

## Modelling of direct motor program learning in fast human arm motions

Dimitry M. Gorinevsky \*

Lehrstuhl B für Mechanik, Technische Universität München, Postfach 20 24 20, W-8000 München, Germany

Received: 6 April 1992/Accepted in revised form: 18 January 1993

**Abstract.** We propose and simulate a new paradigm for organization of motor control in fast and accurate human arm motions. We call the paradigm "*direct motor program learning*" since the control programs are learned directly without knowing or learning the dynamics of a controlled system.

The idea is to approximate the dependence of the motor control programs on the vector of the task parameters rather than to use a model of the system dynamics. We apply iterative learning control and scattered data multivariate approximation techniques to achieve the goal. The advantage of the paradigm is that the control complexity depends neither on the order nor on the nonlinearity of the system dynamics.

We simulate the direct motor program learning paradigm in the task of point-to-point control of fast planar human arm motions. Simulation takes into account nonlinear arm dynamics, muscle force dynamics, delay in low-level reflex feedback, time dependence of the feedback gains and coactivation of antagonist muscles. Despite highly nonlinear time-variant dynamics of the controlled system, reasonably good motion precision is obtained over a wide range of the task parameters (initial and final positions of the arm). The simulation results demonstrate that the paradigm is indeed viable and could be considered as a possible explanation for the organization of motor control of fast motions.

### 1 Introduction

Physiological studies of human motor control give strong evidence that at least fast human arm motions are open-loop controlled; see, for example, the survey by Adamovitch et al. (1990). The most general evidence

could be derived from the fact that a reflex positional feedback has a significant delay and a large feedback gain cannot be achieved. Thus, though being important for disturbance compensation, the reflex feedback alone could hardly provide the required precision of fast motions. A feedforward control (motor command) should be added to the feedback. The relation of the activation pattern of muscles to the resulting motion is rather complex, and it is not clear how this motor command is computed. A widespread belief is that it is somehow learned.

Several researchers have recently simulated, or experimentally studied, some paradigms of motor control learning, either in application to a human arm or to a robot manipulator motion; e.g. see Kawato (1989); Guez and Selinsky (1988); Kano and Takayama (1990); Yabuta and Yamada (1990); Dornay et al. (1991); Katayama and Kawato (1991) and the survey of Sanches and Hirzinger (1991). These papers consider some schemes for learning the inverse dynamics of the arm. With known inverse dynamics, a feedforward control could be computed from the pre-planned arm motion. Though providing important insights into the problem, the cited papers use oversimplified models for fast human arm motions. Delayed reflex feedback with time-variant gains and nonlinear dynamics of muscle actuation in a biological system make inverse dynamics compensation a far more difficult problem than considered there.

In this paper we propose a new paradigm for motor control learning of fast human arm motions. The paradigm that we call "*direct motor program learning*" is based on a rather straightforward approach related to some physiological ideas on (*generalized*) motor program control of human movements see, e.g. Adamovitch et al. (1990).

To explain the paradigm, let us consider a motor control task described by a vector of parameters; for instance, the initial and final positions of a moving arm could be components of such a vector. Suppose that we have already learned and stored in memory (open-loop) motor control programs for some values of the parame-

\* Present address: Robotics and Automation Laboratory, University of Toronto, 5 King's College Road, Toronto M5S 1A4, Canada

ter vector. Then we can use these programs in an approximation (interpolation) procedure to compute control for other values of the parameters.

The direct motor program learning paradigm was applied by Gorinevsky (1991) to the point-to-point control of a simulated two-link manipulator with joint elasticity. Gorinevsky (1992a) experimentally implemented the paradigm to solve a manipulator path tracking problem.

In this paper we consider how control of fast point-to-point motions of a simulated human arm could be organized within the paradigm. Unlike other papers on the simulation of motor control learning, we regard an arm model that includes highly nonlinear actuators (muscles), reflex positional feedback featuring a significant delay, nonlinear dynamics and time-varying gains. It is very hard to cope with such a system using standard automatic control approaches, even if all parameters of the model are known precisely. It is difficult, if possible at all, to learn the inverse dynamics of such a "bad" object with one of the previously proposed approaches, since a system with delayed feedback has infinite-dimensional state-space representation. Furthermore, not all components of the state vector are being directly measured. Our primary goal is not to provide a more accurate explanation of the available physiological data, but rather to demonstrate that the proposed paradigm copes with control of such a human arm model, which includes realistic features, without using a mathematical model of the arm dynamics. However, we hope that the results could provide a key to better understanding of human motor control principles.

We implement the paradigm in several steps, description of which comprises the main body of this paper. The three steps are:

1. Compact representation of the control programs that are needed to store the programs in a limited memory space
2. Iterative learning of the feedforward control programs without using a model for the system dynamics
3. Approximation of a vector-valued multivariate mapping; feedforward control program dependence on several task parameters for the mapping values being known at some scattered points

All three steps are further described. The layout of the paper is as follows: In Sect. 2 we introduce input and output parameters for the controlled object – a biomechanical model for human arm – and present a formal statement of the control problem. As a result of discretization, we build a finite input/output representation of the control task. Section 3 considers a procedure for adaptive learning of the motor program for a single motion. Section 4 briefly explains a method for approximating control programs over the task parameter domain. Section 5 describes in some detail the arm model used in the simulation. Finally, Sect. 6 presents and discusses the results of the simulation in learning motor control programs for fast arm motions.

## 2 Representation of the control task

As a controlled system, we consider a biomechanical model for two-joint human arm motions in a horizontal plane. Our first objective in the model choice was to take into account major features of human arm dynamics that make it a much more complicated system than a manipulator arm. The second objective was to use a standard model formulation previously discussed in other papers in order not to distract the reader's attention from the control paradigm formulation. Our biomechanical model mostly follows the model of van Soderen and Denier van der Gon (1990) and we formulate it in more detail in Sect. 5. In this section we formulate an input/output representation of the considered motor control task that is used in our paradigm.

### 2.1 General

Let us consider a two-link arm moving in a horizontal plane. The arm has two simple rotational joints: shoulder and elbow. We denote by  $\phi^{(1)}$  and  $\phi^{(2)}$  the respective joint angles that we count as the angles between the next link and the extension of the previous link. We assume that each of the two arm joints is powered by a pair of equivalent muscles: an equivalent flexor and an equivalent extensor. This muscle pair exerts a joint torque that is defined by an activation pattern of the muscles.

We consider a motor control task of moving the arm fast from one given position to another. At initial time  $t = 0$  the arm rests, and the joint angles are  $\phi_0^{(1)}$  and  $\phi_0^{(2)}$ . At final time  $t = T$  it should be resting in a new position with joint angles  $\phi_T^{(1)}$  and  $\phi_T^{(2)}$ . We assume that to control the torque of joint  $i$ , an upper-level input signal  $\Phi^{(i)}$  is used which is distributed between the flexor and extensor muscles of the joint so that both muscles are nonnegatively activated. The signal  $\Phi^{(i)}$  is a sum of a positional feedback and a feedforward input (motor command)  $u^{(i)}$ . For more details of the model see Sect. 5.

If we apply no feedforward control to the arm, it will still move to the final position due to the reflex feedback. However, this motion is relatively slow and with an overshoot or undershoot. The control task is to find two feedforward control programs  $u^{(1)}(\cdot)$  and  $u^{(2)}(\cdot)$  for the shoulder and elbow joints so that the arm comes to a standstill at the final point precisely at the given time  $T$ .

Our feedforward learning paradigm uses a black-box representation of the controlled system, so we do not need to describe the system in detail now. The algorithm rather needs a description of the input and output variables.

### 2.2 Parameterization of the input and output

Continuous-time functions  $u^{(i)}(\cdot)$  describing the feedforward input are defined by an infinite number of parameters. We search for the solution of the control problem among a certain family of feedforward programs depending on a reasonably small number of parameters.

Let us consider  $N_u$  sampling instants

$$0 \leq t_j \leq T, \quad (j = 1, N_u) \quad (1)$$

and feedforward control of the form

$$\begin{aligned} u^{(1)}(t) &= \sum_{k=1}^{N_u} U_{2k-1} B_k(t) \\ u^{(2)}(t) &= \sum_{k=1}^{N_u} U_{2k} B_k(t) \end{aligned} \quad (2)$$

where  $U_k$  are parameters of the control and  $B_k(t)$  are the "shape functions". For instance, one can choose  $B_k$  to be B-spline functions (see Gorinevsky (1992a)), but here we suppose that  $B_k$  is an indicator function for the interval  $[t_{k-1}, t_k]$ .

$$B_k(t) = \begin{cases} 0, & \text{if } t < t_{k-1} \text{ or } t \geq t_k \\ 1, & \text{if } t_{k-1} \leq t < t_k \end{cases} \quad (3)$$

According to (1)–(3), feedforward control programs are piece-wise constant functions with values  $U_{2k-1}$  and  $U_{2k}$  between sampling instants  $t_{k-1}$  and  $t_k$ . The feedforward controls (2) are defined by a  $2N_u$ -dimensional vector

$$U = \text{col}(\{U_k\}_{k=1}^{2N_u}) \quad (4)$$

that we further call an input or control vector of the task.

A motor command pattern for a fast human arm movement is modeled with a train of rectangular pulses also in a paper by Wu et al. (1992), where results of such modeling are fit to experimental data.

We further consider arm motion at the time interval  $[0, T_f]$ ,  $T_f > T$ . For  $t > T$  we compute feedforward as  $u^{(i)}(t) = 0$ , ( $i = 1, 2$ ). To discretize the output of the control task, we consider  $N_y$  equidistant sampling instants  $\theta_j$  just after the control interval  $[0, T]$ .

$$T \leq \theta_j \leq T_f, \quad (j = 1, N_y), \quad \theta_j - \theta_{j-1} = \tau \quad (5)$$

Let us consider an output vector  $Y \in \mathbb{R}^{2N_y}$

$$Y = \left\{ \begin{bmatrix} \phi^{(1)}(\theta_j) - \phi_s^{(1)} \\ \phi^{(2)}(\theta_j) - \phi_s^{(2)} \end{bmatrix} \right\}_{j=1}^{N_y} \quad (6)$$

that consists of the sampled joint-angle deviations from their final values. If the arm comes to the final position at time  $T$  without an overshoot, then vector  $Y$  is zero. Note that within our paradigm we can consider other forms of output vector  $Y$ , e.g., one that includes deviation of the tip motion from a preplanned path (see Gorinevsky 1992a).

The input vector  $U$  and the output vector  $Y$  are related by a complex non-linear mapping  $H: \mathbb{R}^{2N_u} \mapsto \mathbb{R}^{2N_y}$

$$Y = H(U) \quad (7)$$

Each point of the mapping could be obtained by applying feedforward control (1)–(4) to an arm motion and observing the output data (5), (6).

### 2.3 Performance index

We can now reformulate the control problem as searching for input  $U$  (4) so that output  $Y$  (6) is in some sense minimal. Assuming that the controlled system is observ-

able, we can conclude that if the output sampling interval  $\tau$  is sufficiently small,  $\|Y\| \rightarrow 0$  means that the arm comes to a standstill at the desired final state. The controlled system includes a delay and, thus, is infinite-dimensional from the control theory viewpoint. Therefore, no control of the form (2) can move it precisely to the desired final state. However, we only need to solve the problem with some (not very high) finite accuracy.

We proceed as in Gorinevsky (1991, 1992a). Let us introduce a performance index minimization problem that describes the task

$$J = \frac{1}{2} \|Y\|^2 + \frac{\rho}{2} \|U\|^2 \rightarrow \min \quad (8)$$

where  $\rho > 0$  is a scalar and  $\|\cdot\|$  denotes the Euclidean norm of a vector.

If the controlled system is finite-dimensional, for  $\rho \rightarrow 0$  the solution to (8) is close to a quadratic-optimal solution of the terminal control problem with  $\|Y\| = 0$ ; however, for finite  $\rho > 0$  the problem (8) is better posed and could be solved for an infinite-dimensional system (see Tikhonov and Arsenin 1979; Gorinevsky 1992b).

The quadratic optimality criterion (8) is of course an arbitrary assignment. According to some models (e.g., Flash and Hogan 1985; Uno et al. 1989; Dornay et al. 1991), the human arm is controlled to minimize squared jerk, torque change rate or muscle tension change rate rather than energy. (Jerk is the rate of acceleration change.) Since the acceleration is proportional to the control torque, the mentioned criteria are closely interrelated. Our framework can be easily adapted to other forms of the optimality criterion (8). For instance, by changing  $\|U\|^2$  in (8) to  $U^T W U$ , where  $W$  is a convenient tridiagonal weighting matrix, one can obtain a criterion for a minimal squared torque change rate. Our simulation results demonstrate that though the performance index (8) does not explicitly include arm trajectory, it results in sufficiently smooth and realistic arm motion.

We further consider the feedback gains, other control parameters and the control time  $T$  (1) to be the same for different motions. Under these conditions we can describe our motor control task by a four-dimensional parameter vector

$$p = \text{col}(\phi_s^{(1)}, \phi_s^{(2)}, \phi_b^{(2)}, \phi_b^{(3)}) \quad (9)$$

where  $\phi_s^{(1)}$  and  $\phi_s^{(2)}$  are the shoulder and elbow joint angles at the beginning of the motion and  $\phi_b^{(2)}$  and  $\phi_b^{(3)}$  are the desired joint angles at the end of the motion. Of course, the mapping (7) depends on the task parameter vector (9). We do not write this dependence explicitly, but rather imply it.

Our goal is to be able to design feedforward (motor command) programs  $u^{(i)}(\cdot)$ , ( $i = 1, 2$ ), for an arbitrary parameter vector (9) in a given domain.

## 3 Learning

### 3.1 Iterative method

We can minimize the performance index (8) iteratively. Let  $U_i$  be a guess at the solution to (8). Let an output

vector (6)  $Y_i = H(U_i)$  be experimentally obtained by applying the control  $U_i$  (2)–(4) during the arm motion.

We can build the next, better approximation  $U_o$  to the solution of (8) with the Newton–Raphson method

$$U_o = U_i - \left[ \frac{d^2 J}{dU^2} \right]^{-1} \frac{dJ}{dU} \quad (10)$$

where

$$\frac{dJ}{dU} = G^T \frac{\partial J}{\partial Y} + \frac{\partial J}{\partial U} = G^T Y + qU, \quad G = \frac{\partial Y}{\partial U} \quad (11)$$

The matrix  $G = \partial Y / \partial U$  in (11) is an input/output sensitivity matrix of the controlled plant. Columns of  $G$  mean the variation of the sampled output (7) corresponding to the unit variation of the respective component  $U_k$  of the input vector (4), i.e. variation of the control value at the interval  $[t_{k-1}, t_k]$ . The Hessian matrix in (10) has the form

$$\frac{d^2 J}{dU^2} = qI + G^T G + \frac{\partial G^T}{\partial U} Y \quad (12)$$

We neglect the last term in (12) since for  $\rho \ll 1$  the solution to (8) gives  $\|Y\| \ll 1$ . Furthermore, if we change the Hessian matrix in (10) to another positive-definite matrix, this does not spoil the convergence but only reduces its rate. So we use a modified Newton–Raphson method of the form

$$U^{(i+1)} = U^{(i)} - (qI + G^T G)^{-1} (qU^{(i)} + G^T Y^{(i)}) \quad (13)$$

where the upper index denotes iteration number. Each iteration includes trying the arm motion with control  $U^{(i)}$  (2)–(4) and obtaining output  $Y^{(i)}$  (5), (6).

We can only use the iterative learning method (13) if we know the input/output sensitivity matrix  $G$  for the controlled system. Section 3.2 and Sect. 4 describe methods for obtaining an estimate of  $G$ . The learning algorithm (13) is conceptually close to the repetitive control algorithms of Arimoto et al. (1984), Arimoto (1990), Togai and Yamano (1986), Oh et al. (1988) and Messner et al. (1991). The robustness of convergence of the proposed iterative method and the influence of the imprecise knowledge of  $G$  on the convergence point have been considered by Gorinevsky (1992a).

### 3.2 Adaptation (on-line estimation of $G$ )

As follows from Sect. 3.1, to achieve convergence of the learning procedure to the solution of (8) we should know the input/output sensitivity matrix  $G$  of the task. Though we do not precisely know the plant input/output properties in advance, we can try to estimate them (or to improve an available estimate) in the course of learning.

Let us denote

$$w^{(i)} = U^{(i)} - U^{(i-1)}, \quad z^{(i)} = Y^{(i)} - Y^{(i-1)} \quad (14)$$

variations of the plant input and output, respectively, at learning iteration  $i$ . We suppose that

$$z^{(i)} = G w^{(i)} \quad (15)$$

and try to improve an estimate  $\hat{G}^{(i)}$  of the matrix  $G$  from data (14). We modify the estimate so that (15) holds precisely for it and that for any  $w$  orthogonal to  $w^{(i)}$  the estimate of  $Gw$  does not change. We obtain the following kind of stochastic approximation procedure:

$$\hat{G}^{(i+1)} = \hat{G}^{(i)} - \frac{w^{(i)T}}{\beta + \|w^{(i)}\|^2} [\hat{G}^{(i)} w^{(i)} - z^{(i)}], \quad (16)$$

where  $\beta$  is a small positive constant.

Local convergence of the adaptive learning algorithm (13) follows from the fact that if the mapping (7) is linearized in the vicinity of the optimum, the error of estimate (16) does not increase. Thus the Newton–Raphson method is convergent in the vicinity of the extremum for any  $G$  in (13).

In fact, (16) is a variation of the recursive projection estimation algorithm popular in adaptive control (Goodwin and Sin 1984) and (13), (16) could be considered as a one-step ahead adaptive control of the system (7). This makes available proofs of the adaptive control algorithm convergence applicable. To ensure persistency of excitation, we add a small self-excitation signal to (13).

## 4 Approximation

Let us assume that we already know a feedforward vector  $U$  (2)–(4) and an input/output sensitivity matrix  $G$  (11) for some motions, i.e. for some values of the parameter vector (9). We can use the learned data to build an approximation for the vector  $U$  and the matrix  $G$  over the domain of vector  $p$ . If the computed approximation for  $U$  does not provide a sufficiently small motion error, only a few learning iterations (13) with known approximation of  $G$  would be sufficient to achieve the desired accuracy.

The considered approximation problem is nontrivial. The typical dimension of the parameter vector could be between 2 and 10. (In the control task that we are considering,  $\dim p = 4$ .) The dimension of vector  $U$  could be between 10 and 100 and the dimension of matrix  $G$  could be up to  $100 \times 200 = 2 \cdot 10^4$ . A number of papers deal with scattered data approximation of multivariate functions, but they mostly consider scalar-valued functions. Such an approximation problem is nowadays considered as a typical problem for application of artificial neural network (ANN) approaches. Some of the ANN schemes have evolved from scattered data approximation methods that were developed earlier. We can use any of the suitable methods within our paradigm. A comparison made by Gorinevsky and Connolly (1992) shows that the method described below provides superior approximation accuracy and robustness to the inaccuracy of the data.

We follow McLain (1976), Foley (1986), Franke (1986), Farwig (1987) and Renka (1988) and approximate dependencies  $U(p)$  and  $G(p)$  by fitting a multivariate polynomial to the data.

First, let us consider the approximation problem for a scalar-valued function  $f(p)$ , where  $p \in \mathbb{R}^K$ . Suppose

that we know function values at some given points in the parameter space  $y^{(i)} = f(\mathbf{p}^{(i)})$ ,  $(i = 1, \dots, m)$ . The problem is to find an estimate  $\hat{y}^{(0)}$  for  $y^{(0)} = f(\mathbf{p}^{(0)})$ . We consider the estimate of the form

$$\hat{y}^{(0)} = \sum_{i=1}^m a_i y^{(i)} \quad (17)$$

where the weights  $a_i$  depend only on vectors  $\mathbf{p}^{(0)}$  and  $\mathbf{p}^{(i)}$  ( $i = 1, \dots, m$ ) and do not depend on the function values  $y^{(i)}$ . For a vector or matrix-valued functions  $U(\mathbf{p})$  and  $G(\mathbf{p})$  we can use estimates of the same form (17) for each vector component or matrix entry.

We suppose that the points  $\mathbf{p}^{(i)}$ ,  $(i = 1, \dots, m)$  lie "in the vicinity" of the point  $\mathbf{p}^{(0)}$ . Otherwise we could choose such points from the whole set. Let us formulate the problem as a classical regression problem.

Let us write the Taylor expansion in the vicinity of the point  $\mathbf{p}^{(0)}$ :

$$y^{(i)} = f(\mathbf{p}^{(0)}) + \sum_{q=1}^K \frac{\partial f}{\partial p_q} d_q^{(i)} + \frac{1}{2} \sum_{q,r=1}^K \frac{\partial^2 f}{\partial p_q \partial p_r} d_q^{(i)} d_r^{(i)} + e^{(i)} + \varepsilon^{(i)} \quad (18)$$

where  $d^{(i)} = \mathbf{p}^{(i)} - \mathbf{p}^{(0)}$ ,  $\varepsilon^{(i)}$  is an error of measuring  $y^{(i)}$  and  $e^{(i)}$  is a mismatch of the Taylor expansion.

In fact, we know neither  $f(\mathbf{p}^{(0)})$  nor derivatives of the function  $f$ . We can assume, however, that we know some bounds on their values. To compute an estimate of  $y^{(0)} = f(\mathbf{p}^{(0)})$ , let us consider  $f(\mathbf{p}^{(0)})$ ,  $\partial f / \partial p_q(\mathbf{p}^{(0)})$ ,  $\partial^2 f / (\partial p_q \partial p_r)(\mathbf{p}^{(0)})$ ,  $e^{(i)}$  and  $\varepsilon^{(i)}$  as independent zero-mean random variables. We further assume that we know covariances of these variables that give an idea of their value bounds:

$$\begin{aligned} \langle f^2(\mathbf{p}) \rangle &= \psi^2 \\ \left\langle \left( \frac{\partial f}{\partial p_q} \right)^2 \right\rangle &= \alpha^2 \psi^2 \\ \left\langle \left( \frac{\partial^2 f}{\partial p_q \partial p_r} \right)^2 \right\rangle &= \alpha^4 \psi^2 \\ \langle (e^{(i)})^2 \rangle &= \frac{1}{36} \alpha^6 \psi^2 \|d^{(i)}\|^6 \\ \langle (\varepsilon^{(i)})^2 \rangle &= \chi^2 \end{aligned} \quad (19)$$

( $q, r = 1, \dots, K$ )

where  $\langle \cdot \rangle$  denotes mathematical expectation. Parameter  $\alpha$  has the meaning of "wavelength" of the function  $f(\mathbf{p})$ . This is, for a variation  $\Delta \mathbf{p}$  of the parameter vector (9) so that  $\|\Delta \mathbf{p}\| = \alpha$ , the function value significantly changes. Parameter  $\alpha$  could be assigned a value by considering a physical meaning of the task parameters. Expression for  $\langle (e^{(i)})^2 \rangle$  in (19) follows from the estimate of the Taylor expansion residual in (18) that have the form

$$e^{(i)} = \frac{1}{6} \sum_{q,r,s=1}^K \frac{\partial^3 f}{\partial p_q \partial p_r \partial p_s} (\mathbf{p}^{(0)}) d_q^{(i)} d_r^{(i)} d_s^{(i)} \quad (20)$$

where  $\mathbf{p}^{(0)}$  is a point in the vicinity of  $\mathbf{p}^{(i)}$  and  $\mathbf{p}^{(0)}$ .

Let us write (18) in the vector form as

$$y^{(i)} = \mathbf{F}^T D^{(i)} + e^{(i)} + \varepsilon^{(i)} \quad (21)$$

$$D^{(i)} = \text{col}(1, d_1^{(i)}, \dots, d_q^{(i)}, \dots, d_r^{(i)} d_s^{(i)}, \dots),$$

$$\mathbf{F} = \text{col} \left( f(\mathbf{p}^{(0)}), \frac{\partial f}{\partial p_1}(\mathbf{p}^{(0)}), \dots, \frac{\partial f}{\partial p_q}(\mathbf{p}^{(0)}), \dots, \frac{\partial^2 f}{\partial p_r \partial p_s}(\mathbf{p}^{(0)}), \dots \right) \quad (22)$$

By introducing a matrix  $D = [D^{(1)}, \dots, D^{(m)}]$  and vectors  $\mathbf{Y} = \text{col}(y^{(1)}, \dots, y^{(m)})$  and  $\boldsymbol{\varepsilon} = \text{col}(e^{(1)} + \varepsilon^{(1)}, \dots, e^{(m)} + \varepsilon^{(m)})$  we can represent our regression problem in the form

$$\mathbf{Y}^T = \mathbf{F}^T D + \boldsymbol{\varepsilon}^T \quad (23)$$

where the matrix  $D$  and vector  $\mathbf{Y}$  are known and  $\boldsymbol{\varepsilon}$  and  $\mathbf{F}$  are unknown zero-mean vectors with known covariances.

We search for the least covariance estimate of the form

$$\hat{y}^{(0)} = \sum_{i=1}^m a_i y^{(i)} = \mathbf{Y}^T \mathbf{a}, \quad \mathbf{a} = \text{col}(\{a_i\}_{i=1}^m)$$

Since  $y^{(0)} = f(\mathbf{p}^{(0)}) = \mathbf{F}^T \mathbf{l}$ , where  $\mathbf{l} = \text{col}(1, 0, \dots, 0)$ , we may write the estimate in the form

$$\hat{y}^{(0)} = \mathbf{Y}^T \mathbf{a} = \mathbf{F}^T \mathbf{l} + \eta \quad (24)$$

where  $\eta$  is an estimation error.

Solving (23) and (24) for the vector  $\mathbf{a}$  that provides least-covariance zero-mean  $\eta$  results in

$$\mathbf{a} = (\Lambda + D^T \Psi D)^{-1} D^T \Psi \mathbf{l}, \quad (25)$$

$$\Lambda = \langle \boldsymbol{\varepsilon} \boldsymbol{\varepsilon}^T \rangle, \quad \Psi = \langle \mathbf{F} \mathbf{F}^T \rangle$$

where  $\Lambda$  and  $\Psi$  are diagonal matrices with entries defined by (22) and (19).

A similar-looking solution to a polynomial fitting problem was proposed by Atkeson (1991). However, Atkeson (1991) obtained an expression of the form (25) as a regularized solution to the ill-posed polynomial fitting problem. Our solution is based on the stochastic model (19) of the mapping to be approximated. Therefore the regularization parameters have clear physical meaning. This helps in choosing them.

Here we describe a method for fitting a quadratic Taylor expansion to the data. However, the method could be easily generalized for any order of the expansion.

The result of (25) depends on some parameters of (19) that describe function  $f(\mathbf{p})$ , i.e. the "wave length"  $\alpha$  and the relative inaccuracy of the available data  $\chi/\psi$ . Regression model (19) gives a very general and rough description of the mapping to be approximated. As we have little a priori information about the mapping, it is quite reasonable to assume the same stochastic model (19) for each vector component or matrix of the entry vector and a matrix-valued function  $U(\mathbf{p})$  and  $G(\mathbf{p})$ . In this case we can compute the weight vector  $\mathbf{a}$  (25) only once for all components and the considered approximation method has complexity that is only *linear* in the

dimension of the approximated function. With the parameter vector  $\mathbf{p}$  composed of the joint angles at the beginning and the end of the motion, it is quite natural to choose  $\alpha = 1$  radian in (19). We have assumed that the relative inaccuracy of the data is  $\chi/\psi = 10^{-2}$ .

The described approximation method is local in the sense that it supposes points  $\mathbf{p}^{(i)}$  to lie close to  $\mathbf{p}^{(0)}$ . In the learning process described in Sect. 6, the points where the function value is known are scattered over the whole parameter vector domain and we use a fixed number of the closest points for the approximation. The built approximation does not smoothly depend on the parameter vector  $\mathbf{p}^{(0)}$ , since the set of closest points could change as  $\mathbf{p}^{(0)}$  shifts. However, high precision is more important to us than the smoothness of approximation.

## 5 Biomechanical model of the human arm

This section describes a biomechanical model for planar arm motion that we used to simulate the work of the algorithm in Sects. 2-4. Much of the model and the data coincide with those used by van Sondern and Denier van der Gon (1990) but our model differs in how the control is organized. Below is a brief model formulation that we present for completeness' sake. It can be omitted on the first reading.

### 5.1 Muscle

As stated in Sect. 2, we consider the total of four fictitious equivalent muscles making two antagonist pairs, each powering a single joint. The four muscles have the same parameters and properties.

We assume that the force  $P$  exerted by a muscle is independent of the muscle length and depends on the muscle shortening velocity  $v$  as

$$P = \begin{cases} F_i(b - a/F_0)/(v + b), & v \geq 0, \\ F_i(x - kv)/(x - v), & v < 0, \end{cases}$$

$$x = (k - 1)b/(1 + a/F_0) \quad (26)$$

For muscle shortening ( $v > 0$ ), (26) gives the Hill equation modified to include isometric force  $F_i$  for the current levels of the muscle activation. We use the same parameters of the Hill equation as van Sondern and Denier van der Gon (1990):  $a/F_0 = 0.25$  and  $b = 0.2$  m/s. However unlike van Sondern and Denier van der Gon, who assumed force during muscle lengthening to be the same as isometric, (26) assumes that for lengthening ( $v < 0$ ) the force-velocity curve is also a hyperbolic that approaches the value of  $kF_i$  for fast lengthening and smoothly matches the Hill curve as  $v$  tends to zero. In simulations we set  $k = 1.1$ . This part of the force-velocity dependence is needed to get smooth right-hand sides of the governing differential equations of the system and ensure that the input/output mapping (7) is smooth. The Hill curve modification of the form (26) is used by many authors, though real muscle behavior at lengthening is much more complicated (see, e.g. Morgan 1990 for discussion).

We consider the isometric force  $F_i$  to be obtained from muscle activation signal  $\Phi$  after the first-order low-pass filtering with time constant of 40 ms. This gives us a typical form of the muscle force twitch.

### 5.2 Joint control

**5.2.1 Kinematics.** Following van Sondern and Denier van der Gon (1990) we assume kinematics of the muscle attachment to be the same for the shoulder and elbow joints. Let  $\phi$  be the angle between two adjoined links;  $\phi = \phi^{(1)}$  for the shoulder joint or  $\phi = \phi^{(2)}$  for the elbow. The flexor is attached to the links at the distances  $l$  and  $h_f$  from the joint center and pulls along the line passing via the attachment points. Once  $\phi$ ,  $l$  and  $h_f$  are known, we can compute the mechanical advantage  $d_f$  of the flexor by simple trigonometry. We consider  $d_f$  to be not less than 1 cm. The mechanical advantage  $d_e$  of the extensor was taken to be constant. The kinematical data used in the simulation are the same for the two joints:  $l = 32$  cm,  $d_e = 2$  cm and  $h_f = 5$  cm.

The torque generated in the joint by the muscle pair can be computed as  $T = P_f d_f - P_e d_e$ , where the flexor and extensor forces  $P_f$  and  $P_e$  are computed as described in Sect. 5.1. One can find the velocities  $v_f$  and  $v_e$  of the flexor and extensor shortening from the joint angular velocity  $\dot{\phi}$  as  $v_f = \dot{\phi} d_f$  and  $v_e = -\dot{\phi} d_e$ .

**5.2.2 Control of antagonist muscle pair.** Following Feldman (1979) we consider the muscles powering a joint to be controlled as a whole and an input signal to be distributed between flexor and extensor muscles. We assume that a single input signal  $Q$  defines the flexor and extensor activation levels  $\Phi_f$  and  $\Phi_e$  (before the low-pass filtering) as

$$\Phi_f = \Psi_\Delta(Q/d_f), \quad \Phi_e = -\Psi_\Delta(-Q/d_e), \quad (27)$$

where  $\Psi_\Delta(x)$  is a smooth function such that  $\Psi_\Delta(x) = 0$  for  $x \rightarrow -\infty$ ,  $\Psi_\Delta(x) = x$  for  $x \rightarrow \infty$  and  $\Psi_\Delta(x) - \Psi_\Delta(-x) = x$ . We assume  $\Psi_\Delta$  to be a piece-wise quadratic function of the form

$$\Psi_\Delta(x) = \begin{cases} x, & \text{for } x > 4\Delta \\ (x + 4\Delta)^2/(16\Delta), & \text{for } |x| < 4\Delta \\ 0, & \text{for } x \leq -4\Delta \end{cases} \quad (28)$$

Expressions (27) and (28) mean that for large negative activation, only the extensor is activated, and for large positive activation, only the flexor; in the middle, smooth transition of activation levels takes place. In stationary state the joint torque generated by the two muscles according to (27) is just  $Q$ . For zero input  $Q$ , the antagonist muscles exert opposite sign torques of the magnitude  $\Delta$ . Thus,  $\Delta$  could be considered as a muscle tonus parameter. In the terminology of Feldman (1979),  $\Delta$  is a coactivation command. If computed by a positional reflex feedback,  $Q$  corresponds to a reciprocity command.

Presence of the muscle tonus increases dissipation in the system due to negative slope of the Hill curve. This stabilizes the system and prevents oscillatory behavior that otherwise occurs in simulation due to the reflex feedback delay.

**5.2.3 Joint torque control.** We consider the joint input torque  $Q$  to consist of a delayed PD positional (reflex) feedback and a feedforward term

$$Q(t) = k[\phi_e - \phi(t - \tau)] - b\dot{\phi}(t - \tau) + u(t) \quad (29)$$

where  $\tau$  is the stretch-reflex feedback delay,  $\phi$  is the measured joint angle,  $\phi_e$  is the desired joint angle at the end of the motion,  $k$  and  $b$  are reflex feedback gains and  $u$  is a feedforward input computed as described in Sects. 2-4. The input  $Q$  is further distributed between the antagonist muscles according to (27).

Van Sondern and Denier van der Gon (1990) considered (velocity) feedback without a delay because otherwise they have had problems with oscillations appearing in the simulated motion. Our model takes into account coactivation of the antagonist pair (see Sect. 5.2.2), so the system has sufficient dissipation and remains stable despite the delay.

Using the model of van Sondern and Denier van der Gon (1990) we supposed the feedback stiffness to be zero during the first 100 ms of the motion, grow linearly to 10 Nm/rad during the next 100 ms and then remain constant until the motion ends.

We used the relation between positional and velocity feedback and reflex delay obtained for human elbow joint movements by Zahalak and Pramod (1985) though we did not count the delayed acceleration feedback term that was included in their model. So, we supposed that in (29)  $b = k \cdot 0.12$  s and  $\tau = 27$  ms always holds.

As to the value  $\Delta$  of the coactivation command, we supposed that it is 2.5 Nm during the first 100 ms of motion, then linearly grows to 10 Nm during the next 100 ms and remains at 10 Nm thereafter. The growth of the coactivation was instrumental in keeping the system stable as the feedback gains grow, and correlates with a model of Feldman (1979).

### 5.3 Arm dynamics

When considering the dynamics of the planar arm motion we neglect the motion of the muscles with respect to the arm links, and model the arm as two rigid bodies connected by two cylindrical joints. By introducing an arm configuration vector  $q = \text{col}(\phi^{(1)}, \phi^{(2)})$  one can write an equation of the arm dynamics in the form (see Craig 1986)

$$M(q)\ddot{q} + C(q, \dot{q}) = T \quad (30)$$

where  $M(q)$  is an inertia matrix, vector  $C(q, \dot{q})$  gives Coriolis and centrifugal forces, and  $T$  is a vector of joint torques generated by the muscles. The equations of motion are given in more detail, together with the parameters we used, in van Sondern and Denier van der Gon (1990).

The complete model of the arm motion used in the simulation includes equations of motion (30), where torques are computed as described in Sects. 5.1 and 5.2. As stated in Sect. 5.2, four muscle forces depend on the states of the four respective low-pass filters. The delayed feedback law (29) means that we have also to

keep the history of the joint angles in the computer memory during the simulation. The overall simulated system is rather complex and nonlinear.

## 6 Simulation results

### 6.1 Learning a single motion

In the simulation we used the algorithms of Sects. 2-4 to control the system described in Sect. 5. We considered the feedforward control interval (1) to be  $T = 0.3$  s and the observation interval (5) to be  $T_f = 0.6$  s. Figure 4 demonstrates a typical form of the piece-wise constant feedforward input (2). The feedforward value changes at time points 0.00, 0.06, 0.12, 0.18, 0.24, 0.27 and 0.30 s and is described by a 12-dimensional vector  $U$  (4). We choose times of control switching without any relation to physiological data, just so that they allow reasonably good control of the simulated arm motions. Two last pulses have shorter duration and are needed to better dampen the overshoot and vibrations due to delays in muscle activation and in the reflex feedback loop.

To monitor how precisely the arm comes to the final position, we used the position measurements sampled between  $t = T = 0.3$  s and  $t = T_f = 0.6$  s with interval  $\tau = 0.03$  s. They give us a measurement vector  $Y$  (7) of dimension 20.

Figure 1 shows the end-point trajectory for the learned motion from  $20^\circ$  to  $60^\circ$  in the shoulder joint and from  $60^\circ$  to  $120^\circ$  in the elbow joint. The overshoot is negligible and the motion smooth. Simulation shows that the learned feedforward reduces error  $\|Y\|$  more than 20-fold. Figure 2 presents time histories of joint angular velocities for the same motion which look similar to those observed experimentally by other authors.

Figure 3 displays the learned feedforward input and the activation pattern of the muscles (combined feedback and feedforward) for the motion. The activation signals could be considered as an analogue of muscle activation level monitored via electromyography in a respective physiological experiment. In our simulation,

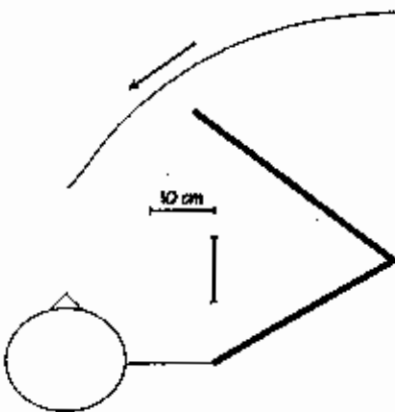


Fig. 1. Example of an end-point trajectory for learned motion

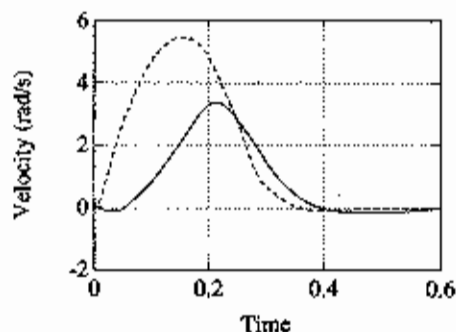


Fig. 2. Joint angular velocity history in radians per second. *Solid line*, shoulder angle; *dashed line*, elbow angle

coactivation of the antagonist muscles is kept constant after the end of the motion. This can partly explain the difference between the presented plots and the experimentally recorded EMG appearance. In human arm motion the coactivation diminishes at the end of the motion (Adamovitch et al. 1990).

### 6.2 Filling the databases

We simulated the process of learning control for all motions in a certain domain. The simulation proceeded as follows. First, the motion parameter vector  $\mathbf{p}$  (9) defining the motion trajectory was generated as a random vector with components in the domain  $29^\circ \leq \phi_b^{(1)}$ ,  $\phi_e^{(1)} \leq 142^\circ$ ;  $14^\circ \leq \phi_b^{(2)}$ ,  $\phi_e^{(2)} \leq 158^\circ$ .

Next, the data for the closest values of  $\mathbf{p}$  (9) were extracted from the databases that contain the already learned control vector  $\mathbf{U}$  and the sensitivity matrix  $G$ .

The data were used in an approximation procedure of Sect. 5 to find estimates for  $\mathbf{U}$  and  $G$ . These estimates were used and further improved in an iterative learning procedure described in Sect. 3. The learning is stopped when the error  $\|\mathbf{Y}\|$  (6) of coming to the final position is small enough to give a mean hand deviation of less than 2 cm. If more than one learning iteration is done, the new learned control is added to the database, and if more than four, the improved estimate for the sensitivity matrix  $G$  is also stored. Then the process repeats for a new random point  $\mathbf{p}$  in the task parameter space.

The learning process is illustrated in Fig. 4, where the error  $\|\mathbf{Y}\|$  is shown vs a number of the randomly generated task parameter vector (9). A tolerable error  $\|\mathbf{Y}\|$  is shown by a dashed line. The error diminishes as more data are stored in the databases. In fact, after some 250 generated points the error is mostly within the prescribed bounds, and points with greater error become increasingly rare.

For many generated points, the approximated control gives sufficiently accurate motion, and no repetition of the motion is done at all. Figure 5 shows the percentage of the points where learning was needed in each set of 100 consecutive random points. One can see that the percentage diminishes fast.

At the end of the process shown in Figs. 4 and 5, the learned control vectors  $\mathbf{U}$  were stored at 150 points and the sensitivity matrices  $G$  at 39 points, resulting in 51 kbytes of storage memory. Figure 6 represents the motions, for which vectors  $\mathbf{U}$  are stored, by lines connecting initial and final arm tip position.

Let us assume that for control of a three-dimensional point-to-point motion the density of the stored data in the parameter space should be the same as in

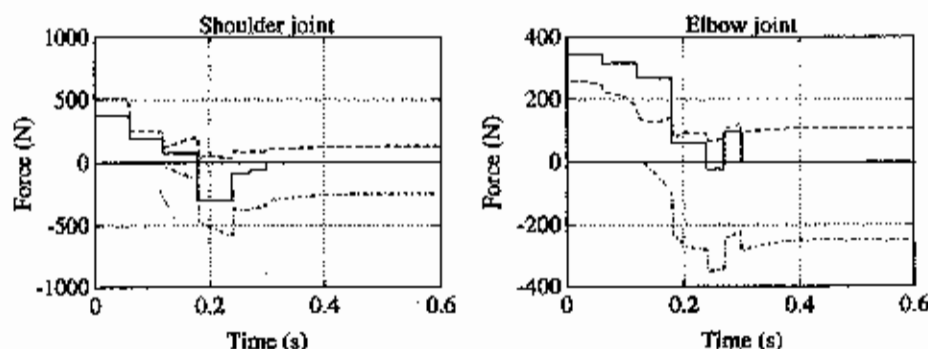


Fig. 3. Pattern of the muscle activation. *Left plot*, shoulder joint; *right plot*, elbow. *Solid line* presents feedforward joint input torque  $u$  (29) scaled as  $u/3$  (cm). *Dashed line*, flexor force, *dash-dotted line*, extensor force with reversed sign

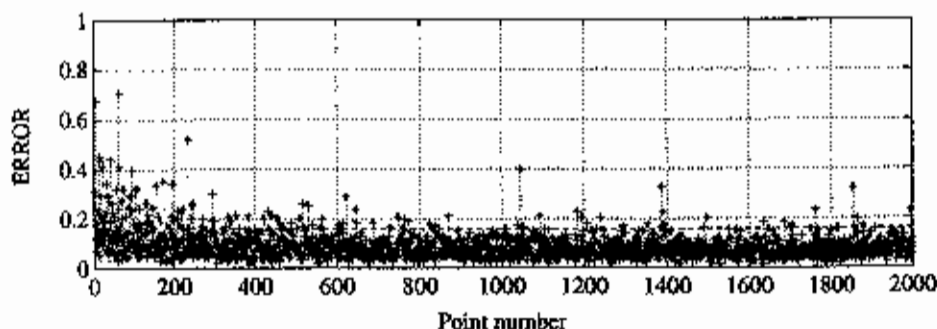


Fig. 4. Dependence of the motion error for the approximated control on the number of the generated set of the motion parameters. *Vertical dotted lines* show error diminishing in learning. *Horizontal dashed line* gives an acceptable error

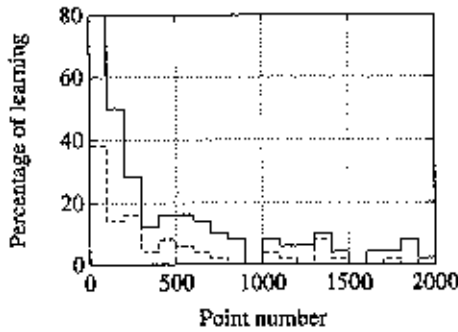


Fig. 5. Percentage of points where learning is needed among the generated sets of the motion parameters. *Solid line*, learning of  $U$ ; *dashed line*, learning of  $G$

the presented case. In that case we obtain that vector  $U$  should be stored for about  $150^{3/2} \approx 1800$  trajectories. This is obviously a reasonable amount. In fact, even fewer movements should be learned since the accuracy achieved in our simulation is significantly higher than that of fast human movements.

Of course, fast human motions involve coordinated work of many more muscle groups and joints than we are considering. However, we believe that our paradigm is backed by the synergy concept of Bernstein (1947). According to this concept, the variability of fast coordinated human motions is due to variations of a few parameters, the number of which is much less than the number of body degrees of freedom involved in the motion. And for a moderate number of the task parameters our paradigm provides a possible means of motor control organization.

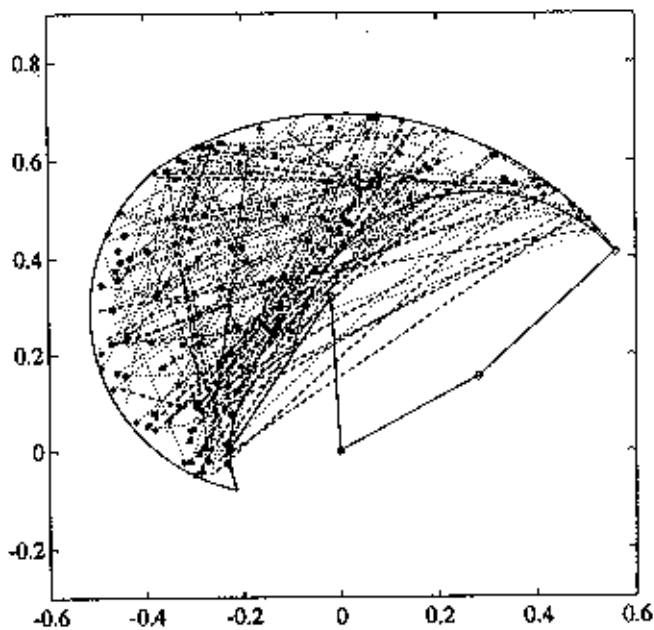


Fig. 6. Percentage of points where learning is needed among the generated sets of the motion parameters. *Solid line*, learning of  $U$ ; *dashed line*, learning of  $G$

### 6.3 Discussion of the results

**6.3.1 General.** The results of this paper prove that the proposed paradigm of direct motor program learning can provide control of fast human arm motions. The model used for arm motion incorporates elements resembling the nonlinearities, delays and time dependencies that are typical for biological systems and that make them very difficult to control with methods conventionally used for control of technical systems.

The present results could help us to understand the basic principles of motor control of fast human motions. The paradigm could be also of practical use for robotics and control of other complex technical systems. Some robotic applications of the concept have been considered in Gorinevsky (1991, 1992a), and other will be discussed in future publications.

Though our aim was to avoid using a model for the system dynamics, we should recognize that, in fact, the databases for  $U$  and  $G$  that are used in the learning process do constitute such a model. This model is specific for a control task. The advantage of having such a task-dependent model instead of learning a model for system dynamics is that, in our paradigm, the amount of data stored depends only on the number of the task parameters and not on the complexity of the dynamics.

Unlike our paradigm, a dynamical model could be used for control of motions other than learned. This, however, requires extensive computations during or before the motion. Our paradigm relies on the memory instead, which we believe is more typical for biological systems. One can also speculate that humans who are able to perfectly control motions in some tasks require additional training and experience to master a new task. This is in line with our paradigm.

This work has left many questions open for further research. One of them is how to organize experience accumulated in the databases so that it could be used in several different tasks. A possible solution is to divide tasks into some elementary or primitive motions and learn control of these motions with our paradigm. Whatever the advantages and properties of the proposed paradigm are, we do not consider it an ultimate explanation or solution. The goal of this work is rather to add a new dimension to the understanding of possible ways of human motor control organization.

**6.3.2 Algorithms.** We implement our concept of direct motor program control in the algorithms of Sects. 3 and 4 with some degree of looseness. There are several parts and parameters of the algorithms that could be changed, replaced or elaborated.

First, we do not discuss in Sections. 2 and 5 how to choose the parameterization of control or, at least, timing of the control switching. Next, it is clear that the presented algorithms for the control learning and input/output sensitivity estimation are not the only ones applicable or similar to those used in nature. The approximation algorithm of Sect. 4 is also not the only one that could be used. However we consider it important that all algorithms do successfully work together and solve an otherwise very complicated control task.

*Acknowledgements.* Part of this work was completed in the Institute for Problems of Information Transmission, USSR National Academy of Sciences, Moscow. Another part was supported by an Alexander von Humboldt Research Fellowship held by the author at the Institute B for Mechanics, Technical University in Munich, Germany. The author is grateful to Professors M. Berkenblit, M. Shik, F. Pfeiffer, and S. Edelman for helpful discussions and to an anonymous referee for many useful suggestions.

## References

- Adamovitch SB, Berkenblit MB, Feldman AG (1990) Principles of human motor control, part I (in Russian). *Itogi Nauki i Tekhniki, Ser Fiziologiya Cheloveka i Zhivotnykh*, Tom 43. VINITI, Moscow
- Arimoto S (1990) Learning control theory for robotic motion. *Int J Adaptive Control Signal Processing* 4:543-564
- Arimoto S, Kawamura S, Miyazaki F (1984) Bettering operation of robots by learning. *J Robotic Syst* 1:123-140
- Atkeson CG (1991) Using locally weighted regression for robot learning. In: *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, Calif, p 963
- Bernstein NA (1947) On composition of motions (in Russian). *Medgiz*, Moscow
- Craig JJ (1986) *Introduction to robotics*. Addison-Wesley, Reading, Mass
- Dornay M et al. (1991) Simulation of optimal movements using the minimum-muscle/tension-change model. In: Lipmann RP, Moody JE and Touretzky DS (eds) *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann, San Mateo, CA, vol 3, pp 627-634
- Farwig R (1987) Multivariable interpolation of arbitrary spaced data by moving least squares methods. *J Comp Applied Math* 16:79-93
- Feldman A (1979) Central and reflex mechanisms of motor control (in Russian). Nauka, Moscow
- Flash T, Hogan N (1985) The coordination of arm movements: an experimentally confirmed mathematical model. *J Neurosci* 5:1688-1703
- Foley TA (1986) Scattered data interpolation and approximation with error bounds. *Comp Aided Geom Design* 3:163-177
- Franke R (1986) Recent advances in the approximation of surfaces from scattered data. In: Chui CK et al. (ed) *Topics in multivariable approximation*. Academic Press, Boston, pp 79-98
- Goodwin GC, Sin KS (1984) *Adaptive filtering, prediction and control*. Prentice-Hall, Englewood Cliffs, NJ
- Gorinevsky DM (1991) Learning and approximation in database for feed-forward control of flexible-joint manipulator. In: *Proc '91 ICAR: 5th International Conf on Advanced Robotics*, Pisa, pp 688-692
- Gorinevsky DM (1992a) Experiments in direct learning of feedforward control for manipulator trajectory tracking. *Robotersysteme* 8:139-147
- Gorinevsky DM (1992b) On the approximate inversion of linear system and quadratic-optimal control. *Sov J Comp Syst Sci* (in press)
- Gorinevsky DM, Connolly TH (1992) Comparison of artificial neural network and scattered data approximations: the inverse manipulator kinematics example. *Technical Report. Lehrstuhl B für Mechanik, TU-München*, August 1992 (also submitted to *Neural Computation*)
- Guez A, Selinsky J (1988) A trainable neuromorphic controller. *J Robotic Syst* 5:363-388
- Kano H, Takayama K (1990) Learning control of robotic manipulator based on neurological model CMAC. In: *11th IFAC Congress*, Tallinn, pp 268-273
- Katayama M, Kawato M (1991) Learning trajectory and force control of an artificial muscle arm by parallel-hierarchical neural network model. *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann, San Mateo, CA, vol 3, pp 436-442
- Kawato M (1989) Adaptation and learning in control of voluntary movement by the central nervous system (tutorial). *Adv Robotics* 3:229-249
- McLain DH (1976) Two dimensional interpolation from random data. *Comput J* 19:178-181
- Messner W, Horowitz R, Kao WW, Boals M (1991) A new adaptive learning rule. *IEEE Trans Automat Contr* 36:188-197
- Morgan DL (1990) New insights into the behavior of muscle during active lengthening. *Biophys J* 57:209-221
- Oh SR, Bien Z, Suh IH (1988) An iterative learning control method with application for the robot manipulator. *IEEE J Robotics Automation* 4:508-514
- Renka RJ (1988) Multivariable interpolation of large sets of scattered data. *ACM Trans Math Software* 14:139-148
- Sanchez VDA, Hirzinger G (1991) State-of-the-art robotic learning control based on artificial neural networks. An overview. In: Khatib O et al. (ed) *The robotic review 2*. MIT Press, Cambridge, Mass
- Sonderm JF van, Denier van der Gon JJ (1990) A simulation study of a programme generator for centrally programmed fast two-joint arm movement: responses to single- and double-step target displacements. *Biol Cybern* 63:35-44
- Togai M, Yamano O (1986) Learning control and its optimality. In: *Proceedings of the 1986 IEEE Conference on Robotics and Automation*, San Francisco, Calif, pp 248-253
- Tikhonov AN, Arsenin VYa (1979) *Methods for solution of ill-posed problems* (in Russian). Nauka, Moscow
- Uno Y, Suzuki R, Kawato M (1989) Formation and control of optimal trajectory in human multijoint arm movement - minimum torque change model. *Biol Cybern* 61:89-101
- Wu CH, Young KY, Hwang KS, Leman S (1992) Voluntary movements for robotic control. *IEEE Contr Syst Magazine* 12:8-14
- Yabuta T, Yamada T (1990) Possibility of neural network controller for robot manipulators. *Proceeding of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp 1686-1691
- Zahalak GI, Pramod R (1985) Myoelectric response of the human triceps brachii to displacement-controlled oscillations of the forearm. *Exp Brain Res* 58:305-317