MODELING, IDENTIFICATION AND CONTROL, 1990, VOL. 11, NO. 2, 109–121
doi:10.4173/mic.1990.2.4

Practical trajectory learning algorithms for robot manipulators

ERLING LUNDE† and JENS G. BALCHEN

Several alternative learning control algorithms are discussed, both from an inverse dynamics and an optimization point of view. The learning laws are derived in discrete time and do not need acceleration measurements. A simple algorithm using a constant learning operator is proposed to run in addition to a simple (PD) feedback controller. Its performance is comparable to other algorithms, and it works under non-ideal conditions where the others fail. Two simulation examples on (1) learning dynamic control, and (2) learning optimal redundancy resolution, are presented.

1. Introduction

Industrial manipulators usually perform the same task over and over again. In most systems the task is defined in terms of a joint space reference trajectory which is tracked by means of conventional single loop controllers. It is commonly known that even for almost rigid manipulators the tracking may not be very accurate. In practice, trajectory programming can be considered as an 'art' where the manipulator's lack of accuracy is compensated when designing the reference trajectory: Through an iterative (and manual) process the reference input is adjusted so that the actual response of the manipulator is close to the desired trajectory (which may be different from the reference trajectory given as input).

During the last years, most research on manipulator control has resulted in advanced, computationally expensive, feedback algorithms such as the inverse dynamics method and adaptive control strategies. However, industrial applications of these methods are rare, mainly because of their complexity.

On the other hand, iterative learning algorithms for robot trajectory tracking have been proposed by a number of researchers (e.g. Arimoto *et al.* 1984; Hauser 1987 and Craig 1988). The idea is to reduce tracking errors by a trial-and-error procedure: The same task is repeated several times, after each repetition a feedforward control signal is improved by some learning rule. Provided that the conditions under which the manipulator is performing are sufficiently deterministic and invariant from repetition to repetition, the control will converge to the 'ideal' one. The learning control strategy can serve as an effective means for

- improving the performance (accuracy, speed) of tracking tasks when the manipulator dynamics are partly unknown, and when feedback control is insufficient.
- solving difficult optimization problems by trial and error; such as optimal redundancy resolution.

Received 15 August 1990.

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

This paper was presented at the IEEE International Conference in Robotics and Automation, Cincinnati, Ohio, USA, May 13–18, 1990, and is reprinted with permission, from the Proceedings of the conference.

In this paper, several learning algorithms are proposed. The robustness and the performance of the learning algorithms are improved by adding feedback loops to the control system. We demonstrate the feasibility of the algorithms through simulation experiments using two planar manipulators with two and three degrees of freedom respectively.

2. The control problem

The rigid manipulator dynamics are given by

$$\left.\begin{aligned} M(q)\ddot{q}+n(q,\dot{q})&=\tau \\ n(q,\dot{q})&=C(q,\dot{q})\dot{q}+F\dot{q}+gr(q) \end{aligned}\right\} \tag{1}$$

where q is the vector of joint coordinates, τ the generalized joint forces, $M(q)$ the inertia matrix, $C(q,\dot{q})\dot{q}$ the vector of centrifugal and Corioli's forces, $F\dot{q}$ viscous friction forces, and $gr(q)$ the gravitational forces. The joint position and torque vectors are restricted by $q\in\mathbf{Q}\subset\mathbf{R}^n$, $\tau\in\mathbf{T}\subset\mathbf{R}^n$. The initial state is $q(0)=q_0$, $\dot{q}(0)=\dot{q}_0$. The kinematic position and velocity transformations are given by

$$p=h(q), \qquad \dim(q)=n \tag{2}$$

$$\dot{p}=J(q)\dot{q}, \qquad \dim(p)=m \tag{3}$$

where p is the task space position vector and $J(q)=\partial h(q)/\partial q$ is the manipulator Jacobian.

For the subsequent discussion we reformulate the above equations on a general state space form

$$\left.\begin{aligned} \dot{x}&=f(x)+B(x)u \\ y&=g(x) \end{aligned}\right\} \tag{4}$$

where

$$x=\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}=\begin{bmatrix} q \\ \dot{q} \end{bmatrix},\ u=\tau$$

$$f(x)=\begin{bmatrix} \dot{q} \\ -M^{-1}(q)n(q,\dot{q}) \end{bmatrix},\ B(x)=\begin{bmatrix} 0 \\ M^{-1}(q) \end{bmatrix}$$

$$y=\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}=\begin{bmatrix} p \\ \dot{p} \end{bmatrix},\ g(x)=\begin{bmatrix} h(q) \\ J(q)\dot{q} \end{bmatrix}$$

We will also need the *output* Jacobian matrix

$$G(x)=\frac{\partial g(x)}{\partial x}=\begin{bmatrix} J(x) & 0 \\ \dot{J}(x) & J(x) \end{bmatrix}$$

Actuator models can easily be included, and for most motors (e.g. DC-motors) this will not change the structure of (4). The control signal u will then be the motor input. However, we will assume that $u=\tau$ throughout this paper.

The manipulator task is to track a given task space reference trajectory $r(t)$, $t\in[0, T]$, and the corresponding tracking error is defined as $e=r-y$. The reference trajectory is *realizable* if it can be tracked perfectly: $e(t)=0$ with $q(t)\in\mathbf{Q}$ and $\tau(t)\in\mathbf{T}$ for all $t\in[0, T]$. Corresponding to $r(t)$ are the trajectories $x^*(t)$ and $u^*(t)$, we call these the *desired* trajectories, the error terms are denoted $\delta x=x^*-x$ and $\delta u=u^*-u$.

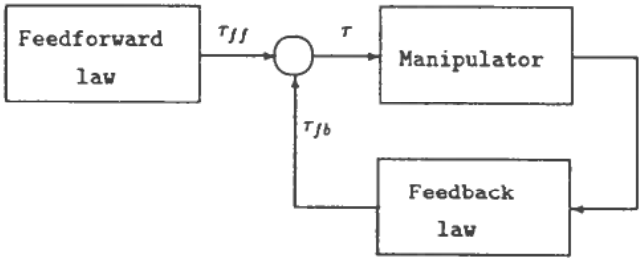

Figure 1. Feedforward and feedback control.

3. Feedback versus feedforward control

Even though feedback methods are the most used for manipulator control, feedforward control has some important advantages that should be utilized. For example, when solving finite time, optimal control problems we usually get open-loop solutions, while infinite time LQ-problems give closed-loop solutions.

For a perfectly rigid manipulator, the tracking error can quite easily be made satisfactorily small using any of several possible feedback algorithms: high-gain PD-controllers, the inverse dynamics method, or adaptive control (Samson 1987, Craig 1988). A feedback controlled system is robust with respect to disturbances and unmodeled effects such as gear backlash, non-viscous friction and varying load. On the other hand, because of unmodeled high frequency dynamics the feedback gains (and the system bandwidth) are upward bounded to ensure closed-loop stability.

Open-loop algorithms compute the control trajectory $u_{ff}(t)$ for all $t\in[0, T]$, prior to execution of the task—and then simply feed it forward when executing the task. Open-loop control does not introduce stability problems (but might of course excite resonant modes of the manipulator). Hence, the system bandwidth may be considerably higher than for the closed-loop case. However, an open-loop algorithm is very sensitive to disturbances and unmodeled dynamics.

Real-life robotic systems work under conditions of both deterministic and non-deterministic nature. The control system could therefore quite reasonably contain two parts: (i) a feedforward law effectively solving the a priori known details of the problem, or improving tracking by learning the feedforward torque, and (ii) a feedback law providing inaccurate, but robust tracking (see Figure 1).

4. Review of learning trajectory control

Several researchers have been investigating the problem of learning to track a reference trajectory. The learning algorithms are almost invariantly based on calculating an additive correction term to the feedforward control trajectory. The underlying 'philosophy' is to invert the dynamics (1) or (4) (which is also the idea behind the computed torque method):

$$u=B^{+}(x)(\dot{x}-f(x)) \tag{5}$$

where B^{+} is a generalized inverse of B. Most probably the dynamic model will not be completely known, we may then investigate whether an approximation will produce satisfactory results.

Most learning algorithms can be summarized by the following expression

$$u^{i+1}(t)=u^i(t)+L(.)De^i(t)$$

where the super-script denotes the iteration number, $L(.)$ is a learning operator—linear or nonlinear, and D is a differential operator, e.g.,

$$D=\frac{d}{dt}I, \quad \text{or} \quad D=\frac{d}{dt}+\lambda I,\ \lambda\geqslant 0$$

(Arimoto *et al.* 1984).

A concise and general description of a typical algorithm and its convergence properties has been given by Hauser (1987). The algorithm for system (4) takes the following form

$$u^{i+1}(t)=u^i(t)+L(y^i(t))\dot{e}^i(t) \tag{6}$$

where the (realizable) reference trajectory $r(t)$ is given on a finite interval $[0, T]$, and $L(y^i(t))$ is a nonlinear learning operator. The initial state is assûmed to be equal to the desired: $e(0)=0$. A condition for convergence is found as

$$\|I-L(y^i(t))G(x^i(t))B(x^i(t))\|\leqslant\rho<1, \quad \text{for all } t\in[0, T] \tag{7}$$

Note that (6) requires a reliable measurement or estimate of the acceleration signal ($\ddot{q}$ or $\ddot{p}$).

4.1. *Linearized dynamics*

If for some reason we have a control $\tilde{u}$ close to the desired control trajectory u^* (i.e. the one that gives perfect tracking), then the manipulator dynamics can be approximated by a linear time-varying equation (Oh *et al.* 1988). Linearize equation (4) around the state and control trajectories $\tilde{x}$ and $\tilde{u}$

$$\begin{aligned}\delta\dot{x}&\simeq(f_x(\tilde{x})+B_x(\tilde{x})\tilde{u})\delta x+B(\tilde{x})\delta u\\&=A(t)\delta x+B(t)\delta u\end{aligned}$$

where f_x denotes the partial derivative of f with respect to x. An obvious learning rule might now be, assuming that $y=x$ and $e=\delta x$,

$$u^{i+1}(t)=u^i(t)+B^+(t)(\dot{e}^i(t)-A(t)e^i(t))$$

where B^+ should be chosen as the Moore-Penrose inverse $B^+=(B^{\mathrm{T}}B)^{-1}B^{\mathrm{T}}=[0, M]$. The matrix $A(t)$ is quite complicated and therefore often omitted. The learning operator $L=B^+$ satisfies condition (7), since $G(x)=I$ and $I-LB=I-B^+B=I-MM^{-1}=0$. Also, observe that $B^+\dot{e}=M\delta\ddot{q}$ which makes sense.

Craig (1988) and Atkeson and McIntyre (1986) have proposed to apply the computed torque method (inverse dynamics)

$$\tau=\hat{M}(q)v+\hat{n}(q, \dot{q}) \tag{8}$$

where v is a new control input, and the hats indicate estimates of the real functions. Craig added a feedback loop $v_{fb}=\ddot{r}+K_v\delta\dot{q}+K_p\delta q$. Then the simple learning law

$$v_{ff}^{i+1}=v_{ff}^i+\delta\ddot{q}+\tilde{K}_v\delta\dot{q}+\tilde{K}_p\delta q$$

where $0<\tilde{K}_\alpha\leqslant K_\alpha$, is shown to compensate the effect of model errors.

Bondi *et al.* (1988) proposed a high-gain feedback law of the kind

$$\tau_{fb} = -K_a\ddot{q} - K_v\dot{q} - K_p q + \hat{g}r$$

where $\hat{g}r = gr(q_0)$, i.e., gravity is exactly compensated at $t=0$. The learning law

$$\tau_{ff}^{i+1} = \tau_{ff}^{i} + K_a\delta\ddot{q}^i + K_v\delta\dot{q}^i + K_p\delta q^i$$

is shown to converge when K_a, K_v, $K_p \to \infty$ (large 'enough').

4.2. *Performance limitations*

Most authors present convergence proofs for their learning algorithms. These are quite often based on methods from functional analysis, e.g., showing that (6) defines a contraction mapping (Arimoto *et al.* 1984; Hauser 1987). Others rely on the near-linearity of the feedback controlled system (Atkeson and McIntyre 1986, Craig 1988), then allowing techniques from the frequency analysis. Some limitations of the results are listed below:

The algorithms require acceleration measurements which may not be available. Numerical differentiation is noise sensitive and should be used with care.

Uniform convergence can only be ensured when $e(0)=0$, but can we expect reasonable performance when this is not the case? The algorithms including only $\dot{e}$ will give a tracking offset.

The reference trajectory is assumed to be realizable, what if this is not true (e.g., for a step reference)? Position errors should be taken into account.

Even though the convergence is proved, this may not ensure good performance: Most proofs based on contraction mappings show convergence in terms of the norm

$$\|e\|_\mu = \sup_{t\in[0,T]} e^{-\mu t}\|e(t)\|,\ \mu>0$$

where $\|e\|_\mu \to 0$ for μ large 'enough'. This means that the weight on errors decreases exponentially with time.

5. Issues for practical learning algorithms

We will in this section discuss some issues important for the implementation and the robustness of trajectory learning algorithms. The first practical consideration deals with the implementation of the algorithms on a digital computer.

5.1. *Discrete-time algorithms*

Assume that the control system operates with sampling time ΔT. The time interval $[0, T]$ is partitioned into N intervals given by $\{t_0, t_1, \ldots, t_N\}$ where $t_k = k\Delta T$. The measurements $y(t)$ are sampled at $t_0, t_1, \ldots$, and the control $u(t)$ is held constant through the intervals $[t_k, t_{k+1})$. The state space equations (4) can then be approximated by the difference equations

$$\left.\begin{aligned} x_{k+1} &= \psi(x_k) + \Phi(x_k)u_k \\ y_k &= g(x_k) \end{aligned}\right\} \tag{9}$$

where $x_k = x(t_k)$ etc. Clearly, the input u_k can only affect the outputs $y_{k+1}, y_{k+2}, \ldots$.

The approximation $\dot{e}(t_k)=(e(t_{k+1})-e(t_{k-1}))/2\Delta T$ is valid for small ΔT. Hence, the acceleration based learning law (6) has the discrete-time counterpart

$$u^{i+1}(t_k)=u^i(t_k)+L(y^i(t_k))\frac{e^i(t_{k+1})-e^i(t_{k-1})}{2} \tag{10}$$

with the convergence condition, given $e_0=0$,

$$\|I-L(y_k^i)G(x_k)\Phi(x_k)\|\leqslant\rho<1, \quad 0\leqslant k<N \tag{11}$$

Note that $\Phi(x_k)\to B(x_k)/\Delta T$ as $\Delta T\to 0$, and (10) converges to (6).

Remark. The central difference $\dot{e}(t_k)\approx(e_{k+1}-e_{k-1})/2\Delta T$ is a quadratic approximation, while the more common forward difference $\dot{e}(t_k)\approx(e_{k+1}-e_k)/\Delta T$ is a less accurate linear approximation (Dahlquist and Björk 1974).

So far, the discussion on the learning problem has been based on the inverse dynamics strategy. Further insight on the learning problem is obtained by an optimization approach (see also Togai and Yamano 1986).

5.2. *Local optimization*

Let us define a criterion function to be minimized

$$V(u_k)=\frac{1}{2}\|e_{k+1}\|_Q^2=\frac{1}{2}e_{k+1}^{\mathrm{T}}Qe_{k+1} \tag{12}$$

where Q is a positive definite weight matrix. The first and second partial derivatives of V with respect to u_k becomes

$$\nabla_u V_k=-(G(x_k)\Phi(x_k))^{\mathrm{T}}Qe_{k+1} \tag{13}$$

$$\nabla_{uu}V_k=(G(x_k)\Phi(x_k))^{\mathrm{T}}QG(x_k)\Phi(x_k) \tag{14}$$

Higher order derivatives vanish.

Observe that since the inertia matrix $M(q)$ is positive definite (Craig 1988) the matrix $\Phi(x_k)$ will have full column rank for all x_k. The physical interpretation is obvious: every non-zero input vector gives a non-zero contribution to the state vector at the next time instance: $\Phi(x_k)u_k=0\Leftrightarrow u_k=0$. The following important result indicates that most gradient based optimization methods will converge:

Result 1. The criterion function $V(u_k)$ is convex for all $G(x_k)$ with full column rank.

That is, the Hessian $\nabla_{uu}V_k$ is positive definite. For example, Newton's method (Luenberger 1984) gives the learning law

$$\begin{aligned}u_k^{i+1}&=u_k^i-[\nabla_{uu}V(x_k^i)]^{-1}\nabla_u V(x_k^i)\\&=u_k^i+L_k^i e_{k+1}^i\end{aligned} \tag{15}$$

which satisfies (11) with $\rho=0$.

The convergence properties of (15) can now be studied from an optimization point of view.

If $e_0=0$, then $e_1=G(x_0)\Phi(x_0)\delta u_0$ and $V(u_0)$ is purely quadratic. Hence, Newton's method is known to converge in one iteration, i.e., $e_1^1=0$. In the next iteration e_2 is

eliminated, and so on. The error correction propagates forward in time. By subtracting u^* from both sides of (15) we get

$$\delta u_0^{i+1} = \delta u_0^i - L_0^i e_1^i = (I - L_0^i G(x_0^i)\Phi(x_0^i))\delta u_0^i$$

which defines a contraction mapping when

$$\|I - L_0^i G(x_0^i)\Phi(x_0^i)\| < 1$$

For a bounded, but non-realizable reference trajectory, e.g., for $e_0 \neq 0$, the learning algorithm should emphasize position errors, and the input constraints must be taken into account. In the learning law (15), this can be achieved by tuning the weight matrix Q properly (see section 6).

Remark. The criterion function (12) measures the output errors, if it is replaced with $\frac{1}{2}\|e_{k+1} - e_{k-1}\|_Q^2$ the acceleration learning scheme (10) results. The latter criterion gives the same Hessian matrix $\nabla_{uu} V_k$, and similar local convergence.

However, the local analysis tells us little about the global properties of the algorithms. As the learning propagates from t_0, how does it behave near the end of the trajectories? The control update $L_k^i e_{k+1}^i$ may not be reasonable when being applied at the $i+1$th iteration since the state vector x_{k+1}^{i+1} may have changed significantly from x_{k+1}^i. A major concern is therefore to keep the output trajectories within reasonable bounds.

5.3. *Including feedback loops*

Second order systems without damping oscillate. This will definitely be harmful to the learning algorithm, since it means that the part of the trajectories that have not yet been learned might oscillate more or less randomly. In addition to the natural damping (friction), velocity feedback helps smoothing the response, e.g., we may apply the feedforward/feedback control law

$$u = u_{ff} - u_{fb} = u_{ff} - K_v \dot{q}$$

where K_v is positive definite.

More advanced feedback laws may be applied to improve the learning performance, e.g., the inverse dynamics method or decoupled PD-controllers. In this way the nominal performance (without learning) will be closer to the optimum, and the learning algorithms will be more robust and converge faster.

5.4. *Task space tracking*

When tracking a reference in the task space, two situations need special attention.

The output Jacobian $G(x)$ and the manipulator Jacobian $J(q)$ both loose rank in singular configurations. Consequently, a numerical learning algorithm using the inverse $G^{-1}(x)$ will break down in near-singular configurations. Assume that the reference can be tracked without entering singularities, our task is then to keep the state trajectories away from the singularities during learning.

For a redundant manipulator, the Hessian $\nabla_{uu} V_k$ is singular, and Newton's method (15) can not be realized. However, any 'sub-optimal' learning operator satisfying the condition (11) can be used. Several approaches for redundancy resolution have been

proposed in the literature (see Nenchev (1989) for a review), of which the method of gradient projection is one of the best known. The learning law can now be extended to

$$u_k^{i+1} = u_k^i + L_k^i e_{k+1}^i - \alpha P(x_k^i) \frac{\partial}{\partial u_k} w(x_{k+1}^i) \tag{16}$$

where $w(x)$ is a criterion function to be maximized (minimized), $P(x)$ is a projection matrix into the null space of $G(x)\Phi(x)$, and α is a step size parameter.

The inverse dynamics strategy gives the learning operator $L_k = (G(x_k)\Phi(x_k))^+$ (the pseudoinverse) which satisfies

$$\|I - (G(x_k^i)\Phi(x_k^i))^+ G(x_k^i)\Phi(x_k^i)\| = \|P(x_k^i)\| \leqslant 1$$

That is, if $e_k = 0$, then $\delta u_k^{i+1} = P(x_k^i)\delta u_k^i$ lies in the null space of $G(x_k^i)\Phi(x_k^i)$, such that the output error

$$e_{k+1}^{i+1} = G(x_k^{i+1})\Phi(x_k^{i+1})\delta u_k^{i+1} = 0$$

Furthermore, the above pseudoinverse solution minimizes the torque measure $\|u_k^i\|^2$.

6. Simulation examples

Several experiments have been carried out comparing the performance of the different learning algorithms proposed in the previous section. Two learning laws were derived:

Law 1: $u_k^{i+1} = u_k^i + L_k^i(e_{k+1}^i - e_{k-1}^i)/2$

Law 2: $u_k^{i+1} = u_k^i + L_k^i e_{k+1}^i$

with three candidates for the learning operator

Operator *a*: $L_k^i = [G(x_k^i)\Phi(x_k^i)]^+$, the pseudoinverse.

Operator *b*: $L_k^i = [G(x_k^i)]^+ L$, where L is a constant matrix.

Operator *c*: $L_k^i = [G(x_k^i)\Phi(x_k^i)]^T Q$, the steepest descent method.

The global positional tracking performance was measured in terms of the criterion function

$$J = \frac{1}{T}\int_0^T \|e_1(t)\|^2 \, dt$$

A reference trajectory was chosen as

$$r_1(t) = \begin{bmatrix} \alpha(1 - \cos(\omega t)) \\ \alpha \sin(\omega t)) \end{bmatrix} + r_{1,0} \tag{17}$$

The initial guess for the feedforward control trajectory was $u_{ff}^0 = 0$. No acceleration measurement was available. The iterative algorithm was terminated when $J^i < 10^{-4}$, when $i = 50$, or when no further improvement was achieved. For the discrete-time simulation we used the fourth-order Runge–Kutta method with sample time $\Delta T = 0{\cdot}005$. The learning algorithms were realized with $\Phi = \Delta T \cdot B$.

Example 1. Joint space tracking

The two-link manipulator shown in Fig. 2 was used for experiments on dynamic control in joint space, $G(x) = I$. The reference parameters were: $\alpha = 0{\cdot}5$, $\omega = 2\pi$, $T = 0{\cdot}5$. The initial state was equal to the desired, $x_0 = [1, -1, 0, 0]^T$, and $e_0 = 0$. No gravity was present.

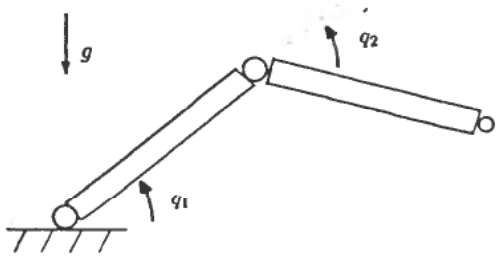

Figure 2. The test manipulator. Link mass: $m_1 = m_2 = 10{\cdot}0$, link length: $l_1 = l_2 = 1{\cdot}0$, both links are homogeneous bars.

Learning algorithm	Performance index	Number of iterations
Law 1, Op. a	4·1	50
Law 2, Op. a	∞	—
Law 2, Op. b	0·9	27
Law 2, Op. c	7·7	50
1.a & 2.b	0·7	5
Feedback only	198·0	—

Table 1. Example 1.B: Joint trajectory learning. (The performance index is normalized by $J/10^{-4}$.)

A: The manipulator joints were weakly damped assuming a viscous friction coefficient $F_{ii} = 1$ for each joint. Learning law 1 with Operator *a* converged in 7 iterations to $J = 1{\cdot}3 \cdot 10^{-5}$. All possible learning algorithms gave strongly oscillating behaviour. In fact, all the algorithms would break down for high frequency references—when the floating-point processor reached its upper limit.

B: The manipulator was equipped with the feedback control $u_{fb} = K_p e_1 + K_v e_2$ with $K_p = \text{diag}(10, 10)$ and $K_v = \text{diag}(50, 50)$.

The acceleration based law, 1.*a* converged fast to a near-optimal solution (in 5 iterations), before proceeding monotonically but slowly towards the minimum. This deficiency is due to approximation errors in the discrete-time learning law. See table 1 for the simulation results. The algorithm performed almost equally well when assuming a constant inertia matrix $M(q(0))$: The assumption is valid for bounded variations in q_2; the convergence condition (7) is satisfied for $-1{\cdot}8 < q_2 < -0{\cdot}5$ and for all q_1.

Three alternative algorithms based on law 2, with learning operators *a*, *b* and *c*, respectively, were also tested. The Newton method 2.*a* now diverged. The constant learning operator (2.*b*) was tuned to

$$L = \begin{bmatrix} 500 & 0 & 50 & 0 \\ 0 & 500 & 0 & 50 \end{bmatrix}$$

which satisfies $0{\cdot}83 < \|I - L\Phi^{-1}(q)\| < 0{\cdot}96$ for all q. Figure 3 shows the propagation of the angular error for joint 1 using algorithm 2.*b*. The weight matrix of the steepest descent operator was chosen as

$$Q = \begin{bmatrix} Q_1 & Q_1 \\ Q_1 & Q_2 \end{bmatrix}, \; Q_1 = \text{diag}(0{\cdot}5, 0{\cdot}5), \; Q_2 = \text{diag}(0{\cdot}05, 0{\cdot}05)$$

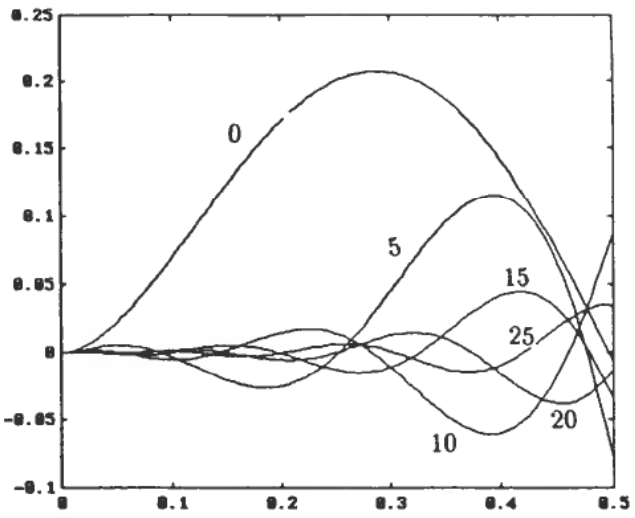

Figure 3. Example 1.B: Error compensation, angle joint 1 (after 0, 5, 10, 15, 20, 25 iterations).

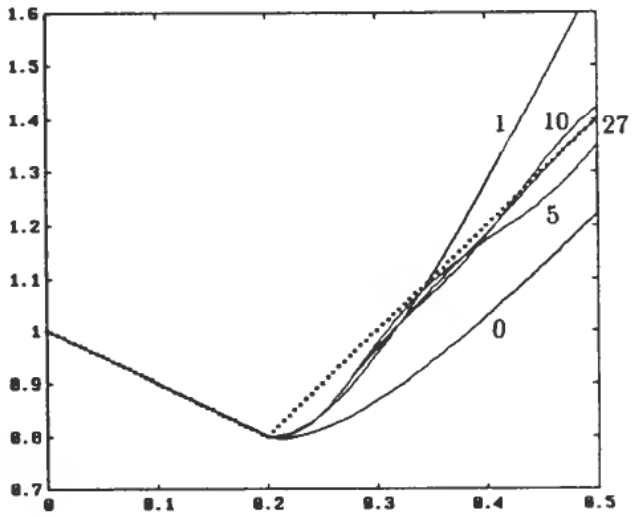

Figure 4. Example 1.C: Step reference tracking, angle joint 1 (after 0, 1, 5, 10, 27 iterations).

The slow near-optimum convergence of the latter algorithm is typical for fixed gain steepest descent methods (Luenberger 1984).

Finally, the algorithms 1.*a* and 2.*b* were combined giving the fastest convergence so far.

C: A non-realizable reference trajectory was specified

$$r_1(t)=\begin{bmatrix}-t\\ t\end{bmatrix}+q_{1,0}, t\leqslant t_s;\quad \begin{bmatrix}2(t-t_s)-t_s\\ -2(t-t_s)+t_s\end{bmatrix}+q_{1,0}, t>t_s$$

The step occurred at $t_s=0{\cdot}2$, the other simulation parameters were the same as the above. The inputs are constrained by $|u_i|<U$, $i=1,2$, with the choice $U=200$.

The corresponding desired acceleration is zero everywhere, except for $t=t_s$ where it is infinitely large. Consequently, learning Law 1 cannot be used. Therefore, the output error driven Law 2 with the constant learning operator 2 was applied. The algorithm converged in 27 iterations, and the result for joint 1 is shown in figure 4.

Remark 1. The criterion function J does in general not decrease monotonically, $J^{i+1} \not< J^i$. Thus, the intermediate solutions occurring before the learning is completed may be non-feasible.

Remark 2. Including gravity does not essentially change the learning behaviour. This is due to the integral effect of the iterative algorithms.

Example 2. Task space tracking. Redundant manipulator.

A planar three link manipulator was simulated ($l_i=1{\cdot}0$, $i=1,2,3$). When considering the kinematics only (a trajectory generation problem), the model can be written

$$\dot{x}_1=x_2, \dot{x}_2=u, y=g(x) \tag{18}$$

where u is a joint acceleration input. The initial configuration was $q_0=[\pi/3, -2\pi/3, 2\pi/3]^T$. The reference trajectory (17) was used with $\alpha=0{\cdot}25$, $\omega=2\pi$, $T=3{\cdot}0$, i.e., drawing a circle of radius α three times in the (p_1, p_2)-plane. The sample time was $\Delta T=0{\cdot}02$.

We used the following simplified output Jacobian in the learning algorithms:

$$G(x)=\begin{bmatrix} J(x) & 0 \\ 0 & J(x) \end{bmatrix}$$

The approximation is accurate for low velocity tracking.

A: The tracking task was attempted learned using the algorithms 1.*a* and 2.*b*. First, learning was applied to the undamped model (18). Both algorithms forced the manipulator into singular configurations after a few iterations.

Then, the response was damped by adding a velocity feedback $\dot{x}_2=-Dx_2+u$, where $D=\text{diag}(10, 10, 10)$. Algorithm 1.*a* converged in 33 iterations to $J=7{\cdot}3\cdot10^{-5}$. Algorithm 2.*b* converged in 9 iterations to $J=5{\cdot}5\cdot10^{-5}$.

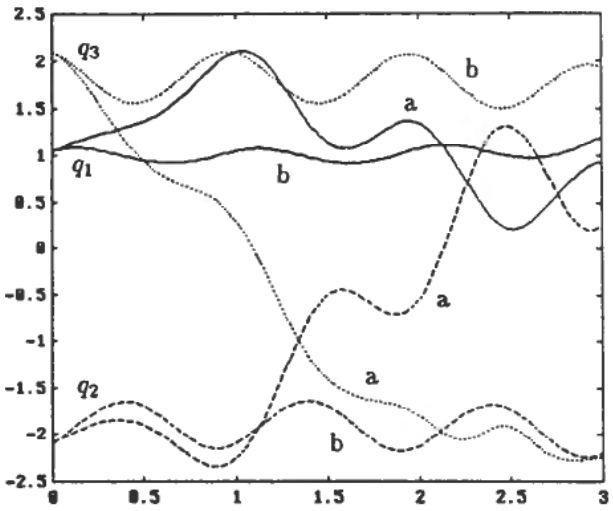

Figure 5. Example 2.B: Maximizing the manipulability measure. Joint trajectories before (b) and after (a) maximizing the manipulability measure.

B: Since the manipulator has redundant degrees of freedom, an additional cost function may be optimized along the trajectory. The manipulability measure

$$w(x)=\sqrt{(\det(J(x)J^T(x)))}$$

is maximized in the configurations 'furtnest away' from the singular ones (Nenchev 1989).

Based on the above solution with algorithm 2.*b*, the learning law (16) was applied. The gradient $\partial w(x)/\partial x$ was calculated numerically. The optimization was run for 25 iterations with the step size parameter $\alpha=5$, followed by a new learning session to re-establish the tracking accuracy. The time integral of $w(x)$ was used as a global measure of manipulability: it was improved from 1·66, to 2·10 after optimization. The joint trajectories before and after optimization is shown in Fig. 6. They clearly indicate that the initial configuration was not optimal.

7. Conclusions

From the previous sections, and in particular section 6, we might draw some conclusions regarding trajectory learning algorithms:

Feedback loops give more robust learning, and remove the need for 'intelligent' initial guesses for the feedforward trajectory τ_{ff}^0.

The algorithms including the acceleration error are sensitive to 'non-regularities', such as initial errors $e_0 \neq 0$, which leads to very high accelerations and divergency.

All algorithms compensate gravity quickly—this is due to the integral effect of the iterative algorithms. Nonlinearities and coupling forces were taken care of even for algorithm 2.*b* which was diagonal in nature.

Experience from simulation experiments shows that learning convergence is sensitive to tuning parameters such as learning and feedback gains.

The learning law based on output errors and constant learning gain (2.*b*) demonstrated good convergence properties and robust performance.

The learning strategy presented is local as opposed to a global method where the control trajectory is updated using information of the overall response (Lunde and Balchen 1988). Though more complex, the global approach is able to describe general optimal control problems.

Acknowledgment

This work was supported by a grant from the Royal Norwegian Council for Scientific and Industrial Research.

References

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.*, **3**, 1737–1742.

Bondi, P., Casalino, G. and Gambardella, L. (1988). On the iterative learning control theory for robotic manipulators. *IEEE J. Rob. and Autom.*, **4**, 14–22.

CRAIG, J. J. (1988). Adaptive control of mechanical manipulators. Addison-Wesley, Reading, Massachusetts.

DAHLQUIST, G. and BJÖRK, Å. (1974). Numerical methods. Prentice-Hall, Englewood Cliffs, New Jersey.

HAUSER, J. E. (1987). Learning control for a class of nonlinear systems. *Proc. 26th CDC*, 859–860.

LUENBERGER, D. G. (1984). Linear and nonlinear programming. Addison-Wesley, Reading, Massachusetts.

LUNDE, E. and BALCHEN, J. G. (1988) Learning control of redundant degrees of freedom robots by optimization in parameterized control space. *Modeling, Identification and Control*, **9**, 207–222.

NENCHEV, D. N. (1989). Redundancy resolution through local optimization: a review. *J. of Robotic Systems*, **6**, 769–798.

OH, S.-R., BIEN, Z. and SUH, H. H. (1988). An iterative learning control method with application for the robot manipulator. *IEEE J. Rob. and Autom.*, **4**, 508–514.

SAMSON, C. (1987). Robust control of a class of non-linear systems and application to robotics. *Int. J. Adaptive Control and Signal Processing*, **1**, 49–68.

TOGIA, M. and YAMOMOTO, O. (1986). Learning control and its optimality: analysis and its application to controlling industrial robots. *Proc. IEEE Int. Conf. on Rob. and Autom.*, **1**, 248–253.