Modelling of direct motor program learning in fast human arm motions

Didier M. Gorienski
Kiehn (1992) (Received 21 March 1991; Revised 29 March 1991)

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 multicollinearity 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, arm dependence of the feedback gains and cocontraction 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 speed-dependent controlled: see, for example, the survey by Adamovich et al. (1990). The most general evidence could be derived from the fact that a reflex postural feedback has a significant delay and a large feedback gain cannot be achieved. Thus, while being important for disturbance compensation, the reflex feedback alone could hardly provide the required precision of fast motions. A feedforward control (motor commands) should be added to the feedback. The relation of the activation pattern of muscles to the resulting position 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 in the arm. Control of robot manipulators, e.g. see Kawato (1989), Guiraud and Sejnowski (1988), Kano and Takayama (1990), Yabuta and Yamada (1990), Doyon et al. (1991), Kawato and Kawato (1991) and the survey of Sandblom and Hirose (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 used oversimplified models for fast human arm motions. Delayed reflex feedback with time-variant gains and nonlinear dynamics of muscle activation 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, e.g. Adamovich 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 commands, programs for some values of the parameter...
tor 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 Gottlieb (1997) to the point-to-point control of a simulated two-link manipulator with joint elasticity. Gottlieb (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 this 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 "hard" 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, descriptions of which comprise 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 preselected 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 programs 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 that no account major features of the arm, an arm dynamics that make it much more complicated system than a manipulator arm. The second objective was to use a standard model formulation precisely 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 Soest and Denier van der Gon (1989) 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 \( \theta_1 \) and \( \theta_2 \) the respective joint angles that we control 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 \( \theta_1^0 \) and \( \theta_2^0 \). At final time \( t = T \) it should be resting in a new position with joint angles \( \theta_1^f \) and \( \theta_2^f \). We assume that to control the torque of joint 1, an angle-level input signal \( u_1(t) \) is used which is distributed between the flexor and extensor muscles of the joint so that both muscles are nonnegatively activated. The signal \( u_1(t) \) is a sum of a positional feedback and a feedforward input (motor command) \( w_1(t) \). For more details of the model see Sect. 5.

If we apply feedforward control to the arm, it will still move to the final position due to the reflex activity. However, this motion will be significantly slower and with an overshoot or undershoot. The control task is to find two feedforward control programs \( w_1(t) \) and \( u_1(t) \) for the shoulders 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 inputs and output variables.

2.2 Parameterization of the input and output

Continuous-time functions \( w_1(t) \) describing the feedforward input are defined by an infinite number of parameters. We search for the solution of the control problem and the certain family of feedforward programs depending on a reasonably small number of parameters.
Let us consider \( N_0 \) sampling instants
\[
0 \leq t \leq T, \quad (j - 1, N_0)
\]
and feedforward control of the form
\[
U(t) = \sum_{i=1}^{N_0} \beta_i U_{i-1}(t), \quad U_0(t) = \beta_0 U_{-1}(t)
\]
where \( U_0 \) are parameters of the control and \( \beta_i \) are the \( i \)-th order feedback functions. For instance, one can choose \( \beta_i \) to be B-spline functions (see Gorenkevych 1992a), but here we suppose that \( \beta_1 \) is an indicator function for the interval \( [t_i, t_{i+1}] \).

According to (1)–(3), feedforward control programs are piece-wise constant functions with values \( U_{n-1} \) and \( U_n \) between sampling instants \( t_{n-1} \) and \( t_n \). The feedforward controls \( \beta_i \) are defined by a \( 2N \)-dimensional vector
\[
\beta = \text{col}(U_0, U_1, \ldots, U_{2N_0})
\]
that 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 Wen et al. (1992), where results of such modeling are fit to experimental data.

We further consider arm motion at the time intervals [0, T], \( j \leq T \). For \( j > T \) we compute feedforward as
\[
U(t) = \beta(t), \quad (j = 1, 2, \ldots)
\]
To discretize the output of the control task, we consider \( N_0 \) equidistant sampling instants \( t_j \) just after the control interval \( [0, T] \).

Let us consider an output vector \( Y \in \mathbb{R}^{2N_0} \)
\[
Y = \begin{bmatrix} \delta Y(t_1 ; \beta_1) \\ \delta Y(t_2 ; \beta_2) \\ \vdots \\ \delta Y(t_{2N_0} ; \beta_{2N_0}) \end{bmatrix}
\]
that contains the sampled joint-angle deviation from their final value. 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 describes deviation of the hip motion from a preplanned path (see Gorenkevych 1992a).

The input vector \( U \) and the output vector \( Y \) are related by a complex non-linear mapping
\[
Y = H(U)
\]
Each point of the mapping could be obtained by applying feedforward control (1)–(6) to an arm motion and observing the output data (5, 6).

3 Performance index

We can now formulate 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 observable, we can conclude that if the output sampling interval \( s \) is sufficiently small, \( |Y| = 0 \) means that the arm comes to a steady state as 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 Gorenkevych (1991, 1992a). Let us introduce a performance index minimization problem that describes the task
\[
J = \frac{1}{2} \| Y(t) \|^2 - \min \quad (8)
\]
where \( \alpha > 0 \) is a scalar and \( \| \cdot \| \) denotes the Euclidean norm of a vector.

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

The quadratic optimality criterion (8) is of course an arbitrary assignment. According to some models (e.g., Flach and Hogan 1985; Uno et al. 1988; Donay 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 \( U^2 \) to \( U^2 + \gamma U^2 \), where \( \gamma \) is a convenient diagonal 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
\[
\beta = \text{col}(\beta_1, \beta_2, \beta_2, \beta_{2N_0})
\]
where \( \beta_1 \) and \( \beta_2 \) are the shoulder and elbow joint angles at the beginning of the motion and \( \beta_2 \) and \( \beta_{2N_0} \) are the desired joint angles at the end of the motion. Of course, the mapping (7) depends on the task parameter vector \( \beta \). We do not write this dependence explicitly, but rather implicitly.

Our goal is to be able to design feedforward (motor command) programs \( U(t) \), \( j = 1, 2, \ldots \), for an arbitrary parameter vector \( \beta \) in a given domain.

4 Learning

4.1 Iterative method

We can minimize the performance index (8) iteratively. Let \( U(t) \) be a guess at the solution to (8). Let an output
vector \( V \) is analytically obtained by applying the control law \( U \) (2)–(16) during the open loop. We can find the next, better approximation \( U_c \) to the solution of (8) with the Newton–Raphson method

\[
U_c = U - \left[ \frac{\partial Y}{\partial U} \right]^{-1} \frac{\partial Y}{\partial U}
\]

where

\[
\frac{\partial Y}{\partial U} = \frac{\partial Y}{\partial G} \frac{\partial G}{\partial U} + \frac{\partial Y}{\partial U}
\]

The matrix \( G = Y(U|U) \) in (11) is the input/output sensitivity matrix of the controlled plant. Column of \( G \) means the variation of the sampled output (7) corresponding to the unit variation of the respective component \( V_0 \) of the input vector (4), i.e., variation of the control value at the interval \( T_0 \). The Hessian matrix in (10) has the form

\[
\frac{\partial^2 Y}{\partial U^2} = \frac{\partial G}{\partial U} \frac{\partial G}{\partial U} + \frac{\partial G}{\partial U} \frac{\partial G}{\partial U}
\]

We neglect the last term in (12) since for \( p < 1 \) the solution to (8) gives \( \frac{\partial Y}{\partial U} \) straight. Furthermore, if we change the Hessian matrix in (10) to another positive-definite matrix, the algorithm does not spoil the convergence but only reduces its rate. So we use a modified Newton–Raphson method of the form

\[
U^{n+1} = U^n - \frac{\partial G}{\partial U} \frac{\partial G}{\partial U} \frac{\partial G}{\partial U}
\]

where the upper index denotes iteration number. Each iteration includes trying the new model with control \( U^{n+1} \) and obtaining output \( Y^{n+1} \), (4).

We can only use this iterative learning method (13) if we know the input/output sensitivity matrix \( G \) of 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 recursive control algorithms of Ansimato et al. (1980), Arimoto (1990), Suen and Yamada (1988), Oh et al. (1988) and Matsui et al. (1995). The robustness of convergence of the proposed iterative method and the influence of the imprecision knowledge of \( G \) on the convergence point have been considered by CuthBERT (1992).

3.2. Adaptation for-line estimates 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 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

\[
\hat{G} = G^{(n)} - \frac{1}{2} \hat{G}^{(n-1)} - \frac{1}{2} \hat{G}^{(n-2)} - \frac{1}{2} \hat{G}^{(n-3)}
\]

variations of the plant input and output, respectively, at learning iteration \( n \). We suppose that

\[
\hat{G} = G^{(n)}
\]

and try to improve an estimate \( \hat{G} \) of the matrix \( G \) from data (4). We modify the estimate so that (12) holds precisely for \( \hat{G} \) and that for any \( \varphi \) orthogonal to \( \hat{G} \) the estimate of \( \varphi \) does not change. We obtain the following kind of iterative approximation procedure:

\[
\hat{G}^{(n+1)} = \hat{G}^{(n)} + \frac{1}{\sigma} \hat{G}^{(n)} \hat{G}^{(n)} - \frac{1}{\sigma} \hat{G}^{(n)} \hat{G}^{(n)}
\]

where \( \sigma \) is a small positive constant.

Local convergence of the adaptive learning algorithm (16) follows from the fact that if the mapping (7) is linear in the vicinity of the minimum, then the cost of the 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 variable projection gradient descent 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 \( G(z) \) (11) for some model, i.e., for some values of the parameter vector \( \varphi \). We can use the learned data to build an approximation for the vector \( U \) and the matrix \( G \) over the domain of \( \varphi \), i.e., the constructed approximation for \( U \) does not provide a sufficiently small mean square 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 multivariable, the typical dimension of the parameter vector \( \varphi \) could be between 2 and 10. In the control task that we are considering, \( p - 4 \). The dimension of the vector \( U \) could be between 10 and 100 and the dimension of matrix \( G \) could be up to \( 100 \times 200 \approx 2 \cdot 10^9 \). A number of papers deal with standard 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 networks (ANN) approach.

Some of the ANN advances 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 CuthBERT and Connolly (1992) shows that the methods described below provide superior approximation accuracy and robustness to the inaccuracy of the data.

We follow McLain (1976), Foley (1986), Franke (1989), Fannes (1987) and Rama (1988) and approximate functions \( U(\varphi) \) and \( G(\varphi) \) by lifting a multivariate polynomials data points.

First, let us consider the approximation problem for a scalar-valued function \( f(\varphi) \), where \( \varphi \in \mathbb{R}^p \). Suppose
that we know function values at some given points in the parameter space \( y^0 = f(p^0) \), \( i = 1, \ldots, n \). The problem is to find an estimate \( \hat{y}(p) \) for \( y(p) = f(p) \). We consider the estimate of the form

\[
y(p) = \sum_{i=1}^{n} a_i y^0_i \tag{17}
\]

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

We suppose that the points \( p^0_i \) \( i = 1, \ldots, n \) lie "in the vicinity" of the point \( p^0 \). 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 \( p^0 \):

\[
y(p) = f(p^0) + \sum_{i=1}^{n} \frac{\partial f}{\partial p_i} (p^0 - p^0_i) + \epsilon \tag{18}
\]

where \( \epsilon = y(p) - f(p^0) \) and \( \epsilon \) is an error of measuring \( y(p) \) and \( \epsilon \) is a residue of the Taylor expansion.

In fact, we know neither \( f(p^0) \) nor derivatives of the function \( f \). We can assume, however, that we know some bounds on their value. To compute an estimate of \( y(p) \) \( f(p^0) \), we can use another \( \hat{f}(p^0) \), \( \hat{\partial f}/\partial p \), \( \hat{\partial^2 f}/\partial p \), \( \hat{\partial^3 f}/\partial p \), \( \hat{\partial^4 f}/\partial p \), and \( \hat{\partial^5 f}/\partial p \) as independent random variables. We further assume that we know bounds of these variables that give an idea of their value bounds:

\[
\langle \hat{f}(p) \rangle = \hat{f} \tag{19}
\]

\[
\langle \hat{\partial f}/\partial p \rangle = \hat{\partial f}/\partial p \tag{20}
\]

\[
\langle \hat{\partial^2 f}/\partial p \rangle = \hat{\partial^2 f}/\partial p \tag{21}
\]

\[
\langle \hat{\partial^3 f}/\partial p \rangle = \hat{\partial^3 f}/\partial p \tag{22}
\]

\[
\langle \hat{\partial^4 f}/\partial p \rangle = \hat{\partial^4 f}/\partial p \tag{23}
\]

\[
\langle \hat{\partial^5 f}/\partial p \rangle = \hat{\partial^5 f}/\partial p \tag{24}
\]

where \( \hat{f} \) and \( \hat{\partial f}/\partial p \) are diagonal matrices with entries defined by (19) and (19).

A similar-looking solution to a polynomial fitting problem was proposed by Aronsen (1991). However, Aronsen (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 \( f(p) \), i.e., the "wave lengths" \( \alpha \) and the relative inaccuracy of the available data \( \epsilon \). Regression model (19) gives a very general and flexible description of the mapping to be approximated. As we have little a priori information about the mapping, it is quite reasonable to assume some stochastic model (19) for each vector component or matrix of the entry vector and a matrix-valued function \( U(p) \) and \( G(p) \). In the case we can consider the weight vector \( \epsilon \) (25) only once for all components and the considered approximation method has simplicity 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 $\theta = 1$ radian in (19). We have assumed that the relative inaccuracy of the data is $\bar{\epsilon} = 10^{-4}$.

The described approximation method is local in the same that it supposes points $p_0^i$ to be close to $p_0^j$. 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 $\mathbf{p}$, since the set of closest points could change as $\mathbf{p}$ 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 algorithms in Sects. 2-4. Much of the model and the data coincide with those used by van Someren and Denier van der Gon (1998) 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 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 point. 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

$$
P = \begin{cases}
P_0 (v) & |v| > b, \\
P_0 (b) - b \dot{v} & |v| < b,
\end{cases}
$$

where $P_0 (v)$ is a smooth function such that $P_0 (v)$ is $\frac{dP_0}{dv} = 0$ for $v = -b$, $P_0 (x) = 0$ for $x = -b$, and $P_0 (x) = \frac{dP_0}{dx} = 0$ for $x = 0$. We assume $P_0$ to be a piece-wise quadratic function of the form

$$
P_0 (x) = (x^2 + 2ax + a^2)(16d^2), \quad \text{for } |x| < 4a
$$

Expression (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 antagonistic muscles exert opposite signs torques of the magnitude $\Delta$. Then, $\Delta$ could be considered as a muscle torque parameter, in the terminology of Pineda (1979). It is a conduction command. If computed by a positional reflex feedback, $Q$ corresponds to a reciprocity command.

Presence of the muscle torques 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.

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 $\theta$. This gives us a typical form of the muscle force network.

5.2 Joint control

5.2.1Kinematics. Following van Someren and Denier van der Gon (1998) we assume kinematics of the musculoskeletal attachment to be the same for the shoulder and elbow joints. Let $\phi$ be the angle between two adjacent links: $\phi = \phi^i - \phi^j$ for the shoulder joint or $\phi = \phi^j - \phi_0$ for the elbow. The shoulder is attached to the links at the distances $l_1$ and $l_j$ from the joint centers and pulls along the line passing via the attachment points. Once $\phi$, $\phi^j$, and $\phi_0$ are known, we can compute the mechanical advantage $\alpha_j$ of the flexor by simple trigonometry. We consider $\alpha_j$ to be not less than 1 cm. The mechanical advantage $\alpha_j$ of the extensor was taken to be constant. The kinematical data used in the simulation are the same for the two joints: $l_1 = 32$ cm, $l_2 = 24$ cm and $l_3 = 5$ cm.

The torque generated in the joint by the muscle pair can be computed as $T = P \alpha_0 - P \alpha_j$, where the flexor and extensor forces $P_0$ and $P_j$ are computed as described in Sect. 5.1. One can find the velocities $\alpha_0$ and $\alpha_j$ of the flexor and extensor shortening from the joint singular velocity $\delta$ as $\alpha_0 = \phi_0 / \delta$ and $\alpha_j = -\phi / \delta$.
5.3 Joint torque control. We consider the joint input torque \( T \) to consist of a delayed PD positional (reflex) feedback and a feedforward term

\[
Q(t) = \dot{\phi}_r - \dot{\phi}_r(t - \tau) + \ddot{\phi}_r(t - \tau) + \dddot{\phi}_r(t - \tau)
\]

where \( \dot{\phi}_r \) is the muscle-reflex feedback delay, \( \dot{\phi}_r(t - \tau) \) is the desired joint angle at the end of the motor, \( \dot{\phi}_r \) and \( \ddot{\phi}_r \) are reflex feedback gains and \( \dddot{\phi}_r(t - \tau) \) is a feedforward input computed as described in Sect. 2.5. The input \( Q(t) \) is further distributed among the antagonistic muscles according to (27).

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

Using the model of van Strien 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/1rad during the next 100 ms and then remain constant until the motion ends.

We used the relative between positional and velocity feedback and reflex delay obtained for human elbow joint movements by Zehabek and Pinedo (1985) though we did not use the delayed acceleration feedback term that was included in their model. So, we supposed that in (29) \( \dot{\phi}_r = 0.12 \dot{\phi}_r \) and \( \ddot{\phi}_r = 27 \dot{\phi}_r \) always holds.

As to the value of the control 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 control was instrumental in keeping the system stable as the feedback gains grow, and correlates with a model of Fedeltgen (1979).

5.3 Arm dynamics

When considering the dynamics of the planar arm modeled as two rigid bodies connected by two cylindrical joints, and introducing an arm configuration vector \( \mathbf{q} = \mathbf{q}_m \) three of joint angles, we can write an equation of the arm dynamics in the form (see Craig 1986)

\[
M(q) \ddot{q} + C(q, \dot{q}) = \tau
\]

where \( M(q) \) is an inertia matrix, vector \( C(q, \dot{q}) \) gives Coriolis and centrifugal forces, and \( \tau \) 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 Strien 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 forms depend on the states of the flexor, respectively 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 simulations we used the algorithms of Sects. 2.4 to control the system described in Sect. 5.3. We considered the feedforward control interval (1) to be \( T = 0.3 s \) and the observation interval (5) to be \( T = 0.6 s \). Figure 4 demonstrates a typical form of the non-score-based feedforward input \( \tau \). The feedforward value changes at time points 0.0, 0.05, 0.12, 0.15, 0.21, 0.27 and 0.36 s and is described by a 12-dimensional vector \( \mathbf{v} \) (4). We choose these 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 damp the overshoot and vibrations due to delays in muscle activation and in the feedback filter loop.

To monitor how precisely the arm comes to the final position, we used the position error parameters measured between \( r = T = 0.3 s \) and \( r = T = 0.6 s \). They are described in Section 2.4. Figure 1 shows the end-point trajectory for the learned motion from 20° to 60° in the shoulder joint and from 60° to 120° in the elbow joint. The overshoot is negligible and the motion smooth. Simulation shows that the learned feedforward reduces error \( E \) more than 30-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 analogues of muscle antagonist level monitored via electromyography is a non-physiological experiment. In our simulation,
cocontraction 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 cocontraction diminishes at the end of the motion (Adamsen and others 1990).

6.2 Fitting 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 $p$ (9) defining the motion trajectory was generated as a random vector with components in the domain $29^\circ \leq q_i \leq 147^\circ$; $14^\circ \leq q_i \leq 158^\circ$.

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

The data were used in an approximation procedure of Sect. 5 to find estimates for $U$ 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 $[V]_{ij}$ 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 $p$ in the task parameter space.

The learning process is illustrated in Fig. 4, where the error $[V]_{ij}$ is shown as a number of the randomly generated task parameter vector $p$. A tolerable error $[V]_{ij}$ is shown by a dashed line. The error diminishes as more data are stored in the database. 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 $U$ were stored at 150 points and the sensitivity matrices $G$ at 29 points, resulting in 51 kbytes of storage memory. Figure 6 represents the motions, for which vectors $U$ are stored, by lines connecting initial and final arm tip positions.

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

**Fig. 3.** Pattern of the muscle activation: Left panel, shoulder joint; right panel, elbow joint. Solid line presents the integral area under the (3) bounded as if (7 mm. Dashed line, lower trapezoid, dashed-dotted line, integral area under the inverse sign.

**Fig. 4.** Dependence of the motion error for the approximated motion on the number of the generated set of the motion parameters. Vertical dotted lines show error dependence on learning. Horizontal dashed line gives an acceptable error.
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 muscle activation delays and time dependences 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 potential use for robotics and control of other complex technical systems. Some robotic applications of the concepts have been considered in Usher (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 task 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 Algorithm. We implement our concepts 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 3 how to choose the parameters of control or, at least, timing of the control rewarding. Next, it is clear that the presented algorithm 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. Moreover, we consider it important that all algorithms do successfully work together and solve an otherwise very complicated control task.