10

Learning Control Systems

Jay Farrell and Walter Baker
The Charles Stark Draper Laboratory, Inc.
555 Technology Square
Cambridge, MA 02139

Abstract

An important capability of an intelligent system is the ability to improve its future performance based on experience within its environment. The concept of learning *is usually used to describe the process by which this capability is achieved. The present chapter will focus on control systems that are explicitly designed to have and to exploit this capability. In this context, the control system can be viewed as a (generally nonlinear) mapping from plant outputs to actuation commands so as to achieve certain control objectives, with learning as the process of modifying this mapping to improve future closed-loop system performance.*

The information required for learning, that is, the information required to correctly generate the desired control system mapping, is obtained through direct interactions with the plant (and its environment). Thus, learning can be used to compensate for a lack of a priori *design information by exploiting empirical information that is gained experientially. In this setting, the principal benefit of learning control is its ability to accommodate poorly modeled, nonlinear dynamical behavior.*

Contemporary learning control methodologies based on this perspective will be described and compared to traditional control approaches such as gain scheduled robust and adaptive control, as well as to earlier learning control paradigms. The discussion and examples that follow will identify both the distinguishing characteristics of learning control systems and the benefits of augmenting traditional control approaches with learning capabilities.

1. Introduction

Control systems for autonomous vehicles must maintain closed-loop system integrity and performance over a wide range of operating conditions. This objective can be difficult to achieve due to a number of circumstances including the complexity of both the plant and the performance objectives, and due to the presence of uncertainty. From a design standpoint, these difficulties may result from nonlinear or time-varying behavior, poorly modeled plant dynamics, high dimensionality, multiple inputs and outputs, complex objective functions, constraints, and the possibility of actuator or sensor failures. Each of these effects, if present, must be addressed if the system is to operate properly in an autonomous fashion for extended periods of time. Although learning control systems may be used to address several of these design difficulties, the main focus of this chapter will be the *accommodation of poorly modeled nonlinear dynamical behavior.* The reasons for this special emphasis are straightforward, and are outlined in the next section.

Most, if not all, researchers in the intelligent systems field would accept the general statement that the ability to learn is a key attribute of an intelligent system. Even so, very few would be able to agree on any particular statement that attempted to be more precise. The stumbling blocks, of course, are the words "learn" and (ironically) "intelligent." Similarly, it is difficult to provide a precise and completely satisfactory definition for the term "learning control system." One interpretation that is, however, consistent with the prevailing literature is that:

> A *learning control system* is one that has the ability to improve its performance in the future, based on experiential information it has gained in the past, through closed-loop interactions with the plant and its environment.[1]

There are several immediate implications of this statement. One implication is that a learning control system has some *autonomous* capability, since it has the ability to improve its own performance. Another is that it is *dynamic*, since it can vary. Yet another implication is that it incorporates *memory*, since the retention of past information is critical to the improvement of future performance.[2] Finally, to improve its performance, the learning system must receive *performance feedback* information based on the objective function that it seeks to optimize.

2. Learning Control Issues

At this point, a number of questions come to mind. What is the role of learning in the context of intelligent control? How does it work? Are there alternatives to learning? How does learning differ from adaptation? How can a learning control system

---

1. To help focus the discussion that follows, we will limit our discussion to learning for the control of dynamical system behavior. We will not discuss control of reasoning processes or learning in artificial intelligence knowledge basis; although both areas of research are important for the development of autonomous vehicle behavior.

2. The use of memory is a key distinction between learning and adaptation (Landau (1979)) as will be discussed in the sequel.

be realized or implemented? We will consider each of these questions briefly in the remainder of this section; later, in the sections that follow, we will develop more detailed and complete answers to these same questions.

Role of Learning. Following Tsypkin [20], the necessity for applying learning arises in situations where a system must operate in conditions of uncertainty, and when the available *a priori* information is so limited that it is impossible or impractical to design in advance a system that has fixed properties and also performs sufficiently well. In the context of intelligent control, learning can be viewed as a means for solving those problems that lack sufficient *a priori* information to allow a complete and fixed[3] control system design to be derived in advance. Thus, a central role of learning in intelligent control is to enable a wider class of problems to be solved, by reducing the prior uncertainty to the point where satisfactory solutions can be obtained on-line. This result is achieved empirically, by means of reinforcement, association, and memory adjustment. A goal of this chapter is to explain the role that each of these items plays in the formulation of a learning control approach.

Learning Control Basics. The benefits of learning control (given the present state of its development) can be realized through the automatic synthesis of the mappings that are used within a control system architecture. Examples of such mappings include a "control mapping" that relates command and measured plant outputs to a desirable control action, or a "model mapping" that relates the plant operating condition to an accurate set of model or controller parameters (in the latter example, the learning architecture is similar to that of gain scheduling, with the proviso that learning occurs on-line with the actual plant, while gain scheduling is developed off-line via a model). Learning is required when these mappings cannot be determined completely in advance because of *a priori* uncertainty (e.g., poorly modeled nonlinear dynamical behavior). In a typical learning control application, the desired mapping is stationary, and is expressed (implicitly) in terms of an objective function involving the outputs of both the plant and the learning system. The objective function is used to provide *performance feedback* to the learning system, which must then *associate* this feedback with specific *adjustable* elements of the mapping that is currently stored in its *memory*. The underlying idea is that *performance feedback* can be used to improve the mapping furnished by the learning system.

Relation to Alternative Approaches. There are, of course, several alternative approaches that have traditionally been used for the accommodation of poorly modeled nonlinear dynamical behavior. These include gain scheduled robust, adaptive, and "manual learning" techniques. The relationships between these approaches and the learning control approach are important and are discussed in the following paragraphs.

---

3. The term "fixed" is used here to indicate those control systems for which the parameters and structure are determined in an open-loop (performance independent) fashion. Examples include fixed gain and gain scheduled controllers. With this terminology, non-fixed control systems employ performance feedback to adjust the parameters or structure of the control law. Note that this definition requires a somewhat arbitrary distinction between the adjustable *parameters* and *states* of the controller.

Robust control system design techniques attempt to treat the problem of model uncertainty as best as possible in advance, so as to produce a fixed design with guaranteed stability and performance properties for any specific scenario contained within a given uncertainty set. A tradeoff exists between performance and robustness, since robust control designs are usually achieved at the expense of resulting closed-loop system performance (relative to a control design based on a perfect model). Advanced robust control system design methods have been developed to minimize this inherent performance / robustness tradeoff. Although robust design methods are currently limited to linear problems, nonlinear problems with model uncertainty can sometimes be approached by gain scheduling a representative set of robust point designs over the full operating envelope of the plant, thus decreasing the amount of model uncertainty that each linear point design must accommodate. Nevertheless, the performance resulting from any fixed control design is always limited by the availability and accuracy of the *a priori* design information. If there is sufficient complexity or uncertainty so that a fixed control design will not suffice, then satisfactory closed-loop system performance can only be obtained in one of three ways: (i) through improved modeling to reduce the uncertainty, (ii) via an automatic on-line adjustment technique, or (iii) through manual tuning of the nominal control law design.

In contrast to the above "fixed" approach, adaptive control approaches attempt to treat the problem of uncertainty through on-line means. An adaptive control system can adjust itself to accommodate new situations, such as changes in the observed dynamical behavior of a plant. In essence, adaptive techniques monitor the input-output behavior of the plant to identify, either explicitly or implicitly, the parameters of an assumed dynamical model. The control system parameters are then adjusted to achieve some desired performance objective. Thus, adaptive techniques seek to achieve increased performance by updating or refining some representation, which is determined (in whole or in part) by a model of the plant's structure, based on on-line measurement information. An adaptive control system will attempt to adapt if the behavior of the plant changes by a significant degree. However, if the dynamical characteristics of the plant vary considerably over its operating envelope (e.g., due to nonlinear dynamics), then the control system may be required to adapt continually. During these adaptation periods, closed-loop performance cannot be guaranteed. Note that this adaptation can occur even in the absence of time-varying dynamics and disturbances, since the control system must readapt *every* time a different dynamical regime is encountered (i.e., one for which the current control law is inadequate), even if it is returning to an operating condition it has encountered and successfully adapted to before.

The motivation for the gain scheduled approach is the expectation that plant dynamics can be represented by a known, fixed function of a set of scheduling variables. As the scheduling variables are normally a subset of the variables describing the operational envelope of the plant, these will be referred to as *spatially* dependent dynamics. In contrast, the focus of adaptive techniques is on *temporal* changes in the plant dynamics. As previously indicated, learning (in the context of control) can be construed as the ability to automatically develop and retain a desired control law for a given dy-

namical plant, based on closed-loop interactions with the plant (and its environment). The ability to *develop* the requisite control law on-line clearly differentiates learning control approaches from fixed robust design approaches (including those which are gain scheduled). The ability to *retain* the control law as a function of operating condition differentiates learning control strategies from adaptive control ones. These distinctions will be expanded on in the following subsection.

A third approach that has been used in applications with complex nonlinear dynamical behavior is based on a kind of manually executed learning control system. In fact, this is the predominant design practice used to develop flight control systems for aircraft. In this approach, the control law is developed through an iterative process that integrates multiple control law designs to approximate the required nonlinear control law. This approach often results in numerous design iterations, each involving manual redesign of the nominal control law (for certain operating conditions), followed by extensive computer simulation to evaluate the modified control law. After the initial control system has been designed, extensive empirical evaluation and tuning (manual adjustment and redesign) of the nominal control system is often required. This arises because the models used during the design process do not always accurately reflect the actual plant dynamics. In this process, the role of the learning system (as described above) is played by a combination of control system design engineers, modeling specialists, and system analysts. An interesting perspective to consider is that learning control offers a means of automating this manual design and tuning process for certain applications.

Adaptive vs. Learning Capabilities. In the discussion that follows we will see that both adaptive and learning control systems can be implemented using parameter adjustment algorithms, and that both make use of performance feedback information gained through closed-loop interactions with the plant and its environment. Nevertheless, we intend to clearly differentiate the goals of adaptation from those of learning. The key differences are essentially a matter of degree, emphasis, and intended purpose. A control system that treats every distinct operating situation as novel is limited to *adaptive* operation, whereas a system that correlates past experiences with past situations, and that can recall and exploit those experiences, is capable of *learning*. Since, in the process of learning, the learning system must be capable of adjusting its memory to accommodate new experiences, a learning system must, in some sense, incorporate an adaptive capability. It should be noted, however, that the design and intended purpose of a learning system requires capabilities beyond that of adaptation.

Adaptive control has a temporal emphasis: its objective is to maintain some desired closed-loop behavior in the face of disturbances and dynamics *that appear to be time-varying*. In actuality, the changing dynamics may be caused by nonlinearities, when the operating point of the system is changing with time (i.e., temporal changes in the linearized plant dynamics). Because the functional form of adaptive control laws is generally incapable of representing, over a wide range of operating conditions, the required control action as a function of the current plant state, it can be said that adap-

tive controllers lack "memory" in the sense that they must readapt to compensate for all changing dynamics, *even those which are due to (time-invariant) nonlinearity and have been experienced previously*. This inefficiency results in degraded performance, since transient behavior due to parameter adjustment will occur every time the (recently) observed dynamical behavior of the plant changes by a sufficient degree.

In general, adaptive controllers operate by optimizing a small set of adjustable parameters to account for plant behavior that is local in both state-space and time. To be effective, adaptive controllers must have relatively fast dynamics so that they can react quickly to changing plant behavior. In some instances, the plant parameters may vary so fast (perhaps due to nonlinearity) that the adaptive system cannot maintain desired performance through adaptive action alone. As argued by Fu [8], it is in this type of situation that a learning system is preferable. Because the learning system retains information, it can, in principle, react more rapidly to spatial variations once it has learned.

Learning controllers exploit an automatic mechanism that *associates*, throughout some operating envelope, a suitable control action or set of control system parameters with the current operating condition. In this way, the presence and effect of previously unknown nonlinearities can be accounted for and anticipated, based on past experience. Once such a control system has "learned," transient behavior that would otherwise be induced by spatial variations in the dynamics no longer occurs, resulting in greater efficiency and improved performance over adaptive control strategies.

Learning systems can operate by optimizing a large set of adjustable parameters (and potentially variable structural elements) to construct a mapping that captures the spatial dependencies of the problem throughout the operating envelope. To successfully execute this optimization process, learning systems make extensive use of past information and employ relatively slow learning dynamics.

As defined, the processes of adaptation and learning are complementary: each has unique desirable characteristics from the point of view of intelligent control. For example, adaptive capabilities are capable of accommodating slowly time-varying dynamics and novel situations (e.g., those which have never been experienced before), but are often inefficient for problems involving significant nonlinear dynamics. Learning approaches, in contrast, have the opposite characteristic: they are well-equipped to accommodate poorly modeled nonlinear dynamical behavior, but are not well-suited to applications involving time-varying dynamics.

Implementation of Learning Systems. From the previous discussion, it is clear that a learning system must be capable of accumulating and manipulating experiential information, storing and retrieving compiled information, and adapting its stored knowledge to accommodate new experiences. A key implementation point is that a learning system will require an efficient information storage and retrieval (i.e., memory) system to retain empirically derived knowledge. One simple way to implement a learning system is to partition the input-space into a number of disjoint regions, so

that the current output is determined by "looking up" the output value associated with the current input region. Many early learning control systems were based on this type of architecture. Assuming that the learning system output is used directly for control action, a nonlinear control law can be developed by learning the appropriate output value for each input region, resulting in a "staircase" approximation to the actual desired control law. The relation between learning and function approximation is readily apparent in this example. One drawback of this approach is the combinatorial growth in the number of regions required, as either the state-space dimension or the number of partitions per state-space dimension is increased.

More advanced learning systems (particularly for control applications) can be developed via an appropriate mathematical framework that is capable of representing a family of continuous functions; this framework can have a fixed or variable structure, and a potentially large number of free parameters. The structure and operational attributes of a learning system are determined, in part, by the quantity and quality of the *a priori* design information that is available, including the anticipated characteristics of the experiential information that is expected to be available on-line. Such architectures are often used in contemporary learning control systems. In this case, the learning process is designed to automatically adjust the parameters (or structure) of the functional form to achieve the desired input / output mapping. Such representations have important advantages over simple look-up table approaches; for instance, continuous functional forms are generally more efficient in terms of the number of free parameters, and hence, the amount of memory, needed to represent a smooth (or piecewise smooth) function. Furthermore, they are capable of providing generalization (i.e., interpolation between nearby input points) automatically. We will return to the subject of learning control system implementation in the Contemporary Implementations section of this chapter.

3. Past Implementations

In this section we briefly review learning control implementations that have been investigated over the last forty years. As this review is by no means exhaustive, the interested reader is referred to the survey papers of Sklansky [19] and Fu [9].

The typical early learning control systems subdivided the envelope over which the system was expected to operate into a set of regions called *control situations*. Each control situation corresponded to a different region for the features of the problem under consideration. *Features* could for example be elements of the state vector or parameters of the plant model. A *feature classifier* was used to characterize the current plant features as belonging to a particular *control situation*. Then, an admissible *control action* was associated with each control situation. The *association* (or mapping) from control situations to desired control actions was the task of the learning system. The accumulated knowledge of the learning system, embodied in the current control mapping, was maintained in the *memory* of the learning system. Based on this mapping, the current control situation was used to generate a control action, which was then applied to the plant. Subsequently, *performance feedback* was pro-

vided to the learning control system to adjust the control mapping, and hence, the future behavior of the system.

The majority of early learning control approaches focused on the development of algorithms capable of selecting the appropriate control action a_k from a *finite* set A of admissible control actions, for each element s_k of a *finite* set S of possible control situations. The elements of the finite action set often represented binary control signals suitable for bang-bang control (i.e., $A = \{a_1, a_2\}$), although more generally, they might represent any finite number of distinct control signals or even different control laws (e.g., different linear state feedback laws). The problem was to determine (learn) an optimal mapping $C^*: S \rightarrow A$ based on some predefined objective function.[4] Once learned, this control mapping could be used to generate the appropriate action a_i for the current situation s_k.

With these definitions in mind, the critical learning control research questions of the 1960s included: How should performance be evaluated and communicated to the learning system? How should an action be selected for each control situation? How might the action set be improved? How should the control situations be defined or learned? Each of these issues is examined more closely in the following subsections.

Performance Evaluation and Feedback. The driving force behind the learning process is a feedback signal that is based on the performance of the closed-loop system, as measured by some objective function.[5] Traditionally, the objective functions used in conjunction with learning control research have belonged to two classes: instantaneous and dynamic. An *instantaneous* objective function is one that can be completely defined in terms of a scalar function involving only the current characterization of the environment and the current inputs and outputs of the plant and the learning system. In this case, the ultimate goal is to maximize or minimize this function over the entire input domain of the control mapping by incrementally improving it with each learning experience. One example of this type is model-following control, in which the goal is to make the closed-loop system mimic the input-output behavior of a reference model. A common objective function in this case is a norm of the instantaneous error between the desired and the actual plant outputs. In contrast, a

4. Note that more than one type of optimization problem can be posed. For instance, the problem might be to determine an optimal mapping based on *fixed* control action and situation sets (this is the simplest case), or alternatively, to determine an optimal mapping based on *variable* control action and situation sets. In this latter case, an important part of the optimization problem is to modify the *interpretation* of the elements of these two sets, so as to achieve a better overall mapping. For control situations, this is achieved by modifying the feature classifier, whereas for control actions this is achieved by modifying the control signal associated with each action.

5. The question of how to specify desired system performance is an important and long-standing one in the field of control theory. There are a few basic specifications in terms of linear systems that are widely used because they lead to tractable design problems. The desire to allow a wider variety of performance specifications (e.g., nonlinear reference models) to be applied to a wider variety of plants (e.g., nonlinear systems with saturating actuators) was one of the early motivations for learning control.

dynamic objective function is a scalar function that cannot be completely defined in terms of the current environment, plant and learning system information, due to the fact that the objective function has internal state. An example of such an objective function is the familiar integral quadratic cost function from optimal control theory. It should be noted that the learning control problem is considerably more difficult in the case of a dynamic objective function than an instantaneous one. This is discussed below.

Given a particular objective function, it is possible to provide different types of performance feedback to the learning system. A common scenario in early learning control system implementations was the discrete case in which the output of the objective function was mapped to a binary *reinforcement* signal. In such situations, the reinforcement signals could be thought of as being "positive" or "negative." Positive reinforcement was designed to encourage the same behavior in future visits to the same control situation, while negative reinforcement was designed to discourage that behavior. Although most early approaches employed binary (or discrete) valued reinforcement signals, continuous valued signals could also be used. In either case, performance feedback of this type provides a direct indication of the appropriateness of the selected control action, but does not provide any (direct) indication as to whether an alternative control action might be more appropriate. As a result, reinforcement learning systems lead to stochastic learning rules that are used to search for the best control action among the set of applicable control actions.

Whether the objective function is instantaneous or involves state, the performance feedback signal it provides may be separated temporally from the application of the control signal; this phenomenon is referred to as *delayed reinforcement*. A classic and often used example of delayed reinforcement is the problem of balancing an inverted pendulum on a moving cart. In this problem, the only reinforcement that the system receives is negative, when the pole falls; at all other times, no reinforcement signal is provided. This situation gives rise to what is known as a (temporal) credit assignment problem. The crux of this credit assignment problem is to determine the set of control actions (which were applied in previous control situations) that were ultimately responsible for the failure, so that they can be corrected. An early paper addressing this type of problem (in a different context) is due to Samuel [17], subsequent work includes Michie & Chambers [14], and Barto, *et al.* [5].

In cases where the reinforcement signal is based on a dynamic objective function, the appropriateness of a particular control action can no longer be easily ascertained from the performance feedback signal. The reason for this is that the performance feedback signal is not necessarily a unique function of the most recently selected control action; instead, it may now depend on the entire history of control actions that have been applied up to that point in time. Therein lies the difficulty, for it is no longer immediately clear which control action(s) contributed most strongly (and in what direction) to the performance feedback that was received. This is another version of the temporal credit assignment problem.

Control Action Selection. There are two aspects to the problem of selecting an appropriate control action. The first involves the use of the control mapping that has been learned up to that point in time to *generate* a control action. The second involves the *adjustment* of the current control mapping, based on the latest performance feedback information. Stochastic learning methods are required when derivative information relating the objective function to the control mapping is not available. When gradient information is available, the desired control mapping can usually be developed more efficiently, since the derivative information concerning the objective function provides *directional* information for improving the outputs of the learning system. We will refer to this special class of learning control approaches as being *gradient based.* In this section, we discuss only the stochastic reinforcement learning methods. The gradient based techniques are fully discussed in the subsequent sections of this chapter. The stochastic learning control problem can be treated as a pattern classification task, in the sense that a large number of possible control situations are to be classified according to a smaller number of admissible control actions. Two different implementation techniques will be discussed to exemplify these approaches.

In one approach, see [14], each control situation maintains a probability vector for the selection of discrete control actions. When the state of the plant enters a control situation, a control action is randomly generated according to the probability density function defined over the ensemble of control alternatives for that control situation. If the chosen control action (eventually) receives positive (negative) reinforcement then the probability of that action is adjusted to be more (less) likely relative to the alternative control actions available in that control situation. Thus, a stochastic competition takes place to evolve the best control action in each control situation. In addition to the learning issues involved with credit assignment, this stochastic search approach must ensure adequate exploration of all control actions for each control situation.

The choice between the alternative control actions represented by the set A can also be determined through a decision-theoretic analysis if the sample experiences are used to approximate the joint probability density functions $p_i(x)$ for all i, that control action a_i is correct at position x. Using these estimated joint distributions, decision-theoretic switching boundaries can be identified, see [19]. Control action generation is then determined by the location of the current state relative to the current switching boundaries. The switching boundaries are adjusted automatically as the probability density functions are updated.

The differences between these two examples are interesting. The former implementation fixes the control situation boundaries and adjusts the probability of each admissible control action within each control situation. The latter defines the control situations implicitly as a function of the joint probability density functions $p_i(x)$; thus, the switching curves are not constrained by *a priori* control situation boundary definitions.

In both implementations, the finite action set is fixed. This limitation on the output resolution of the control actions constrains the accuracy of the resulting control law. These accuracy limitations can be decreased, at the expense of increased memory requirements, by allowing different interpretations of the action set in different control situations or by allowing new control actions to be added to the action set. Such adjustments to the action set were not the main focus of early research efforts. In contrast, continuous adjustment of the control actions is a priority for the function synthesis approaches currently being developed.

Control Situation Definition. In early applications (e.g., [14]), where the interpretation of each control action and control situation was defined *a priori* and did not change during the learning process, the expected input domain of the control law was partitioned in advance. The boundaries of the control situations described potential switching curves (lines of discontinuity) in the control mapping. This had the desirable effect of considerably simplifying the implementation of a learning control system, but had the undesirable effect of limiting its performance in all but the most exceptional cases. There are several reasons for this. First, an obvious drawback to using fixed control situations is the inability to resolve learning conflicts in cases where the performance feedback indicates that a switching curve should (ideally) pass *through* a control situation, rather than along its periphery. Second, if nothing at all is known about the nature of the solution in advance, then a rational approach is simply to create a uniform (and relatively fine) partitioning of the control mapping input domain, which implies a relatively large number of different control situations.[6] Frequently, however, a significantly better partition exists that exploits specific characteristics of the problem. In a regulator control problem, for example, control situations corresponding to plant states far from the setpoint can be large (coarsely quantized) compared to control situations in which the plant state is close to the setpoint. Such variable resolution would allow for improved transient and steady-state closed-loop system behavior in the vicinity of the setpoint. Generally speaking, a learning control system can be more efficient and achieve better performance if control situations are adjusted on-line during the learning process.

Several techniques were developed to address this need. Waltz & Fu [21] expanded on a pattern recognition idea by Sebestyen [18] to create a learning control system capable of refining its control situations. Initially no control situations were defined. As new experience was obtained during a training session, the idea was to create new control situations with decreasing radii, as required, in the vicinity of the perceived switching curves. Thus, memory was not wasted on representing control situations that might never occur, and increased resolution was expected along perceived switching curves that emerged as learning progressed. Sklansky [19] reviews two approaches aimed at providing more general solutions to the problem of learning the control situations. The first approach, based on decision theory, was discussed in the

6. Worse still, a uniform partition of a multidimensional space grows exponentially with the number of subdivisions in each dimension, so that even modest problems can require prohibitively large amounts of memory.

previous subsection. As a means of overcoming the burden of estimating joint probability densities, the second approach attempted to explicitly represent control situation boundaries using linear or nonlinear functions. The boundaries were represented as parameterized combinations of these functions and performance feedback was used to adjust the parameters.

Comment. Much of the early research in learning control was shaped by the computational resources that were available (or likely to become available) at the time (circa 1960). This led to the major limitation of most early learning control approaches — the need to rely on a representational structure that had a discrete domain and corresponding discrete range. This created many problems, as outlined above, and was particularly troublesome in situations where the ideal control mapping was actually better approximated by a smooth nonlinear function having a continuous domain and a continuous range. Recent work in the area of connectionist learning systems has provided a new strategy for the development of learning control systems with substantially improved properties, relative to early learning control systems. A key idea espoused by this new strategy is that the on-line synthesis of control mappings can be treated as a type of smooth function approximation problem.

4. Contemporary Implementations

A commonly held notion is that learning results in an *association* between input stimuli and desirable output actions. By interpreting the word "association" in a mathematical sense, one is naturally led to the central idea underlying many contemporary implementations of learning control systems. "Learning" can be interpreted as a process of automatically synthesizing multivariable functional mappings, based on a criterion for optimality and experiential information that is gained incrementally over time [3]. Most importantly, this process can be realized through the selective adjustment of the parameters and structure of an appropriate representational framework. The advantages of contemporary implementation schemes relative to those of the past are numerous and will be considered in this section. Key benefits are derived from the smoothness, generality, and representational efficiency of the mappings that can be obtained (i.e., learned).

To further develop the central theme of this section, we will proceed by elaborating on the notion of learning as automatic function synthesis. After a more formal introduction to the concept and its key issues, the applicability of *connectionist* learning systems in this context will be described, followed by other subsections covering the issues of incremental learning and spatially localized learning. To allow for a more concrete discussion of the key concepts, we will make use of the following definitions. Let M denote a generalized memory device[7] that will be incorporated into a learning system, and let D represent the domain over which this memory is applicable. If $z \in D$, then the expression $u = M(z)$ will be used to signify the "recall" of

7. By generalized, we mean that the usual concept of a discrete input, discrete output memory has been extended to allow for more general continuous input / output relationships.

item u by "situation" z from the memory M. The desired mapping to be stored by this memory (via learning) will be denoted by M^*. Since the desired mapping is not usually known, it must be learned. This is the subject of the following subsection.

Learning as Automatic Function Synthesis. The function synthesis concept is easily visualized in an idealized situation where M is an infinite capacity memory capable of storing an independent value of u for each instance of z. As an example, we will consider a simple associative memory (defined below) problem. Let $z(k)$ be a concatenated vector comprised of the current plant state $x(k)$ and the desired plant state on the next time-step $x_d(k+1)$ (i.e., $z(k) = [x(k), x_d(k+1)]$), let $u(k)$ (the output of M) be the control signal applied at time k, and let the objective function be a norm of the difference between the desired and actual plant states at time k. In this example, the purpose of the learning system is to output an appropriate control signal, given the current state and desired next state of the plant, so as to cause the value of the objective function to be minimized at the next time instant. Assuming there is no measurement noise or plant disturbances and the problem is stationary, then each closed-loop interaction can be interpreted as a learning experience in which the application of the control signal $u(k)$ to the plant in state $x(k)$ is seen to result in a change of the plant state to $x(k+1)$. By setting the value of $M(z)$ at $z(k) = [x(k), x(k+1)]$ to $u(k)$, the learning system will have stored the appropriate control action for use in the future, whenever the plant state is $x = x(k)$ and the desired next state is $x_d = x(k+1)$. With continued use, many such learning experiences could be incorporated into the memory, allowing the desired function to be constructed. In this idealized setting M^* is accurately represented by M, albeit on a point by point basis.[8]

It is interesting to reconsider this example in terms of the material presented in the previous section. In this case, each control situation has been reduced to a distinct point (which is only possible given the assumption of infinite memory capacity). Except for finite state problems, this approach would actually be unusable since the probability of entering the same control situation twice is effectively zero; hence, the learning control system would never benefit from its past experience. Nearest neighbor or local interpolation techniques can be used to circumvent this problem — the idea being that previous learning under similar circumstances can be combined to provide a suitable response for the current situation. This type of fusion process effectively broadens the scope and influence of each learning experience, and is referred to as *generalization* of the training data.

There are several important ramifications of generalization. First, it has the effect of eliminating "blank spots" in the memory (i.e., specific points at which no learning has occurred), since *some* response (not necessarily the desired one) will always be generated. Second, it has the effect of constraining the set of possible input / output

8. The general strategy suggested by these remarks is commonly referred to as an *associative memory* approach (see [2]).

mappings that can be achieved by the memory, since in most cases neighboring input situations will result in similar outputs. Finally, generalization complicates the learning process, since the adjustment of the mapping following a learning experience can no longer be considered as an independent, point by point process. In spite of this, the advantages accorded by generalization far outweigh the difficulties it introduces.

For a wide and important class of learning control problems, the desired mapping is known to be continuous in advance. In such situations, memory implementations with more efficient recall mechanisms can be proposed. If we are willing to assume that the desired memory M^* is continuous, which is an adequate assumption for most physical systems, then an approximate mapping M can be implemented by any scheme capable of approximating arbitrary continuous functions. In this case, the memory M is implemented as a continuous function parameterized by the vector θ ; i.e., $M = M(z;\theta)$. To apply this approach to the architecture of the previous associative memory example, the value of $M(z;\theta)$ at $z(k) = [x(k), x(k+1)]$ is set to $u(k)$ not by adding an additional memory item, but by appropriately adjusting the parameter vector θ. As learning experiences become available, the mapping will be incrementally improved.

By constraining the mapping structure to have a finite number of parameters, while still requiring an accurate approximation over the entire input domain, each parameter is forced to affect the realized function over a region of non-zero measure. When a parameter is adjusted to improve the approximation accuracy at a specific point, the mapping will also be affected throughout the region of influence of that parameter. Thus, by constraining the mapping structure the training data is automatically generalized. The nature of this generalization may or may not be beneficial to the learning process depending on whether or not the extent of the generalization is local or global. These issues are further discussed in the Incremental Learning and Spatially Localized Learning subsections.

The point by point storage that was described as a learning process for the associative memory implementation is not feasible for these parametric function synthesis approaches. Instead, the parameters are adjusted based on the derivative information for the objective function. More insight can be gained into gradient learning algorithms through an application of the chain rule, which yields

$$\Delta\theta = -\mathrm{W} \cdot \partial M^{\mathrm{T}} / \partial\theta \cdot \partial J / \partial M$$

where W is a positive definite weighting matrix, J is the objective function, M is the output of the approximating function, and θ denotes the parameter vector for the approximating function (the Jacobian $\partial M^{\mathrm{T}} / \partial\theta$ is defined as a matrix of gradient column vectors $\partial M_i / \partial\theta$, so that $\partial J / \partial\theta = \partial M^{\mathrm{T}} / \partial\theta \cdot \partial J / \partial M$). This form of the gradient learning rule involves two types of information: the Jacobian of the outputs of the mapping with respect to the adjustable parameters, and the gradient of the objective function with respect to the mapping outputs. The gradient $\partial J / \partial M$ is deter-

mined both by the specification of the objective function J and the manner in which the mapping outputs affect this function—which is determined by the way in which the learning system is used within the control system architecture. The Jacobian $\partial M^{\mathrm{T}}/\partial\theta$ is completely determined by the approximation structure; and, hence, is known *a priori* as a function of the mapping input. Note that the performance feedback information provided to the learning system is the output gradient $\partial J/\partial M$. This gradient vector provides the learning system with considerably more information than the scalar J in particular, $\partial J/\partial M$ indicates both a direction and magnitude for $\Delta\mathrm{p}$ (since $\partial M^{\mathrm{T}}/\partial\theta$ is known), whereas performance feedback based solely on the scalar J does neither.

Notice that, conceptually, the associative memory based techniques offer simple training with complex recall mechanisms, while the parametric approximation techniques require more complex training but offer simple recall. The actual training algorithms for parametric function synthesis methods are control architecture dependent. Two specific instances are presented in the examples.

Connectionist Learning Systems. Connectionist systems, including what are often called "artificial neural networks," have been suggested by many authors to be ideal structures for the implementation of learning control systems. A typical connectionist system is organized in a network architecture that is comprised of nodes and connections between nodes. Each node can be thought of as a simple processing unit. Typically, the number of different node types in a network is small compared to the total number of nodes. Common examples of connectionist systems include multilayer perceptron and radial basis function networks. The popularity of such systems arises, in part, because they are relatively simple in form, are amenable to gradient learning methods, and can be implemented in parallel computational hardware. More importantly, however, it is well known that several classes of connectionist systems have the *universal approximation property*. This property implies that any continuous multivariable function can be approximated to a given degree of accuracy by a sufficiently large network (see [10], [11]).

Although the universal approximation property is important, it is held by so many different approximation structures that it does not form a suitable basis for distinguishing them. Thus, we must ask what other attributes are important in the context of learning control. In particular, we must look beyond the initial biological motivations for connectionist systems and determine whether they indeed hold any advantages over more traditional approximation schemes. An important factor to consider is the environment in which learning will occur; that is, the information that is likely to be available to the learning system.

Incremental Learning Issues. Incremental function synthesis using data obtained during closed-loop experiments constrains the choice of approximation approaches. In closed-loop control, the training examples cannot be selected freely, as the plant out-

puts are constrained by the system dynamics, and the desired plant outputs are constrained by the specifications of the control problem. Under these conditions, the training examples often remain in small regions of the domain of the mapping for extended periods of time. This training sample "fixation" can have deleterious effects in situations where parameter adjustments can affect the input / output map globally. For example, if a parameter that has a global effect on the mapping is repeatedly adjusted to correct the mapping in a particular input region, this may cause the map in other regions to deteriorate and, thus, can effectively "erase" the learning that has previously taken place. This characteristic of learning control problems has led to the development and analysis of *spatially localized* architectures and learning rules.

Spatially Localized Learning. The basic idea underlying spatially localized learning arises from the observation that learning is facilitated in situations where a clear association can be made between a subset of the adjustable elements of the learning system and a localized region of the input-space. Further consideration of this point in the context of the difficulties described above, suggests a few *desired* traits for learning systems that rely on incremental gradient learning algorithms. These objectives can be expressed in terms of the "sensitivity" functions $|\partial M_i/\partial\theta_j|$, which are the partial derivatives of the mapping outputs M_i with respect to the adjustable parameters θ_j. At each point x in the input domain of the mapping, it is desired that the following properties hold:

- for each M_i, there exists at least one θ_j such that the function $|\partial M_i/\partial\theta_j|$ is relatively large in the vicinity of x (*coverage*)
- for all M_i and θ_j, if the function $|\partial M_i/\partial\theta_j|$ is relatively large in the vicinity of x, then it must be relatively small elsewhere (*localization*)

Under these conditions, incremental gradient learning is supported throughout the input domain of the mapping, but its effects are limited to the local region in the vicinity of each training point. Thus, experience and consequent learning in one part of the input domain have only a marginal effect on the knowledge that has already been accrued in other parts of the mapping. Although sigmoidal networks do not have the localization characteristic, several other network architectures, including BOXES [14], CMAC [1], radial basis function networks [15], and basis / influence function networks [3-4] do have this characteristic.

Spatially localized learning rules capitalize on localization in two ways. First, localization implies that at each instant of time only a small subset of all the network parameters have a significant effect on the network output. Thus, both the efficiency of calculating the network output and of updating the network parameters can be improved by neglecting the insignificant parameters. This approach can greatly increase the throughput of a network when implemented in digital hardware. Furthermore, since training examples may remain in particular regions of the input domain for extended periods of time, it is expected that the approximation error will not tend *uniformly* to zero. Instead, the error will be lowest in those areas where the greatest amount of training has occurred while still large in other regions. This leads to conflicting constraints on the learning rate: it should be small, to filter the effects of noise, in those regions where the approximation error is small; at the same time, it

should be large, for fast learning, in those regions where the approximating error is large. Resolution of this conflict is possible by the use of spatially dependent learning rates.

The memory requirements for spatially localized architectures fall somewhere in between those of associative memory and non-local function approximation architectures. When we ask for each parameter to have a localized effect on the overall approximation, we should expect an increase in the number of parameters required to cover the entire input domain. However, in control applications, training speed and approximation accuracy should have priority over memory requirements as memory is inexpensive relative to the cost of inaccurate or inappropriate control actions.

6. Architectures and Examples

The previous sections have discussed several issues related to the implementation of effective learning control systems. The purpose of this section is to present example architectures for the implementation of these systems. For each architecture, we present an illustrative example and identify the algorithms necessary for implementing the training process. Each example is designed to demonstrate the desired principles, yet be simple enough and include enough detail for interested readers to reproduce.

Example 1. This example demonstrates a technique designed for applications, such as aerospace or underwater vehicle control, where the linearized dynamics are known to change significantly as a function of a measurable set of scheduling variables. The gain scheduling method is the state-of-the-art approach when the underlying system is well modeled. The present methodology is applicable when the schedule for the linearized dynamics is difficult to accurately predict *a priori* due to model uncertainty. The block diagram in Figure 1 depicts the various components of the implementation we will discuss. If the performance evaluator was not implemented and the learning system was replaced with a static gain schedule, this block diagram would represent the usual gain scheduled approach. Alternatively, if the learning system was eliminated and a single (globally applicable) set of control system parameters was adjusted independent of operating condition, then this block diagram would represent a direct adaptive control system. The implementation of the entire system represented in Figure 1, can be viewed as either a traditional adaptive system augmented with memory or a traditional gain scheduled system augmented with the ability to adjust its

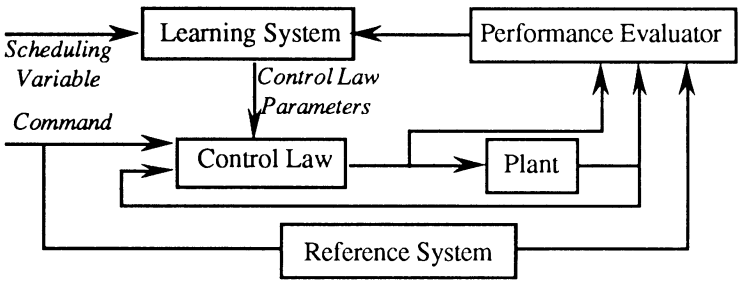


Figure 1. Block diagram representation of the direct learning approach.

performance on-line.

For this example, the plant is represented as

$$x(k+1) = A(v)x(k) + B(v)u(k)$$

$$y(k) = Cx(k)$$

where v is the scheduling variable. Specifically, let

$$A(v) = \begin{bmatrix} 0.665v(v-2) & -0.95 \\ 1.0 & 0.0 \end{bmatrix},$$

$$B(v) = \begin{bmatrix} 0.3\left(\dfrac{e^{-v}-1}{v} + 1.1\right) \\ 0.0 \end{bmatrix}, \qquad C = [1.0, 0.0].$$

For this example, we assume that the dynamics of the scheduling variable v are modeled by

$$v(k+1) = 0.9v(k) + 0.1\mu(k).$$

Thus, the scheduling variable will change slowly and smoothly, as would be the case if v represented, for example, the forward velocity of a vehicle. The variable $\mu(k)$ will be used in the simulation to change the operating point $v(k)$ of the system. The assumed range for μ is [0,1]. We assume knowledge of the system order, but assume no knowledge of the plant structure or the manner in which the scheduling variable affects the dynamics.

The control system goal is to track an input signal, with a specified transient response. The desired dynamics for the closed loop system are implemented as a reference system as indicated in Figure 1, so that an error can be formed at each step for performance evaluation. We specify the characteristic equation of the linear reference system by the discrete time pole locations, $p = 0.4 \pm 0.05i$, where i represents the square root of -1. In most real applications, the pole locations would also be scheduled since the performance attainable by the system with reasonable actuator commands is not expected to be constant. For simplicity, constant desired pole locations were used in this example. The numerator of the reference system was selected to result in a unity gain system with a zero at the origin.

The control law is assumed to have a state feedback structure with variable parameters. The goal of the learning control system is then to identify and store the controller parameter vector as a function of the scheduling variable. The learning system incorporates a 3 output Radial Basis Function (RBF) network with 31 nodes equally spaced over the interval [–0.0357,1.0357]. The three network outputs are interpreted as the first and second state feedback gains and the feedforward gain for the reference input, respectively. At each time instant the following operations are performed:

- the scheduling variable, $v(k)$, is measured,
- the control gains, $K(v(k))$, are read from the RBF network,
- the control signal is calculated and applied,

- the effect of the control is compared with the desired effect via the reference model
- this error is used to determine a control gain correction vector
- the control gain correction vector is used to improve the radial basis function approximation.

The performance evaluator is responsible for converting the control loop variables into the learning system training signal. This is accomplished in two stages. In the first stage, the Lyapunov argument typical for model reference adaptive approaches is used to determine the desired correction for the control gains output by the RBF for the current value of the scheduling variable:

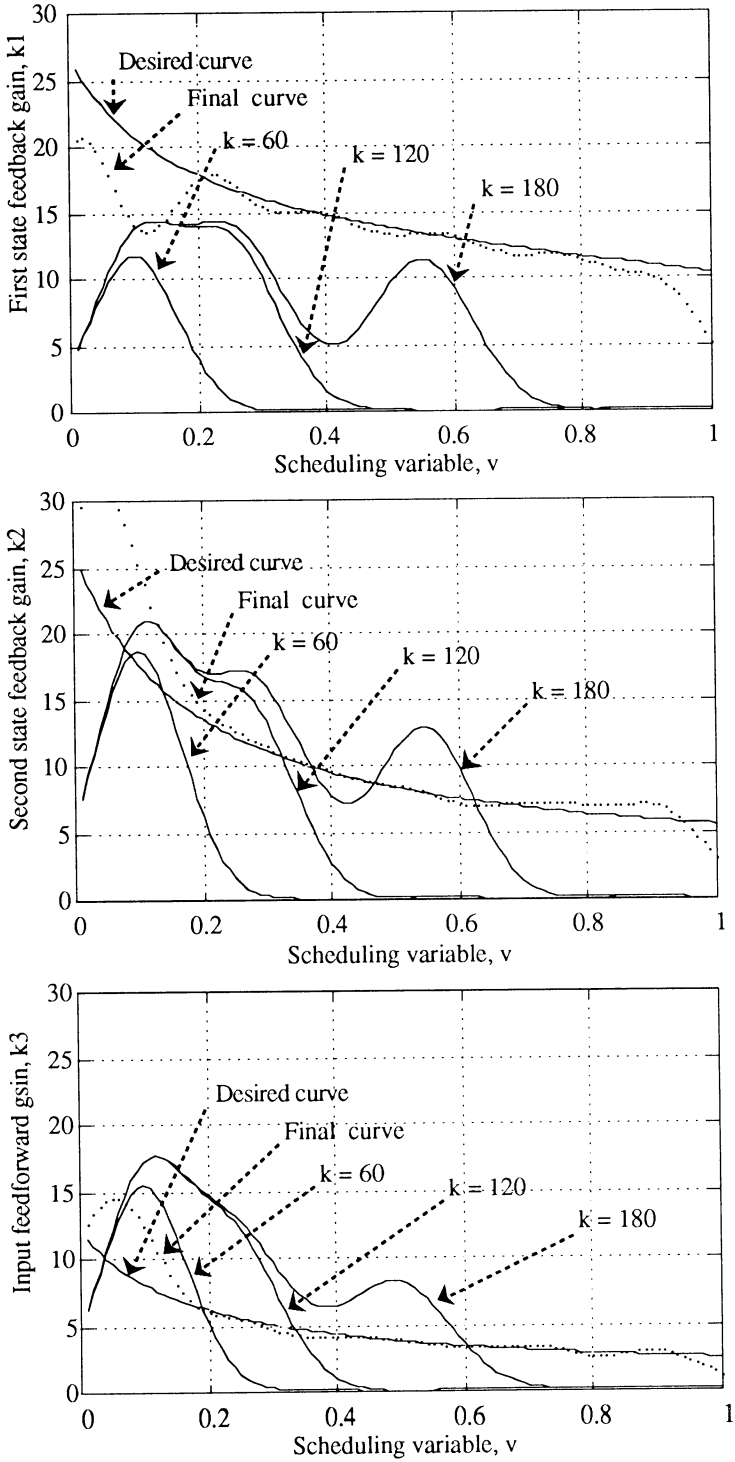
$$\delta K(v[k]) = \alpha(error)\phi,$$

where $error = x_1^{ref} - x_1)$, $\phi = \{x_1[k], x_2[k], r[k]\}^T)$, α is small and positive, and $K(v(k))$ are the control law parameters output by the RBF evaluated at $v(k)$. In the second stage, the RBF output parameters are changed, by gradient descent, to completely implement the change $\delta K(v[k])$.

Figures 2 a-c illustrate the control parameter schedules (first and second state feedback gains and the feedforward gain, respectively) learned in simulation trials of this methodology. In the example, the value of $\mu(k)$ was changed every 60 iterations, and 50 different values of μ were issued. The first three values of $\mu(k)$ were set to 0.1, 0.3, and 0.6. The remaining values were generated by a uniform random number generator; however, as the dynamics of the scheduling variable constrain it to change continuously, many more training examples are generated for midrange values of $v(k)$ than for the extreme values. Each figure shows the form of the RBF approximation after 60, 120, 180, and 3000 iterations. The 3000 iteration graph (i.e., final curve) demonstrates the accuracy with which this methodology was capable of learning the desired gain schedule. Note that the approximation error is least for midrange values of $v(k)$, where the most training examples were generated.

The 60, 120, and 180 iteration curves correspond to the RBF approximations after training in the vicinity of v equal 0.1, 0.3, and 0.6, respectively. Notice that while v is near any one of these points, only the control gains in the vicinity of that value are adjusted. This is the effect of using a function approximation system which has the spatial localization property. The issues of training fixation and spatial localization of training are critical in applications of this type, where we expect prolonged training at particular operating points within the operational envelope and do not want training in a particular region to affect previously stored information in other regions. We also note that training at specific operating points generalizes to the surrounding regions.

This example demonstrates that augmenting an adaptive system with memory allows the adaptive mechanism to progress from continually reacting to changing linearized dynamics to constructing the appropriate control actions for measurable operational conditions. Various technical issues would have to be addressed to guarantee success



Figures 2 a-c. The control system parameter schedules learned for Example 2.

ful on-line operation. These issues include sufficiency of excitation, disturbances, and model-order errors, etc. They can be accommodated by a supervisory logic system as is the case in a conventional adaptive approach [6], [16]. These issues are beyond the scope of the present chapter; however, in some instances the learning approach appears to offer implementation advantages over the corresponding adaptive approach. For example, in the present type of application an adaptive system would require persistent excitation to ensure accurate control parameters as the operating condition changes. The learning system would only require sufficient excitation, during some training phase, until the gain schedule had been identified. Similar comments also apply in the following example.

Example 2. Recurrent networks are of interest due to their ability to represent a large class of *dynamic* systems. By appropriate manipulations, any recurrent network can be transformed to the form

$$\left.\begin{aligned}\hat{x}(k+1) &= f_{net}(\hat{x}(k), u(k); \theta)\\ \hat{y}(k) &= h_{net}(\hat{x}(k), u(k); \theta)\end{aligned}\right\} \tag{1}$$

where u and y represent the network inputs and outputs respectively, θ represents the network parameters, and $\hat{x}$ represents the state of the recurrent system. Using the function approximation philosophy, the connectionist network can be viewed as a parameterized model of a dynamic system that is to be learned. Since the network and system structures are generally different, the system map cannot be represented exactly over the domain of interest. Instead, the goal is to find network structure and training algorithms that will be able to approximate these systems adequately. In addition to the parameter adjustment required for learning in feedforward systems, recurrent networks have the added difficulty that the state of the network must be corrected to ensure its accuracy and stability in spite of noise, disturbances, and arbitrary initial conditions.

This example demonstrates the use of a recurrent function approximation scheme to identify the form of an unknown nonlinear plant. The identified nonlinear model is then used in a model based control system, see Figure 3. The training of recurrent connectionist systems is an area of significant current interest. For the training of

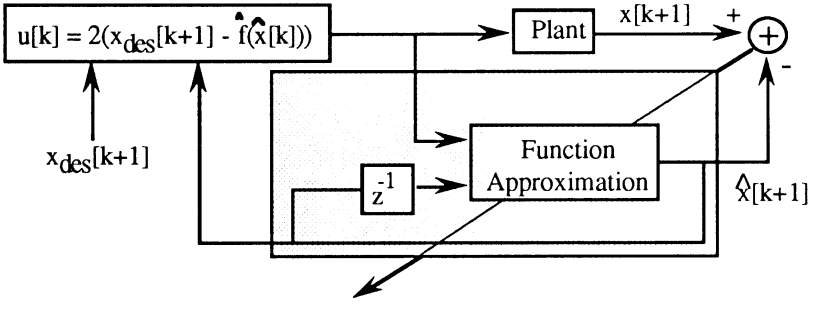


Figure 3. Block Diagram of Control Architecture for Example 2.

this example we use the Spatially Localized Extended Kalman Filter Algorithm [13]. In other extended Kalman filter (EKF) formulations, all the network parameters must be changed at each training instance. These EKF algorithms require the manipulation of matrices whose dimensions are on the order of the total number of parameters in the network. The drawbacks of computational complexity and memory requirements are significant enough to render such implementations impractical for applications requiring on-line learning. The Spatially Localized Extended Kalman Filter Algorithm was developed as a local modification of the Extended Kalman Filter for the class of spatially localized recurrent networks. Computational efficiency of the learning scheme is achieved by exploiting local properties of the network structure.

As a particular example, assume that the only *a priori* information about the plant is that it is of the form

$$x(k+1) = f(x) + 0.5u(k)$$
$$y(k) = x(k)$$

and that the goal is to control the true plant such that it tracks some desired reference input. For this example the plant nonlinearity will be

$$f(x(k)) = 0.5\sin(x(k)).$$

The network model is

$$\hat{x}(k+1) = f_{net}(\hat{x}(k)) + 0.5u(k)$$
$$\hat{y}(k) = \hat{x}(k)$$

Since the control goal is $x(k+1) = x_{des}(k+1)$, the control law based on the current system model is

$$u(k) = 2(-f_{net}(\hat{x}(k)) + x_{des}(k+1))$$

which will yield perfect tracking if the network learns the nonlinear dynamics $x(k+1) = 0.5\sin(x(k))$ exactly, and the state estimate $\hat{x}$ converges.

For simplicity, a one dimensional BOXES structure is used to implement the function approximator. The input domain, D = [-1.2,1.2], is divided into 50 equally sized intervals. One real valued parameter corresponds to each interval. The approximation of f is then calculated for any value of $\hat{x}$ by determining which interval contains $\hat{x}$, and averaging the parameters associated with that interval and its two nearest neighbors. Thus, neighboring intervals will share two parameters. The sharing of parameters between intervals allows a compromise between approximation accuracy and training time. Alternatively, the accuracy could be increased by associating a linear or higher order function with each interval, but the constant parameters were selected to maintain the simplicity of the example.

The Spatially Local Extended Kalman algorithm is used to update the parameters and state of the approximation system. This algorithm simplifies the EKF algorithm calculations for spatially localized architectures. At each time instant the state estimate vector is augmented with the three model parameters pertinent to the calculation of $\hat{f}(\hat{x})$. To calculate the Kalman filter feedback gains, we require the Jacobian matrix

$$F[k] = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial \theta} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x}_{(1\times1)} & \frac{\partial f}{\partial \theta}_{(1\times3)} \\ 0_{(3\times1)} & I_{(3\times3)} \end{bmatrix}$$

for the augmented nonlinear dynamic system, where θ denotes the current three elements of the model parameter vector. Due to the approximation structure that was selected,

$$\frac{\partial f}{\partial \theta} = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right] \text{ and } \frac{\partial f}{\partial x} = 0.$$

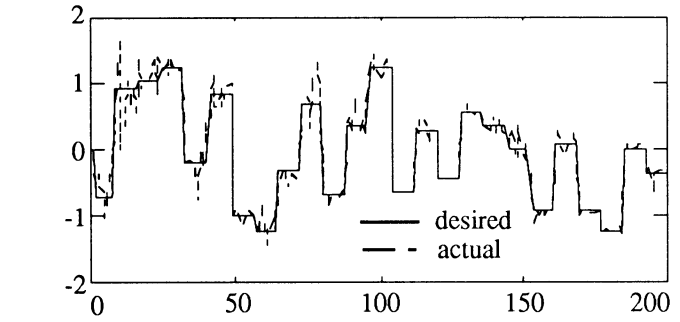
Alternatively, $\frac{\partial f}{\partial x}$ could be approximated by a first difference of the parameters associated with neighboring intervals. Several alternative methods are discussed in [13] for simplifying the EKF implementation and decreasing its memory requirements.

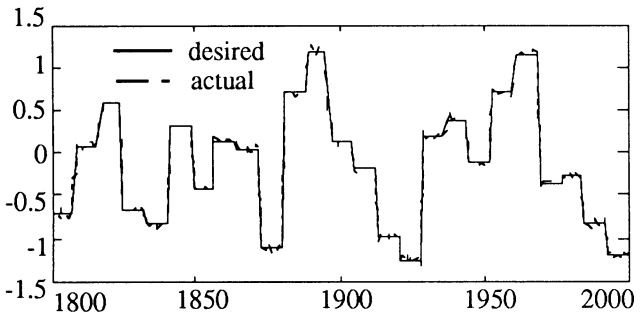
The parameters associated with each interval were initialized to zero and a simulation was carried out for 2000 time steps. The training of the network and the computation of the control law were performed at every time step. Figure 4a shows the initial poor tracking caused by an incorrect model of the true plant, and Figure 4b shows the improved performance after the network has sufficiently learned the nonlinear dynamics. Figure 4c shows that the initial control effort is reasonable. The nonlinearity learned by the network after 2000 iterations is shown in Figure 4d. The data presented in Figure 4 is from a noise-free simulation; however, adding white noise to the simulation yields similar results.

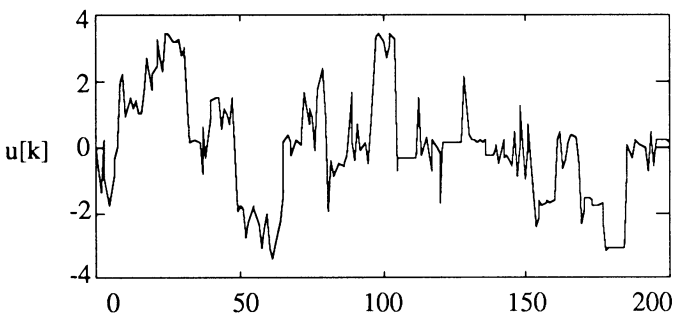
6. Conclusion

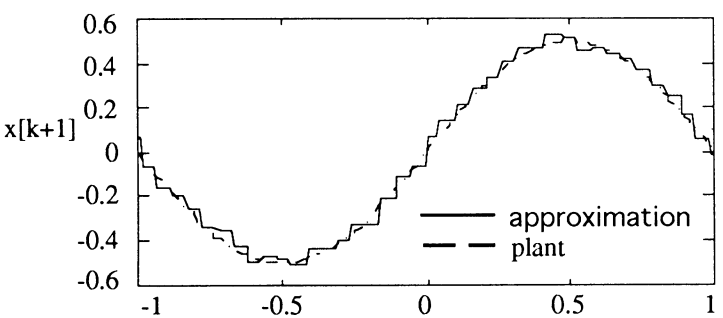
The principal objectives of this chapter were to describe the salient features of learning control systems, to differentiate these systems from related approaches, and to suggest a general implementation methodology. We have intentionally avoided the urge to identify and categorize the ever-growing variety of learning control system architectures. Instead, we have focused on the main motivation, operation, and implementation theoretic issues.

In summary, intelligent control systems incorporating learning capabilities have the potential to (1) *facilitate the control system design and tuning process*, (2) *accommodate uncertainty* through on-line interaction with the actual plant, and (3) *improve efficiency and performance* through on-line self-optimization. Learning control systems can be implemented as general automatic function synthesis mechanisms, and are suggested for applications involving either poorly modeled nonlinear dynamics or tedious manual optimization.









Figures 4 a - d: (4a) Initial tracking accuracy; (4b) Final tracking accuracy; (4c) Simulation control signal; (4d) Approximated nonlinearity

Two examples have been offered to demonstrate possible learning control architectures and to underscore the issues related to spatial localization, generalization, and recurrent training. Numerous alternative control implementation architectures have been suggested in the literature. At the present stage in the development of learning control systems, implementation and experimentation is critical both to demonstrate their feasibility and possible benefits, and to identify future research directions. The fact that current computational facilities are sufficient to support these implementations is demonstrated in Farrell & Baker [7].

To close this chapter, we suggest the following three subjects as areas of research offering significant opportunities to improve the future capabilities of learning control systems. (1) *Investigation of important characteristics for learning control system function synthesis:* How large a network is required to adequately represent a desired mapping? What resources (computational as well as storage) are required for implementation? What conditions are required to ensure convergence of the parameter vector? If so, how fast will it occur? Are there potential pitfalls associated with certain types of representation schemes and learning algorithms? (2) *Investigation of techniques for variable structure representation schemes:* The attainable approximation accuracy and the implementation requirements are determined by the structure of the function synthesis system. When the structure is determined *a priori*, an inefficient use of resources may occur. Either too few resources may be assigned to approximate complex regions of the learning domain, thus limiting the approximation accuracy, or too many resources may be applied to approximate simple portions of the function, resulting in wasteful use computational resources. Variable structure learning schemes are one solution to this dilemma; however, efficient modification rules, those for which the solution is less complex than the original problem, are still unidentified. (3) *Higher level learning: A priori* specification of a performance index over an entire operational envelope is a difficult task. This line of research is directed at adjusting the objective function on-line to request increased performance where possible, and to decrease strain on the system where necessary.

7. Acknowledgment

This work has been supported, in part, by the National Science Foundation under Grant No. ECS-9014065, the Charles Stark Draper Laboratory, Inc. Independent Research and Development Program under Project No. 276, and the Air Force Wright Laboratories under contract F33615-88-C-1740. The sponsors have certain rights in this material. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring agencies or the Charles Stark Draper Laboratory, Inc.

The authors would also like to acknowledge the contributions to the development of this approach by the past and present members of the Draper Connectionist Learning Control Research Group.

8. Bibliography

[1] Albus, J., "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), *"Trans. ASME, J. Dynamic Syst., Meas., Contr.*, Vol. 97, pp. 220-227, Sept., 1975.

[2] Atkeson, C., "Using Associative Content-Addressable Memories to Control Robots," *Proceedings, IEEE International Conference on Robotics and Automation*, May, 1989.

[3] Baker, W. & Farrell, J., "Connectionist Learning Systems for Control," *Proceedings, SPIE OE/Boston '90*, November, 1990.

[4] Baker, W., and J. Farrell, "An Introduction to Connectionist Learning Control Systems," *Handbook of Intelligent Control: Neural Network, Fuzzy Logic, and Adaptive Implementations*, Van Nostrand Reinhold, 1992.

[5] Barto, A., Sutton, R. & Anderson, C., "Neuronlike Adaptive Elements that can Solve Difficult Learning Control Problems," *IEEE Transactions of Systems, Man, and Cybernetics*, Vol. SMC-13, No. 5, September / October 1983.

[6] Clauberg, B. & Farrell, J. "Issues in the Implementation of an Indirect Adaptive Control System," Draper Laboratory Report CSDL-P-3136, Cambridge, MA, 1991.

[7] Farrell, J. & Baker, W., "Learning Augmented Control for Advanced Autonomous Underwater Vehicles," *Proceedings of the 18th Annual AUVS Technical Symposium and Exhibit*, August , 1991.

[8] Fu, K., "Learning Control Systems," in Tou, J. & R. Wilcox, eds., *Computer and Information Sciences*, Spartan, 1964.

[9] Fu, K., "Learning Control Systems — Review and Outlook," *IEEE Transactions on Automatic Control*, Vol. AC-15, No. 2, April, 1970.

[10] Funahashi, K., "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, Vol. 2, pp. 183–192, 1989.

[11] Hornik, K., Stinchcombe, M. & White, H., "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks*, Vol. 2, pp. 359–366, 1989.

[12] Landau, Y., *Adaptive Control: The Model Reference Approach*, Marcel Dekker, 1979.

[13] Livstone, M., Farrell, J. & Baker, W."A Computationally Efficient Algorithm for Training Recurrent Connectionist Networks," submitted to the 1992 American Control Conference.

[14] Michie, D. & Chambers, R., "BOXES: An Experiment in Adaptive Control," in Dale, E. & Michie, D., eds., *Machine Intelligence 2*, Oliver and Boyd, 1968.

[15] Poggio, T. & F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1481-1497, September 1990.

[16] Narendra, K. S., R. Ortega, & P. Dorato, eds., *Advances in Adaptive Control*, IEEE Press, New York, 1991.

[17] Samuel, A., "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, Vol. 3, pp. 210–229, 1959.

[18] Sebestyen, G., "Pattern Recognition by an Adaptive Process of Sample Set Construction," *IRE Transactions on Information Theory*, Vol. IT-8, September, 1962.

[19] Sklansky, J., "Learning Systems for Automatic Control," *IEEE Transactions on Automatic Control*, Vol. AC-11, No. 1, January, 1966.

[20] Tsypkin, Y., *Foundations of the Theory of Learning Systems*, Academic Press, 1973.

[21] Waltz, M. & Fu, K., "A Heuristic Approach to Reinforcement Learning Control Systems," *IEEE Transactions on Automatic Control*, Vol. AC-10, No. 4, October, 1965.