

Learning Control of Robot Manipulators ¹

ROBERTO HOROWITZ
Department of Mechanical Engineering
University of California at Berkeley
Berkeley, CA 94720, U.S.A.
Phone: (510) 642-4675
e-mail: horowitz@canaima.berkeley.edu

Abstract

Learning control encompasses a class of control algorithms for programmable machines such as robots which attain, through an iterative process, the motor dexterity that enables the machine to execute complex tasks. In this paper we discuss the use of function identification and adaptive control algorithms in learning controllers for robot manipulators. In particular, we discuss the similarities and differences between betterment learning schemes, repetitive controllers and adaptive learning schemes based on integral transforms. The stability and convergence properties of adaptive learning algorithms based on integral transforms are highlighted and experimental results illustrating some of these properties are presented.

Key words: Learning control, adaptive control, repetitive control, robotics.

1 Introduction

The emulation of human learning has long been among the most sought after and elusive goals in robotics and artificial intelligence. Many aspects of human learning are still not well understood. However, much progress has been achieved in robotics motion control toward emulating how humans develop the necessary motor skills to execute complex motions. In this paper we will refer to learning controllers as the class of control systems that generate a control action in an iterative manner using a function adaptation algorithm, in order to execute a prescribed action. In typical learning control applications the machine under control repeatedly attempts to execute a prescribed task while the adaptation algorithm successively improves the control system's performance from one trial to the next by updating the control input based on the error signals from previous trials.

The term *learning control* in the robot motion control context was perhaps first used by Arimoto and his colleagues (c.f. (Arimoto et al., 1984; Arimoto et al., 1988)). Arimoto

¹To appear in the *ASME Journal of Dynamic Systems, Measurement and Control* 50th Anniversary Issue

defined learning control as the class of control algorithms that achieve asymptotic zero error tracking by an iterative *betterment process*, which Arimoto called learning. In this process a single finite horizon tracking task is repeatedly performed by the robot, starting always from the same initial condition. The control action at each trial is equal to the control action of the previous trial plus terms proportional to the tracking error and its time derivative respectively.

Parallel to the development of the learning and betterment control schemes, a significant amount of research has been directed toward the application of *repetitive control* algorithms for robot trajectory tracking and other motion control problems (c.f. (Hara et al., 1988; Tomizuka et al., 1989; Tomizuka, 1992)). The basic objective in repetitive control is to cancel an unknown periodic disturbance or to track an unknown periodic reference trajectory. In its simplest form, the periodic signal generator of many repetitive control algorithm closely resembles the betterment learning laws in (Arimoto et al., 1984; Arimoto et al., 1988). However, while the learning betterment controller acts during a finite time horizon, the repetitive controller acts continuously as regulator. Moreover, in the learning betterment approach, it is assumed that, at every learning trial, the robot starts executing the task from the same initial condition. This is not the case in the repetitive control approach.

My interest in learning and repetitive control arose in 1987, as a consequence of studying the stability of a class of adaptive and repetitive controllers for robot manipulators with my former student and colleague Nader Sadegh. My colleague and friend Masayoshi Tomizuka had been working very actively in the area of repetitive control and he introduced me to this problem. At that time there was much activity in the robotics and control communities toward finding adaptive control algorithms for robot manipulators which were rigorously proven to be asymptotically stable. The problem had been recently solved using passivity by Slotine and Li (1986) , Sadegh and Horowitz (1987) and Wen and Baynard (1988) . In contrast, most of the stability results in learning and repetitive control of that period relied on several unrealistic assumptions: either the dynamics of the robot was assumed linear, or it was assumed that it could be at least partially linearized with feedback control. Moreover, it was assumed in most works that the actual response of the robot was periodic or repeatable, even during the learning transient, and that joint accelerations could be directly measured. Nader and I had recently overcome some of these problems in our adaptive control research, and concluded that learning controllers could be synthesized and analyzed, using a similar approach. For us the main appeal of learning and repetitive controllers lied in their simplicity. In most of the other approaches to robot trajectory control, including parametric adaptive control, it is necessary to compute the so called inverse dynamic equations of the robot. In many of these schemes these equations have to be computed in real time. In contrast, in the betterment learning and repetitive control schemes, the control action is generated by relatively simple functional adaptation algorithms. Moreover, since most robot applications in industry involve the repeated execution of the same task, the idea of implementing a control algorithm which would “learn” through practice, without requiring any a-priori knowledge of the structure of the robot equations of motion, was also very appealing. Our approach to the synthesis of learning controllers relied on the following insights: i) In learning control,

motor dexterity is not gained through the use of feedback. This was the approach used in most adaptive controllers at the time (c.f. (Slotine and Li, 1987; Sadegh and Horowitz, 1987; Ortega and Spong, 1989)). In these adaptive schemes both the nonlinear control law and the parameter adaptation algorithm regressor are functions of the *actual* robot joint coordinates and velocities. ii) In contrast, in learning control algorithms, motor dexterity is gained through the use of a feedforward control action which is stored in memory and is retrieved as the task is executed. The learning process involves the functional adaptation of the feedforward action. iii) Feedback plays a fundamental role in stabilizing the system and in guaranteeing that the map between the feedforward function error and the tracking errors is strictly passive. It therefore became apparent to us that, in order to use passivity based adaptive control results in the synthesis and analysis of learning and repetitive algorithms, it was necessary to formulate and to prove the stability of an adaptive control law which accomplishes the linearization of the robot dynamics by feedforward rather than feedback control. We presented these results in (Sadegh and Horowitz, 1990) with the introduction of the so called Desired Compensation Adaptive Law (DCAL). In this adaptive scheme both the nonlinear control law and the parameter adaptation algorithm regressor are functions of the *desired* trajectories and velocities. Subsequently we were able to synthesize repetitive controllers for robot arms by replacing the adaptive law in the DCAL with a repetitive control law (Sadegh et al., 1990).

Unfortunately, as discussed in (Hara et al., 1988; Tomizuka et al., 1989; Sadegh et al., 1990), the asymptotic convergence of the basic repetitive control system can only be guaranteed under restrictive conditions in the plant dynamics or restrictions in the nature of the disturbance signals. These conditions are generally not satisfied in robot control applications. Most often, modifications in the update schemes are introduced, such as the so called Q filter modification (Hara et al., 1988; Tomizuka et al., 1989), that enhance the robustness of the repetitive controller, at the expense of limiting its tracking performance. Likewise, the convergence of betterment learning schemes is proven by appealing to strict assumptions regarding the initial condition of the robot at the beginning of each learning trial. Another shortcoming of the betterment learning and repetitive control schemes discussed so far, is that these algorithms were developed for the iterative learning of a *single task*. None of the research works in these areas provided a mechanism for extending the learning process so that a family of tasks can be simultaneously learned by the machine, or provide a systematic mechanism for using the *dexterity* gained in learning a particular task to subsequently perform a somewhat different task of a similar nature. After Nader left Berkeley to become a faculty member in the Georgia Institute of Technology, I began working on these problems with Bill Messner.

Our research has revealed that the robustness limitation of the basic betterment and repetitive control laws and the inability of these algorithms to learn multiple tasks in part stem from the fact that all these schemes use point to point function adaptation algorithms. These algorithms only update the value of the control input at the current instant of time and do not provide a mechanism for updating the control input at neighboring points. However, in most applications, the control function that must be identified is usually at least

piecewise continuous. Thus, the value of the control at a given point will be almost the same as those of nearby points. Point to point function update laws do not take advantage of this situation. This issue has implications in more general learning problems and content addressable memories. Let us consider as an example the case of multi-task learning control algorithms for robot manipulators. In this application, a function of several variables must be identified, namely the robot inverse dynamics. The trajectory used for training in betterment control cannot visit every point (or vector) in the domain of the function in a finite amount of time. Thus, the perfect identification of a control input function for one task using a point to point update law will not provide any information for generating the control input for other similar tasks unless the trajectories intersect, or some sort of interpolation is used. Similarly, in content addressable memories, it is desirable that the learning algorithm have an “interpolating” property, so that input vectors which are similar to previously learned input vectors, but are novel to the system, will generate output vectors that are similar to previously learned output vectors.

One solution to the interpolation problem in robot learning control was presented in (Miller, 1987) with the use of the so called “cerebellar model arithmetic computer” (CMAC). In this algorithm an input vector is mapped to several locations in an intermediate memory, and the output vector is computed by summing over the values stored in all the locations to which the input vector was mapped. The mapping of input vectors has the property that inputs near to each other map to overlapping regions in intermediate memory. This causes interpolation to be performed automatically.

In (Messner et al., 1991a) we introduced a class of function identification algorithms for learning control systems based on integral transforms, in order to address the robustness and interpolation problems of point to point repetitive and learning betterment controllers mentioned above. In these adaptive learning algorithms unknown functions are defined in terms of integral equations of the first kind which consist of *known* kernels and *unknown* influence functions. The learning process involves the indirect estimation of the unknown functions by estimating the influence functions. The entire influence function is modified in proportion to the value of the kernel at each point. Thus, the use of the kernel in both the update of the influence function and in the generation of the function estimate provides these algorithms with desirable interpolation and smoothing properties and overcomes many of the limitations concerning the estimation of multivariable functions of prior point to point betterment and repetitive control schemes. Moreover, the use of integral transforms makes it possible to demonstrate strong stability and convergence results for these learning algorithms.

The remainder of the paper we discuss the use of learning control in the robot tracking control context, and stress the similarities and differences between betterment learning schemes, repetitive control schemes and learning schemes based on integral transforms. Conclusions and reflections on some of the outstanding problems in this area are included in the last section.

2 Robot Tracking Control

We begin our discussion on learning control by considering its application to robotics. Specifically, we consider the trajectory tracking control of an n degree of freedom robot arm composed of rigid links which are connected by revolute joints. A schematic drawing of a two degree of freedom arm is shown in Fig. 1. The joint angular rotations between consecutive links are commonly called the joint coordinates. We denote by $q = (q_1, q_2, \dots, q_n)$ the joint coordinate vector, where q_i is the angular rotation at the i th joint. The last link in the robot linkage consists of the end effector, which can be a gripper or another tool. We assume that each degree of freedom in the robot is actuated by an ideal torque source and denote by $\tau = (\tau_1, \tau_2, \dots, \tau_n)$ the input torque vector, where τ_i is the torque produced by the i th actuator.

Most existing industrial robot manipulation tasks involve either the precise positioning of the end effector, such as in spot welding tasks, or the precise motion in space of the end effector, such as in painting tasks. We will assume in this paper that a robot task is defined by the specification of *desired* joint coordinates, velocities and acceleration trajectories during an interval of finite duration. We denote the desired joint coordinate, velocity and acceleration vectors by q_d, \dot{q}_d and \ddot{q}_d respectively and define the *task vector* u by

$$u = (q_d, \dot{q}_d, \ddot{q}_d), \quad (2.1)$$

in order to make our notation compact. For simplicity we will consider a deterministic environment and assume that the robot must execute a task defined by the task trajectory, $u_T = u(t)$ for $t \in [t_o, t_o + T]$, where t_o is the time at which the task begins and T is the duration of the task. We also assume that the desired trajectories are selected such that $u(t)$ is bounded for all t i.e., $u \in L_{3n}^\infty$.²

We assume in this paper that both the joint coordinate vector, q and joint velocity vector \dot{q} are available to the control system, as well as the task trajectory vector, u and define the tracking joint coordinate and velocity error vectors by

$$e_p = q_d - q, \quad \dot{e}_p = \dot{q}_d - \dot{q} \quad (2.2)$$

respectively. To simplify our subsequent discussion, it is convenient to define the so called *reference velocity error vector*, e_v , (Slotine and Li, 1987; Sadegh and Horowitz, 1990; Slotine and Li, 1991) by the linear combination

$$e_v = \dot{e}_p + \lambda_p e_p, \quad \lambda_p > 0, \quad (2.3)$$

and to use the vector

$$e = (e_p, e_v) \quad (2.4)$$

as the state of the manipulator tracking error dynamics. Notice that e is related to the vector (e_p, \dot{e}_p) by a nonsingular, constant homogeneous transformation. Thus, $e = 0$ implies that $e_p = \dot{e}_p = 0$.

²The reader who is unfamiliar with the notation used in this paper to denote L^p spaces and norms is referred to the appendix for definitions.

We consider that the robot has executed the commanded task without error if $e(t) = 0$ for $t \in [t_o, t_o + T]$. In practice such exact tracking is rarely possible since the feedback signals are commonly contaminated by some form of noise. In noisy environments we can evaluate the performance of the tracking control system using the truncated norms $\|e_T\|_2$ and $\|e_T\|_\infty$, where e_T is defined by

$$e_T(t) = \begin{cases} e(t) & \text{for } t \in [t_o, t_o + T] \\ 0 & \text{otherwise} \end{cases} . \quad (2.5)$$

We now examine the ideal feedforward control action which must be generated by the control system, if the robot is to execute a task without error. The dynamic equations of motion for rigid link robots are

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + d(q, \dot{q}) = \tau , \quad (2.6)$$

where q , \dot{q} and \ddot{q} are respectively the $n \times 1$ joint position, velocity and acceleration vectors, τ is the $n \times 1$ input torque vector, $M(q)$ is the symmetric and positive definite generalized inertia matrix, $C(q, \dot{q})\dot{q}$ is a vector resulting from Coriolis and centripetal accelerations, $g(q)$ is the generalized gravitational force vector and $d(q, \dot{q})$ is due to friction forces. The term $d(q, \dot{q})$ in (2.6) will be separated into two terms

$$d(q, \dot{q}) = d_l(q, \dot{q}) + d_c(q, \dot{q}) , \quad (2.7)$$

where the term $d_l(\cdot)$ represents the component of $d(\cdot)$ which are smooth functions with bounded derivatives, such as viscous damping and the term $d_c(\cdot)$ represents the component of $d(\cdot)$ formed by bounded but not necessarily smooth functions, such as Coulomb friction. In the remainder of the paper we will neglect the friction component $d_c(\cdot)$ in the formulation of the control law and will consider it to be a bounded disturbance to the control system.

Neglecting the component $d_c(\cdot)$ of the friction forces, and assuming that at the initial time t_o the robot is located at the exact location so that $e_p(t_o) = \dot{e}_p(t_o) = 0$, an open loop control action which achieves the desired motion is given by

$$\tau(t) = w(u(t)) , \quad t \in [t_o, t_o + T] , \quad (2.8)$$

where the function, $w(\cdot) : \mathcal{R}^{3n} \rightarrow \mathcal{R}^n$, is given by

$$w(u) = M(q_d)\ddot{q}_d + C(q_d, \dot{q}_d)\dot{q}_d + g(q_d) + d_l(q_d, \dot{q}_d) \quad (2.9)$$

and will be called in this paper the *inverse dynamics* function. This form of *feedforward* control belongs to the class of control algorithms which are frequently lumped in the robotics literature under the common heading of the *computed torque method*. Notice that the control in (2.8) is only a function of the desired trajectories, and is therefore a purely feedforward action. Under the ideal conditions that $e_p(t_o) = \dot{e}_p(t_o) = 0$, the inverse dynamics function $w(\cdot)$ is known exactly, and there are no unmodeled disturbances acting on the system, the control (2.8) linearizes the robot dynamics and achieves perfect robot tracking.

There are many variants of the computed torque method in the robot control literature (c.f. (Craig, 1986; Spong and Vidyasagar, 1989)). These variants differ on the combination of feedback and feedforward signals that are used as the arguments of the inverse dynamics function. In this paper we will only consider feedforward linearization actions in the form of (2.8), since feedforward control is readily extended to learning control.

It is well known that a purely feedforward control action of the form of (2.8) is not effective since the inverse dynamics function $w(\cdot)$ is rarely known with sufficient accuracy, the system is subjected to unmodeled dynamics and disturbances, and moreover, the robot dynamics in (2.6) is not stable, particularly when $d = 0$. Thus, it is necessary to incorporate to the control in (2.8) a feedback action to stabilize the robot closed loop dynamics. As previously mentioned, we assume in this paper that the error signal, $e(t) = (e_p(t), e_v(t))$ is available to the control system, as well as the task vector signal $u(t)$ in (2.1). Notice that we do not assume that the joint accelerations, \ddot{q} , can be measured, since it is impractical and difficult to instrument industrial robot arms with sensors which can directly measure joint accelerations.

We now consider the Desired Compensation Control Law (DCCL) introduced in (Sadegh and Horowitz, 1990). The control is given by

$$\tau = \hat{w}(u) + F_v e_v + F_p e_p + \sigma_n |e_p|^2 e_v, \quad (2.10)$$

with F_v and F_p being positive definite matrices and $\sigma_n > 0$. The term $\hat{w}(u)$ in (2.10) constitutes the feedforward control action. The function $\hat{w}(\cdot)$ is an estimate of the inverse dynamics function $w(\cdot)$ in (2.9). The last three terms in (2.10) are linear and nonlinear PD actions. In particular, notice that the term $\sigma_n |e_p|^2 e_v$ can be interpreted as a nonlinear PD action in which the feedback gains are proportional to the square of the Euclidean norm of the position error vector, e_p . This term is necessary to guarantee that the error dynamics of the learning control systems which will be discussed herein are globally asymptotically stable. Local stability results have been proven using only a linear PD control (Bayard and Wen, 1988).

In the expression for the control law in (2.10) we are accounting for the possibility of a mismatch between the true and estimated inverse dynamics functions. Let us denote this term by

$$\tilde{w}(\cdot) = w(\cdot) - \hat{w}(\cdot). \quad (2.11)$$

It is desirable that the inverse dynamics function error estimate, $\tilde{w}(\cdot)$, be as small as possible since, as we will show subsequently, the tracking error norm, $\|e\|_p$ is proportional to the feedforward error norm $\|\tilde{w}(u)\|_p$, for any p in $[1, \infty]$.

It is interesting at this point to discuss the role of the feedback action in maintaining the stability and robustness of the control system. To this end we can express the closed loop error dynamics form by applying the control law (2.10) to robot described by the dynamics (2.6) as follows

$$\dot{e} = f(u, e) + B(u, e) \tilde{w}_c, \quad \tilde{w}_c = [\tilde{w}(u) + d_c(u, e)], \quad (2.12)$$

where the vector $e = (e_p, e_v)$ is a state of the error dynamics, and the task vector, u , which was defined in (2.1) is an exogenous input to the system. The functions $f(\cdot, \cdot)$ and $B(\cdot, \cdot)$ are smooth functions of their arguments, and $f(u, e)$, $B(u, e)$ are bounded if e and u are bounded. Their expressions are given in (Sadegh and Horowitz, 1990; Messner et al., 1991a), but need not be detailed in this paper. The inverse dynamics function error, $\tilde{w}(\cdot)$, was defined in (2.11) and the Coulomb friction term d_c was defined in (2.7). The term \tilde{w}_c in (2.12) can be considered as the disturbance input to the tracking error dynamics (2.12). This is the term which gets minimized through the learning process. As the following results show, the feedback control law in (2.10) achieves the input/output stability of the error tracking dynamics to the mismatch term \tilde{w}_c .

Theorem 2.1 (Sadegh and Horowitz, 1990) *For the error dynamics (2.12), define the Lyapunov function candidate*

$$V_1(u, e) = \frac{1}{2}e_v^T M(u, e)e_v + \frac{1}{2}e_p^T [F_p + \lambda_p^2 \bar{\lambda} I]e_p, \quad (2.13)$$

where $M(u, e) = M(q)$, $\bar{\lambda}$ is the average of the maximum and minimum eigenvalues of the generalized inertia matrix, F_p is the position error feedback gain in the control law (2.10), and λ_p is the constant used in (2.3).

Then, for any bounded task vector trajectory, $u(t)$, it is always possible to find sufficiently large feedback gains, F_v , F_p , and σ_n in the control law (2.10) such that

$$\frac{d}{dt}V_1 \leq -\alpha_1 V_1 + e_v^T \tilde{w}_c, \quad (2.14)$$

for some $\alpha_1 > 0$.

The following results are readily derived from Theorem 2.1 (Sadegh and Horowitz, 1990):

- Exponential stability of the unperturbed system ($\tilde{w}_c = 0$).
- L^p input/output stability of the pair (\tilde{w}_c, e) , for $p \in [1, \infty]$ and $0 \leq \beta_1, \beta_2, \beta_3 < \infty$

$$\|e\|_p \leq \beta_1 \|\tilde{w}_c\|_p + \beta_2, \quad \lim_{t \rightarrow \infty} |e(t)| \leq \beta_3 \lim_{t \rightarrow \infty} |\tilde{w}_c(t)|. \quad (2.15)$$

β_3 is proportional to the magnitude of the initial error state, $e(0)$. Moreover, if $e(0) = 0$, then $\beta_2 = 0$.

• Strict passivity of the map between the input/output pair (\tilde{w}_c, e_v) . This result is immediate by integrating (2.14).

The following observations regarding the role of the feedback action can be stated.

1. The task vector trajectory $u(t)$, and Coulomb disturbances $d_c(u(t), e(t))$ are always bounded functions of time. Thus, if the inverse dynamics estimate, $\hat{w}(\cdot)$, is a bounded function of its arguments, then $\tilde{w}_c \in L_n^\infty$, and the tracking error vector trajectory, $e(t)$, also remains bounded. Thus, the control system is robust to the imprecise estimation of the robot inverse dynamics.

2. The signal $w(u(t))$ is only a function of the desired trajectories $u(t)$. Thus, the error signal $\tilde{w}(u(t))$ can be considered as an exogenous input to the error dynamics (2.12) and this is the signal which is minimized by the learning process. Moreover, the passivity of the map between the input/output pair (\tilde{w}_c, e_v) , can be exploited in the formulation of learning and adaptation laws by using the reference velocity error signal, $e_v(t)$, as the learning error signal.

3 Learning Control

In this section we discuss learning mechanisms for improving robot motor dexterity through practice. Consider a robot executing a task trajectory $u_T = u(t)$ for $t \in [t_o, t_o + T]$, such as the task shown in Fig. 1. We will evaluate the *dexterity* of the robot by measuring of the truncated error norm $\|e_T\|_p$, for some $p \in [1, \infty]$, where e_T , was defined in (2.5). As was pointed out in the previous section, for the control law (2.10) with fixed feedback gains, the error norm $\|e\|_p$ is proportional to the feedforward error norm $\|\tilde{w}(u)\|_p$. Hence, the smaller the norm $\|\tilde{w}(u_T)\|_p$, the more dexterous is the robot system in executing the task trajectory, u_T . We will assume that the robot can attempt to execute the same task repeatedly and will use the index k to denote the trial number. The learning algorithms that will be introduced subsequently attempt to improve the dexterity of the robot system, by updating in a recursive manner the feedforward inverse dynamics function estimate, $\hat{w}(u_T)$.

3.1 Learning betterment control

The first work in robot learning motion control is perhaps that of Uchiyama (Uchiyama, 1978). However, it is the pioneering work by Arimoto, Kawamura and Miyazaki (Arimoto et al., 1984; Arimoto et al., 1988) which in my opinion popularized the idea of learning and betterment control in robotics and formulated it in a rigorous framework. Other works by (Craig, 1984; Casalino and Bartolini, 1984; Atkeson and McIntyre, 1986) independently developed similar ideas. Arimoto and his colleagues defined learning control as the class of control algorithms that achieve asymptotic zero error tracking by an iterative *betterment process*. In this betterment process the robot repeatedly attempts to execute the same task trajectory, u_T , starting each attempt from the same initial condition, $e(t_o) = 0$. Under these conditions, the ideal feedforward action $\hat{w}(u)$ in the control law (2.10) is given by $w(u)$ in (2.9). This feedforward action can be expressed as a function of time with finite support $[0, T]$. Defining $t' = t - t_{o(k)}$ for $t \in [t_{o(k)}, t_{o(k)} + T]$, where $t_{o(k)}$ is the time at which the k th attempt begins, and T is the duration of the task, we can define the function $w_b(\cdot) : [0, T] \rightarrow \mathcal{R}^n$ by

$$w_b(t') = w(u(t)), \quad \text{for } t \in [t_{o(k)}, t_{o(k)} + T]. \quad (3.1)$$

The tracking controller proposed by Arimoto is a desired compensation control law of the form

$$\tau(t) = \hat{w}_{b(k)}(t - t_{o(k)}) + F_v e_v(t) + F_p e_p(t) + \hat{g}(q(t)), \quad (3.2)$$

for $t \in [t_{o(k)}, t_{o(k)} + T]$, where $\hat{g}(q)$ is a term which compensates for gravitational forces and $\hat{w}_{b(k)}(t')$ is the estimate of the ideal feedforward control action, $w_b(t')$ in (3.1) at the k th learning attempt. Arimoto proposed several learning algorithms for recursively updating the feedforward action $\hat{w}_b(t')$, depending on whether the joint acceleration vector, \ddot{q} , is assumed measurable or not. In this paper we discuss the so called *PI-type learning algorithm*, which uses only position and velocity error signals.³ Utilizing the subscript k to denote the feedforward action, \hat{w}_b , at the k th trial, the betterment learning law can be expressed as follows

$$\hat{w}_{b(k)}(t') = \hat{w}_{b(k-1)}(t') + K_L e_v(t' + t_{o(k)}) \quad \text{for } t' \in [0, T], \quad (3.3)$$

where $t_{o(k)}$ is the time at which the k th attempt begins and $K_L > 0$ is the learning gain. The feedforward control action $\hat{w}_{b(k)}(t')$ for $t' \in [0, T]$ is stored in memory after each attempt at executing the task, and is retrieved on the next attempt to generate the new feedforward action. Notice that (3.3) is a *point to point* functional adaptation algorithm in which the point $\hat{w}_b(t')$ of the function $\hat{w}_b(\cdot) : [0, T] \rightarrow \mathcal{R}^n$ gets updated by the term $K_L e_v(t)$ at the instance $t = t' + t_{o(k)}$. This adaptation law is illustrated by the sketch in Fig. 2. Arimoto has shown, under the strict assumption that the initial condition for each trial be $e(t_{o(k)}) = 0$, and the assumptions of invariance of the robot dynamics and existence of a noise free environment, that the combination of control law (3.2) and betterment learning algorithm (3.3) is successful in improving the robot's motor dexterity in the execution of the task trajectory, u_T , i.e.,

$$\|e_{T(k)}\|_\infty \leq \theta \|e_{T(k-1)}\|_\infty, \quad 0 \leq \theta < 1,$$

where $e_{T(k)}$ is defined in (2.5) with $t_{o(k)}$ in place of t_o , and the subscript k denotes the trial number. In the original stability analysis presented in (Arimoto et al., 1984; Arimoto et al., 1988), the robot dynamics was linearized around the desired trajectories and higher order terms were ignored. Moreover, the uniform convergence of e_v was not assured. Recently (Arimoto, 1990) has presented a rigorous stability analysis of the betterment scheme in (3.2) and (3.3) where these problems are addressed. Unfortunately, the robustness of this betterment scheme to varying initial conditions, i.e., $e(t_{o(k)}) \neq e(t_{o(k-1)}) \neq 0$, remains an open question. Examples showing the lack of robustness of these algorithms to variations in the initial conditions have been presented in (Heinzinger et al., 1989). On the other hand, the experimental results presented in (Kawamura et al., 1988) tend to indicate that, in some applications, these betterment schemes have some degree of robustness to small variations in the initial conditions. To address the robustness issues of the betterment algorithms to variations in the initial conditions, the following modified betterment learning law with *forgetting factor* was proposed in (Arimoto, 1990), following some of the ideas introduced in (Heinzinger et al., 1989).

$$\hat{w}_{b(k)}(t') = (1 - \alpha) \hat{w}_{b(k-1)}(t') + \alpha \hat{w}_{b(0)}(t') + K_L e_v(t' + t_{o(k)}), \quad (3.4)$$

where $t = t' + t_{o(k)}$, $0 \leq \alpha \ll 1$ is the forgetting factor and $\hat{w}_{b(0)}(\cdot)$ is initially the function $\hat{w}_b(\cdot)$ at the *first trial*. Unfortunately, the convergence of the tracking errors to zero can no

³Arimoto used the ‘‘PI’’ nomenclature for this algorithm because he defined the joint velocity vector, \dot{q} , as the output of the manipulator.

longer be guaranteed using the betterment scheme in (3.2) and (3.4), even under ideal conditions. To improve the asymptotic convergence of this scheme, it is suggested in (Arimoto, 1990) that the function, $\hat{w}_{b(0)}(\cdot)$ in (3.4) be periodically set to $\hat{w}_{b(k)}(\cdot)$, after a number of learning iterations.

3.2 Repetitive control

In the betterment learning schemes that were discussed in section 3.1, a trajectory, $u(t)$, of finite duration is used to specify the task that must be learned by the robot and it is required that the robot return to the desired initial configuration after completing a learning trial, before a new trial can be attempted. This condition may be unrealistic, particularly in light of the fact that a high degree of dexterity may be necessary to position the robot back to the desired initial configuration, which the robot may not have at the beginning of the learning process. In this section we discuss the repetitive control approach for training robots to execute periodic motions. In this approach it is not required that the robot periodically return to the exact desired initial configuration. The repetitive control approach to robot tracking control is based on the premise that the task that the robot must execute involves tracking a *periodic* task trajectory

$$u(t) = u(t - T), \quad (3.1)$$

with known period T . In other words, the desired robot configuration at the end of a task is equal to the desired initial configuration, and the robot must track this cyclic motion repeatedly.

Consider the robot control law given by (2.10) and neglect the Coulomb friction term d_c in (2.7). If the feedforward action $\hat{w}(u)$ in the control law (2.10) is given by $w(u)$ in (2.9) then, for any bounded robot initial configuration, the tracking error vector $e(t)$, converges exponentially to zero and the robot eventually follows the periodic desired trajectory. Notice that, because $u(t)$ is periodic, the ideal feedforward action, $w(u(t))$ can be expressed as a periodic function of time.

$$w_r(t) = w(u(t)), \quad w_r(t) = w_r(t - T). \quad (3.2)$$

Since the ideal repetitive feedforward function $w_r(t)$ is generally unknown, it must be estimated using some form of a functional estimation algorithm. The following prototype periodic generator, originally proposed by (Hara et al., 1988), can be used for estimating the periodic function $w_r(t)$

$$\hat{w}_r(t) = \hat{w}_r(t - T) + K_L e_v(t) \quad t \geq 0, \quad K_L > 0. \quad (3.3)$$

The initial condition of the function $\hat{w}(\tau)$, $\tau \in [-T, 0]$ is generally set to zero. Notice that, as in the case the betterment learning laws in Section 3.1, the repetitive control law, (3.3) is a point to point functional adaptation algorithm. At time t only the point $\hat{w}_r(t)$ of the function estimate, $\hat{w}_r(\cdot)$ is updated while points $\hat{w}_r(t + \delta t)$, $0 < |\delta t| \ll 1$, neighboring $\hat{w}_r(t)$ are not updated. As a consequence, the update law (3.3) will not generate a smooth function $\hat{w}_r(t)$

when the error signal $e_v(t)$ is contaminated by noise. The first stability analysis for repetitive control systems was presented by (Hara et al., 1988; Omata et al., 1987) for continuous time systems and by (Tomizuka et al., 1989; Tsai et al., 1988) for discrete time systems. The first rigorous stability analysis of the prototype repetitive algorithm (3.3) applied to robot manipulator was presented in (Sadegh et al., 1990). The analysis presented in these works reveals that the asymptotic convergence of this algorithm can only be guaranteed under restrictive conditions in the controlled plant dynamics, which are not satisfied by the robot arm.

An effective modification which enhances the stability of repetitive control systems is to use the so called Q filter (cf. (Hara et al., 1988; Tomizuka et al., 1989; Tomizuka, 1992)) in the repetitive control law.

$$\hat{w}(t) = \hat{w}_{ave}(t - T) + K_L e_v(t), \quad \hat{w}_{ave}(t) = Q(s)\hat{w}(t), > 0 \quad (3.4)$$

As discussed in (Tomizuka et al., 1989; Tomizuka, 1992), a good selection for the filter $Q(s)$, is to a moving average filter with zero phase characteristics. The essential difference between (3.4) and (3.3), is that in (3.4) the filter signal $\hat{w}_{ave}(t - T)$ is used instead of $\hat{w}(t - T)$, where $\hat{w}_{ave}(t - T)$ is a weighted average of all the values of the function $\hat{w}(\cdot)$ in a neighborhood of $t - T$. This is best illustrated in the discrete time domain. Let us define Δt as the repetitive controller sampling time, and the integer N by $N = T/\Delta t$. Denoting $Q(z)$ as the discrete time Q filter, where z is the one step advance operator, i.e., $z\hat{w}(t) = \hat{w}(t + \Delta t)$, the discrete time version of (3.4) is given by

$$\hat{w}_{ave}(t) = Q(z)\hat{w}(t) = \frac{\gamma_0 \hat{w}(t) + \sum_{j=1}^p \gamma_j [\hat{w}(t - j\Delta t) + \hat{w}(t + j\Delta t)]}{\gamma_0 + \sum_{j=1}^p 2\gamma_j}, \quad (3.5)$$

where the moving average filter $Q(z)$ is given by

$$Q(z) = \frac{\gamma_0 + \gamma_1(z^{-1} + z^1) + \dots + \gamma_p(z^{-p} + z^p)}{\gamma_0 + 2\gamma_1 + \dots + 2\gamma_p}, \quad (3.6)$$

and the constants γ_i 's are such that $\gamma_0 > \gamma_1 > \dots > \gamma_p \geq 0$, $N \geq p$. Typically, a Q filter with p small compared to N has proven to be effective.

Numerous experimental results (Chew and Tomizuka, 1990a) have verified that the robustness of the repetitive control algorithm is greatly enhanced with the introduction of the Q filter modification. Notice that the robustness of the algorithm is achieved by the averaging effect of (3.5). As discussed in (Chew and Tomizuka, 1990b), for linear time invariant discrete time systems, the Q filter modification enhances the robustness of the control system to unmodeled dynamics by placing the poles of the repetitive control law (3.4) and (3.5), which are given by $1 - z^{-N}Q(z) = 0$, inside the unit circle. Notice that if $Q(z) = 1$ these poles are placed at the unit circle. The reader is referred to (Chew and Tomizuka, 1990b), for further details on the stability and robustness analysis. Unfortunately, the convergence of the tracking errors to zero is no longer possible. This is easily verified in the discrete time domain by noticing that, since the poles of the repetitive law (3.4) and (3.5) are inside the

unit circle, $e = 0$ implies that $\hat{w} \rightarrow 0$. In the continuous time case, the term $\hat{w}_{ave}(t)$ in (3.4) can be expressed in terms of the integral transform

$$\hat{w}_{ave}(t) = Q(s)\hat{w}(t) = \int_{t-T}^{t+T} \Gamma(t, \tau)\hat{w}(\tau)d\tau . \quad (3.7)$$

where the function $\Gamma(\cdot, \cdot)$ is a positive definite symmetric *kernel* which satisfies $\int_{t-T}^{t+T} \Gamma(t, \tau)^2 d\tau = 1$, for all $t \in \mathcal{R}$. Fig. 3 shows a typical plot of the coefficients γ_j 's in (3.6), as a function of j and a typical plot of the kernel $\Gamma(t, \tau)$ in (3.7) for two different times: t_1 and t_2 . Note that the Q filter is an anticipative operator. There has been much recent interest toward the formulation of robust repetitive control schemes. The reader is referred to the works by (Tomizuka, 1992; Sadegh, 1991; Messner et al., 1991b) and their bibliographies for new results on the synthesis, and analysis of repetitive control algorithms.

3.3 Learning control based on integral transforms

In this section an integral transform for the representation of the inverse dynamics function $w(\cdot)$, originally introduced in (Messner et al., 1991a), which can be used in both repetitive and learning control, is presented.

3.3.1 Repetitive control

One method of assuring the piecewise continuity of the repetitive feedforward action $\hat{w}_r(t)$ in the control law (2.10) is to represent $\hat{w}_r(t)$ using an integral transform similar to the one used in (3.7). To this end, we will assume that the ideal feedforward action, $w_r(t)$ in (3.2) can be represented by the following integral equation

$$w_r(t) = \int_0^T K(t, \tau)c(\tau)d\tau , \quad (3.1)$$

where the function $K(\cdot, \cdot) : \mathcal{R} \times [0, T] \rightarrow \mathcal{R}$ is a **known** Hilbert-Schmidt kernel which satisfies

$$\int_0^T K(t, \tau)^2 d\tau = \kappa_j < \infty , \quad K(t, \tau) = K(t + T, \tau) , \quad (3.2)$$

and the *influence function* $c(\cdot) : [0, T] \rightarrow \mathcal{R}^n$ is **unknown**. If the kernel $K(t, \tau)$ in (3.1) is a truncated Gaussian function such as the one shown in Fig. 4, for a given time t' , the value $w_r(t')$ in Eq. (3.1) can be thought of as the weighted average of the values of the influence function $c(\tau)$ evaluated in a neighborhood centered at $\tau' = t - NT$, where N is the repetitive cycle and $t \geq NT$. As known in Fig. 4, the kernel $K(t, \tau)$ is periodic in t and it wraps around the interval $[0, T]$. Notice however that the influence function $c(\tau)$ for $\tau \in [0, T]$ is not equal to the function $w_r(t)$ for $t \in [0, T]$. Fig. 4 illustrates this fact. The figure shows the functions $w_1(\cdot)$ and $w_2(\cdot)$ which are respectively obtained from Eq. (3.1) using the influence functions $c_1(\cdot)$ and $c_2(\cdot)$. In (3.1) we are assuming that both $w_r(\cdot)$ and $c(\cdot)$ are unknown functions and that a kernel function, $K(\cdot, \cdot)$ can be selected rather arbitrarily such that (3.1) is satisfied. It is therefore useful to know what types of periodic functions can

be represented by (3.1) for a given kernel. (Messner et al., 1991a) provides conditions under which periodic functions $w_r(\cdot)$ with an infinite eigenfunction expansion can be represented by (3.1), for a wide class of kernel functions, such as the truncated Gaussian function in Fig. 4. Essentially it is necessary that the task trajectory, u_T be sufficiently smooth so that $\ddot{w}_r(t)$ exists and satisfies the Dirichlet conditions. We now introduce the following function adaptation law for estimating the unknown functions $w_r(t)$ and $c(\cdot)$:

$$\hat{w}_r(t) = \int_0^T K(t, \tau) \hat{c}(t, \tau) d\tau, \quad (3.3)$$

$$\frac{\partial \hat{c}(t, \tau)}{\partial t} = K(t, \tau) e_v(t), \quad (3.4)$$

where $e_v(t)$ is the reference velocity error signal in (2.3). In (3.3) the feedforward action $\hat{w}_r(t)$ is updated indirectly by the adaptation of the influence function estimate, $\hat{c}(t, \cdot)$ through (3.4). Fig. 5 illustrates the adaptation mechanism of the influence function estimate $\hat{c}(t, \cdot)$. Notice that the functions $\hat{w}_r(t)$ and $\hat{c}(t, \cdot)$ are no longer update in a point to point manner. As shown in (Messner et al., 1991a), the system formed by the error dynamics (2.6) and the function update law (3.3) is stable in the sense of Lyapunov and the error signal, $e(t)$ converges to zero asymptotically when $d_c = 0$. Moreover, as shown in (Horowitz et al., 1991), if the kernel $K(\cdot, \cdot)$ has a finite eigenfunction expansion, the equilibrium state, $(e, c(\cdot) - \hat{c}(\cdot)) = (0, 0)$, is globally exponentially stable. This result in turn guarantees some degree of robustness of the learning system to bounded disturbances and unmodelled dynamics (c.f. (Sastry and Bodson, 1989)). An important difference between the function adaptation law in (3.3) and (3.4) and the repetitive control law in (3.3) is that in (3.3) and (3.4), $\hat{w}_r(\cdot)$ is no updated in a point to point manner. The fact that a Hilbert-Schmidt kernel which satisfies (3.2) is used in the adaptation algorithm (3.3) and (3.4), allows us to guarantee the boundedness of the repetitive control input, $\hat{w}_r(t)$, and consequently, the uniform continuity of the error signal, $e(t)$. The repetitive control law in (3.3) can be represented as a degenerate case of the adaptation algorithm given by (3.3) and (3.4), were the kernel $K(t, \tau)$ is a periodic *Dirac delta impulse*. However, the Dirac delta impulse does not satisfy the integral bound in (3.2).

3.3.2 General learning control

In the previous sections on learning control we considered the situation in which the robot is only required to learn a single task. The learning betterment approach discussed in section 3.1 and the repetitive control approaches discussed in sections 3.2 and 3.3.1 were developed specifically for this situation. As a consequence, the feedforward control action, $\hat{w}_r(t) = \hat{w}(u(t))$, can be estimated as the function of time. However, the feedforward action must be relearned every time that the task is changed. In this section we consider the situation in which the robot must learn how to execute several tasks. One approach to the general learning control problem is to estimate the inverse dynamics function as an

explicit function of the task vector, i.e., $w(\cdot) : \mathcal{R}^{3n} \rightarrow \mathcal{R}^n$. Notice that in this approach, the feedforward action, $\hat{w}_r(t) = \hat{w}(u(t))$, is no longer directly updated as an explicit function of time. In principle, if the learning process is successful in estimating the robot dynamics so that the inverse dynamics function error $\tilde{w}(\cdot)$ in Eq. (2.11) is zero, the robot learning system will have acquired the necessary dexterity for the robot to execute any arbitrary task trajectory, $u(t)$. Unfortunately, it may be difficult to devise a sufficiently complex training task for the robot to practice during the learning process, which results in the complete identification of the robot dynamics.

There are many learning and adaptive robot control works published in the robotics literature which are based on the estimation of the inverse dynamics function, $w(\cdot)$. The estimation techniques in these works vary according to the degree of structure that is assumed known about the inverse dynamics function $w(\cdot)$, and on the type of algorithm used for updating $\hat{w}(\cdot)$. These algorithms include parametric adaptation algorithms (c.f. (Slotine and Li, 1987; Slotine and Li, 1991; Sadegh and Horowitz, 1990; Ortega and Spong, 1989; Wen and Bayard, 1988) and their bibliographies) and function estimation algorithms which use basis expansion functions, content addressable memories, neural networks and self organizing maps (c.f. (Miller, 1987; ?; Messner et al., 1991a; Atkeson, 1990; Barto et al., 1983; Ritter et al., 1992; ?; ?; Shoureshi, 1993) and their bibliographies). In this section we will describe a learning control system for robot manipulators based on integral transforms (Messner et al., 1991a). We assume that there exist L tasks which the robot must learn to execute, each of which is described by a corresponding task vector trajectory, u_{T_i} , where the subscript i denotes the i th task and each trajectory will be assumed to be periodic for simplicity.

$$u_{T_i} := u_i(t) = u_i(t + T_i) \quad t \geq t_{o_i}, \quad (3.5)$$

with t_{o_i} being the time at which the i th task starts and T_i the time that it takes the robot to complete the task. We also assume for simplicity that all tasks begin and end at the same configuration, i.e., $u_1(t_{o_1}) = u_2(t_{o_2}) = \dots = u_L(t_{o_L})$. Let us define the task vector space A by the smallest closed, bounded and simply connected subset of \mathcal{R}^{3n} such that all task vectors lie inside this space, i.e.,

$$u_i(t) \in A, \quad i = 1, 2, \dots, L \quad t \geq 0. \quad (3.6)$$

Thus, the domain of the inverse dynamics function which must be learned by the robot in order to execute all L tasks can be restricted to A , i.e., $w(\cdot) : A \rightarrow \mathcal{R}^n$. We now assume that the inverse dynamics function $w(\cdot)$ can be represented as a linear integral equation of the first kind, i.e. there exists a *known* scalar, symmetric, non-degenerate kernel $K(\cdot, \cdot) : A \times \Gamma \rightarrow R$ and an *unknown* influence function $c(\cdot) : \Gamma \rightarrow R^{3n}$ such that

$$w(u) = \int_{\Gamma} K(u, \gamma) c(\gamma) d\gamma. \quad (3.7)$$

for all $u \in A$, where the domain of integration, Γ , is compact and $A \subset \Gamma$.

There may be many kernels which can be constructed such that (3.7) is satisfied. We assume that at least one such kernel is known. Moreover, we will impose, in order to guarantee the stability of the learning algorithm, that the kernel be squared integrally bounded

$$\int_{\Gamma} |K(u, \gamma)|^2 d\gamma \leq \kappa < \infty \quad ;, \quad (3.8)$$

and have bounded partial derivatives. This condition is not too severe since the inverse dynamics function $w(\cdot)$ is infinitely smooth. An example of such a kernel function is a multivariable truncated Gaussian Kernel function. Such a kernel was used in the experimental results which will be shown at the end of this section. The learning process proceeds by having the robot attempt to execute a training task defined by a periodic trajectory $u_{TR} := u_R(t) \in A$, under the DCCL control law (2.10), using the following functional adaptation algorithm for updating the inverse dynamics function estimate, $\hat{w}(\cdot)$.

$$\hat{w}(\cdot) = \int_{\Gamma} K(\cdot, \gamma) \hat{c}(t, \gamma) d\gamma, \quad (3.9)$$

$$\frac{\partial}{\partial t} c(t, \gamma) = K(u_R(t), \gamma) K_L e_v(t) \quad (3.10)$$

where, e_v is the reference velocity error and K_L is a symmetric, positive definite matrix. This adaptation algorithm is similar to the repetitive algorithm discussed in section 3.3. The major difference resides in the fact that the adaptation algorithm in (3.3) and (3.4) is used to identify a function of one variable, while the algorithm in (3.9) and (3.10) is used to estimate a multivariable function. As shown in (Messner et al., 1991a), the system formed by the error dynamics (2.6) and the function update law (3.9) and (3.10), driven by the training trajectory $u_R(t)$ is stable in the sense of Lyapunov, and the error signal, $e(t)$ converges to zero asymptotically. Moreover, under the additional assumption that the training trajectory u_{TR} is uniformly continuous, $\lim_{t \rightarrow \infty} \tilde{w}(u_R(t)) = 0$.

It is important to determine under what conditions does the robot acquire a sufficient level of dexterity to execute all other L tasks, when it learns to execute the training task. Let us assume that the required robot dexterity for all tasks is specified by a positive constant ϵ_p and a function norm $\|\cdot\|_p$ so that

$$\|e_{Ti}\|_p \leq \epsilon_p, \quad \text{for } i = 1, \dots, L, \quad (3.11)$$

where e_T was defined in (2.5) and (2.4), and the subscript i denotes the robot's error state for the i th task. We also assume that the initial error state for each task, $e_i(t_{oi})$, is small enough so that it is possible to attain (3.11). We need to find the conditions on the training trajectory u_{TR} , and the kernel function $K(\cdot, \cdot)$ so that there exist a $\delta_p \geq 0$ such that, if $\|\tilde{w}(u_{TR})\|_p \leq \delta_p$, (3.11) is satisfied. As shown in (Moore et al., 1992; Horowitz et al., 1991), these conditions are that the kernel function $K(\cdot, \cdot)$ in (3.7) must have a *finite eigenfunction expansion* and the training task u_{TR} must be *persistently exciting*. The finite dimensionality condition on the kernel, $K(\cdot, \cdot)$ can be expressed as follows.

$$K(u, \gamma) = \sum_{n=1}^N \lambda_n \psi_n(u) \psi_n(\gamma) = \Psi^T(u) \Lambda \Psi(\gamma), \quad N < \infty \quad (3.12)$$

where $\Psi(\cdot) = [\psi_1(\cdot), \dots, \psi_N(\cdot)]^T$, and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_N)$, $\lambda_i \neq 0$ and the eigenfunctions $\psi_n(\cdot)$'s in (3.12) are orthonormal over Γ . This condition is not very restrictive in practice since the inverse dynamics function $w(\cdot)$ can be decomposed into two components $w(\cdot) = w_F(\cdot) + w_D(\cdot)$, where $w_F(\cdot)$ has a finite eigenfunction expansion and satisfies (3.7), while the term $w_D(\cdot)$, contains high frequency components and can be consider as a disturbance input to the control system. Notice that it is not necessary to explicitly determine the orthonormal eigenfunction set $\Psi(\cdot)$ in (3.12), only a kernel with a finite eigenfunction expansion must be generated. In actual implementations, where the functions, $\hat{w}(\cdot)$, $K(\cdot, \cdot)$ and $\hat{c}(\cdot, \cdot)$ are discretized into finite elements, this condition is always satisfied. Thus, it is only necessary to determine a sufficiently high degree of discretization so that the term $w_D(\cdot)$ is small enough for condition (3.11) to be satisfied. The condition that the training task trajectory u_{TR} be persistently exciting can be expressed as follows. The trajectory $u_R(t) \in A$ is persistently exciting when, for some scalars $\alpha_1, \alpha_2, T > 0$ and *for all influence functions* $c(\gamma) : A \rightarrow R^n$

$$\alpha_2 \int_A |c(\gamma)|^2 d\gamma \geq \int_t^{t+T} |w(u_R(\tau))|^2 d\tau \geq \alpha_1 \int_A |c(\gamma)|^2 d\gamma \quad (3.13)$$

for all t . (Moore et al., 1992) presents a technique for testing if a given training trajectory will be persistently exciting. One way of satisfying this condition is to make the training trajectory, u_{TR} a concatenation of all the trajectories which must be learned by the robot. As shown in (Horowitz et al., 1991), under persistence of excitation, the unperturbed robot learning system (i.e., $w_D(\cdot) = 0$) form by the robot error dynamics (2.12) and the learning laws (3.9) and (3.10) is globally exponentially stable. Thus, the dexterity condition (3.11) can be satisfied by discretizing the functions $\hat{w}(\cdot)$, $K(\cdot, \cdot)$ and $\hat{c}(\cdot, \cdot)$ with sufficiently small spatial finite elements.

The real time implementation of the learning algorithm in equations (3.9) and (3.10) requires that both functions $\hat{w}(\cdot)$ and $\hat{c}(\cdot)$ be updated in parallel. These algorithms are ideally implemented with computational devices with massive parallel processing capabilities, such as neural networks. Even if neural networks are used, it is reasonable to expect that some computational delays will exist in updating the influence functions $\hat{c}(\cdot)$. To account for this fact the following *delayed learning algorithm* (Messner et al., 1991a) can be implemented instead of the learning rule in (3.9) and (3.10).

$$\hat{w}(u) = \int_A K(u, \gamma) \hat{c}_k(\gamma) d\gamma, \quad k\bar{\Delta}t \leq t < (k+1)\bar{\Delta}t \quad (3.14)$$

and

$$\hat{c}_k(\gamma) = \hat{c}_{k-1}(\gamma) + \int_{(k-1)\bar{\Delta}t}^{k\bar{\Delta}t} K(u, \gamma) K_L e_v(\sigma) d\sigma, \quad (3.15)$$

where K_L is a positive definite matrix.

In the delayed learning rule the influence function $\hat{c}(\cdot, \cdot)$ is not updated continuously. Rather, it is updated at discrete time intervals (i.e. at $t = k\bar{\Delta}t, \forall k = 0, 1, 2, \dots$) and it remains unchanged for $k\bar{\Delta}t \leq t < (k+1)\bar{\Delta}t$. Moreover, there is a one step delay in the

computation of $\hat{c}_k(\gamma)$. Since the functions $\hat{w}(\cdot)$ and $\hat{c}(\cdot)$ are no longer updated in parallel, the algorithm can be implemented using existing digital computers.

The learning and repetitive control algorithms discussed in this section have been extensively tested by both simulation and experiments on the Berkeley/NSK SCARA two axis manipulator (Sadegh et al., 1990; Messner et al., 1991a; Kao et al., 1989; Horowitz et al., 1990). These results confirm the stability and convergence properties of the repetitive and learning algorithms discussed in this paper, and show that the learning algorithms based on integral transforms are robust to both unmodeled dynamics and measurement and input disturbances. The unmodelled dynamics in the above works include the actuator and velocity sensor parasitic dynamics. The measurement and input disturbances included random measurement and input noise, sensor quantization, and friction forces. For details the reader is referred to the above mentioned works and to (Kang et al., 1988), where a detailed description of the robot and control system is presented.

Here we present experimental results obtained in training the robot to execute a task using the general delayed learning algorithm given by (3.14) and (3.15). The effectiveness of the learning process is tested by subsequently commanding the robot to execute a task which is different from the training task, without any prior learning. The training task consists in commanding the robot to follow the spiral figure shown in Fig. 6. Fig. 7 shows how the robot executes the training task, during the first trial of the learning process and after the 20th learning trial. The initial condition for the inverse dynamics estimate is $\hat{w}(\cdot) = 0$. Once the learning process is completed the robot is commanded to follow the circle shown in Fig. 6, without any prior practice. Fig. 8 shows the execution of the circle when no feedforward action is used in the control action and when the inverse dynamics function $\hat{w}(\cdot)$, which was learned during the training process, is utilized in the control action. These results illustrate that a robot can learn how to execute several tasks, if the training task is persistently exciting. It should be noted that a fairly coarse spatial discretization was used for the functions $\hat{w}(\cdot)$, $K(\cdot, \cdot)$ and $\hat{c}(\cdot, \cdot)$ in the results presented above. For the Berkeley/NSK robot, $n = 2$ and the inverse dynamics function, $w(\cdot)$, is a map from $A \subset \mathcal{R}^6$ to \mathcal{R}^2 . The domain A was discretized into a hyper-rectangular grid of 7^6 elements.

4 Conclusion

In this paper I discussed the use of function identification and adaptive control algorithms in the formulation of learning controllers for robot manipulators. Several learning algorithms including betterment learning schemes, repetitive control schemes and the adaptive learning schemes based on integral transforms were analyzed and compared. There are other function identification algorithms were not addressed in this paper, due to space constraints, such as algorithms which are based on the representation of functions using radial basis functions and other expansions (cf. (?; ?; ?) and their bibliographies). These algorithms have similar properties to the learning algorithms based on integral transforms which were discussed in this paper. Another class of adaptive learning algorithms which have been proposed for robot control are those based on feedforward neural networks and the generalized delta learning

rule, and those based on the use of self-organizing neural maps (cf. (Shoureshi, 1993; Ritter et al., 1992) and their bibliographies). The convergence properties of these schemes are difficult to analyze, due to the inherent nonlinearities which are present in the generalized delta learning rule and the rules used to update self-organizing maps. An attractive feature of self-organizing neural maps is that they appear to be capable of identifying the domain of a function as well as the function itself, and can therefore store the learned information more efficiently. This area of research is currently under much investigation. A common shortcoming of the learning control schemes discussed in this paper is that the determination of the desired trajectory which the robot must track is not part of the learning process, and it must be specified a-priori. A more comprehensive learning control formulation should also involve the determination of the robot's desired trajectory, by specifying the task which the robot must learn using a cost function. There are a growing number of works in which learning algorithms and neural networks are used to solve optimal control and credit assignment problems (Werbos, 1989; Jordan, 1989; Barto et al., 1989; ?). These systems are in principle capable of determining an optimal robot control action. However, to my knowledge, there does not exist a formal and rigorous analysis of the stability and convergence properties of these systems and the formulation of a robot self optimizing learning controller remains an open problem.

5 Acknowledgements

This work was partially supported by the National Science Foundation under grant MSM-8511955. The author wishes to acknowledge the collaboration of Nader Sadegh, Bill Messner, John Moore, Masayoshi Tomizuka, Wei-Wen Kao and Mike Boals in many of the results presented in this paper. The experimental results presented in this paper were conducted by Perry Li.

References

- Arimoto, S. (1990). Robustness of learning control for robot manipulators. In *Proceedings of 1990 IEEE International Conference on Robotics and Automation*. Cincinnati, Ohio.
- Arimoto, S., Kawamura, S., and Miyazaki, F. (1984). Bettering operation of robots by learning. *Journal of Robotic Systems*, 1(2):123–140.
- Arimoto, S., Kawamura, S., and Miyazaki, F. (1988). Realization of robot motion based on a learning method. *IEEE TSMC*, 18(1):126–134.
- Atkeson, C. G. (1990). Memory based techniques for task-level learning in robots and smart machines. In *Proceedings of the 1990 ACC Conference*, volume 3, pages 2815–2819. San Diego.

- Atkeson, C. G. and McIntyre, J. (1986). Robot trajectory learning through practice. *Proceedings of 1986 IEEE International Conference on Robotics and Automation*, 3:1737–1742. San Francisco.
- Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:834–846.
- Barto, A. G., Sutton, R. S., and Watkins, C. J. C. H. (1989). Learning and sequential decision making. Technical report, COINS Technical Report 89-95.
- Bayard, D. S. and Wen, J. T. (1988). New class of control laws for robotic manipulators - part 1. non-adaptive case. *International Journal of Control*, 47(5):1361–1385.
- Casalino, G. and Bartolini, G. (1984). A learning procedure for the control of movements of robotic manipulators. In *IASTED Symposium on Robotics and Automation*, pages 108–111. Amsterdam.
- Chew, K. and Tomizuka, M. (1990a). Digital control of repetitive errors in disk-drive systems. *IEEE Control Systems Magazine*, 10(1):16–20.
- Chew, K. and Tomizuka, M. (1990b). Steady-state and stochastic performance of a modified discrete-time prototype repetitive controller. *ASME Journal of Dynamic Systems, Measurement and Control*, 112(1):35–41.
- Craig, J. J. (1984). Adaptive control of manipulators through repeated trials. In *Proceedings of the American Control Conference*, pages 1566–1573. San Diego.
- Craig, J. J. (1986). *Introduction to Robotics: Mechanics and Control*. Addison-Wesley.
- Hara, S., Yamamoto, Y., Omata, T., and Nakano, M. (1988). Repetitive control system: A new type servo system for periodic exogenous signals. *IEEE Transactions on Automatic Control*, 33(7):659–668.
- Heinzinger, D., Fenwick, B., Paden, B., and Miyazaki, F. (1989). Robot learning control. In *Proceedings of the IEEE CDC*. Tampa, Florida.
- Horowitz, R., Messner, W., and Moore, J. (1991). Exponential convergence of a learning controller for robot manipulators. *IEEE Transaction on Automatic Control*, 36(7):890–894.
- Horowitz, R., Messner, W., Moore, J., and Kao, W.-W. (1990). Convergence properties of learning controllers for robot manipulators. In *Proceedings of the 1990 Japan-USA Symposium on Flexible Automation*.

- Jordan, M., I. (1989). Generic constraints on unspecified target trajectories. In *IJCNN International Joint Conference on Neural Networks*, volume 1, pages I217–I226. San Diego.
- Kang, C. G., Kao, W. W., Boals, M., and Horowitz, R. (1988). Modeling and identification of a two link scara manipulator. In Youcef-Toumi, K. and Kazerroni, H., editors, *Symposium on Robotics, ASME Winter Annual Meeting*, pages 393–407, Chicago, IL.
- Kao, W. W., Horowitz, R., Tomizuka, M., and Boals, M. (1989). Repetitive control of a two degree of freedom scara manipulators. In *Proceedings of the American Control Conference*, pages 1457–1462.
- Kawamura, S., Miyazaki, F., and Arimoto, S. (1988). Realization of robot motion based on a learning method. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-18(1):126–134.
- Messner, W., Horowitz, R., Kao, W. W., and Boals, M. (1991a). A new adaptive learning rule. *IEEE Transaction on Automatic Control*, 36(2):188–197.
- Messner, W., Kempf, C., Tomizuka, M., and Horowitz, R. (1991b). A comparison of four discrete-time repetitive control algorithms. In *Proceedings of the 1991 American Control Conference*. To appear in the *IEEE Control Magazine*.
- Miller, T. W. (1987). Sensor-based control of robotic manipulators using a general learning algorithm. *IEEE Journal of Robotics and Automation*, RA-3(2):157–165.
- Moore, J., Horowitz, R., and Messner, W. (1992). Functional persistence of excitation and observability for learning control systems. *ASME Journal of Dynamic System Measurements and Control*, 114(3):500–507.
- Omata, T., Hara, S., and Nakano, M. (1987). Nonlinear repetitive control with application to trajectory control of manipulators. *Journal of Robotic Systems*, 4(5):631–652.
- Ortega, R. and Spong, M. W. (1989). Adaptive motion control of rigid robots: a tutorial. *Automatica*, 25(6):877–888.
- Ritter, H., Martinez, T., and Schulten, K. (1992). *Neural Computation and Self-Organizing Maps*. Addison Wesley.
- Sadegh, N. (1991). Synthesis and stability analysis of repetitive control systems. In *Proceedings of the American Control Conference*, volume 3, pages 2634–2639.
- Sadegh, N. and Horowitz, R. (1987). Stability analysis of an adaptive controller for robotic manipulators. In *Proceedings of 1987 IEEE Conference on Robotics and Automation*, volume 3, pages 1223–1229.

- Sadegh, N. and Horowitz, R. (1990). Stability and robustness analysis of a class of adaptive controllers for robotic manipulators. *International Journal of Robotics Research*, 9(3):74–92.
- Sadegh, N., Horowitz, R., Kao, W. W., and Tomizuka, M. (1990). A unified approach to design of adaptive and repetitive controllers for robotic manipulators. *ASME Journal of Dynamic System Measurements and Control*, 112(4):618–629.
- Sastry, S. S. and Bodson, M. (1989). *Adaptive Control: Stability, Convergence, and Robustness*. Prentice Hall.
- Shoureshi, R. (1993). Intelligent control systems: Are they for real? *ASME Journal of Dynamic Systems Measurement and Control*, 50th Anniversary Issue.
- Slotine, J. and Li, W. (1986). On the adaptive control of robot manipulators. In Paul, F. and Youcef-Toumi, K., editors, *Robots: Theory and Applications*, *ASME Winter Annual Meeting*.
- Slotine, J. and Li, W. (1987). On the adaptive control of robot manipulators. *International Journal of Robotics Research*, 3(6).
- Slotine, J. and Li, W. (1991). *Applied Nonlinear Control*. Prentice Hall.
- Spong, M. W. and Vidyasagar, M. (1989). *Robot Dynamics and Control*. John Willey.
- Tomizuka, M. (1992). On the design of digital tracking controllers. *ASME Journal of Dynamic Systems, Measurement and Control*, 50th Anniversary Issue.
- Tomizuka, M., Tsao, T. C., and Chew, K. K. (1989). Discrete-time domain analysis and synthesis of repetitive controllers. *ASME Journal of Dynamic Systems Measurement and Control*, 111:353–358.
- Tsai, M. C., Anwar, G., and Tomizuka, M. (1988). Discrete time repetitive control for robot manipulators. In *Proceedings of 1988 IEEE International Conf. on Robotics and Automation*, volume 3, pages 1341–1347. Philadelphia.
- Uchiyama, M. (1978). Formulation of high speed motion pattern of a mechanical arm by trial. *Transactions of the Society of Instrumentation and Control Engineers*, 14:706–712. in Japanese.
- Wen, J. T. and Bayard, D. S. (1988). New class of control laws for robotic manipulators -part 2. adaptive case. *International Journal of Control*, 47(5):1387–1406.
- Werbos, P. J. (1989). Backpropagation and neural control: A review and prospectus. In *IJCNN International Joint Conference on Neural Networks*, volume 1, pages I208–I216. San Diego.

Appendix - Norms and Normed Spaces

For a vector $w \in \mathcal{R}^n$ we define the norm of $|w|$ to be the Euclidean norm. For a Lebesgue measurable function $f(\cdot) : \mathcal{R}_+ \rightarrow \mathcal{R}^n$ the L_n^p norm for $p \in [1, \infty)$ is defined to be $\|f\|_p := \{\int_0^\infty |f(\tau)|^p d\tau\}^{1/p}$. For $p = \infty$ the norm is defined to be $\|f\|_\infty := \sup_{t \geq 0} |f(t)|$ almost everywhere. The normed space L_n^p is defined by $L_n^p := \{f(\cdot) : \mathcal{R}_+ \rightarrow \mathcal{R}^n \mid \|f\|_p < \infty\}$.

6 Figures

List of Figures

1	2 Degree of Freedom Robot Arm	25
2	Arimoto's learning law	26
3	γ coefficients and $\Gamma(t, \tau)$ kernel	27
4	Truncated Gaussian kernel and integral representation	28
5	Influence function estimate update	29
6	Robot Tasks	30
7	Execution of the training task at the beginning of the learning process and after 20 cycles	31
8	Execution of a Circle with and without a Feedforward Action	32

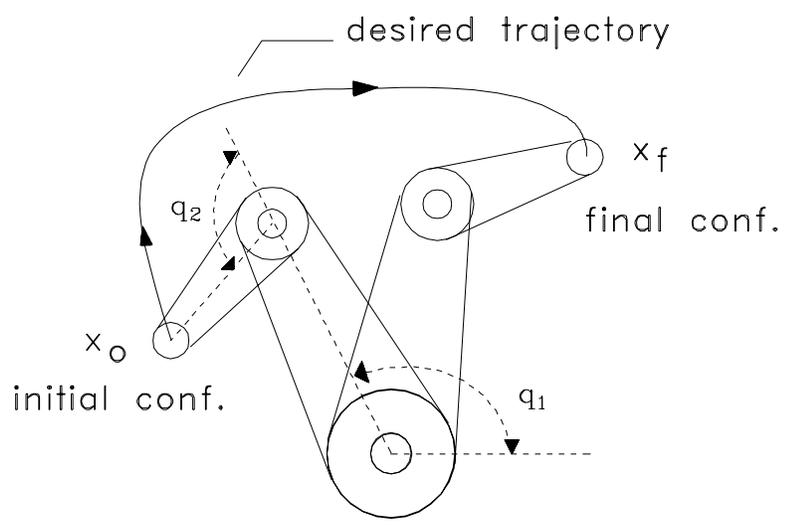


Figure 1: 2 Degree of Freedom Robot Arm

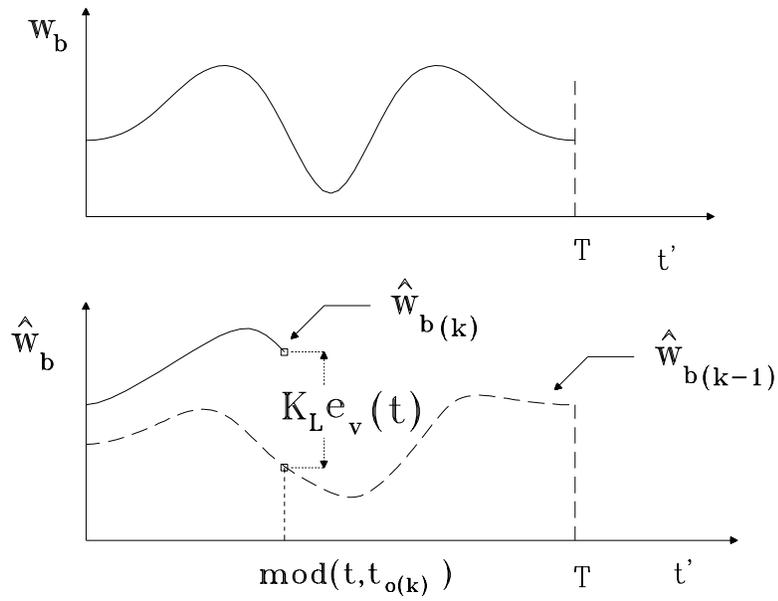


Figure 2: Arimoto's learning law

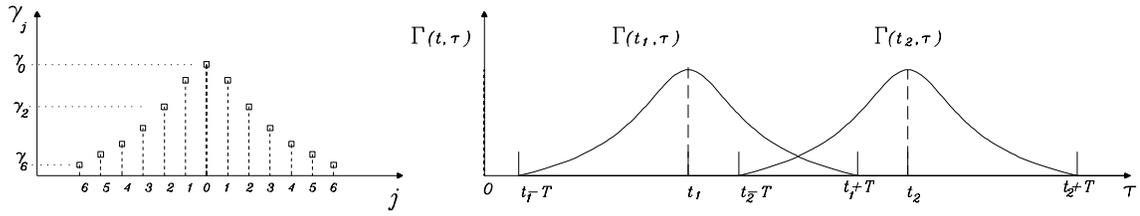


Figure 3: γ coefficients and $\Gamma(t, \tau)$ kernel

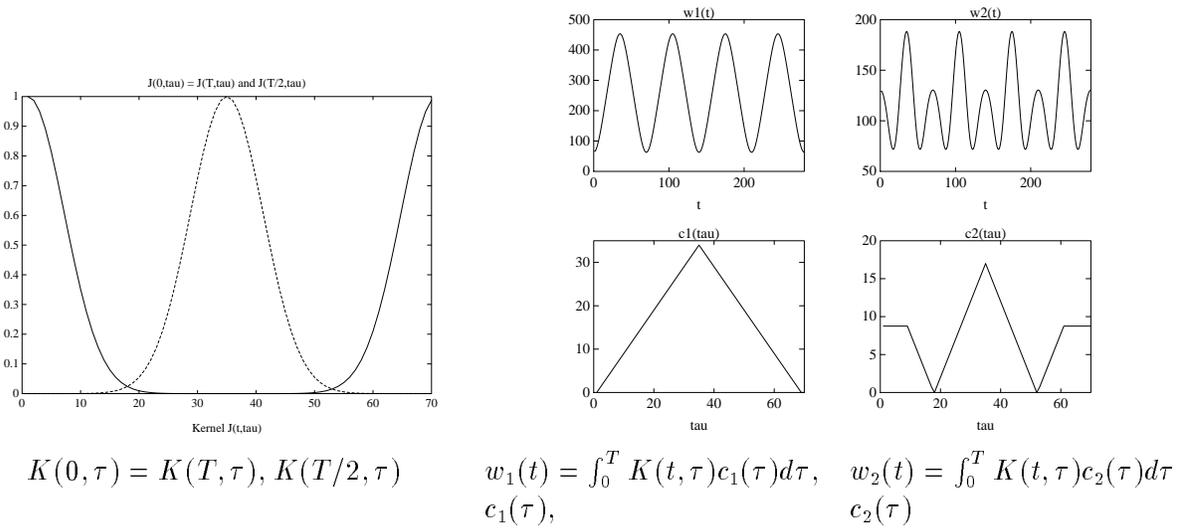


Figure 4: Truncated Gaussian kernel and integral representation

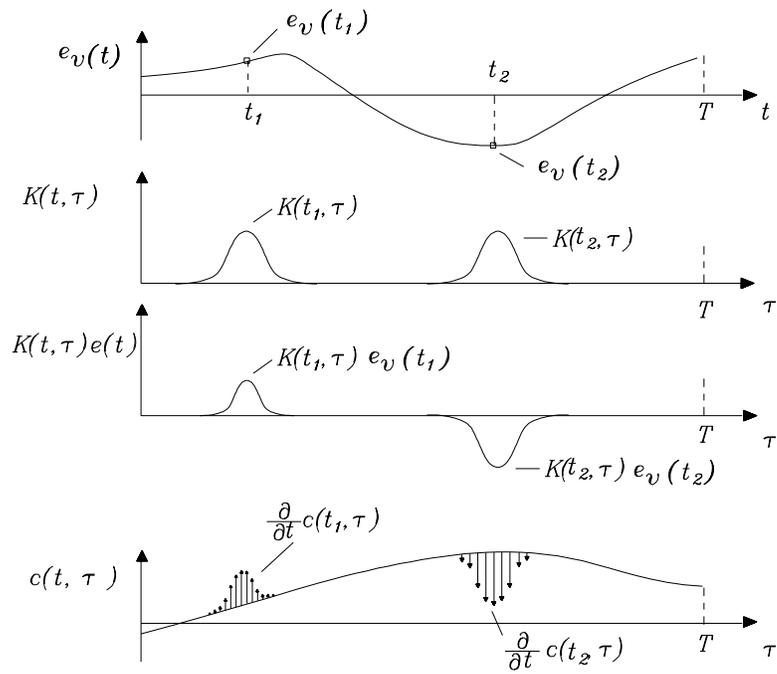


Figure 5: Influence function estimate update

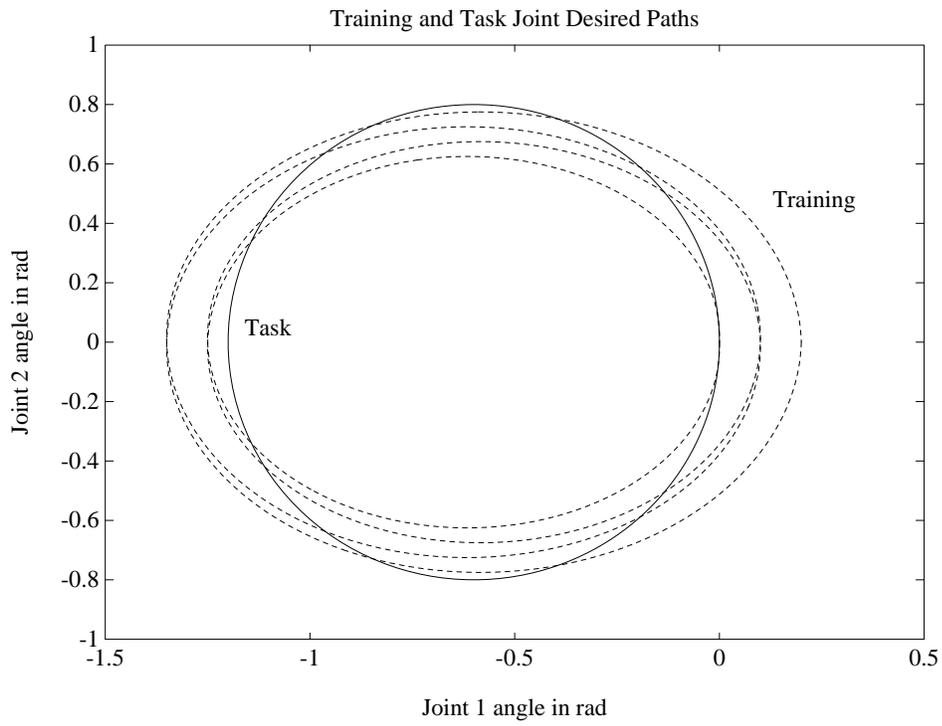


Figure 6: Robot Tasks

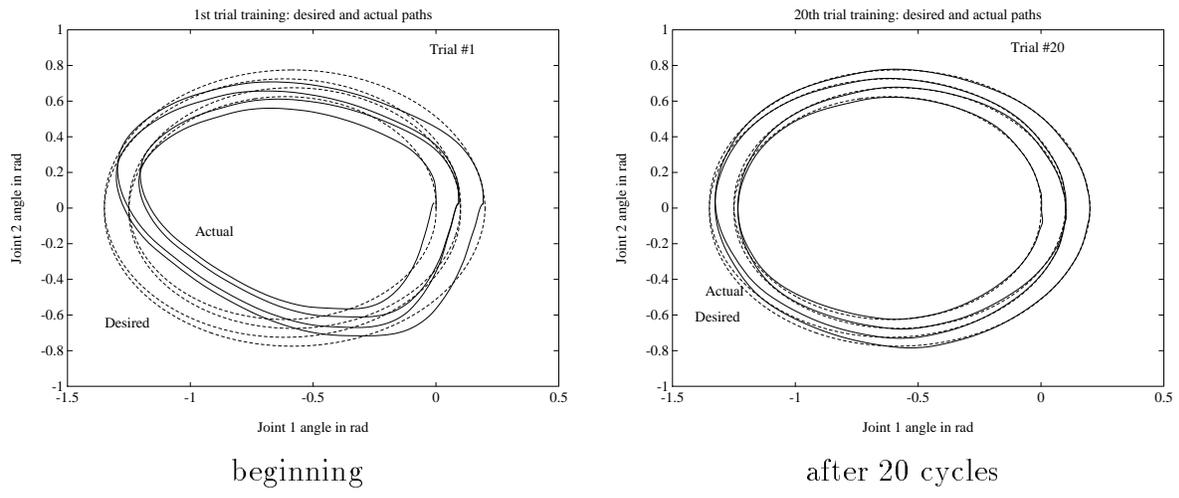
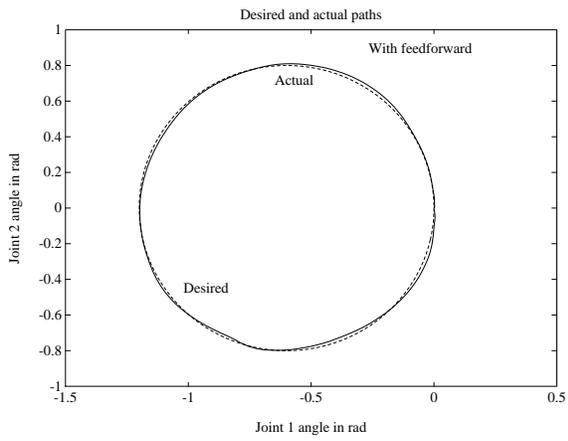
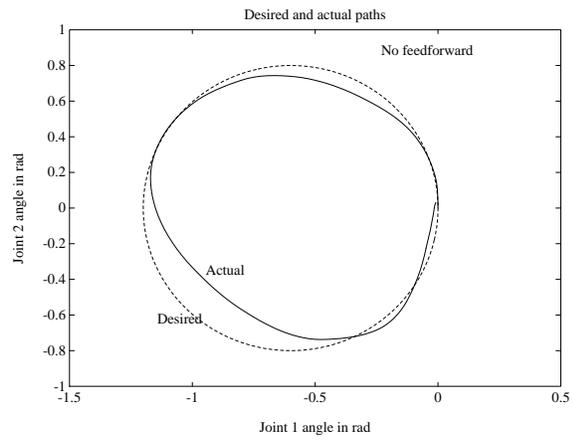


Figure 7: Execution of the training task at the beginning of the learning process and after 20 cycles



with feedforward



without feedforward

Figure 8: Execution of a Circle with and without a Feedforward Action