As an example, the approach taken in [2] was determined by the target audience, embedded systems programmers who lack a background in signals and systems. The text is informal, stressing intuition over the theoretical, and quite non-mathematical given the subject. (In fact, only one equation in the book explicitly involves an integral, to note but one heretical element.) Nonetheless, an effort is made to provide enough information to design and use digital filters in real applications, as well as interpret the FFT and use other DSP tools.Since the coverage of fundamental material such as linear signals and systems is limited in depth, the "obvious" connections among concepts (such as the relationship between convolution and FIR filters mentioned earlier) must be made explicitly. The emphasis on reaching a practical implementation leads, in some cases, to material that is usually postponed to later DSP courses (e.g., scaling IIR filter coefficients). This skewing—a kind of "subject warping"—has parallels with the statistics or electronics courses discussed above; subjects may be compressed or expanded greatly with respect to the usual progression, generally in favor of the practical, and at the expense of the theoretical.

Cross-fertilization of ideas from books and courses aimed at both non-EEs and beginning EEs are likely, and will lead to approaches that better address the needs of all DSP learners without a traditional EE background. Courses like the basic statistics and electronics courses mentioned above are also a source of useful general principles. Given the diversity of backgrounds and needs, several different approaches to "just DSP" may eventually co-exist, with varying content, mathematical rigor, and formality. For example, biomedical engineers may benefit from a different approach than, say, computer scientists. The books and courses now available suggest that it is at least possible to construct a coherent presentation of DSP topics with a minimum of prerequisites; it is now a matter of taking this content and packaging it in the most effective way (or ways).

To this novice educator, the trends point to a future in which fundamental DSP is made available to a wider range of professionals and students. Not that the world is made up of embedded systems programmers, or that librarians worldwide will be applying the FFT to their data every day, but there is a role for DSP education of people who are not traditional EEs. Even if teaching non-EEs reveals little about how to teach EEs—though it likely will—this task is consistent with the obligation of the engineer to develop and make available tools and insights into the physical world. Not only that, it may legitimately be demanded of us.

References

[1] C. Foster, *Real Time Programming---Neglected Topics*, Reading, MA: Addison-Wesley, 1981.

[2] D. Grover, and J.R. Deller, Jr., *Digital Signal Processing and the Microcontroller*, Upper Saddle River, NJ: Prentice Hall, 1999.

[3] P. Horowitz, and W. Hill, *The Art of Electronics* (2nd edition), New York: Cambridge University Press, 1989.

[4] R.G. Lyons, *Understanding Digital Signal Processing*, Reading MA: Addison Wesley Longman, 1997.

[5] J.H. McClellan, R.W. Schafer, and M.A. Yoder, *DSP First: A Multimedia Approach*, Upper Saddle River, NJ: Prentice Hall, 1997.

Lessons on Adaptive Systems for Signal Processing, Communications, and Control

Simon Haykin

McMaster University

In nearly every undergraduate electrical engineering (EE) program, the properties of linearity, Gaussianity, and stationarity are emphasized whenever the issue of statistical signal processing or statistical communication theory is discussed. These assumptions are made for the sake of deriving model-based (parametric) algorithms in a mathematically tractable manner. Yet, many of the information-bearing signals (e.g., speech, radar, and sonar signals) encountered in applications are nonstationary, and non-Gaussian [12]. The traditional approach to statistical signal processing was justified in the pre-computer age when the best we could do was to rely almost exclusively on the use of idealized mathematical *models*. However, with ever-increasingly powerful computers at the disposal of our undergraduate students, the time is right for us to find a place in our undergraduate electrical engineering programs for an innovative course that is not limited by the assumptions of linearity, Gaussianity, and stationarity. Indeed, we owe it to our students to introduce them to the practical realities of nonlinearity, *non*-Gaussianity, and nonstationarity in an integrated fashion. The perfect candidate for addressing the limitations of our current undergraduate EE programs is adaptive systems.

Adaptive filtering is recognized as a core of digital signal processing [14]. Most importantly, adaptive filtering or identification does not require knowledge of the underlying statistics of the input data. Indeed, it is for this reason that the need for adaptivity arises in the design and application of antenna systems, speech- and video-coding systems, communication systems, radar and sonar systems, control systems, biomedical engineering, seismology, and so on [13], [20], [31]. All of these facts, put together, make a very compelling case for the introduction of a course on adaptive systems in the final year of undergraduate EE programs.

In this article, we describe a series of lessons on adaptive systems, each of which highlights a specific aspect of this theoretically fascinating and practically important course. Table 1 summarizes the list of lessons proposed for such a new course. The material presented herein is written with the educator of adaptive systems in mind. Much, if not all, of this material also has standalone value for the prospective student of the course.

Table 1. Summary of Lessons on Adaptive Systems.*	
Lesson 1	Why adaptive systems?
Lesson 2	Unconstrained optimization: the method of steepest descent
Lesson 3	Wiener filter Application: Coherent sidelobe cancellation
Lesson 4	Least-mean-square (LMS) algorithm: Applications: Linear prediction Channel equalization Noise cancellation
Lesson 5	Adaptation in the frequency domain
Lesson 6	Back-propagation learning (static) Application: Pattern recognition
Lesson 7	Back-propagation learning (dynamic) Application: Nonlinear system identification
Lesson 8	Adaptive systems are function approximators
*Naturally, the lessons summarized herein occupy different numbers of hours of presentation.	

Lesson 1: Why Adaptive Systems?

Adaptability is defined as "the ability of a system to cope with unexpected disturbances of the environment." The process by which this capability manifests itself is referred to as adaptation, and the system itself is said to adapt to its environment or just said to be adaptive.

In a biological context, the adaptability of biological systems is one of nature's most striking properties [6]. Indeed, biological systems provide well-proven examples of effectively coping with nonstationary environments. A nonstationary environment is one whose characteristics vary with time. The relevant point to note here is that animals are capable of adapting to changes in their environment through a learning process rather than a model-based approach.

The computing power for the biological form of adaptation resides in the brain. The human brain is a highly complex, nonlinear, and parallel computer (information-processing system), which operates in an entirely different way from the conventional von Neumann computer. Most importantly, the brain has the capability to organize its structural constituents, known as neurons, to perform certain tasks, namely, pattern recognition, pattern association, perception, and motor control, many times faster than the fastest digital computer in existence today. The brain can accomplish these tasks through a learning process that is performed in one of two fundamentally different ways: supervised and unsupervised. This is the same with artificial neural networks.

In an artificial neural network, supervised learning presupposes the availability of a "teacher" that oversees the learning process by presenting a desired response to the neural network for every input signal. In effect, the neural network learns about an unknown environment in which it is embedded through a set of examples, with each example consisting of an input signal and a desired response. The information contained in such a set of examples is transferred to the neural network and stored in a set of adjustable coefficients called synaptic weights or just weights. In unsupervised learning, on the other hand, the learning process, (i.e., weight adjustments) proceeds without a teacher.

From this discussion, it is apparent that the processes of adaptation and learning are intimately related. In this article, we confine ourselves to adaptive systems that operate through supervised learning.

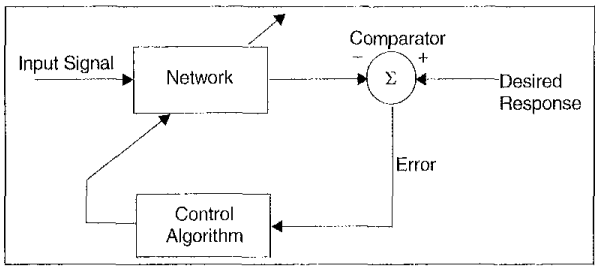
From an engineering perspective, an adaptive system consists of three functional blocks, as depicted in Fig. 1:

- A network consisting of a set of adjustable weights.
- A comparator for measuring the difference or error between the output of the network due to an input signal and the corresponding desired response.
- A control algorithm for adjusting the weights of the network in response to the error signal.

The type of supervised learning described herein is recognized as error-correction learning, whose goal is to optimize the adjustments made to the network's weights so as to minimize a prescribed cost function. This formulation sets the stage for the next couple of lessons.

Lesson 2: Unconstrained Optimization

Consider a set of input-output pairs of data (examples) described by the mapping function



▲ 1. Block diagram of adaptive system.

$$d_i = f(\mathbf{x}_i), \qquad i = 1,2,\cdots,N, \tag{1}$$

where the function $f(\cdot)$ is unknown. To simplify the presentation without loss of generality, this unknown function is assumed to be real-valued scalar, but the input $\mathbf{x}_i$ may be vector valued. The requirement is to approximate the unknown function $f(\cdot)$ by a function $F(\cdot,\mathbf{w})$ in an optimum manner. The function $F(\cdot,\mathbf{w})$ pertains to a network of known structure, which is characterized by a set of p adjustable weights denoted by the vector $\mathbf{w}$. As an index of performance, define the cost function

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(d_i - F(\mathbf{x}_i,\mathbf{w}))^2, \tag{2}$$

which may be viewed as the total error energy. A class of unconstrained optimization techniques, exemplified by the method of steepest descent, is particularly well suited for finding an optimum value of the weight vector $\mathbf{w}$ that minimizes the cost function $\mathcal{E}(\mathbf{w})$. Specifically, starting with an initial guess denoted by $\mathbf{w}(0)$, we generate a sequence of weight vectors $\mathbf{w}(1),\mathbf{w}(2),\cdots$, in accordance with the following recursion [2]

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta\mathbf{g}(n), \quad n = 0,1,2,\cdots, \tag{3}$$

where η is a positive constant called the step-size or learning-rate parameter, and $\mathbf{g}(n)$ is the gradient vector of the cost function $\mathcal{E}(\mathbf{w})$, evaluated at the point $\mathbf{w}(n)$:

$$\mathbf{g}(n) = \left.\frac{\partial\mathcal{E}(\mathbf{w})}{\partial\mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}(n)} = \sum_{i=1}^{n}(d_i - F(\mathbf{x}_i,\mathbf{w}))\left.\frac{\partial F(\mathbf{x}_i,\mathbf{w})}{\partial\mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}(n)}. \tag{4}$$

If η is small enough, the method of steepest descent converges to a stationary point of the cost function $\mathcal{E}(\mathbf{w})$ at which point $\mathbf{g}(n) = 0$. This stationary point can be a local minimum, for example.

The method of steepest descent described in (3) is a recursive system whose behavior depends on the value assigned to the step-size parameter η:

▲ 1. When η is small compared to a certain critical value η_{crit}, the trajectory traced by the weight vector $\mathbf{w}(n)$ for increasing number of iterations, n, is overdamped.

▲ 2. When η is allowed to approach (but remain less than) the critical value η_{crit}, the trajectory is oscillatory or underdamped.

▲ 3. When η exceeds η_{crit}, the trajectory becomes unstable.

Conditions 1 and 2 correspond to a convergent or stable system, whereas condition 3 corresponds to a divergent or unstable system. These conditions are demonstrated for the two-dimensional case by plotting $w_1(n)$ versus $w_2(n)$, elements of $\mathbf{w}(n)$, for increasing n and different values of η [13].

Depending on whether the approximating function $F(\cdot,\mathbf{w})$ is linear or nonlinear in $\mathbf{w}$, we may classify adaptive systems as linear or nonlinear, respectively. The distinguishing feature here is that it is possible for a linear adaptive system to find a global minimum of the cost function $\mathcal{E}(\mathbf{w})$, whereas in a nonlinear adaptive system, we can have cases of global or local minima. Both linear and nonlinear adaptive systems are considered in this article. We begin with the linear case.

Lesson 3: The Wiener Filter

Suppose the network in Fig. 1 is modeled as a finite-duration impulse response (FIR) filter, in which case, the function $F(\mathbf{x},\mathbf{w})$ is simply the inner product of the input vector $\mathbf{x}$ and weight vector $\mathbf{w}$, that is, $F(\mathbf{x},\mathbf{w}) = \mathbf{w}^T\mathbf{x}$. The cost function $\mathcal{E}(\mathbf{w})$ is now a quadratic function of $\mathbf{w}$, with a well-defined minimum, at which point, the weight vector attains its optimum value. To find this optimum value, we first differentiate $F(\mathbf{x},\mathbf{w})$ with respect to $\mathbf{w}$, obtaining

$$\frac{\partial F(\mathbf{x},\mathbf{w})}{\partial\mathbf{w}} = \mathbf{x}. \tag{5}$$

Correspondingly, the gradient vector for a set of N examples takes the form

$$\mathbf{g} = -\sum_{i=1}^{N}(d_i - \mathbf{w}^T\mathbf{x}_i)\mathbf{x}_i. \tag{6}$$

Setting $\mathbf{g}$ equal to the null vector, recognizing that $\mathbf{w}^T\mathbf{x}_i = \mathbf{x}_i^T\mathbf{w}$, and solving for $\mathbf{w}$, we obtain

$$\hat{\mathbf{w}} = \left(\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T\right)^{-1}\left(\sum_{i=1}^{N}\mathbf{x}_i d_i\right), \tag{7}$$

where $\mathbf{x}_i\mathbf{x}_i^T$ is the outer product of $\mathbf{x}_i$ with itself. At this point in the discussion, it is convenient to introduce the following two definitions:

▲ 1. Sample correlation matrix of the input vector $\mathbf{x}$:

$$\hat{\mathbf{R}}_x = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T. \tag{8}$$

▲ 2. Sample cross-correlation vector between the input vector $\mathbf{x}$ and desired response d:

$$\hat{\mathbf{r}}_{xd} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i d_i . \tag{9}$$

Correspondingly, we may rewrite (7) as

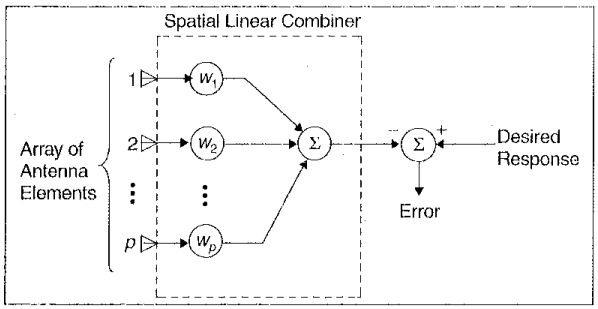
$$\hat{\mathbf{w}} = \hat{\mathbf{R}}_x^{-1}\hat{\mathbf{r}}_{xd} . \tag{10}$$

The formula of (10), based on a single snapshot of input data, is the least-squares (LS) estimation algorithm [13], also known as the direct matrix inversion algorithm [21]; its computational complexity grows exponentially as p^3, where p is the dimension of $\mathbf{w}$. If desired, the algorithm can be implemented in a recursive fashion, wherein exponential weighting of the data is used to control the memory length [3], [5]; computational complexity of the resulting algorithm, known as the recursive least-squares (RLS) algorithm, grows as p^2.

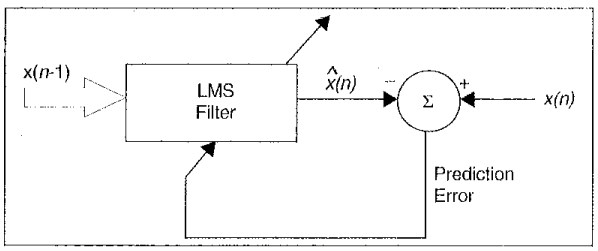
From (8) to (10), we readily see that the estimated weight vector $\hat{\mathbf{w}}$ is a function of the sample size N. When the input vector $\mathbf{x}$ and desired response d are jointly ergodic, the sample correlation matrix $\hat{\mathbf{R}}_x$ and sample cross-correlation vector $\hat{\mathbf{r}}_{xd}$ approach constant values denoted by $\mathbf{R}_x$ and $\mathbf{r}_{xd}$, respectively, as we let N approach infinity [19]. In a corresponding way, as N approaches infinity the estimated weight vector $\hat{\mathbf{w}}$ approaches a constant value defined by

$$\mathbf{w}_{opt} = \lim_{N\to\infty} \hat{\mathbf{w}} = \mathbf{R}_x^{-1}\mathbf{r}_{xd} . \tag{11}$$

The optimum linear filter defined by (11) is called the Wiener filter in recognition of the pioneering work done by Norbert Wiener in the 1940s on the "filtering" problem of a process of interest corrupted by additive noise [30]. The Wiener filter provides the optimum framework for assessing the performance of linear adaptive systems.



▲ 2. Block diagram of array antenna system.



▲ 3. Adaptive linear predictor.

An important property of the Wiener filter is that it satisfies the principle of orthogonality. When the cost function $\mathcal{E}(\mathbf{w})$ attains its minimum value, the estimation error, defined as the difference between the desired response d and actual response of the filter, is orthogonal to each element of the input vector $\mathbf{x}$ that enters into the estimation of the desired response [13].

Coherent Sidelobe Cancellation

The LS estimation algorithm finds application in the design of adaptive array antennas for radar and communications due to its rapid convergence property [5], [21], [32]. In such applications, we determine an estimate of the weight vector $\mathbf{w}_{opt}$ by using (10), where the estimates $\hat{\mathbf{R}}_x$ and $\hat{\mathbf{r}}_{xd}$ are themselves computed by using (8) and (9), respectively. To characterize the performance of the adaptive antenna so designed, consider the system depicted in Fig. 2. For example, the desired response $d(n)$ may represent the output of a reference antenna in the form of a high-gain antenna pointed in the direction of a desired signal source. In such a setting, the antenna system of Fig. 3 operates as a coherent sidelobe canceller (CSLC), for which the Wiener solution minimizes the interference power component of the array antenna output [21]. To characterize the transient response of the antenna system of Fig. 2, we may compute the output error power and plot it as a function of the number of data samples, N. The results of this investigation show that the output error power is within 3 dB of the Wiener solution after only $N = 2p$ distinct time samples, and within 1 dB of the Wiener solution after $N = 5p$ samples, where p is the number of antenna elements (excluding the reference antenna). These results demonstrate the rapid convergence property of the LS estimation algorithm, independent of the operating environment or array configuration [21].

Lesson 4: Least-Mean-Square Algorithm

As mentioned in Lesson 3, the computational complexity of the LS estimation algorithm grows exponentially with the dimension p of the weight vector $\mathbf{w}$. To overcome this computational difficulty, we may combine two ideas:

▲ 1. Use the method of steepest descent for a recursive computation of the weight vector $\mathbf{w}$, as described in Lesson 2.

▲ 2. Simplify the computation by using an instantaneous estimate of the gradient vector, defined by $-e_n\mathbf{x}_n$, where e_n is the estimation error:

Subsequently, we obtain the least-mean-square (LMS) algorithm, summarized in Table 2. The simplicity of the LMS algorithm is clearly evident from this table, hence its popularity in the design of linear adaptive systems. Moreover, the LMS algorithm is robust, which means that a small uncertainty and small disturbance (i.e., disturbance with small energy) can only result in small estimation errors [11].

Table 2. Summary of the LMS Algorithm.

Set $\mathbf{w}(0) = \mathbf{0}$

Compute for $n = 0,1,2,\cdots$

$$e_n = d_n - \hat{\mathbf{w}}^T(n)\mathbf{x}_n$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta e_n \mathbf{x}_n$$

Despite the computational simplicity of the LMS algorithm, its convergence (stability) analysis remains a research topic to this day [14]. Several useful results and methods of analysis exist for the case of sufficiently small step-sizes [13], [31].

The LMS algorithm just described is the standard form of the algorithm. Other variants of the algorithm include the normalized LMS algorithm [13] and leaky LMS algorithm [31]. In one form or another, the LMS algorithm has been successfully applied to solve highly different signal-processing tasks exemplified by the following:

Linear Prediction

Prediction is basic to model building [13]. Assuming that the source responsible for generating the input vector $\mathbf{x}(n)$ admits a linear autoregressive model (i.e., a model whose output regresses on a set of its past values), then the LMS algorithm provides a simple and yet effective method for estimating the model coefficients. In such a framework, the set of past values $x_{n-1}, x_{n-2}, \cdots, x_{n-p}$ provides the input to the LMS algorithm and the present value x_n serves as the desired response as illustrated in Fig. 3; p denotes the autoregressive model order.

Channel Equalization

A voiceband modem (modulator/demodulator) is a device that makes it possible to transmit digital data over a telephone channel. Typically, the statistical characteristics of the channel are unknown, requiring the use of an equalizer to compensate for distortion produced by the channel [26]. In such an application, supervised training of the adaptive filter proceeds by using a pseudo-noise (PN) sequence [26] as the channel input. An adaptive filter, operated using the LMS algorithm, is connected in cascade with the channel. The desired response for the algorithm in the receiver is provided by a second PN sequence generator synchronized with the PN sequence generator used in the transmitter, as illustrated in Fig. 4. Once the training of the equalizer is completed, the modem is ready for the transmission of information-bearing data.

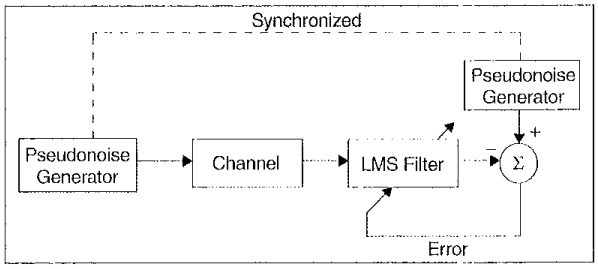
Noise Cancellation

As the name implies, an adaptive noise canceller subtracts an estimate of the noise from the received signal, with the whole operation being controlled in an adaptive manner for the purpose of improved signal-to-noise ratio. Ordinarily, it is inadvisable to subtract noise from a received signal because such an operation could produce disastrous results by causing an increase in the average noise power. For an adaptive system, however, it is precisely this increase in noise power that can be exploited to remove the unwanted noise. Consequently, when the noise cancellation is performed adaptively in a proper fashion, it is possible to achieve a superior performance compared to direct filtering of the received signal [31]. Basically, an adaptive noise canceller is a dual-input, closed-loop, adaptive feedback system as illustrated in Fig. 5. The two inputs of the system are derived from a pair of sensors: a primary sensor and reference sensor. The primary sensor receives the information-bearing signal corrupted by additive noise, whereas the reference sensor receives a correlated version of the additive noise. In effect, the noisy received signal acts as the desired response, and the adaptive filter operates on the noise canceller's output (representing error) so as to produce an estimate of the additive noise component in the received signal. The canceller's output provides a less noisy form of the information-bearing signal than that available at the canceller's input.

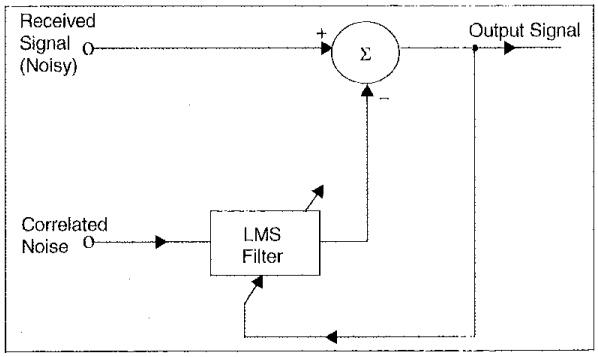
Adaptive noise cancellation, in one form or another, finds application in echo cancellation [31], active noise control [4], [17], and antenna sidelobe cancellation [5], [21], [31].

Lesson 5: Adaptation in the Frequency Domain

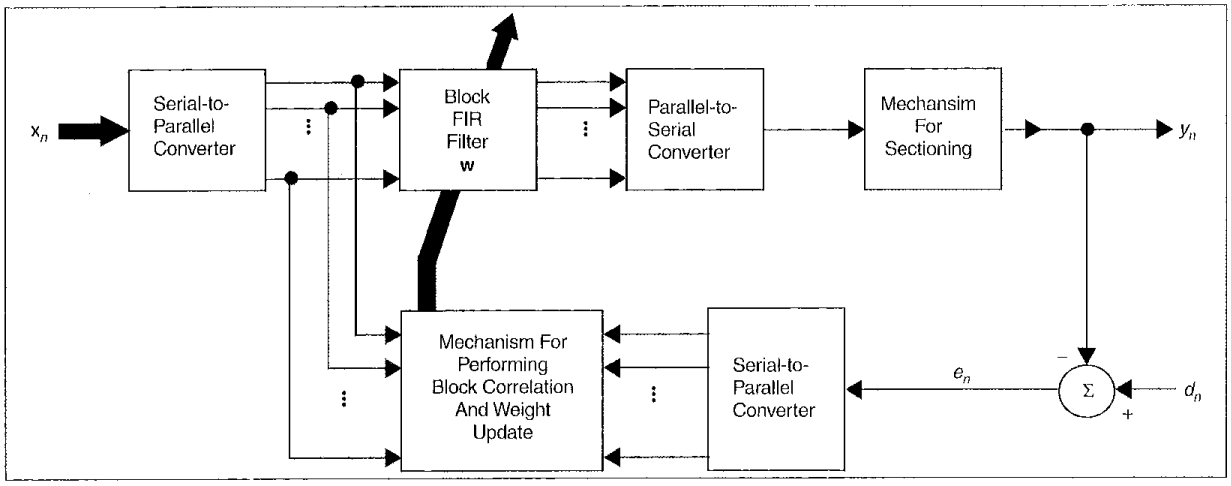
In the LMS algorithm described in Lesson 4, adaptation of the weight vector $\mathbf{w}$ is performed in the discrete-time domain. Recognizing that the discrete Fourier transform (DFT) maps the discrete-time domain into the discrete-frequency domain [23], it is equally feasible to perform adaptation of the weight vector in the discrete-frequency domain. In this new setting, we speak of frequency-domain adaptive systems [7], [13].



▲ *4. Adaptive equalizer.*



▲ *5. Adaptive noise canceller.*



▲ 6. *Block-adaptive filter.*

There are two main reasons for seeking the use of frequency-domain adaptive systems:

▲ 1. In certain applications, such as acoustic echo cancellation in teleconferencing, for example, the adaptive filter is required to have a long impulse response (i.e., long memory) to cope with an equally long echo duration. When the LMS algorithm is adapted in the time domain, the requirement of a long memory results in a significant increase in the algorithm's computational complexity. This difficulty could be alleviated by combining two complementary methods widely used in digital signal processing: block implementation of a FIR filter, and the fast Fourier transform (FFT) algorithm for performing the convolution (i.e., filtering) operation.

▲ 2. Frequency-domain adaptive filtering is used to improve the convergence performance of the standard LMS algorithm by exploiting the orthogonality properties of the DFT.

Figure 6 shows the layout of a block frequency-domain adaptive system. The incoming data sequence is sectioned into L-point blocks, say, by means of a serial-to-parallel converter, and the blocks of input data produced are applied to a FIR filter of length p, one block at a time. The tap weights of the filter are held fixed over each block of data, so that adaptation of the filter proceeds on a block-by-block basis rather than on a sample-by-sample basis as in the standard LMS algorithm. The parallel-to-serial converter followed by a sectioning mechanism allows the computation of the filter output. The filter output y_n produced is compared against the corresponding value of the desired response d_n to produce the error e_n, which varies at the sampling rate as in the standard LMS algorithm. In the feedback path of the block adaptive system, the error sequence is sectioned into L-point blocks in a synchronous manner at the input end of the system, and then used to compute the adjustments to the tap weights of the filter as depicted in Fig. 7. The computation of these adjustments follows a rule similar to that in the standard LMS algorithm, as described here:

$$\begin{pmatrix}\text{Adjustment to the}\\ \text{weight vector}\end{pmatrix} = \begin{pmatrix}\text{Step - size}\\ \text{parameter}\end{pmatrix}\begin{pmatrix}\text{tap - input}\\ \text{vector}\end{pmatrix}(\text{error signal}).$$

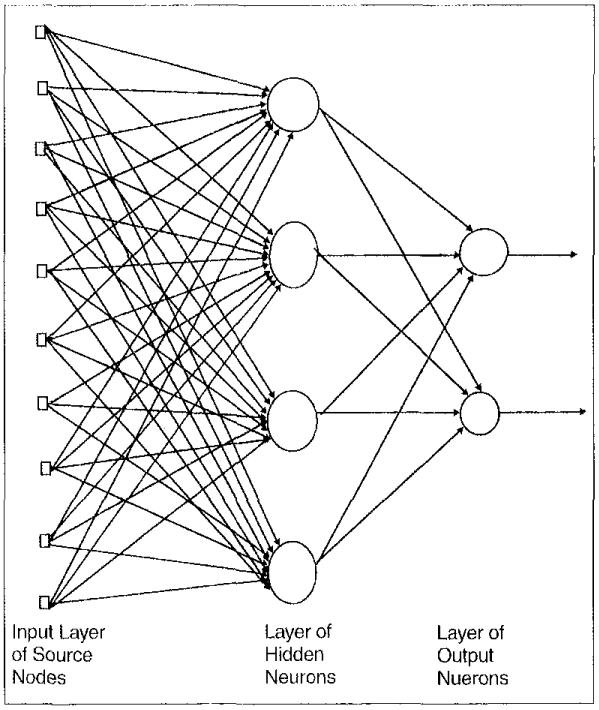
Recognizing that in the block LMS algorithm, the error signal varies at the sampling rate, it follows that for each block of data, we have different values of the error signal for use in the adaptive process.

An important issue that needs to be considered in the design of a block adaptive filter is how to choose the block size L. When L is greater than p, redundant operations are performed in the adaptive process, because the estimation of the gradient vector (computed over L points) uses more information than the filter itself. On the other hand, when L is less than p, some of the tap weights in the filter are wasted, because the sequence of tap inputs is not long enough to feed the whole filter. Thus, the most practical choice is $L = p$.

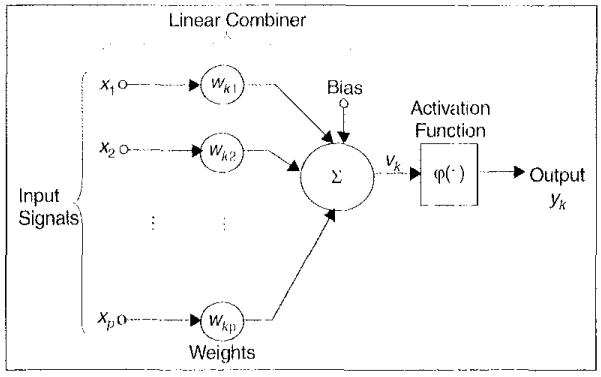
Lesson 6: Backpropagation Learning (Static)

The LMS algorithm (discussed in Lessons 4 and 5) is linear, which confines its suitability to environments that are governed by linear processes. However, when the requirement is governed by nonlinear processes, we should likewise resort to the use of nonlinear adaptive systems. With the re-emergence of (artificial) neural networks in the mid-1980s, we now have powerful tools at our disposal for the design of nonlinear adaptive systems. This lesson studies a highly popular algorithm known as back-propagation (BP) learning [15], [27], [28], which may be viewed as a generalized form of the equally popular LMS algorithm.

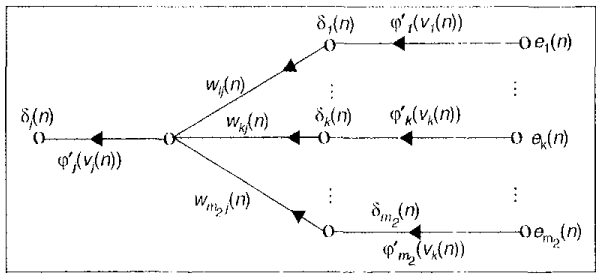
The LMS algorithm pertains to a special structure in the form of a FIR filter, whereas the BP algorithm pertains to a distributed, feedforward, nonlinear system called a multilayer perceptron (MLP). The MLP consists of an input layer, one or more layers of hidden neurons



▲ 7. *Fully connected feedforward network with one hidden layer and one output layer.*



▲ 8. *Nonlinear model of a neuron labeled as neuron k.*



▲ 9. *Signal-flow graph pertaining to a backpropagation of error signals for hidden neuron j as described in (14) and (16); the subscript m_2 denotes the number of output neurons.*

(processing units) and an output layer of neurons. Such a structure is illustrated in Fig. 7, for the case of one hidden layer and two output neurons. Fig. 8 shows the most commonly used model of a neuron. The model consists of a linear combiner followed by a nonlinear activation function $\varphi(\cdot)$. The activation function can be nonsymmetric, as in the case of the logistic function

$$\varphi(v) = \frac{1}{1+e^{-v}} \tag{12}$$

or antisymmetric about the origin, as in the case of the hyperbolic tangent function

$$\varphi(v) = \tanh\left(\frac{v}{2}\right) = \frac{1-e^{-v}}{1+e^{-v}}. \tag{13}$$

As with the LMS algorithm, the objective of BP learning is to minimize a cost function similar to that defined in (2). In such a setting, the adjustments applied to the weights of the output neuron(s) in the MLP follow exactly the same rule as the LMS algorithm. However, when we go on to consider hidden neurons, we run into a new situation. Unlike the output neurons that are supplied with individual desired responses of their own, the hidden neurons do not have such a provision as they are not directly accessible. Nevertheless, the hidden neurons share responsibility for any error made at the output of the MLP. The fundamental question is to know how to penalize or reward hidden neurons for their share of the responsibility. This problem is the credit-assignment problem, which is solved in a highly elegant fashion by back-propagating error terms through the network, hence the name of the algorithm.

In its most basic form, the adjustment $\Delta w_{ji}(n)$ applied to the weight connecting neuron i to neuron j at iteration (time step), n, of the algorithm is defined by the delta rule [15]

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n), \tag{14}$$

where η is, as before, the learning-rate parameter, $y_i(n)$ is the input signal of neuron i (i.e., the output signal supplied by neuron j to neuron i), and $\delta_j(n)$ is the local gradient of neuron j defined by

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_{k=1}^{m_2} \delta_k(n) w_{kj}(n), \quad \text{neuron } j \text{ is hidden}, \tag{15}$$

where the prime denotes differentiation and m_2 is the number of neurons in the output layer. Fig. 9 shows a signal-flow graph representation of (15). If neuron k is an output neuron, we simply have

$$\delta_k(n) = \varphi_k'(v_k(n)) e_k(n), \quad k = 1,2,\cdots m_2. \tag{16}$$

Thus, starting from the output neurons, we may extend the use of (15) to deal with any number of hidden layers.

The BP learning process described herein proceeds in a serial manner (also referred to as the on-line or stochastic mode). Specifically, an example is picked from the training sample at random, for which the network is operated first in the forward direction and then the backward direc-

tion. In the forward direction, the weights of the network are all fixed and the input signal of the example in question is propagated through the MLP layer by layer, until it reaches the output layer. Given the desired response corresponding to the input signal of that example, the errors in the output layer are computed and then propagated backward through the network in accordance with (15) and (16). It is during this backward phase that adjustments are applied to the weights of the MLP in accordance with (14). The forward and backward computations are performed for each example in the training set, on the completion of which we say that the training of the MLP has been completed for one epoch. Next, the examples in the training set are shuffled in a random manner, and the whole set of computations is repeated for another epoch. The training is continued until the network reaches a reasonably stable condition, whereupon the training is terminated. However, care must be exercised in how long the training is performed in order ensure that the network generalizes on test data not seen before [15], [27].

With this background on back-propagation learning, we are ready to consider an important application for which it is very well-suited.

Pattern Recognition

Pattern recognition is formally defined as the process whereby a received pattern/signal is assigned to one of a prescribed number of classes (categories). A neural network performs pattern recognition by first undergoing a training session, during which the network is repeatedly presented a set of input patterns along with a category assignment. Later, a new pattern (not seen before) is presented to the network, but it still belongs to the same population used to train the network. The network can identify the class of that pattern due to the information it has extracted from the training data. Pattern recognition is basically statistical in nature, with the patterns being represented by points in a multidimensional data space. The space is divided into regions, with each one associated with a class. The decision boundaries (between classes) formed are determined by the training process. The construction of these boundaries is made statistical by the inherent variability that exists within and between classes.

Typically, a pattern recognition machine consists of a feature extractor (detector) followed by a pattern classifier [9], [10], as illustrated in Fig. 10. In the case of a multilayer perceptron trained with the BP algorithm, the hidden neurons act as feature detectors. As the learning process progresses, the hidden neurons begin to gradually "discover" the salient features that characterize the training data. They do so by performing a nonlinear transformation on the input data into a new space called the feature space. In other words, the operations of feature extraction and pattern classification of a traditional pattern recognition machine are combined in a single network, namely, the multilayer perceptron.

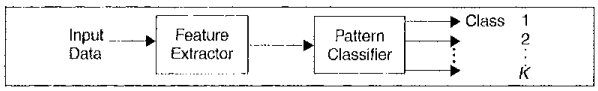
In [18], a novel form of multilayer perceptron known as a convolutional network is described for image processing tasks like handwriting recognition. This network is specifically designed to recognize two-dimensional shapes with a high degree of invariance to translation, scaling, sharing, and other forms of distortion. The development of convolutional networks is neurobiologically motivated by the pioneering work of Hubel and Wiesel described in [1] and [3].

Lesson 7: Back-Propagation Learning (Dynamic)

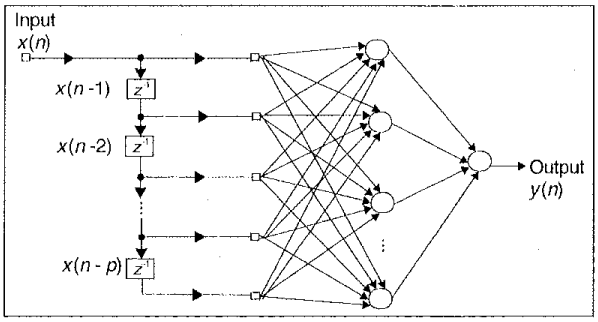
The back-propagation learning process described in Lesson 6 is static in nature in that the spatial information contained in the set of training examples has no dependence on time. In Lesson 7, we describe back-propagation learning for temporal processing. The lesson builds on the following important theorem [15], [29]: Any shift-invariant "myopic" dynamic map (i.e., a map with uniform fading memory) can be approximated arbitrarily well by a structure consisting of two functional blocks: a bank of linear filters feeding a static neural network. The linear filters manage dependence of the input signal on time, and the static neural network handles nonlinear processing. For example, the approximating dynamic map may consist of a tapped-delay line feeding a multilayer perceptron (trained using the back-propagation algorithm), as illustrated in Fig. 11. In such a setting, the tapped-delay line serves the function of short-term memory, and the multilayer perceptron serves the function of long-term memory. The network of Fig. 11 is referred to as a focused time-lagged feedforward network, focused in the sense that the short-term memory is contained entirely in the front end of the network.

Nonlinear System Identification

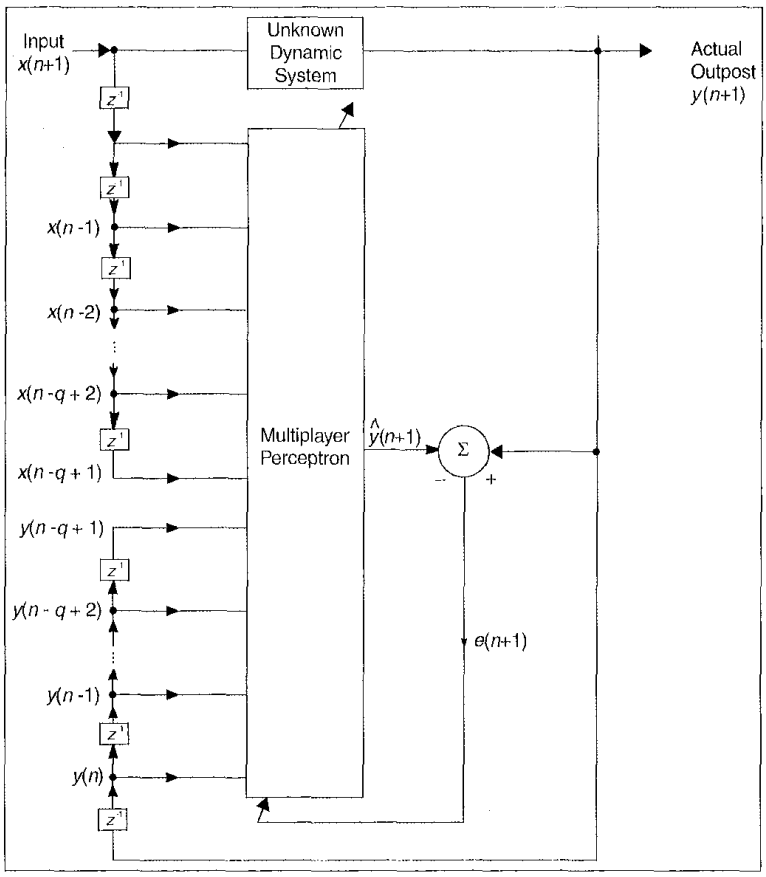
Consider, for example, the problem of having to identify an unknown dynamic system, given only a set of input-output data denoted by $\{x(n+1), y(n+1)\}$, where n denotes discrete-time. For the identification



▲ *10. Block diagram of pattern recognition machine.*



▲ *11. Focused time-lagged feedforward network (TLFN); the bias levels have been omitted for convenience of presentation.*



▲ *12. Nonlinear autoregressive with exogenous inputs (NARX) model; without loss of generality, it is assumed that the numbers of delayed inputs and delayed outputs fed into the multilayer perceptron are equal.*

model, we may use the nonlinear autoregressive with exogenous inputs (NARX) model [15], [22], as shown in Fig. 12. The model output $\hat{y}(n+1)$ represents an estimate of the actual output $y(n+1)$. The estimate $\hat{y}(n+1)$ is subtracted from $y(n+1)$ to produce the error signal

$$e(n+1) = y(n+1) - \hat{y}(n+1),$$

where $y(n+1)$ plays the role of desired response. The error $e(n+1)$ is used to adjust the weights of the multilayer perceptron using a version of the back-propagation algorithm, known as back-propagation through time, to minimize the error in a mean-square error sense. A representation of the unknown system is thereby captured by the NARX model.

Lesson 8: Adaptive Systems Are Function Approximators

The applications of adaptive systems to pattern recognition and system identification, discussed in Lessons 6 and 7, are quite different in nature. Nevertheless, both of these applications exploit a fundamental property of adaptive systems, namely, function approximation. To be more precise, multilayer perceptrons are universal approximators in that, in theory, any nonlinear input-output mapping can be approximated by a multilayer perceptron with a single hidden layer to any desired degree of accuracy, provided that the mapping is continuously differentiable and the hidden layer has enough processing units [18], [15]. For practical reasons, however, we may use a multilayer perceptron with two or more hidden layers, depending on the complexity of the learning task of interest.

In Lesson 8, the course is thus brought to an end by emphasizing the computing power of adaptive systems.

Concluding Remarks

In this article, we have described a series of eight lessons on linear and nonlinear adaptive systems, which provide a framework of a new undergraduate course in EE. A summary of the lessons, including applications, is presented in Table 1.

Lesson 1 is a motivational lesson. Lesson 2 explains the method of steepest descent, which sets the stage for unconstrained optimization of adaptive systems. Lesson 3 describes the Wiener filter, which provides the optimum framework for assessing the performance of linear adaptive systems. Lesson 4 illustrates the least-mean-square (LMS) algorithm, known for its simplicity and robustness. Lesson 5 discusses the idea of linear adaptation in the frequency domain. Lessons 6 and 7 study static and dynamic forms of back-propagation learning for the design of multilayer perceptrons, which are particularly suited for the design of nonlinear adaptive systems. Lessons 3 to 7 also include discussions of representative examples of the linear and nonlinear adaptive systems described therein. Finally, Lesson 8 emphasizes the computing power of adaptive systems.

Bearing in mind the highly mature status of adaptive systems in a theoretical context, the very pervasive scope of their well-proven practical applications, and the wide-scale availability of powerful computers, the time is indeed right for the introduction of an undergraduate course on adaptive systems. In doing so, we will have made room for a truly innovative course that presents a more realistic picture of statistical signal processing than is presently available in undergraduate EE programs.

The fundamental concepts required to understand adaptive systems are not inherently complex, and deserve to be treated as part of the basic undergraduate curricu-

lum. The prerequisite for offering a course on adaptive systems is already in place, recognizing that most undergraduate programs in EE include an introductory course on signals and systems [16]. All that is needed is the vision and willpower to make it happen.

It is noteworthy that at the University of Florida at Gainesville, a new undergraduate course has been introduced where advanced concepts of adaptive systems are taught to students by blending computer simulations with an equation-based approach. A hypertext document has been integrated with a software simulator, which is called an interactive electronic book (i-book) [24], [25]. During every lecture, students have access to the i-book to reinforce relevant concepts with the behavior of an adaptive system simulator. Students learn the material in the context of a particular topic, using real data obtained from the web.

Acknowledgments

The author is grateful to Sue Becker, McMaster University, Hamilton, Ontario; Jose Principe, University of Florida at Gainesville; Ali H. Sayed, UCLA; and Jim Zeidler, University of California at San Diego. I am indebted to them for their careful reviews of this article at very short notice. The comments made by Jack Deller, Michigan State University, are also appreciated. The title for this article and its layout were inspired by Jerry Mendel's graduate textbook [20].

References

[1] J.A. Anderson, *An Introduction to Neural Networks*, MIT Press, 1995.

[2] D.P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1995.

[3] P.S. Churchland and T.J. Sejnowski, *The Computational Brain*, MIT Press, 1992.

[4] R.L. Clark, W.R. Saunders, and G.P. Gibbs, *Adaptive Structures: Dynamics and Control*, Wiley, 1998.

[5] R.T. Compton, Jr., *Adaptive Antennas: Concepts and Performance*, Prentice-Hall, 1988.

[6] M. Conrad, *Adaptibility: The Significance of Variability from Molecule to Ecosystem*, Plenum, 1983.

[7] C.F.N. Cowan and P.M. Grant, *Adaptive Filters*, Prentice-Hall, 1985.

[8] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303-314, 1989.

[9] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley, 1973.

[10] K. Fukanaga, *Statistical Pattern Recognition*, 2nd edition, Academic Press, 1996.

[11] B. Hassibi, A.H. Sayed, and T. Kailath, "Infinite-Quadratic Estimation and Control: A Unified Approach to H^2 and H^∞ Theories" *SIAM Studies in Applied and Numerical Mathematics*, PA, 1999.

[12] S. Haykin, "Neural Networks Expand SP's Horizons," *IEEE Signal Processing Magazine*, vol. 13, no. 2, pp. 24-29, 1996.

[13] S. Haykin, *Adaptive Filter Theory*, 3rd edition, Prentice-Hall, 1996.

[14] S. Haykin, "Adaptive Filters," in "Recent Developments in the Core of Digital Signal Processing," Tsuhan Chen, Guest Editor, *IEEE SP Magazine*, vol. 16, pp. 20-22, 1999.

[15] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd edition, Prentice-Hall, 1999.

[16] S. Haykin and B. Van Veen, *Signals and Systems*, Wiley, 1999.

[17] S.M. Kuo and D.R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations*, Wiley, 1996.

[18] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, M.A. Arbib, Ed., MIT Press, 1995.

[19] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, 2nd edition, Addison-Wesley, 1994.

[20] J.M. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control*, Prentice-Hall, 1995.

[21] R.A. Monzingo and T.A. Miller, *Introduction to Adaptive Arrays*, Wiley, 1980.

[22] K.S. Narendra and K. Parthessarthy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, 1991.

[23] A.V. Oppenheim and R.W. Schafer, *Discrete-time Signal Processing*, Prentice-Hall, 1989.

[24] J.C. Principe, N.R. Euliano, and W.C. Lefebvre, "Innovating adaptive and neural systems instruction with interactive electronic books," *Proc. IEEE*, to appear.

[25] J.C. Principe, N.R. Euliano, and W.C. Lefebvre, *Neural and Adaptive Systems: Fundamentals Through Simulations*, Wiley, to be published.

[26] J.G. Proakis, *Digital Communications*, 3rd edition, McGraw-Hill, 1995.

[27] R.D. Reed and R.J. Marks II, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, MIT Press, 1995.

[28] D.E. Rumelhart and J.L. McClelland, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, MIT Press, 1986.

[29] I.W. Sandberg and L. Xu, "Uniform approximation of multidimensional myopic maps," *IEEE Trans. Circuits and Systems*, vol. 44, pp. 477-485, 1997.

[30] N. Wiener, *Extrapolation, Interpolation, and Smoothing of a Stationary Time Series with Engineering Applications*, MIT Press, 1949.

[31] B. Widrow and S. Stearns, *Adaptive Signal Processing*, Prentice Hall, 1985.

[32] J. Winters, "Smart antennas for wireless systems," *IEEE Personal Communications*, vol. 5, no. 1, pp. 23-27, 1998.

The DSP Learning Environment

Modern DSP Education: The Story of Three Greek Philosophers

Joseph Picone, Jonathan E. Hamaker, and Robert Brown
Institute for Signal and Information Processing
Mississippi State University
John H.L. Hansen and Ronald A. Cole
Center for Spoken Language Understanding
University of Colorado at Boulder

The classroom dynamic has changed—if we are certain of one thing in this changing world, it is this. Ask a student to do a literature search. The enterprising student will return with the results of a web search. If it is an obscure topic, the student will perhaps complain that he or she couldn't find anything really useful on the web, what should they do? Ask your students how many of them have read the textbook they bought for the course. Many will reply that they don't read the textbook as carefully as