MODELING, IDENTIFICATION AND CONTROL, 1988, VOL. 9, NO. 4, 207–222
doi:10.4173/mic.1988.4.4

Learning control of redundant degrees of freedom robots by optimization in parameterized control space

ERLING LUNDE† and JENS G. BALCHEN†

Keywords: *robots, learning control, kinematically redundant manipulators.*

A framework for the learning control of robots based on a parameterized control space is discussed. Emphasis is put on how to utilize stored motion knowledge. The principles are applied to an example of redundancy resolution for a simple manipulator. Global sub-optimal solutions for feedforward control are achieved using a simple optimization algorithm. A time-integral performance criterion is used.

1. Introduction

This work was inspired by observing the fast and accurate movements of a tennis player, and even more impressive, the swift movement of the highly flexible arms of an octopus. Both are examples of systems with redundant degrees of freedom performing skilled movements which obviously are the result of a learning process. This suggests that it should be worth while studying the underlying principles of these processes.

Motor control in man is dominated by learning how to do rather than calculating the proper control signals on-line. The learning mechanism is characterized by (1) using *a priori* knowledge (genetic information or previous experience) and (2) acquisition of knowledge through repeated trials.

From a mathematical/control theoretical point of view, human motion is extremely difficult to explain.

The locomotor system has redundant degrees of freedom for the vast majority of possible movements.

Both the dynamic and the kinematic equations are highly non-linear.

The number of variables/the system dimension is very high.

In spite of all this, the brain is capable of learning the most complex movements in a smooth, optimized way. What is optimized is not very well known even though some features are identified (Brooks 1987, Hogan and Flash 1987). Real life experience indicates that the rate of convergence in the learning process is sometimes very poor, while on the other hand it is usually quite robust.

The human control system seems to contain all the familiar principles: sensory feedback loops, feedforward control, adaptivity (compliance), optimization features. . . . When learning, the brain is in a supervisory state and the task is executed under sensory feedback control. When executing skilled motion, the brain is unat-

Received 1 October 1988.

† Center for Robotic Research‡, Division of Engineering Cybernetics, The Norwegian Institute of Technology, 7034 Trondheim, Norway.

‡ Supported by a grant from the Royal Norwegian Council for Scientific and Industrial Research.

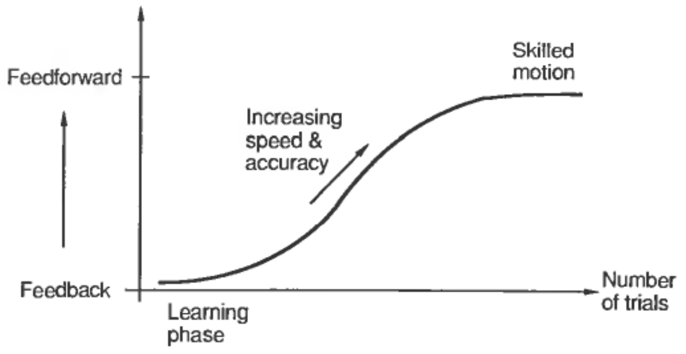

Figure 1. The learning process.

tentative and feedforward control is dominant. Accuracy and execution speed are improved at the same time as the system becomes more independent of sensory information (Fig. 1). Furthermore, learned motion can usually be repeated elsewhere in the task space, even after scaling and rotation.

A learning-based concept for manipulator feedforward control is developed in this paper according to the above-mentioned ideas. The method is based on parametric representations of the task and the system, i.e., introducing a parameterized control space. The main properties sought are:

a simple and robust learning algorithm

dynamic resolution of the redundant manipulator dofs

(sub-)optimization of overall behaviour (throughout the optimization time interval)

a compact and effective parametric representation requiring low storage capacity

utilization of *a priori* knowledge when performing new, but similar tasks

This also reflects the approximate contents of this paper. A demonstration of the final algorithm will be given in a simple example.

1.1. Work on motor control

Research on learning motor control (in humans and/or machines) can be roughly divided into two groups:

(1) Every specific movement is related to a 'motor program', which in general means the motor control trajectories (i.e., a joint space, time series representation) (Arimoto *et al.* 1984, Harokopos 1986, Hauser 1987, Saridis and Lee 1979).

(2) The proper (inverse) transformation from the task description to joint control signals is learned, usually in terms of the inverse dynamics. Raibert (1977) proposed learning and tabularizing linearized representations of the dynamics. Similarly, Albus' (1975) CMAC algorithm uses pure table look-up. Mukerjee (1988) suggests estimating inertial and geometrical parameters and then compensating for model errors by parameter perturbation.

The two groups do not exclude each other: Both in (Craig 1988) and (Atkeson 1986) the inverse plant model is assumed known (*a priori* or by learning) and used in a feedback loop, then additive correction trajectories are learned to compensate for model errors.

The method in the first group is relatively straightforward, and learned motion is reproduced by simply running the corresponding motor program. Unfortunately, the number of stored trajectories may very well become too large to ensure a reasonable repertoire of movements. In addition, the stored knowledge will not generally be useful when executing a new task, irrespective of whether it is similar to a previously learned one or not.

The second group is more general, mainly in the sense that it allows a separation of the task representation and the inverse plant model description. This means that different tasks (movements) can be compared, and previous experience utilized when performing new tasks.

1.2. Work on redundancy resolution

Most reported attempts to resolve manipulator redundancy are based on a kinematic analysis of the arm, using generalized inverses of the manipulator Jacobian (Klein and Huang 1985). By adding nullspace vectors, an instantaneous optimization will be performed; the inverse kinematic equation then becomes a gradient method:

$$\dot{\boldsymbol{p}} = J^+(\boldsymbol{q})\dot{\boldsymbol{q}} - [I - J^+(\boldsymbol{q})J(\boldsymbol{q})]\nabla g(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{1}$$

where $\dot{\boldsymbol{p}}$ and $\dot{\boldsymbol{q}}$ are the task space and the joint space velocities respectively. $J^+ = J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1}$ is the right-inverse (pseudoinverse) of the Jacobian matrix and $\nabla g(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the gradient of a criterion function to be minimized. The criterion function might be chosen as the manipulability measure $g(\boldsymbol{q}) = \det \sqrt{(J^{\mathrm{T}}(\boldsymbol{q})J(\boldsymbol{q}))}$ (Yoshikawa 1985), the distance to an obstacle (Kircanski and Vukobratovic 1984), etc. Both the pseudoinverse and the manipulability measure can be modified to take the manipulator inertias $M(\boldsymbol{q})$ into account, for example $J_M^+ = M^{-1}J^T(JM^{-1}J^T)^{-1}$.

Khatib (1987) uses a similar redundancy resolution scheme, but in terms of joint forces rather than joint velocities. The generalized inverse of the Jacobian is quite analogous to J_M^+ above, minimizing the manipulator's instantaneous kinetic energy.

Despite the simplicity and the optimization feature of Eqn. (1), its disadvantages are obvious. The optimization is instantaneous, which might oppose the overall behaviour during the optimization time interval. Furthermore, the manipulator performance will be strongly dependent on the initial state (joint position and velocity), as well as on the nature of the reference trajectory.

The augmented task space method, proposed in (Lunde *et al.* 1987), is based on a different principle, here the redundancy is eliminated by augmenting the task space to the dimension of the joint space. This method is particularly well suited for a micro-/micro-manipulator. The dynamic properties of the manipulator are then utilized through the control system design, e.g., by optimal control theory.

The global redundancy resolution problem has been addressed by Suh and Hollerbach (1987) and Nakamura and Hanafusa (1987), global in the meaning that optimality is obtained throughout the duration of the motion. Both approaches demand extensive calculations to find the (sub-) optimal solution.

2. The control problem

The manipulator dynamics are given by

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} = \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\tau} \tag{2}$$

where $\boldsymbol{q}$ is the joint coordinates, τ the generalized joint forces, $M(\boldsymbol{q})$ the inertia matrix and $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ the vector of centrifugal, Coriolis, frictional and gravitational forces. The kinematic position and velocity transformations are given by

$$\boldsymbol{p} = \boldsymbol{h}(\boldsymbol{q}), \qquad \dim(\boldsymbol{q}) = \boldsymbol{n} \tag{3}$$

$$\dot{\boldsymbol{p}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \qquad \dim(\boldsymbol{p}) = \boldsymbol{m} \tag{4}$$

where $\boldsymbol{p}$ is the task space position vector. The joint position and torques are restricted by $\boldsymbol{q} \in \boldsymbol{Q} \subset \mathbf{R}^n$, $\tau \in \boldsymbol{T} \subset \mathbf{R}^n$. The initial state is $\boldsymbol{q}(0) = \boldsymbol{q}_0$, $\dot{\boldsymbol{q}}(0) = \dot{q}_0$.

The general optimal control problem is to minimize the performance index

$$J[\boldsymbol{u}] = S(\boldsymbol{y}_T) + \int_0^T L(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{y}, \mathcal{T}, \mathcal{O})\, dt \tag{5}$$

and/or satisfy the endpoint condition $F(\boldsymbol{x}_T, \boldsymbol{y}_T) = 0$, T might be given or free. Here $\boldsymbol{x}_T = \boldsymbol{x}(T)$ and $\boldsymbol{x} = [\boldsymbol{q}^{\mathrm{T}}, \dot{\boldsymbol{q}}^{\mathrm{T}}]^{\mathrm{T}}$ is the state vector, $\boldsymbol{u} = \boldsymbol{\tau}$ is the control vector and $\boldsymbol{y} = [\boldsymbol{p}^{\mathrm{T}}, \dot{\boldsymbol{p}}^{\mathrm{T}}]^{\mathrm{T}}$ is the measurement vector. $\mathcal{T}$ and $\mathcal{O}$ are the task and environment (obstacle) descriptions respectively.

When applying the computed torque (the inverse dynamics) method (Lunde *et al.* 1987) we choose the torque vector $\boldsymbol{\tau} = M(\boldsymbol{q})\tilde{\boldsymbol{u}} - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ which gives the simple system $\ddot{\boldsymbol{q}} = \tilde{\boldsymbol{u}}$.

In this paper, the manipulation variable will not be the control vector $\boldsymbol{u}(t)$ itself, but rather a parameter vector $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbf{R}^d$ determining $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{\theta}, t)$, i.e., the mapping

$$\boldsymbol{u}\colon \boldsymbol{\Theta} \to \boldsymbol{U} \times [0, T]$$

The problem is then to find the optimal solution $\boldsymbol{\theta}^*$ of the control problem described above.

We will assume that any robot task can be represented by the pair $(\mathcal{S}, \boldsymbol{\theta})$ where $\mathcal{S}$ defines the (mathematical) *structure* on which the parameters $\boldsymbol{\theta}$ are applied. The control signal should preferably be a linear combination of $\mathcal{S}$ and $\boldsymbol{\theta}$

$$\boldsymbol{u}(\boldsymbol{\theta}, t) = \mathcal{S}(t)\boldsymbol{\theta} \tag{6}$$

where, for the scalar $\boldsymbol{u}$, we will have

$$\mathcal{S} = [f_1, f_2, \ldots, f_d] \tag{7}$$

Here, the f_is are linearly independent functions of time $f_i = f_i(t)$.

3. Parameterizing the control space

Usually, in the human mind, motion or actions are represented by some abstraction of the real signals (the neural activity), which in our consciousness will be a simple image of the real action, e.g., the command 'throw' triggers a complex series of neural signals that cause the actual throw to be executed. In our subconsciousness (the motor cortex), the representation has to be far more concrete; in a form that transforms the oversimplified image of the action into motor control signals. The transformation is obviously influenced by the relevant sensory information.

It seems probable that the task description–execution transformation should consist of two fundamentally different parts (1) A parametric description of the (abstract) task, (2) A control scheme based on the manipulator dynamics. For a traditional tracking task, the first part can be a trajectory generator, while the second is a feedforward/feedback controller possibly including the inverse manipulator dynamics. Alternatively, there could be only one transformation, from the abstract description to the real control signals.

Why do we want to parameterize the robot control problem?

The original optimal control problem, which is essentially infinite dimensional, is reduced to a d-dimensional non-linear optimization problem.

The parametric representation is well suited for effective storage.

A proper choice of parameterization allows the transfer of knowledge within a class of tasks (see later subsections).

Unfortunately, this also introduces some disadvantages:

The solution space is reduced (giving sub-optimal solutions).

It is difficult to find a 'close-to-optimal' parametric structure.

We will, for the remainder of this paper, restrict our investigation to end-point position control only. However, we believe that the concept is generally applicable, for example to force control and obstacle avoidance problems.

3.1. Parameterized task representation

Parametric representations of (human) motion trajectories have previously been investigated especially in the motor control/neuroscience literature: Flashner *et al.* (1988) proposed least squares curve fitting, while Hogan and Flash (1987) investigated optimized solutions of point-to-point trajectories.

The theory of series expansion states that any time trajector $x(t)$ can be arbitrarily well approximated by a finite series

$$x_n(t) = \theta_0 f_0(t) + \cdots + \theta_n f_n(t)$$

where θ_i is a constant (vector), and f_i a function of time. The error $e(t) = x(t) - x_n(t) \to 0$ uniformly as $n \to \infty$. When analysing these series expansions the question of orthogonality is important. The set of functions $f_0, \dots, f_n$ is *orthogonal* with respect to the weight function $w(t)$, on the interval $[a, b]$, if

$$\int_a^b f_i(t) f_j(t) w(t)\, dt = \begin{cases} 0 & \text{if } i \neq j \\ c_i & \text{if } i = j \end{cases} \tag{8}$$

Furthermore, the set is *orthonormal* if $c_0 = \cdots = c_n = 1$.

Some functions frequently used for mathematical approximation are shown in Table 1 together with their orthogonal intervals and weight functions. The first function $f_i(t) = t^i$ defines the finite Taylor series expansion, which is very simple, but not orthogonal. When orthogonalizing the basic polynominals we get the Legendre polynominals $\mathscr{L}_i(t)$ (a detailed overview of orthogonal polynominals is given by Vlach (1969)). In Vlassenbroeck and van Doren (1988) Chebychev polynominals $T_i(t)$ are successfully used to solve non-linear optimal control problems, here both the control and state signals are parameterized. Laguerre polynominals can be made orthonormal (with $w(t) = 1$) by defining the Laguerre *functions*, $l_i(t) = e^{-t/2} L_i(t)$. In communication theory, the Laguerre functions have been used to represent correlation functions as well as power spectrums (Lee 1960). Fourier series are obviously best suited when approximating periodic functions.

Polynominals		Interval	$w(t)$
Basic	t^i	—	—
Legendre	$\mathscr{L}_i(t)$	$[-1, 1]$	1
Chebychev (1st class)	$T_i(t)$	$[-1, 1]$	$\frac{1}{\sqrt{(1-t^2)}}$
Laguerre	$L_i(t)$	$[0, \infty]$	e^{-t}
Fourier series	sin *it* or cos *it*	$[-\pi, \pi]$	1

Table 1. Some functions used in approximation theory

Alternatively, we might consider non-linear dependence on the parameters, e.g., $x_2(t) = \theta_1 \sin \theta_2 t$.

In particular, the parameter vector might be chosen as a sequence of sampled trajectory values $\boldsymbol{\theta} = [x_0, x_1, \ldots, x_N]^T$, where $x_i = x(i\,\Delta T)$, ΔT is the sampling interval and the final time $T = N\,\Delta T$. This parameterization is simple, though ineffective with regard to the number of parameters, and it allows a general solution (limited only by the choice of the sampling interval) of the optimal control problem which, however, may be very difficult to find. In Hsu and Cheng (1981) this is formalized by using block-pulse functions, applied to a linear optimal control problem.

At this stage the parameterization should be designed with respect to the kinematic properties of the manipulator. Certainly, the dynamics also need to be considered and therefore the choice of parametric structure should allow an additional variation (using an optimization algorithm) of the parameters to utilize these properties as well.

3.1.1. *Smooth motion. Point-to-point problem*

Given the initial values x_0, $\dot{x}_0$ $(\ddot{x}_0)$, move to the point $x_T(\dot{x}_T, \ddot{x}_T)$ at time T. 'Good' ways to do this are characterized by smooth movements causing low joint transmission wear and non-jerky motion (Hogan and Flash 1987). For example, the bang-bang solutions which are typical for minimum time problems are counter-optimal in almost every other respect; they cause maximum strain on the joints and they produce jerky and overshooting motion.

In the mathematical sense, a *smooth* curve $x(t)$ is continuous and has a continuous first derivative $\dot{x}(t)$ for all $t \in [0, T]$. In this paper we will usually assume continuous second and third derivatives as well.

Smooth trajectories can easily be derived analytically, e.g., by the calculus of variation: Let the trajectory be characterized by the criterion functional

$$J[x] = \int_0^T P(x, \dot{x}, \ddot{x}, z)\, dt \tag{9}$$

where the jerk $z = d\ddot{x}/dt$. Euler's equation gives the necessary condition for the minimum:

$$P_x - \frac{d}{dt} P_{\dot{x}} + \frac{d^2}{dt^2} P_{\ddot{x}} - \frac{d^3}{dt^3} P_z = 0 \tag{10}$$

where $P_x = \delta P/\delta x$.

Reasonable integrands might be the quadratic functions

1. $P(\ddot{x}) = \frac{1}{2}\ddot{x}^2$ (min. acceleration)
2. $P(z) = \frac{1}{2}z^2$ (min. jerk)
3. $P(\ddot{x}, z) = \frac{1}{2}\ddot{x}^2 + \frac{\alpha}{2} z^2$

Deriving the analytical solutions is straightforward

1. $x(t) = \theta_0 + \theta_1 t + \theta_2 t^2 + \theta_3 t^3$
2. $x(t) = \theta_0 + \theta_1 t + \theta_2 t^2 + \theta_3 t^3 + \theta_4 t^4 + \theta_5 t^5$
3. $x(t) = \theta_0 + \theta_1 t + \theta_2 t^2 + \theta_3 t^3 + \theta_4 \alpha^2 e^{-t/\alpha} + \theta_5 \alpha^2 e^{t/\alpha}$

The parameters can be determined from the endpoint conditions, if these are known. At least some parameters can be identified from the initial conditions.

If the number of parameters equals the number of endpoint conditions the parameterization is *minimal*, and there is exactly one solution $\boldsymbol{\theta}^*$ that will bring the system to the desired final state. If, however, we choose a larger number of parameters, the problem might be said to be *overparameterized*, which means that an additional optimization with respect to θ is possible.

3.2. *Transferring knowledge*

Once a specific movement is learned, it can of course be repeated with the same degree of accuracy (assuming no external disturbances), but this knowledge should also be of use when performing a new, similar movement. To deal with this problem, we need the following definitions.

Let two tasks be defined by the pairs $\mathscr{T}_A = (\mathscr{S}_A, \boldsymbol{\theta}_A)$ and $\mathscr{T}_B = (\mathscr{S}_B, \boldsymbol{\theta}_B)$, where the structures are assumed to be linear (Eqns. (6), (7)): $\mathscr{S}_A = [a_1, \ldots, a_n]$, $\mathscr{S}_B = [b_1, \ldots, b_m]$.

Definition 1. The task $\mathscr{T}_B$ is said to be *similar*[1] to $\mathscr{T}_A$ if every function a_i can be expressed as a linear combination of the $b_j s$:

$$\forall a_i,\ 1 \leqslant i \leqslant n,\ \exists_{j=1}^{m} \alpha_{ij} \in \mathbf{R} \text{ s.t. } a_i = \sum_{j=1}^{m} \alpha_{ij} b_j$$

The converse is not generally true, i.e., $\mathscr{T}_B$ similar to $\mathscr{T}_A \not\Rightarrow \mathscr{T}_A$ similar to $\mathscr{T}_B$. This means that even though any task defined on $\mathscr{S}_A$ also can be defined on $\mathscr{S}_B$, it is possible to define a task on $\mathscr{S}_B$ which might not be realizable on $\mathscr{S}_A$.

We will say that a learned task $\mathscr{T}_A$ is *transferable* to a new $\mathscr{T}_B$ if $\mathscr{T}_B$ is similar to $\mathscr{T}_A$. Furthermore, it is possible to transfer knowledge both ways if the tasks are structurally equal.

Definition 2. Two tasks $\mathscr{T}_A$ and $\mathscr{T}_B$ are *structurally equal* if both are similar to each other.

[1] Similarity in this context should not be confused with the similarity transformation of linear algebra.

The 'quality' of knowledge transfer from a learned task $\mathcal{T}_A$ to a similar $\mathcal{T}_B$ depends of the distance between them.

Definition 3. The *distance* between two tasks means the corresponding distance in the parameter space

$$d(\mathcal{T}_A, \mathcal{T}_B) = \|\boldsymbol{\theta}_A - \boldsymbol{\theta}_B\|$$

where any suitable norm applies.

If necessary, the definition includes a transformation of $\boldsymbol{\theta}_A$ to fit the structure $\mathcal{S}_B$. For the linear parameterization (Eqn. (6)), let P_{AB} be a transformation matrix performing linear combinations of the elements of $\boldsymbol{\theta}_A$ to make this 'compatible' with $\boldsymbol{\theta}_B$. The matrix P_{AB} will in most cases be very simple: Let $\mathcal{S}_B = [1, t, t^2, t^3]$ and $\mathcal{S}_A = [1, t, t^2]$ ($\mathcal{S}_B$ is similar to $\mathcal{S}_A$ according to Def. 1). Then the transformation $\tilde{\boldsymbol{\theta}}_A = P_{AB}\boldsymbol{\theta}_A$, where

$$P_{AB} = \begin{bmatrix} I_3 \\ 0 \end{bmatrix}$$

(I_3 is a 3×3 unity matrix) fits $\boldsymbol{\theta}_A$ to the structure $\mathcal{S}_B$ such that $\tilde{\boldsymbol{\theta}}_A^{\mathrm{T}} = [\boldsymbol{\theta}_A^{\mathrm{T}} 0]$.

The above definitions applied to a proper parameterization should also be valid after scaling, rotation and translation of the movements (Mukerjee and Sastri 1986). However, identifying which representation is the best for the actual class of tasks may not be simple, and might be a learning process itself (see Sec. 3.3).

3.2.1. Interpolation in the parameter space

Assume that the tasks $\mathcal{T}_i$, $i = 1, \ldots, n$, are learned, how can this knowledge be used when performing a new task $\mathcal{T}_N = (\mathcal{S}_N, \boldsymbol{\theta}_N)$ similar to the $\mathcal{T}_i$s?

Let d_i be an *estimate* of the distance between $\mathcal{T}_N$ and $\mathcal{T}_i$ in the parameter space: $d_i = \hat{d}(\mathcal{T}_N, \mathcal{T}_i)$ (the real measure is obviously not available since $\boldsymbol{\theta}_N$ is unknown). Furthermore, assume that $\mathcal{T}_N$ is 'close' to $\mathcal{T}_i$ for all i. Then, a reasonable initial value for θ_N might be the weighted least squares estimate

$$\boldsymbol{\theta}_N^0 = \arg\min_{\boldsymbol{\theta}_N} \frac{1}{2} \sum_{i=1}^{n} \alpha_i \|\boldsymbol{\theta}_N - \boldsymbol{\theta}_i\|^2 \tag{11}$$

where $\alpha_i = 1/d_i^2$ reflects the uncertainty of using points 'far away' from the estimate. Deriving the analytical solution is straightforward:

$$\boldsymbol{\theta}_N^0 = 1/\Sigma\alpha_i \sum_{i=1}^{n} \alpha_i \boldsymbol{\theta}_i \tag{12}$$

To obtain good estimates of all elements of θ_N we need at least $d + 1$ distinct points in the d-dimensional Θ-space, among these, d of the vectors $\boldsymbol{\theta}_i$ should be linearly independent. Less points or linear dependence means that we get an estimate projected on a subspace of Θ.

For the point-to-point problem (§§ 2 and 3.1.1), the only possible *a priori* measure of distance between the tasks is in terms of the desired final position $\boldsymbol{y}_T^*$

$$d_i = \|\boldsymbol{y}_{TN}^* - \boldsymbol{y}_{Ti}^*\| \tag{13}$$

Because of the non-linear relationship between $\boldsymbol{\theta}$ and $\boldsymbol{y}_T$, the estimate should only be used when $\boldsymbol{y}^*_{TN}$ and $\boldsymbol{y}^*_{Ti}$ are 'close'.

3.2.2. Curve fitting by feedback control

If there is (almost) no previous experience about how to solve a motion problem, and if a feasible solution is needed (e.g., to avoid an obstacle) the following approach might be favourable:

Apply a simple non-optimal feedback control algorithm and record the corresponding control trajectory $u(t)$, $t \in [0, T]$. Then, approximate this by a parameterized signal $u_N(t) = \sum_{i=1}^{n} \theta_i f_i(t)$, e.g., as the minimum mean-square estimate

$$\theta_N^0 = \arg \min_{\theta_N} \frac{1}{2} \int_0^T \|u(t) - u_N(t)\|^2 \, dt \tag{14}$$

Here, the calculations will be considerably simplified if the functions f_i are orthogonal. With the definitions from Eqn. (8) we then get for the scalar $u(t)$:

$$\theta_{Ni}^0 = \frac{1}{c_i} \int_0^T u(t) f_i(t) w(t) \, dt \tag{15}$$

3.3 Generalization

What makes human intelligence superior to machine intelligence is the extensive ability to generalize information: General rules are derived from limited amounts of information by high-level reasoning. In our parameterized world, this means to identify the 'best' parametric structure describing the actual class of tasks. After learning a few specific movements, related to some *a priori* given structure, the structure itself should be updated, if necessary, to a maximum 'level of transferability'. By this we mean that the new structure will be general for the *class* of movements, while the old one was valid for only a few individual trials.

The problem of structure identification includes several important issues:

Should the structure be linear or non-linear?

Should it be based on physical knowledge, or should it be a black-box type model?

Should the representation be defined on the time or the frequency domain? Be continuous or discretized?

Normally these decisions will be made by the system designer based on his/her *a priori* knowledge about system properties, physical relations and heuristics. The choices distinguish between fundamentally different types of representations. Other substantial problems include the questions of system dimension, parameter identifiability and the optimality of the structure with respect to the performance criterion.

The problem can be formalized as follows (Ljung 1987):

Definition 4. A *structure set* is

$$\mathbf{S} = \{\mathscr{S}_\alpha(t) \colon \alpha \in \mathscr{A}\}$$

where the index set $\mathscr{A}$ might be a connected (open) subset $\mathscr{A} \subset \mathbf{R}^a$ or a set of numbers $\mathscr{A} \subset \mathbf{N}^a$($a$ is the dimension of $\mathscr{A}$).

Now, find α giving the 'optimal' structure. $\mathscr{A} \subset \mathbf{R}^a$ allows a differentiation of $\mathscr{S}_\alpha(t) = \mathscr{S}(t, \alpha)$ with respect to α, thereby making possible an analytical or numerical optimization of some performance criterion. However, it will be more natural that $\mathscr{A} \subset \mathbf{N}^a$ making $\boldsymbol{S}$ a countable set of possible structures. The selection mechanism might then be based on performance evaluation as α is varied.

4. Resolving redundancy by learning

This section will consider algorithms for improving performance through repeated trials. The algorithm might be considered to be *learning* if each trial is actually performed, or *iterative* if the solution is found off-line by simulation. The learning approach should give a (close to) feasible solution for every step to prevent disastrous results when executing the motion. The pure iterative method has no such limitation, but demands extensive knowledge about the dynamics and the kinematics of the manipulator.

Throughout this section we will assume that the inverse dynamics are known and may be used in the controller. A *feedforward* algorithm will provide the desired joint position, velocity and acceleration trajectories (Fig. 2) (kinematic learning). Our problem is how to calculate the optimal trajectories.

Specifically, the task is to move to the final point $y(T)$ while minimizing the performance criterion $J_c[\boldsymbol{\theta}]$. The question of resolving redundancy will not be addressed directly, as this is implicitly taken care of by the general optimization problem—which has inherent features of redundancy: The nonlinear relationship $y(T) = \boldsymbol{\Psi}(\boldsymbol{\theta})$ is a many-to-few mapping of possible solutions $\boldsymbol{\theta}$ giving the same result $y(T)$.

4.1. A feasible solution

Given the initial state $\boldsymbol{q}_0 = \boldsymbol{q}(0)$, $\dot{\boldsymbol{q}}_0 = \dot{\boldsymbol{q}}(0)$ find a solution $\boldsymbol{\theta}^*$ that moves the arm to the final position $\boldsymbol{p}_T^* = \boldsymbol{p}^*(T)$ at the given time T. With the computed torque method applied, we have the system

$$\ddot{\boldsymbol{q}}(t) = \boldsymbol{u}(\boldsymbol{\theta}, t) \tag{16}$$

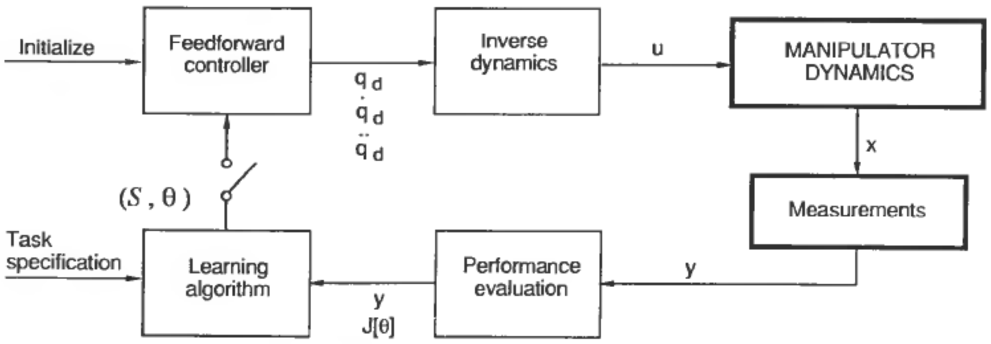

Figure 2. Trajectory learning algorithm.

where $\boldsymbol{u}(\boldsymbol{\theta}, t) = \mathcal{S}(t)\boldsymbol{\theta}$ is the parameterized acceleration vector. Let $\boldsymbol{\theta}^i$ be the solution of the ith iteration, and $\delta \boldsymbol{p}_T^i = \boldsymbol{p}_T^i - \boldsymbol{p}_T^*$ be the endpoint deviation.

From the definition of $\boldsymbol{u}(\boldsymbol{\theta}, t)$ we obtain

$$\dot{\boldsymbol{q}}_T = \int_0^T \mathcal{S}(t)\, dt\boldsymbol{\theta} + \dot{\boldsymbol{q}}_0 \tag{17}$$

$$\boldsymbol{q}_T = \int_0^T \int_0^t \mathcal{S}(\tau)\, d\tau\, dt\boldsymbol{\theta} + T\dot{\boldsymbol{q}}_0 + \boldsymbol{q}_0 \tag{18}$$

We have $\boldsymbol{p}_T^i = h(\boldsymbol{q}_T^i)$, and a Taylor series expansion of $\boldsymbol{p}_T^*$ at $\boldsymbol{q}_T^i$ gives

$$\boldsymbol{p}_T^* = \boldsymbol{h}(\boldsymbol{q}_T^i) + J(\boldsymbol{q}_T^i)(\boldsymbol{q}_T^* - \boldsymbol{q}_T^i) + \tfrac{1}{2}(\boldsymbol{q}_T^* - \boldsymbol{q}_T^i)^{\mathrm{T}} \frac{\delta}{\delta \boldsymbol{q}} J(\boldsymbol{q}_T^i)(\boldsymbol{q}_T^* - \boldsymbol{q}_T^i) + \cdots \tag{19}$$

Combining Eqns. (18) and (19) and introducing $\delta\boldsymbol{\theta}^i = \boldsymbol{\theta}^i - \boldsymbol{\theta}^*$, $\delta\boldsymbol{q}_T^i = \boldsymbol{q}_T^i - \boldsymbol{q}_T^*$, we get

$$\delta \boldsymbol{p}_T^i = \left[J(\boldsymbol{q}_T^i) \int_0^T \int_0^t \mathcal{S}(\tau)\, d\tau\, dt + R(\boldsymbol{\theta}^i) \right] \delta\boldsymbol{\theta}^i \tag{20}$$

where $R(\boldsymbol{\theta}^i)$ contains higher order terms of Eqn. (19) which are practically uncomputable since they contain the unknown value of $\boldsymbol{q}_T^*$.

Consider the following simple iterative algorithm for updating the parameter vector

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i + L\, \delta \boldsymbol{p}_T^i \tag{21}$$

where L is a linear or non-linear ($L = L(\boldsymbol{\theta}^i)$) learning operator (Arimoto *et al.* 1984, Hauser 1987). We can now rewrite Eqn. (21) as follows, by subtracting $\boldsymbol{\theta}^*$ on both sides:

$$\delta\boldsymbol{\theta}^{i+1} = (I + L[H(\boldsymbol{\theta}^i) + R(\boldsymbol{\theta}^i)])\, \delta\theta^i \tag{22}$$

Convergence of the algorithm is assured if Eqn. (22) defines a contraction mapping, i.e.,

$$\|I + L(\boldsymbol{\theta}^i)[H(\boldsymbol{\theta}^i) + R(\boldsymbol{\theta}^i)]\| \leqslant \rho < 1$$

Alternatively, evaluate (estimate) the eigenvalues of the matrix expression in Eqn. (22). The algorithm converges when all the eigenvalues are inside the unit circle. Obviously, the best possible operator would be

$$L(\boldsymbol{\theta}^i) = -[H(\boldsymbol{\theta}^i) + R(\boldsymbol{\theta}^i)]^{-1} \tag{23}$$

which gives $\rho = 0$ and convergence in one step. This is, unfortunately, not generally possible, for two reasons:

$R(\boldsymbol{\theta}^i)$ is unknown.

The matrices $H(\boldsymbol{\theta}^i)$ and $R(\boldsymbol{\theta}^i)$ will have the dimensions $m \times d$ where usually $d > m$, consequently the matrix inversion in Eqn. (23) will not be possible.

We might try a pseudoinverse $L = -T^+$, where $T^+ = T^{\mathrm{T}}(TT^{\mathrm{T}})^{-1}$. For example, choose the sub-optimal operator $L(\boldsymbol{\theta}^i) = -[H(\boldsymbol{\theta}^i)]^+$. However, convergence is not clear (not even for $R(\boldsymbol{\theta}^i)$ 'small'), and it is generally very hard to investigate.

Several methods for non-linear optimization are proposed in the literature. The properties of convergence of the different approaches depend on the nature of the equations, and the method should be chosen and adjusted after investigating the actual problem. For example, to minimize the criterion $J_f = \|\delta p_T\|^2$ (Euclidian norm), we might choose a gradient method: By Eqns. (4) and (17) we find

$$\nabla_\theta J_f = \frac{\delta J_f}{\delta \theta} = \int_0^T \int_0^t S^T(\tau)\, d\tau\, dt\; J^T(\theta)\delta p_T^i \tag{24}$$

We now get, analogously to Eqn. (21)

$$\theta^{i+1} = \theta^i - \alpha^i \nabla_\theta J_f \tag{25}$$

Convergence of the algorithm depends on the choices of α^i and the starting point θ^0; a local minimum can always be found by a line search method (Luenberger 1984).

4.2. Constrained optimization

In addition to satisfying the end-point conditions, we want to minimize the performance criterion $J_c[\theta]$ (i.e., with respect to the constraint $h(q_T^i) = p_T^*$). As argued in the previous section, $T(\theta^i) = H(\theta^i) + R(\theta^i)$ is practically not computable so that we should consider the approximation $T(\theta^i) \approx H(\theta^i)$, which is accurate for θ^i close to θ^*. To find the minimum, let θ 'slide along' the nullspace of $T(\theta^i)$, thus not changing the final position p_T^*:

$$\theta^{i+1} = \theta^i - \beta^i[I - T^+(\theta^i)T(\theta^i)]\nabla_\theta J_c[\theta^i] \tag{26}$$

where $I - T^+T$ is the null space projection matrix (defining a tangential plane in Θ-space at θ^i) and $\beta^i > 0$ is a scalar step-size parameter (the gradient projection method (Luenberger 1984)). The problem is that since $T(\theta^i)$ is non-linear in θ^i, the nullspace projection of $\nabla_\theta J_c[\theta^i]$ will probably not lie on the solution surface for p_T^* for most reasonable values of β^i, thereby causing an error

$$\begin{aligned} \delta p_T^{i+1} &= T(\theta^{i+1})\, \delta\theta^{i+1} \\ &= \beta^i T(\theta^{i+1})[I - T^+(\theta^i)T(\theta^i)]\nabla_\theta J_c[\theta^i] \neq 0 \end{aligned} \tag{27}$$

(assuming $\delta p_T^i = 0$ and $\delta\theta^i = 0$).

To overcome this problem we repeat the positional algorithm Eqns. (24), (25) after every iteration of Eqn. (26). The step-size parameter β_i should be determined such that the value of J_c is reduced and the constraint deviation δp_T^i is kept within 'reasonable' limits. (If $J_c[\theta]$ is non-convex in θ this method might not work at all.)

4.2.1. Minimum acceleration

We want to minimize a joint acceleration performance measure during the time interval $[0, T]$: Let Q be a symmetrical, positive definite weight matrix, and define

$$J_c[\theta] = \frac{1}{2}\int_0^T u^T Q u\, dt = \frac{1}{2}\int_0^T \theta^T \mathcal{S}^T(t) Q \mathcal{S}(t)\theta\, dt \tag{28}$$

Introduce

$$S_Q = \int_0^T \mathcal{S}^T(t) Q \mathcal{S}(t)\, dt \tag{29}$$

which immediately gives the performance criterion

$$J_c[\theta] = \tfrac{1}{2}\theta^T S_Q \theta \tag{30}$$

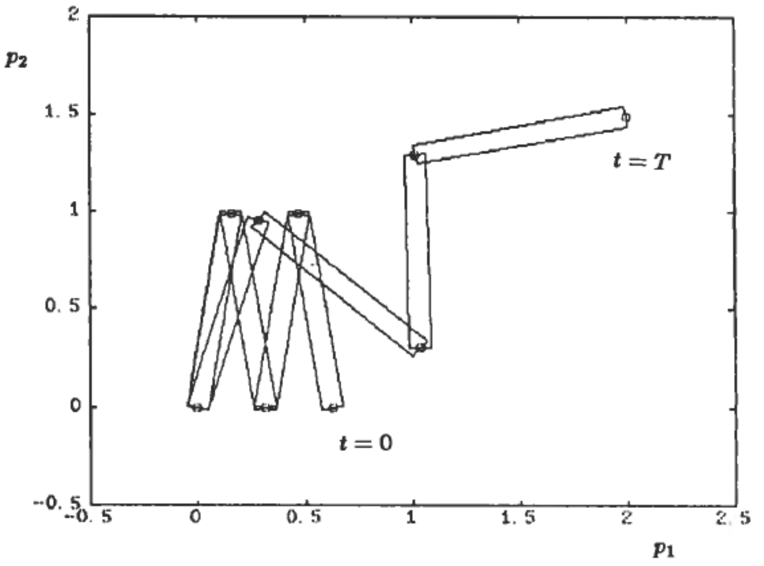

Figure 3. 4 dof. manipulator.

where S_Q is a constant symmetrical matrix. The gradient vector becomes

$$\nabla_\theta J_c = S_Q \theta \tag{31}$$

which can be derived analytically.

The weight matrix Q might be chosen $Q = \text{diag}\,(m_1, \ldots, m_n)$, where $m_1, \ldots, m_n$ are the manipulator inertias (the diagonal terms of the inertia matrix $M(\boldsymbol{q})$)—we get an approximate measure of 'energy' consumption.

5. Example

Consider the 4 dof. manipulator of Fig. 3 (all the links are of equal length: $l_i = 1$). The task is to position the tip of the arm in the (p_1, p_2)-plane, the initial state is given by $\boldsymbol{q}_0 = [0{\cdot}45\pi, -0{\cdot}90\pi, 0{\cdot}90\pi, -0{\cdot}90\pi]^{\mathrm{T}}$ and $\dot{\boldsymbol{q}} = 0$. We will investigate the double integrator model Eqn. (16), with four parameterized control signals

$$u_i(\boldsymbol{\theta}, t) = \theta_{1i} + \theta_{2i} t + \theta_{3i} t^2, \qquad i = 1, \ldots, 4$$

i.e., we have the parameter vector $\boldsymbol{\theta} = [\theta_{11}, \theta_{21}, \theta_{31}, \theta_{12}, \theta_{22}, \theta_{32}, \theta_{13}, \theta_{23}, \theta_{33}, \theta_{14}, \theta_{24}, \theta_{34}]^{\mathrm{T}}$ and the structure matrix

$$\mathscr{S}(t) = \begin{bmatrix} 1 & t & t^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & t & t^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & t & t^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & t & t^2 \end{bmatrix}$$

Assume that the manipulator Jacobian $J(\boldsymbol{q})$ is known.

First, we find a *feasible* solution. The desired final position is $\boldsymbol{p}_T^* = [2{\cdot}0, 1{\cdot}5]^{\mathrm{T}}$ and the final time $T = 1$. A reasonable choice for the initial parameter vector might be $\boldsymbol{\theta}^0 = [-1, -1, -1, 1, 1, 1, -1, -1, -1, 1, 1, 1]^{\mathrm{T}}$—a positive value of u_i is defined to give a rotational acceleration in the counter-clockwise direction. This initial trial gives the final position $\boldsymbol{p}_T = [1{\cdot}89, -0{\cdot}74]^{\mathrm{T}}$. The optimization algorithm

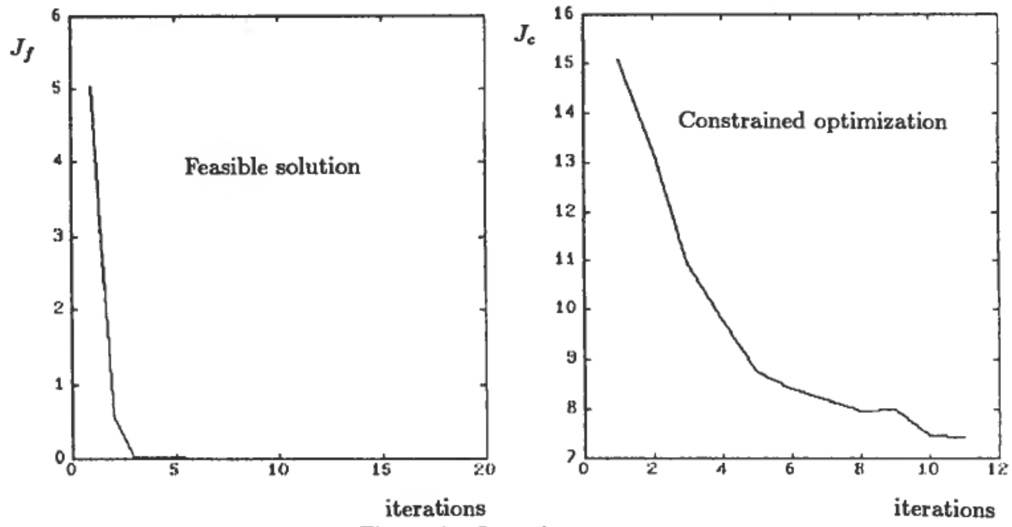

Figure 4. Learning curves.

is given by Eqns. (24) and (25), where the step-size parameter α^i is chosen as the inverse second derivative of $J_f(\theta^i)$ in the gradient direction (a quasi-Newton method). The step size is however bounded by $\|\theta^i - \theta^{i+1}\| < 1$ to prevent divergence when starting far away from the optimum. The convergence condition is $J_f < 0{\cdot}0001$, or an absolute positioning error of less than 0·01. This rather simple algorithm converged in 19 iterations, see the learning curve of Fig. 4.

Given the feasible solution, we want to *minimize acceleration* throughout the time interval (see Eqns. (28) to (31)). The weight matrix is chosen $Q = \text{diag}\,(4, 3, 2, 1)$. The constrained optimization is carried out as given in § 4.2, where the local 'optimum' of each trial is found by line search in the projected gradient direction (allowing only a small endpoint deviation). Starting from the feasible solution $\boldsymbol{\theta}_f$ the algorithm converges in 11 steps to the optimal $\boldsymbol{\theta}_{c1}$ with the corresponding $J_{c1} = 7{\cdot}43$ and endpoint deviation within the previously defined limit. The learning curve is shown in Fig. 4. The final configuration resulting from this (sub)-optimal solution is shown in Fig. 3, notice that the inner joint has hardly moved at all, in accordance with the high cost related to this.

Equivalently, several experiments with different final positions are carried out with the results given in Table 2. The same parametric structure is applied to each experiment; the tasks are obviously structurally equal (see Definition 2).

Now, assume that we want the manipulator to move to the new position $\boldsymbol{p}_{TN}^* = [1{\cdot}7,\ 1{\cdot}6]^{\mathrm{T}}$. By applying the weighted least square estimate $\boldsymbol{\theta}_N^0$ as explained in § 3.2.1, the manipulator moves to the point $\boldsymbol{p}_T = [1{\cdot}67,\ 1{\cdot}65]^{\mathrm{T}}$ which is much closer than

i	$\boldsymbol{p}_{Ti}^*$	J_{ci}
1	$[2{\cdot}0,\ 1{\cdot}5]^{\mathrm{T}}$	7·43
2	$[2{\cdot}0,\ 1{\cdot}0]^{\mathrm{T}}$	5·39
3	$[1{\cdot}5,\ 1{\cdot}2]^{\mathrm{T}}$	2·78
4	$[1{\cdot}0,\ 1{\cdot}5]^{\mathrm{T}}$	3·23
5	$[1{\cdot}5,\ 1{\cdot}8]^{\mathrm{T}}$	6·49

Table 2. Different final points.

when using the initial guess θ^0. Alternatively, using the non-weighted estimate $\alpha_i = 1$ gives the result $p_T = [1{\cdot}62,\ 1{\cdot}44]^T$, a deviation of $d = 0{\cdot}18$ vs. $d = 0{\cdot}05$ for the weighted estimate.

6. Conclusion

The new generations of robots are expected to perform a greater variety of tasks, to be more adaptive to changes in the environment, to be faster and more accurate, to be more 'intelligent'. In this paper, a learning control system based on the parameterization of the control space has been discussed. Stored knowledge about how to solve sample tasks provides good initial solutions to novel situations; fast convergence is achieved with simple algorithms.

By parameterizing, the dimension of the problem is reduced, with the consequence that we should expect some loss of optimality in the solution. However, a careful choice of parametric structure can give a satisfactory approximation with few parameters. Very complex non-linear optimization algorithms can be avoided if we assume that the robot is equipped with some basic motion knowledge ('genetic' information), i.e., a selection of points distributed in Θ-space such that we will always be able to find a starting point 'close' to the desired optimum. This means that we only have to worry about local convergence.

The manipulator dynamics can (and should) be included in the parametric learning algorithm, by learning either the dynamic model or the actual control signals (torques etc.). Non-linear dynamic systems can be linearized around a finite number of points in the state space, each linearized model can be learned and stored in a table for later use (Raibert 1977). Alternatively, the equations can be represented by a Volterra series which is a generalization of the convolution integral of linear systems, containing higher order impulse responses (Eykhoff 1974), here the problem will be to identify the Volterra kernels.

The example presented is simple, but should be fairly representative of the more general problems: Manipulation in a six dof. task space increases the problem dimension substantially, but the geometry will essentially be the same so that similar algorithms should work well. For complex optimization criteria where analytical derivations are unrealistic, the gradient vector can be evaluated by parameter perturbation. Tracking tasks can be treated in a similar way as the point-to-point problem, though requiring more parameters.

The section on redundancy resolution does not discuss this phenomenon explicitly, and we want to emphasize. *Resolving redundancy is basically a optimization problem, rather than a problem of kinematic analysis.* In the present example, singularities are avoided simply because they contribute to increase acceleration (which should be minimized) during the motion.

References

Albus, J. S. (1975). A new approach to robot control: The Cerebellar Model Articulation Controller (CMAC). *ASME J. Dyn. Sys., Meas. and Contr.*, 220–227, Sept. Vol. 97, No. 3.

Arimoto, S., Kawamura, S., and Miyazaki, F. (1984). Bettering operation of robots by learning. *J. of Robotic Systems*, **1**, 123–140.

Atkeson, C. G., and McIntyre, J. (1986). Robot trajectory learning through practice. *Proc. IEEE Int. Conf. on Rob. and Autom.*, 1737–1742, San Francisco.

Brooks, V. B. (1987). *The Neural Basis of Motor Control.* 129–148 (Oxford Univ. Press, New York.)

CRAIG, J. J. (1988). *Adaptive Control of Mechanical Manipulators.* 85–99 (Addison-Wesley, Reading, Massachusetts.)

EYKHOFF, P. (1974). *System Identification* (Wiley, New York.)

FLASHNER, H., BEUTER, A., and ARABYAN, A. (1988). Fitting mathematical functions to joint kinematics during stepping: Implications for motor control. *Biol. Cybern.*, **58**, 91–99.

HAROPOKOS, E. G. (1986). Optimal learning control of mechanical manipulators in repetitive motions. *Proc. IEEE Int. Conf. on Rob. and Autom.*, 396–401, San Francisco.

HAUSER, J. E. (1987). Learning control for a class of non-linear systems. *Proc. 26th CDC*, 859–860, Los Angeles.

HOGAN, N., and FLASH, T. (1987). Moving gracefully: Quantitative theories of motor coordination. *Trends in Neurosci.*, **10**, 170–174.

HSU, N.-S., and CHENG, B. (1981). Analysis and optimal control of time-varying linear systems via block-pulse functions. *Int. J. Control*, **33**, 1107–1122.

KHATIB, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE J. RA*, **3**, 43–53.

KIRCANSKI, M., and VUKOBRATOVIC, M. (1984). Trajectory planning for redundant manipulators in the presence of obstacles. In A. Morecki (ed.), *Theory and practice of robots and manipulators, Proc. RoManSy '84*, 57–63.

KLEIN, C. A., and HUANG, C.-H. (1985). Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Tr. SMC*, **7**, 245–250.

LEE, Y. W. (1960). *Statistical Theory of Communication.* (Wiley, New York.)

LJUNG, L. (1987). *System Identification: Theory for the User.* (Prentice-Hall, Englewood Cliffs, New Jersey.)

LUENBERGER, D. G. (1984). *Linear and Non-linear Programming.* (Addison-Wesley, New York.)

LUNDE, E., EGELAND, O., and BALCHEN, J. G. (1987). Dynamic control of kinematically redundant robotic manipulators. *Modeling, Identification and Control*, **8**, 159–174.

MUKERJEE, A. (1988). Joint force sensing for unified motor learning. *NATO ARW Sensor Dev. and Sys. for Robotics 1987* (Springer Verlag, Berlin.)

MUKERJEE, A., and SASTRI, T. (1986). Robot learning: Transferring knowledge between trajectories. Texas A&M Univ., Comp. Sci. Dept., report TR-87-016.

NAKAMURA, Y., and HANAFUSA, H. (1987). Optimal redundancy control of robot manipulators. *Int. J. Rob. Research*, **6**, 32–42.

RAIBERT, M. H. (1977). Analytical equations vs. table look-up for manipulation: A unifying concept. *Proc. 16th Conference on Decision and Control*, 576–579, New Orleans.

SARIDIS, G. N., and LEE, C.-S. G. (1979). An approximation theory of optimal control for trainable manipulators. *IEEE Tr. SMC*, **9**, 152–159.

SUH, K. S., and HOLLERBACH, J. M. (1987). Local versus global torque optimization of redundant manipulators. *IEEE Int. Conf. on Rob. and Autom.*, 619–624, Raleigh.

VLACH, J. (1969). *Computerized Approximation and Synthesis of Linear Networks.* (Wiley, New York.)

VLASSENBROECK, J., and van DOREN, R. (1988). A Chebychev technique for solving non-linear control problems. *IEEE Tr. AC*, **33**, 333–340.

YOSHIKAWA, T. (1985). Manipulability and redundancy of robotic mechanisms. *Proc. IEEE Int. Conf. on Rob. and Autom.*, 1004–1009, St. Louis.