commonly used radial functions are

$$\begin{aligned} h(p) &= \exp(-||p||^2/d^2) \\ h(p) &= (1 + ||p||^2/d^2)^{1/2} \\ h(p) &= (1 + ||p||^2/d^2)^{-1/2} \end{aligned} \tag{81}$$

where the first radial function is Gaussian, and the last two are called Hardy Multiquadrics and Reverse Multiquadrics, respectively. Usually, the radial function width parameter d in (81) is chosen to be about an average distance between the neighboring node centers.

RBF approximation has a number of advantageous properties which make it very attractive to use in parametric NLS problems. For the first and last functions (81), the expansion (80) is semilocal, i.e., the radial function decays away from its center. This is very advantageous for many applications.

Further, RBF approximation has an excellent accuracy compared to other approximation methods, as it is shown in many applications [11], [18]. This high accuracy has a theoretical justification. It has recently been acknowledged that RBF approximation minimizes a certain regularization performance index that describes the interpolated surface roughness [33], [37]. Different forms of radial functions (81) correspond to a minimization of different regularization indexes. The RBF approximation can be considered as a kind of low-pass spatial filtering of the approximated mapping [38], [39]. Theoretically, the RBF approximation can achieve an arbitrary small error for a smooth mapping [31].

The accuracy of RBF approximation enjoys a high order of convergence with growing density of available data. The most remarkable fact is that the convergence order actually *grows* with the dimension N_p of the argument space [38, Ths. 7.2 and 8.2].

Recently, some authors have treated the RBF approximation in the connectionist (neural) network context [6], [7], [31], [33]. They consider the radial function centers $Q^{(j)}$—which are called the network node centers—that do not coincide with

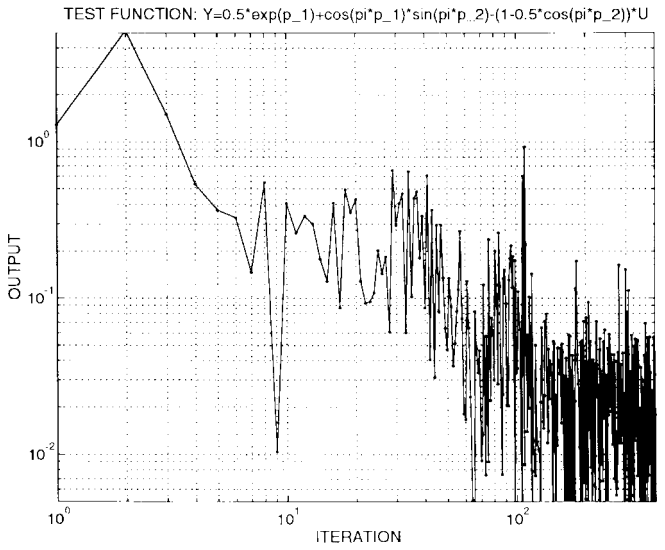

Fig. 1. Evolution of the output error $\|Y\|$ for the illustrative example. Parameter vector sequence $p^{(k)}$ is randomly generated inside the unit square.

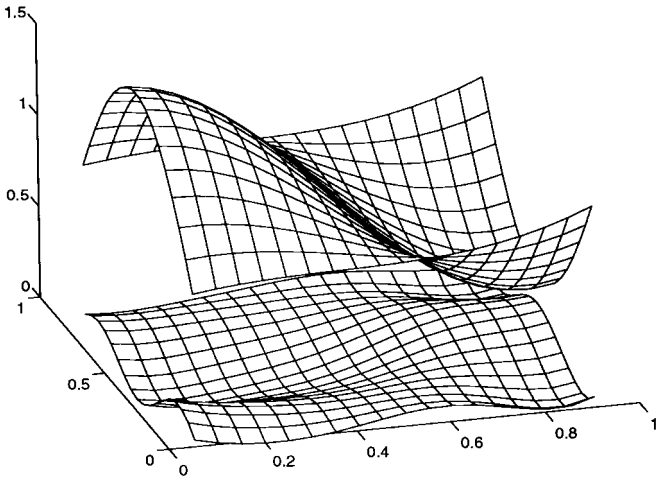

Fig. 2. Output Y for the illustrative example system for $U = 0$ (the upper surface) and for the approximated parametric optimum $U = U_*(p)$ (the lower surface).

the arguments for which function values are available. This coincides with the way we apply RBF expansion (79) in the algorithms of Section III.

In our implementation of the RBF network approximation, we assume that the expansion order N_a and the centers $Q^{(j)}$ $(j = 1, \cdots, N_a)$ in (80) are given and fixed. We use the RBF expansions in the algorithm for parametric NLS optimization described in Section III so that in (36), (38), and (39) $h_j(p) = h(p - Q^{(j)})$. For the RBF network approximation, the regressor vector (41) has the form

$$\Phi(p) = [h(p - Q^{(1)}) \ \cdots \ h(p - Q^{(N_a)})]^T. \qquad (82)$$

The persistence of excitation conditions for the RBF network approximation are analyzed by the author elsewhere [17]. It can be proved that the sequence $\Phi(p^{(j)})$ (82) is PE if the inputs $p^{(j)}$ belong to certain neighborhoods of the RBF centers $Q^{(j)}$. Similar PE conditions are proved in [17] for the affine RBF approximation when the regressor vector has the form (45).

B. Illustrative Example of the Parametric NLS Optimization

Let us illustrate the work of the developed on-line parametric NLS algorithm with a simple example. We consider the parametric NLS problem (2) with the nonlinear function (3) of the form

$$\begin{aligned} Y = S(U, p) &= 0.5e^{p_1} + \cos(\pi p_1)\sin(\pi p_2) \\ &\quad - (1 - 0.5\cos(\pi p_2))U, \qquad Y \in \Re, \qquad U \in \Re \\ p \in \mathcal{P} &\equiv [0, 1]^2 \subset \Re^2, \quad N_Y = 1, \quad N_U = 1, \quad N_p = 2. \end{aligned} \qquad (83)$$

The optimization goal defined by (2) is to find the input $U(p)$ in (83) minimizing the system output Y for each p.

Let us consider an RBF network approximation of the parametric dependencies (36), (38), and (39) with regard to the mapping (83). The regressor vector $\Phi(p)$ in (41) and (45) has the form (82), where $h(p)$ is the Gaussian RBF—the first function in (81). In the numerical results below, we used a uniform 6×6 grid of the RBF network nodes $Q^{(j)}$; thus, $N_a = 36$. The Gaussian RBF width is chosen to be $d = 0.3$, which is 1.5 times the gridsize.

The parametric NLS algorithm (52), (54), (55) described in Section III was coded in MATLAB. We used the following parameters:

$$\rho = 0.0004, \qquad \mu = 5\rho, \qquad \Delta = 0.002$$

and the self-excitation $\eta^{(k)}$ in (52) was a pseudo-random sequence with an absolute value bounded as $1/k$. The sequence of the parameter vectors $p^{(k)}$ was generated randomly inside the unit square $\mathcal{P}$. Fig. 1 illustrates the evolution of the system output $\|Y^{(k)}\|$ in the parametric optimization process. The algorithm converges in 100–200 steps.

Fig. 2 shows the system output $Y = S(0_{N_U}, p)$ for the zero input U (the upper surface) and the output $Y = S(U_*(p), p)$ for the found approximation of the parametric optimum (the lower surface). Despite some remaining error, the approximated input $U_*(p)$ brings the output close to zero. The maximal remaining error $\|Y\|$ is about 40 times smaller than that for zero input.

Fig. 3 displays the approximation (40) for the parametric optimum $U_*(p)$, as computed by the algorithm. Note that the gain (gradient) $G = 1 - 0.5\cos(\pi p_2)$ of (83) varies from 0.5–1.5 in different parts of the unit square $\mathcal{P}$. Despite this, the algorithm copes with the system reasonably well.

The results demonstrate the feasibility and satisfactory performance of the proposed algorithm. Its application to a more comprehensive control problem is considered in the next section.

VI. Application to Learning Control

This section considers an application of the proposed parametric NLS optimization algorithm to a learning control problem. Though we believe that the proposed algorithm can be useful in many different applications, including feedback

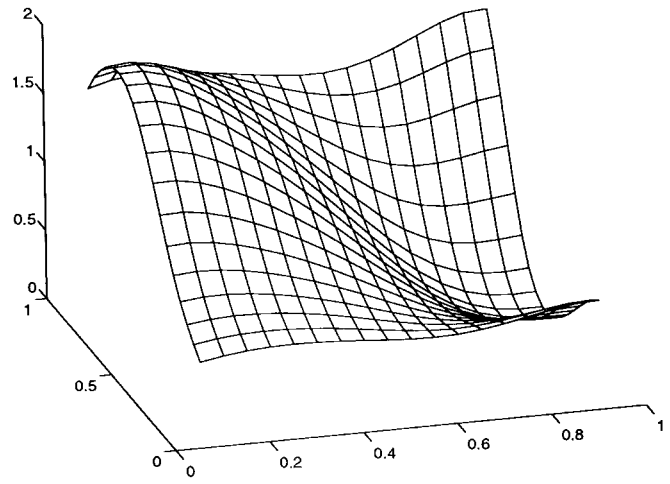

Fig. 3. Parametric optimum $U = U_*(p)$ computed for the illustrative example.

control of nonlinear systems, it was this problem that provided us with the initial inspiration for the algorithm development.

A. Learning Control Problem

Let us consider a nonlinear time-invariant system of the form

$$\dot{x} = g(x, u) \tag{84}$$

$$y = h(x) \tag{85}$$

where $x \in \Re^{n_x}$ is a state vector, $y \in \Re^{n_y}$ is an observation vector, and $u \in \Re^{n_u}$ is a control input vector. We further assume that the nonlinear mappings $g(\cdot, \cdot)$ and $h(\cdot)$ are not known exactly, but are known to be smooth, and the nonlinear system (84), (85) is known to be observable and controllable. We consider a controlled motion of the system (84), (85) in the given time interval $[0, T]$ and assume that the initial state vector $x(0)$ is defined by the vector $\lambda \in \Re^{n_\lambda}$, $n_\lambda \leq n_x$ as $x(0) = \psi(\lambda)$, where $\psi(\cdot)$ is a smooth mapping $\Re^{n_\lambda} \mapsto \Re^{n_x}$.

The control problem is to find a control input $u(\cdot)$ defined in the time interval $[0, T]$ that allows us to achieve

$$J_1(y(\cdot), y_d(\cdot; \lambda, \nu); u(\cdot)) \to \min. \tag{86}$$

Here $\nu \in \Re^{n_\nu}$ is a vector that defines the desired system state at time T, and $y_d(t; \lambda, \nu) \in \Re^{n_y}$ is a preplanned desired output of the controlled plant. The preplanned output y_d depends on the initial condition vector λ and the control goal vector ν. Section VI-B considers the learning control of a rest-to-rest motion for a flexible manipulator arm. In this example, vectors ν and λ correspond to rigid-body initial and final coordinates of the arm, while the state vector x includes both rigid-body and flexible coordinates and velocities.

Let us introduce a task parameter vector p comprising the initial condition vector λ and the vector of the control goal ν

$$p = \begin{bmatrix} \lambda \\ \nu \end{bmatrix} \in \Re^{N_p}, \qquad N_p = n_\lambda + n_\nu. \tag{87}$$

We further consider the learning control problem as stated below.

Parametric Learning Control Problem: We assume that the mappings (84) and (85) are unknown, but we can repeatedly apply the computed feedforward control $u(\cdot)$ to the system (84), (85), observe its output $y(\cdot)$ on the time interval $[0, T]$, and use the obtained observations to update the control. For each run with the feedforward control $u(\cdot)$, we suppose that the system time t is reset to zero, and the initial condition is defined by the vector λ. The control goal is defined by the vector ν. The vectors λ and ν may take different values from run to run. The problem is to minimize the performance index (86) for each value of the parameter vector p (87) in the given domain $p \in \mathcal{P} \subset \Re^{N_p}$.

Let us represent the time-histories of the feedforward input $u(\cdot)$ to the system (84) and the system output $y(\cdot)$ (85) by finite-dimensional vectors. We apply the assumed mode expansion approach and introduce a set of shape functions $\{\phi_j(\cdot)\colon [0, T] \mapsto \Re^{n_u}; (j = 1, \cdots, N)\}$ and consider a feedforward input of the form

$$u(\cdot) = \sum_{j=1}^{N} u^{(j)} \phi(\cdot)$$

$$U = \begin{bmatrix} u^{(1)} \\ \vdots \\ u^{(N)} \end{bmatrix} \in \Re^{N_u} \tag{88}$$

where U is a weight vector of the dimension $N_u = n_u N$. We further assume that the shape function set is given and vector U defines the feedforward control on the interval $[0, T]$. We will call U an *input vector*.

Similarly, we describe the system output with a finite-dimensional output vector Y. We introduce a sampling time sequence $\{t_j\}_{j=1}^{L}$, where $t_L = T_f \geq T$ and an *output vector*

$$Y = \begin{bmatrix} y(t_1) - y_d(t_1; \lambda, \nu) \\ \vdots \\ y(t_L) - y_d(t_L; \lambda, \nu) \end{bmatrix} \in \Re^{N_y} \tag{89}$$

where $y \in \Re^{n_y}$ is the system measurement vector (85), $y_d(t)$ is the desired (planned) output, and $N_y = n_y L$. In most practically important cases, under reasonable observability conditions imposed on the sampling time sequence and on the controlled system, we can evaluate the desired output of the system (84), (85) by monitoring only the sampled output (89). Due to signal sampling in a digital computer, the measured output is of the form (89) in most practical cases.

The input vector U defines the output vector Y in accordance with (84), (85), and (87)–(89). We can write this dependence in the form (3)

$$Y = f(U, p) \tag{90}$$

where f is a smooth mapping, which we assume to be unknown. The task parameter vector p in (3) has the form (87) and comprises the vectors λ and ν that define the initial condition and the control goal.

We will further consider a performance index (86) that has the form (2)

$$J(Y, U) = \|Y\|^2 + \rho\|U\|^2 \to \min. \tag{91}$$

Assume temporarily that the initial and final condition vectors λ and ν are exactly reproducible from run to run (p in (90) is fixed). Then, a solution to (91) can be found using an iterative NLS optimization algorithm (e.g., such as discussed in Section II). Each optimization step of such an algorithm corresponds to a completed motion of the system with the input vector U (88) applied and the output Y (89) collected. Thus, we have a *learning control* algorithm. Some learning control approaches also known as repetitive or "betterment" control were recently studied by a number of authors, mostly with regard to robotics applications, in [2], [3], [22], [25], and [29], and many other papers. A well-known practical limitation of these approaches is that after learning the control to follow a given trajectory, we still do not know control for following other, even close, trajectories.

The above stated parametric learning control problem overcomes this limitation. For this problem, the parameter p changes from motion to motion in accordance with the changing motion goals, which are generated externally to the learning (optimization) procedure. The parametric learning control problem (90), (91) is a parametric NLS optimization problem considered in Sections III and IV. The parametric optimal input $U_*(p)$ for this problem can be found using the algorithm presented in Section III. At each step of the algorithm, the output vector Y is obtained by actually completing the system motion with the computed input U.

A few published papers consider learning of input command *dependence* on task parameters [1], [14], [16], [21]. Unlike the cited work, the algorithm of Section III works for an arbitrary sequence of the task parameter vectors p and thus can be used for improving the system performance without restricting the sequence of the control tasks to be executed. This ability of the algorithm to cope with an arbitrary task parameter sequence is very important for practical implementation of the parametric learning control.

Note that the discretization of the input (88) and output (89) does not restrict the practical scope of the stated learning control problem compared to other published works. This is because a practical implementation of a learning algorithm using a digital computer would work with sampled control and system output data. More discussion on how the formulated parametric learning approach relates to the learning control literature can be found in [21]. Similarly to many on-line optimization and approximation algorithms considered in the neural network literature, the considered learning control application is biologically inspired by some models of human motor control [15].

B. Application Example: Control of Flexible Arm

In this section, we apply the formulated parametric NLS algorithm optimization to the fast motion control of a flexible-joint arm with no *a priori* knowledge of the system dynamics.

Let us consider a two-link articulated arm shown in Fig. 4. We assume that inertial drives placed in the arm joints are connected to the links through lumped elastic elements, and all motion is in a horizontal plane. We employ commonly made assumptions about the elastic-joint manipulator dynamics [40].

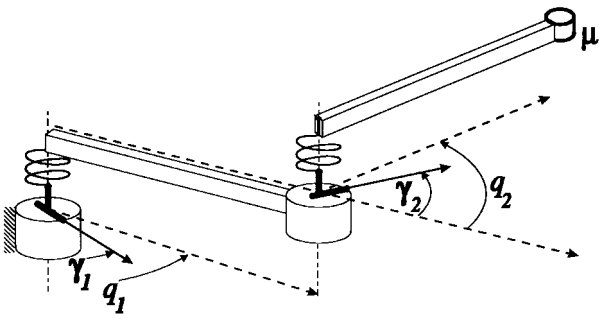

Fig. 4. Two-link planar arm with flexible joints.

In particular, we assume that the damping in the elastic elements is negligible and that angular motion of the drive rotors is decoupled from the arm structure motion. The latter assumption holds for drives with high transmission ratios.

Let $q \in \Re^2$ be the vector of the arm joint angles and $\gamma \in \Re^2$ the vector of the rotation angles for the output shaft of the drive. We further assume that the control torque vector τ applied to the drives is computed in the usual way, as a sum of proportional and derivative (PD) drive position feedback and feedforward compensation $u(t)$

$$\tau(t) = K_*(q_d(t) - \gamma(t)) + B_*(\dot{q}_d(t) - \dot{\gamma}(t)) + u(t) \quad (92)$$

where $q_d(t)$ is the reference drive position, $K_* \in \Re^{2,2}$ is the matrix of the proportional drive position feedback gains, and $B_* \in \Re^{2,2}$ is the matrix of the velocity feedback gains.

We consider the feedback gains K_* and B_* as fixed parameters and the vector of the feedforward joint torques $u(t) \in \Re^2$ as an external (control) input to the system. We assume that we can measure both the drive rotation angles q and the elastic element deformations $q - \gamma$. For the considered flexible manipulator system, the state vector has the form $x = [q^T \ \dot{q}^T \ \gamma^T \ \dot{\gamma}^T]^T \in \Re^8$ and the observation vector, $y = [q^T \ (q-\gamma)^T]^T \in \Re^4$.

Let us assume that the initial state vector has the form

$$x(0) = [\lambda^T \quad 0 \quad 0 \quad \lambda^T \quad 0 \quad 0]^T \quad (93)$$

where $\lambda \in \Re^2$ defines the initial joint angles of the arm and the drive angles. The initial condition (93) means that $q(0) = \gamma(0) = \lambda$ and $\dot{q}(0) = \dot{\gamma}(0) = [0 \ 0]^T$.

Let us consider the control goal parameter vector $\nu \in \Re^2$ that is equal to the desired joint angle vector $q_d(T)$ after the motion of the arm. The preplanned output $y_d(\cdot; \lambda, \nu)$ describes the desired path of the arm motion, which is used as a reference in the PD controller (92) and the planned joint deformations. The joint motion is initially planned as a straight line in the joint angle space and a third-order polynomial in time. We assume that the planned values of the joint deformations $q_d(t) - \gamma_d(t)$ and their derivatives are always zero. The control problem is to compute feedforward $u(\cdot)$ so that the arm comes to the final position $q_d(T) = \nu$ at time $T = 1.5$ without oscillations.

We divide the motion interval $[0, T]$ into seven equal subintervals $[\tau_j, \tau_{j+1}]$, $(j = 0, \cdots, 7)$, $\tau_0 = 0$, and $\tau_j = T$, and consider the feedforward input (88), $U \in \Re^{12}$, that is piecewise linear on these subintervals and zero at times zero and T. In other words, the shape functions $\phi_j(\cdot)$ in (88) are the first-order (triangular) B-splines. We monitor the arm motion on the interval $[T, T_f]$, $T_f = T + 0.5$ at $L = 14$ uniformly spaced output sampling instants $t_1 = T, \cdots, t_{14} = T_f$. The measurement vector y comprises drive angles and joint deformations, $y = [q^T \ (q - \gamma)^T]^T \in \Re^4$. By sampling the vector y at instants t_j, we obtain the output vector $Y \in \Re^{56}$.

The mapping (3) depends on the task parameter vector p (87) that includes the initial and desired final configurations of the arm. Since the system is cyclic, the control depends only on the *variation* of the first joint angle. Thus, we can write the task parameter vector p in the form

$$p = [q_{2d}(0) \quad q_{2d}(T) \quad (q_{1d}(T) - q_{1d}(0))]^T. \tag{94}$$

We consider the parametric NLS problem with the following task parameter vector domain:

$$\mathcal{P} = \left\{ p \in \mathcal{P}: \frac{\pi}{4} \le q_{2d}(0),\, q_{2d}(T) \le \frac{3\pi}{4}; \right. \\ \left. -\frac{\pi}{2} \le q_{1d}(T) - q_{1d}(0) \le \frac{\pi}{2} \right\}. \tag{95}$$

To implement the parametric NLS algorithm of Section III, we use approximation with the Gaussian RBF network with the fixed node centers $Q^{(j)}$ placed on a uniform mesh $5 \times 3 \times 3$ in the task parameter space. When *implementing* the control algorithm, we assume the dynamical model of the system to be completely unknown and set the initial estimate of the parameter matrix Θ in (54) to be zero.

The parametric NLS optimization algorithm was implemented as a Matlab program on a 486/66 computer, and the arm motion simulation was coded in C. The control (optimization) algorithm does not exploit any initial knowledge of the controlled system dynamics and uses just the input–output data. Given the input and output dimensions $U \in \Re^{12}$ and $Y \in \Re^{56}$ and the number of the RBF network nodes $N_a = 45$, the sizes of the matrices in (45) are $\overline{\Phi}(p, U) \in \Re^{585}$ and $\Theta \in \Re^{56,\,585}$. These sizes cause no computational problems, as the updates (52) and (54) only include matrix multiplications and the matrix inverted in (52) has the size 12×12.

For our Matlab implementation of the algorithm, the control update (52) took 0.33 s and the affine model update (54), 0.55 s. These computational delays are acceptable for the feedforward control since the updates need to be done only once for each motion. The computation of control in accordance with (53) takes less than 50 ms, which suggests that the proposed algorithm could also be feasible for feedback control, especially if the updates (52) and (54) are scheduled outside the time-critical feedback loop.

When *simulating* the planar arm motion, we assume that the arm links are uniform rods of unit mass and length. We take the moments of inertia of the drive rotors as $J = \text{diag}\ \{2, 2\}$, the damping in drives as $B = \text{diag}\ \{0, 0\}$, and the angular stiffnesses of the lumped elastic elements

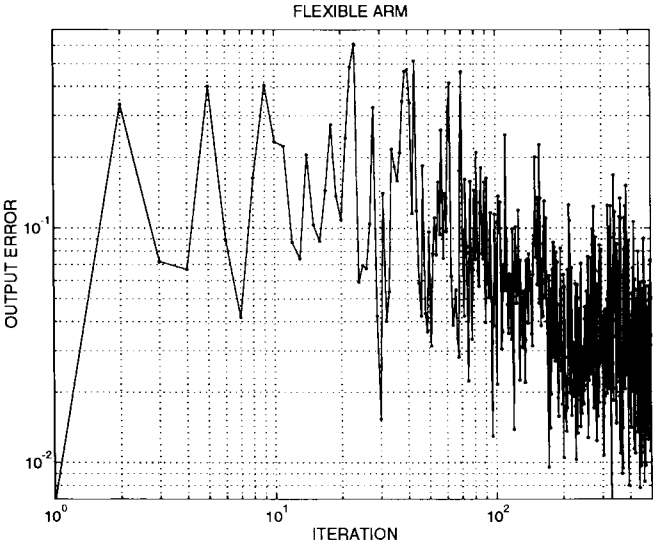

Fig. 5. The progress of the terminal error $\|Y\|$ with the optimization iteration number. The arm moves through a randomly generated goal positions sequence.

in the joints as $K = \text{diag}\ \{200, 200\}$. We further assume that the angular position gain of the PD feedback controller (92) is $K_* = \text{diag}\ \{100, 100\}$, and the angular velocity gain is $B_* = \text{diag}\ \{40, 40\}$. Note that for the above parameters of the system, the period of oscillations with the lowest eigenfrequency is about one if the elbow angle is $3\pi/4$. The motion time $T = 1.5$ is close to this period, which makes the control problem very difficult. We have found that adding a small measurement noise to the simulated system output does not change the algorithm performance in any visible way. The reason is that for a random parameter vector sequence $p^{(k)}$, the deviation of the system mappings from the RBF approximations used by the algorithm acts in the same way as an output noise.

In the numerical experiment, the values of the task parameter vector p are generated so that the initial arm configuration coincides with the final arm configuration at the end of the previous task. Fig. 5 shows the progress of the error $\|Y - Y_d\|$ with the optimization iteration number. One can see that the control error converges to a small acceptable value over the whole parameter vector domain. The error, which is achieved at the end of the optimization process, is about 20 times less than the initial error, which is obtained without feedforward. The oscillations of the motion error in Fig. 5 are related to the variation in the arm motion amplitude as new task parameter vectors p are randomly generated in the course of the learning.

Fig. 6 illustrates the feedforward control computed as a result of the RBF network approximation after the algorithm convergence for the motion with the initial joint angles $\lambda = [0^\circ \ 60^\circ]^T$ and the final angles $\nu = [70^\circ \ 105^\circ]^T$. Fig. 7 shows the joint deformations for the same motion. Thanks to the computed feedforward, the deformation is small after the time $T = 1.5$, which means the arm arrives to the final position without visible oscillations. The acceptable motion accuracy is achieved despite the high motion speed, low feedback gains, moderate network size, and large covered domain of

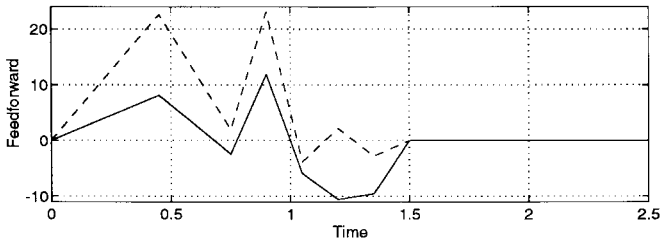

Fig. 6. Feedforward for a test motion after the algorithm convergence: shoulder joint–solid, elbow joint–dashed.

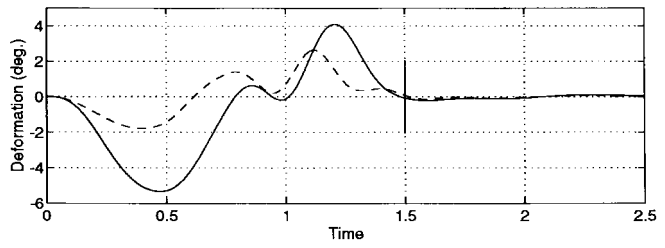

Fig. 7. Joint deformations for the test motion with the approximated feedforward: shoulder joint–solid, elbow joint–dashed.

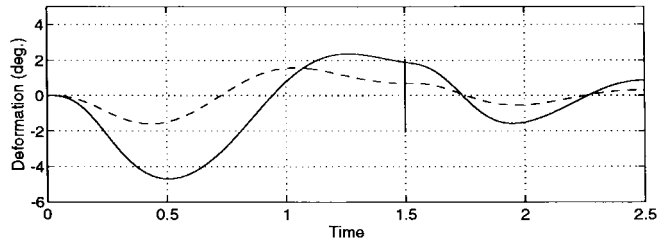

Fig. 8. Joint deformations for the test motion with zero feedforward: shoulder joint–solid, elbow joint–dashed.

the task parameters (95). For comparison, Fig. 8 shows the deformations for the same motion in the absence of the feedforward.

VII. Conclusions

This paper has formulated a problem of on-line parametric NLS optimization, and proposed a technique for its solution. The algorithm proposed and results obtained could be used in nonlinear control, signal processing, and optimization. The problem and the proposed solution are related to recent work in control using on-line function approximation including neural networks, fuzzy, and learning systems.

The proposed algorithm constitutes an extension of the well-known Levenberg–Marquardt algorithm and includes on-line approximation of the nonlinear mappings encountered in the optimization process. We have proved local convergence of the proposed algorithm. Though different types of the approximation techniques can be used with the algorithm, in examples we have successfully used a RBF network expansion.

We have demonstrated an application of the proposed approach to learning control of a flexible-joint arm over the entire workspace of the arm. The controlled motions are very fast and take about 1.5 periods of the lowest eigenfrequency oscillations. The proposed approach solves the problem very efficiently. Without any prior knowledge of the system dynamics, it achieves satisfactory control of arbitrary arm motions after only 500 iterations of learning (optimization). The algorithm can be efficiently implemented for on-line use.

Appendix

Proof of Lemma 1: Let us denote by vec $(\tilde{K}^{(k)})$ the vector built of all entries of the matrix $\tilde{K}^{(k)}$, column by column. In accordance with the Kronecker product properties [4, Sec. 5.6], (70) can be represented in the form

$$\text{vec}\,(\tilde{K}^{(k+1)}) = \Psi_k\ \text{vec}\,(\tilde{K}^{(k)}) + \text{vec}\,(e^{(k)}) \tag{96}$$

$$\Psi_k = F^{(k)} \otimes I_{1+N_U} + (I_{N_a} - F^{(k)}) \otimes A^{(k)}. \tag{97}$$

We will need an l-step solution to (96). This solution has the form

$$\text{vec}\,(\tilde{K}^{(k+l)}) = \Psi(k+l,\, k)\ \text{vec}\,(\tilde{K}^{(k)}) + \sum_{j=0}^{l-1} \Psi(k+l,\, k+l-j)\ \text{vec}\,(e^{(k+l-j)}) \tag{98}$$

$$\Psi(k,\, j) = \prod_{i=k}^{j-1} \Psi_i \tag{99}$$

where $\Psi(k,\, j)$ is the transition matrix for the homogeneous part of the system (96).

Since the matrix $A^{(k)}$ (71) is symmetrical positive definite, and $F^{(k)}$ satisfies (69), the matrix Ψ_k (97) is also symmetrical positive definite with

$$\Psi_k^{1/2} = F^{(k)} \otimes I_{1+N_U} + F_c^{(k)} \otimes A^{(k)^{1/2}}$$
$$F_c^{(k)} = I_{N_a} - F^{(k)}. \tag{100}$$

Let us show that $||\Psi_k|| \leq 1$, which is equivalent to showing that $||\Psi_k\ \text{vec}\,(\tilde{K})|| \leq ||\ \text{vec}\,(\tilde{K})||$ for any vec $(\tilde{K}) \in \Re^{N_U, N_a}$. Note that

$$\begin{aligned} ||\text{vec}\,(\tilde{K})||^2 &= ||\tilde{K}||_F^2 \\ &= \text{trace}(\tilde{K}^T \tilde{K}) \\ &= \text{trace}(\tilde{K} \tilde{K}^T). \end{aligned} \tag{101}$$

By using representation (70) of (96) (with $e^{(k)} \equiv 0$), (69), (71), and (101), we obtain a chain of inequalities

$$\begin{aligned} &||\Psi_k\ \text{vec}\,(\tilde{K})||^2 \\ &\quad = ||\tilde{K} F^{(k)} + A^{(k)} \tilde{K} F_c^{(k)}||_F^2 \\ &\quad = \text{trace}(\tilde{K} F^{(k)} \tilde{K}^T) + \text{trace}(A^{(k)} \tilde{K} F_c^{(k)} \tilde{K}^T A^{(k)}) \\ &\quad = \text{trace}(\tilde{K} F^{(k)} \tilde{K}^T) + \text{trace}(F_c^{(k)} \tilde{K}^T A^{(k)^2} \tilde{K} F_c^{(k)}) \\ &\quad \leq \text{trace}(\tilde{K} F^{(k)} \tilde{K}^T) + \frac{\mu^2}{(\mu+\rho)^2} \text{trace}(F_c^{(k)} \tilde{K}^T \tilde{K} F_c^{(k)}) \\ &\quad = ||\tilde{K}||_F^2 - \phi \text{trace}(\tilde{K} F_c^{(k)} \tilde{K}^T) \leq ||\tilde{K}||_F^2 \end{aligned} \tag{102}$$

where $\phi = 1 - \mu^2/(\mu+\rho)^2 \geq 0$. The fact that $||\Psi_k|| \leq 1$ together with (98) proves that the sum in the right-hand side (RHS) of (98) can be majorated by the sum in the RHS of (75).

Lemma 1 will be proved if we show that for some $\epsilon > 0$ the following inequality is valid:

$$||\Psi(k+l, k)|| < 1 - \epsilon. \tag{103}$$

The rest of the Lemma 1 proof will be devoted to demonstrating (103). Note that proving (103) is equivalent to proving that for $e^{(k)} = 0$ a solution of (70) satisfies

$$||\text{vec } (\tilde{K}^{(k+l)})||^2 \le (1-\varepsilon)||\text{vec } (\tilde{K}^{(k)})||^2 \tag{104}$$

where $\varepsilon = 1 - (1-\epsilon)^{1/2}$.

We will prove the counteropposite to (104). Let us assume that for any $\varepsilon > 0$ we can find $\tilde{K}^{(k)}$ so that

$$||\tilde{K}^{(k+j)}||_F^2 \ge (1-\varepsilon)||\tilde{K}^{(k)}||_F^2, \qquad (j = 1, \cdots, l-1). \tag{105}$$

Then for $j = k, \cdots, k+l-1$, by using the facts that Ψ_k is positive definite and $||\Psi_k|| \le 1$, as well as (105), we get

$$\begin{aligned} &||\text{vec } (\tilde{K}^{(j+1)}) - \text{vec } (\tilde{K}^{(j)})||_F^2 \\ &\quad = ||\text{vec } (\tilde{K}^{(j)})||_F^2 + ||\Psi_j \text{ vec } (\tilde{K})^{(j)}||_F^2 \\ &\qquad - 2 \text{ vec } (\tilde{K}^{(j)})^T \Psi_j \text{ vec } (\tilde{K}^{(j)}) \\ &\quad \le 2||\text{vec } (\tilde{K}^{(j)})||^2 - 2 \text{ vec } (\tilde{K}^{(j)})^T \Psi_j^2 \text{ vec } (\tilde{K})^{(j)} \\ &\quad \le 2||\tilde{K}^{(j)}||_F^2 - 2||\tilde{K}^{(j+1)}||_F^2 \le 2\varepsilon||\tilde{K}^{(j)}||_F^2. \end{aligned} \tag{106}$$

By successively applying (106) for $j = k, \cdots, k+j-1$, we obtain

$$\begin{aligned} \tilde{K}^{(k+j)} &= \tilde{K}^{(k)} + \Delta^{(k,j)} \\ ||\Delta^{(k,j)}||_F &\le j\sqrt{2\varepsilon}||\tilde{K}^{(k)}||_F. \end{aligned} \tag{107}$$

The chain of inequalities (102) gives

$$\begin{aligned} ||\tilde{K}^{(j+1)}||_F^2 &= ||\Psi_j \text{ vec } (\tilde{K}^{(j)})||^2 \\ &\le ||\tilde{K}^{(j)}||_F^2 - \phi \text{trace}(\tilde{K}^{(j)} F_c^{(j)} \tilde{K}^{(j)^T}). \end{aligned} \tag{108}$$

By successively applying (108) for $k, k+1, \cdots, k+l-1$, we obtain

$$\begin{aligned} &||\tilde{K}^{(k+l)}||_F^2 \\ &\quad \le ||\tilde{K}^{(k)}||_F^2 - \sum_{j=1}^{l-1} \phi^j \text{trace}(\tilde{K}^{(k+j)} F_c^{(k+j)} \tilde{K}^{(k+j)^T}) \\ &\quad \le ||\tilde{K}^{(k)}||_F^2 - \phi^{l-1} \sum_{j=1}^{l-1} \text{trace}(\tilde{K}^{(k+j)} F_c^{(k+j)} \tilde{K}^{(k+j)^T}). \end{aligned} \tag{109}$$

Now, by using (105) and (107), we obtain from (109)

$$\begin{aligned} \varepsilon||\tilde{K}^{(k)}||_F^2 &\ge \phi^l \sum_{j=1}^{l-1} \text{trace}((\tilde{K}^{(k)} + \Delta^{(k,j)}) F_c^{(k+j)} \\ &\qquad \cdot (\tilde{K}^{(k)} + \Delta^{(k,j)})^T) \\ &\ge \phi^l \sum_{j=1}^{l-1} \text{trace}(\tilde{K}^{(k)} F_c^{(k+j)} \tilde{K}^{(k)^T}) \\ &\qquad - O(\sqrt{\varepsilon}||\tilde{K}^{(k)}||_F) - O(\varepsilon). \end{aligned} \tag{110}$$

Finally, by remembering that $F_c^{(j)} = I_{N_a} - F_c^{(j)} = \Phi^{(j+1)}\Phi^{(j+1)^T}/||\Phi^{(j+1)}||^2$ and using (74), we obtain

$$\begin{aligned} &\sum_{j=1}^{l} \text{trace}(\tilde{K}^{(k)} F_c^{(k+j)} \tilde{K}^{(k)^T}) \\ &\quad = \text{trace}\left(\tilde{K}^{(k)} \left(\sum_{j=1}^{l} F_c^{(k+j)}\right) \tilde{K}^{(k)^T}\right) \\ &\quad \ge ||\tilde{K}^{(k)}||_F^2 N_a \delta. \end{aligned} \tag{111}$$

Since δ in (111) is fixed and ε in (110) can be made arbitrarily small, we came to a contradiction.

REFERENCES

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

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

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

[4] S. Barnett, *Matrices, Methods and Applications.* Oxford: Clarendon, 1990.

[5] M. S. Branicky, "Task-level learning: Experiments and extensions," in *Proc. 1991 IEEE Int. Conf. Robotics Automation*, Sacramento, CA, Apr. 1991, pp. 266–271.

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

[7] S. Chen, S. A. Billings, and P. M. Grant, "Recursive hybrid algorithm for nonlinear systems identification using radial basis function networks," *Int. J. Contr.*, vol. 55, no. 5, pp. 1051–1070, 1992.

[8] J. Craig, *Introduction to Robotics*, 2nd ed. New York: Addison-Wesley, 1989.

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

[10] J. A. Farrel, T. Berger, and B. D. Appleby, "Using learning technique to accommodate unanticipated faults," *IEEE Contr. Syst. Mag.*, vol. 15, no. 3, pp. 16–24, 1995.

[11] R. Franke, "Scattered data interpolation: Test of some methods," *Math. Comput.*, vol. 38, no. 157, pp. 181–200, 1982.

[12] F. R. Gantmacher, *The Theory of Matrices.* New York: Chelsea, 1959.

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

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

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

[16] ——, "Learning open-loop terminal control," in *Proc. Amer. Contr. Conf.*, Chicago, IL, June 1992, vol. 3, pp. 1900–1901.

[17] ——, "On the persistency of excitation in radial basis function network identification of nonlinear systems," *IEEE Trans. Neural Networks*, vol. 4, no. 5, pp. 1237–1244, 1995.

[18] D. M. Gorinevsky and T. H. Connolly, "Comparison of some neural network and scattered data approximations: The inverse manipulator kinematics example," *Neural Computation*, vol. 6, no. 3, pp. 519–540, 1994.

[19] D. M. Gorinevsky, A. Kapitanovsky, and A. A. Goldenberg, "Radial basis function network architecture for nonholonomic motion planning and control of free-flying manipulators," *IEEE Trans. Robotics Automation*, vol. 12, no. 3, pp. 491–496, 1996.

[20] ——, "Neural network architecture and controller design for automated car parking," *IEEE Trans. Contr. Syst. Technol.*, vol. 4, no. 1, pp. 50–56, 1996.

[21] D. Gorinevsky, D. Torfs, and A. A. Goldenberg, "Learning approximation of feedforward dependence on the task parameters: Experiments in direct-drive manipulator tracking," in *Proc. Amer. Contr. Conf.*, Seattle, WA, June 1995, pp. 883–887.

[22] S. Hara, Y. Yamamoto, T. Omata, and M. Nakano, "Repetitive control systems: A new type of servo systems for periodic exogenous signals," *IEEE Trans. Automat. Contr.*, vol. 33, no. 7, pp. 659–668, 1988.

[23] 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.
[24] H. T. Jongen and G.-W. Weber, "On parametric nonlinear programming," *Ann. Operation Res.,* vol. 27, pp. 253–284, 1990.
[25] W. Messner *et al*., "A new adaptive learning rule," *IEEE Trans. Automat. Contr.,* vol. 36, no. 2, pp. 188–197, 1991.
[26] H. Michalska and D. Q. Mayne, "Robust receding horizon control of constrained nonlinear systems," *IEEE Trans. Automat. Contr.,* vol. 38, no. 11, pp. 1623–1633, 1993.
[27] P. Moraal and J. W. Grizzle, "Observer design for nonlinear systems with discrete-time measurements," *IEEE Trans. Automat. Contr.,* 1995.
[28] B. Bank *et al*., *Non-Linear Parametric Optimization.* Basel: Birkhäuser, 1983.
[29] S. R. Oh, Z. Bien, and I. H. Suh, "An iterative learning control method with application for the robot manipulator," *IEEE J. Robotics Automation,* vol. 4, no. 5, pp. 508–514, 1988.
[30] T. Parizini and R. Zoppoli, "Neural approximations for multistage optimal control of nonlinear stochastic systems," *IEEE Trans. Automat. Contr.,* vol. 41, no. 6, pp. 889–895, 1996.
[31] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation,* vol. 3, pp. 246–257, 1991.
[32] K. M. Passino, "Intelligent control for autonomous systems," *IEEE Spectrum,* vol. 32, pp. 55–62, June 1995.
[33] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE,* vol. 78, no. 9, pp. 1481–1497, 1990.
[34] B. T. Polyak, *Introduction to Optimization.* New York: Optimization Software, 1987.
[35] M. M. Polycarpou and A. T. Vemuri, "Learning methodology for failure detection and accommodation," *IEEE Contr. Syst. Mag.,* vol. 15, no. 3, pp. 16–24, 1995.
[36] A. B. Poorey, "Bifurcations in parametric nonlinear programming," *Ann. Operation Res.,* vol. 27, pp. 443–370, 1990.
[37] M. J. D. Powell, "Radial basis functions for multivariable interpolation: A review," in *Algorithms for Approximation,* J. C. Mason and M. G. Cox, Eds. Oxford: Clarendon, 1987, pp. 143–168.
[38] ——, "The theory of radial basis function approximation in 1990," in *Advances in Numerical Analysis,* W. Light, Ed. Oxford: Clarendon, 1992, vol. 2, pp. 102–205.
[39] R. M. Sanner and J.-J. E. Slotine, "Gaussian networks for direct adaptive control," *IEEE Trans. Neural Networks,* vol. 3, no. 6, pp. 837–863, 1992.
[40] M. Spong, "Modeling and control of elastic joint robots," *Trans. ASME J. Dynam. Syst. Meas. Contr.,* vol. 109, no. 4, pp. 310–319, 1987.
[41] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems.* Washington: Halsted, 1977.
[42] M. Togai and O. Yamano, "Learning control and its optimality," in *Proc. 1986 IEEE Conf. Robotics Automation,* San Francisco, CA, 1986, pp. 248–243.
[43] Y. Z. Tsypkin, *Foundations of the Theory of Learning Systems.* New York: Academic, 1973.

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

He was with the USSR Academy of Sciences, Russia, the Munich Unversity of Technology, Germany, and the University of Toronto, Canada, and he consulted for governmental and industrial organizations. He is currently with Honeywell-Measurex, North Vancouver, B.C., where he develops advanced algorithms and applications for process control. He is also an Adjunct Professor in the Department of Electrical and Computer Engineering at University of British Columbia and a consultant for the Canadian Space Agency. His interests include advanced control applications, intelligent and learning systems, and biological control systems. He has authored and coauthored 80 technical papers and book chapters, a book published in English and Russian, and has six patents received or pending.

Dr. Gorinevsky was a recipient of the Alexander von Humboldt International Research Fellowship and a Canada Research Fellowship. He is a member of several professional organizations and a registered Professional Engineer in Ontario.