Now consider the expectation of Eq. (37):

$$E[\boldsymbol{w}(n+1)] = E[\boldsymbol{w}(n)] + 2\mu E[d(n)\boldsymbol{x}(n)] - 2\mu E[\boldsymbol{x}(n)\boldsymbol{x}(n)^{\mathrm{T}}]E[\boldsymbol{w}(n)] \tag{38}$$

We have assumed that the filter weights are uncorrelated with the input signal. This is not strictly satisfied, because the weights depend on $x(n)$; but we can assume that μ has small values because it is associated with a slow trajectory. So, subtracting the optimum solution from both sides of Eq. (38), and substituting the autocorrelation matrix R and cross-correlation vector $\boldsymbol{p}$, we get

$$E[\boldsymbol{w}(n+1)] - R^{-1}\boldsymbol{p} = E[\boldsymbol{w}(n)] - R^{-1}\boldsymbol{p} + 2\mu R\{R^{-1}\boldsymbol{p} - E[\boldsymbol{w}(n)]\} \tag{39}$$

Next, defining

$$\boldsymbol{\xi}(n+1) = E[\boldsymbol{w}(n+1)] - R^{-1}\boldsymbol{p} \tag{40}$$

from Eq. (39) we obtain

$$\boldsymbol{\xi}(n+1) = (\boldsymbol{I} - 2\mu R)\boldsymbol{\xi}(n) \tag{41}$$

This process is equivalent to translation of coordinates. Next, we define R in terms of an orthogonal transformation (7):

$$R = K^{\mathrm{T}}QK \tag{42}$$

where Q is a diagonal matrix consisting of the eigenvalues $(\lambda_0, \lambda_1, \ldots, \lambda_N)$ of the correlation matrix R, and K is the unitary matrix consisting of the eigenvectors associated with these eigenvalues.

Substituting Eq. (42) in Eq. (41), we have

$$\begin{aligned}\boldsymbol{\xi}(n+1) &= (\boldsymbol{I} - 2\mu K^{\mathrm{T}}QK)\boldsymbol{\xi}(n)\\ &= K^{\mathrm{T}}(\boldsymbol{I} - 2\mu Q)K\boldsymbol{\xi}(n)\end{aligned} \tag{43}$$

Multiplying both sides of the Eq. (43) by K and defining

$$\begin{aligned}\boldsymbol{v}(n+1) &= K\boldsymbol{\xi}(n+1)\\ &= (\boldsymbol{I} - 2\mu Q)\boldsymbol{v}(n)\end{aligned} \tag{44}$$

we may rewrite Eq. (44) in matrix form as

$$\begin{bmatrix} v_0(n)\\ v_1(n)\\ \vdots\\ v_{N-1}(n)\end{bmatrix} = \begin{bmatrix}(1-2\mu\lambda_1)^n & & & \\ & (1-2\mu\lambda_2)^n & & \\ & & \vdots & \\ & & & (1-2\mu\lambda_N)^n\end{bmatrix} \begin{bmatrix} v_0(0)\\ v_1(0)\\ \vdots\\ v_{N-1}(0)\end{bmatrix} \tag{45}$$

For stable convergence each term in Eq. (45) must be less than one, so we must have

$$0 < \mu < \frac{1}{\lambda_{\max}} \tag{46}$$

where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix R, though this is not a sufficient condition for stability under all signal conditions. The final convergence rate of the algorithm is determined by the value of the smallest eigenvalue. An important characteristic of the input signal is therefore the eigenvalue spread or disparity, defined as

$$\lambda_{\max}/\lambda_{\min} \tag{47}$$

So, from the point of view of convergence speed, the ideal value of the eigenvalue spread is unity; the larger the value, the slower will be the final convergence. It can be shown (3) that the eigenvalues of the autocorrelation matrix are bounded by the maximum and minimum values of the power spectral density of the input.

It is therefore concluded that the optimum signal for fastest convergence of the LMS algorithm is white noise, and that any form of coloring in the signal will increase the convergence time. This dependence of convergence on the spectral characteristics of the input signal is a major problem with the LMS algorithm, as discussed in Ref. 6.

LMS-Based Algorithms

The Normalized LMS Algorithm. The normalized LMS (NLMS) algorithm is a variation of the ordinary LMS algorithm. Its objective is to overcome the *gradient noise amplification* problem. This problem is due to the fact that in the standard LMS, the correction $\mu e(n)\boldsymbol{x}(n)$ is directly proportional to the input vector $\boldsymbol{x}(n)$. Therefore, when $\boldsymbol{x}(n)$ is large, the LMS algorithm amplifies the noise.

Consider the LMS algorithm defined by

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + 2\mu e(n)\boldsymbol{x}(n) \tag{48}$$

Now consider the difference between the optimum vector $\boldsymbol{w}^*$ and the current weight vector $\boldsymbol{w}(n)$:

$$\boldsymbol{v}(n) = \boldsymbol{w}^* - \boldsymbol{w}(n) \tag{49}$$

Assume that the reference signal and the error signal are

$$d(n) = \boldsymbol{w}^{*\mathrm{T}}\boldsymbol{x}(n) \tag{50}$$

$$e(n) = d(n) - \boldsymbol{w}(n)^{\mathrm{T}}\boldsymbol{x}(n) \tag{51}$$

Substituting Eq. (50) in Eq. (51), we obtain

$$\begin{aligned}e(n) &= \boldsymbol{w}^{*\mathrm{T}}\boldsymbol{x}(n) - \boldsymbol{w}(n)^{\mathrm{T}}\boldsymbol{x}(n)\\ &= [\boldsymbol{w}^{*\mathrm{T}} - \boldsymbol{w}(n)^{\mathrm{T}}]\boldsymbol{x}(n)\\ &= \boldsymbol{v}^{\mathrm{T}}(n)\boldsymbol{x}(n)\end{aligned} \tag{52}$$

We decompose $\boldsymbol{v}(n)$ into its rectangular components

$$\boldsymbol{v}(n) = \boldsymbol{v}_{\mathrm{o}}(n) + \boldsymbol{v}_{\mathrm{p}}(n) \tag{53}$$

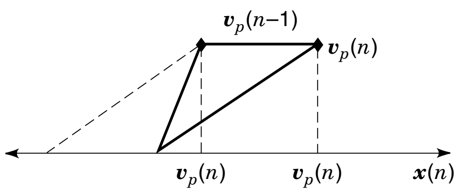

Figure 13. Geometric interpretation of the NLMS algorithm.

where $\boldsymbol{v}_o(n)$ and $\boldsymbol{v}_p(n)$ are the orthogonal component and the parallel component of $\boldsymbol{v}(n)$ with respect to the input vector. This implies

$$\boldsymbol{v}_p(n) = C\boldsymbol{x}(n) \tag{54}$$

where C is a constant. Then substituting Eq. (53) and Eq. (54) in Eq. (52), we get

$$e(n) = [\boldsymbol{v}_o(n) + \boldsymbol{v}_p(n)]^T\boldsymbol{x}(n) \tag{55}$$

$$e(n) = [\boldsymbol{v}_o(n) + C\boldsymbol{x}(n)]^T\boldsymbol{x}(n) \tag{56}$$

Because $\boldsymbol{v}_o(n)$ is orthogonal to $\boldsymbol{x}(n)$, the scalar multiplication is

$$\boldsymbol{v}_o^T\boldsymbol{x}(n) = 0 \tag{57}$$

Then solving for C from Eqs. (56) and (57) yields

$$C = \frac{e(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)} \tag{58}$$

and

$$\boldsymbol{v}_p(n) = \frac{e(n)\boldsymbol{x}(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)} \tag{59}$$

The target now is to make $\boldsymbol{v}(n)$ as orthogonal as possible to $\boldsymbol{x}(n)$ in each iteration, as shown in Fig. 13. The above mentioned can be done by setting

$$\boldsymbol{v}(n+1) = \boldsymbol{v}(n) - \alpha\boldsymbol{v}_p(n) \tag{60}$$

Finally, substituting Eq. (49) and Eq. (59), we get

$$\boldsymbol{w}^* - \boldsymbol{w}(n+1) = \boldsymbol{w}^* - \boldsymbol{w}(n) - \alpha\frac{e(n)\boldsymbol{x}(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)} \tag{61}$$

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \alpha\frac{e(n)\boldsymbol{x}(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)} \tag{62}$$

where, in order to reach the target, α must satisfy (9)

$$0 < \alpha < 2 \tag{63}$$

In this way

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \beta e(n)\boldsymbol{x}(n) \tag{64}$$

where

$$\beta = \frac{\alpha}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)} \tag{65}$$

Therefore, the NLMS algorithm given by Eq. (64) is equivalent to the LMS algorithm if

$$2\mu = \frac{\alpha}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)} \tag{66}$$

NLMS Algorithm

Parameters:	M = filter order
	α = step size
Initialization:	Set $\boldsymbol{w}(0) = 0$
Computation:	For $n = 0, 1, 2, \ldots$, compute
	$y(n) = \boldsymbol{w}(n)^T\boldsymbol{x}(n)$
	$e(n) = d(n) - y(n)$
	$\beta = \frac{\alpha}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)}$
	$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \beta e(n)\boldsymbol{x}(n)$

Time-Variant LMS Algorithms. In the classical LMS algorithm there is a tradeoff between validity of the final solution and convergence speed. Therefore its use is limited for several practical applications, because a small error in the coefficient vector requires a small convergence factor, whereas a high convergence rate requires a large convergence factor.

The search for an optimal solution to the problem of obtaining high convergence rate and small error in the final solution has been an arduous in recent years. Various algorithms have been reported in which time-variable convergence coefficients are used. These coefficients are chosen so as to meet both requirements: high convergence rate and low MSE. Interested readers may refer to Refs. 9–14.

Recursive Least-Squares Algorithm

The recursive least-squares (RLS) algorithm is required for rapidly tracking adaptive filters when neither the reference-signal nor the input-signal characteristics can be controlled. An important feature of the RLS algorithm is that it utilizes information contained in the input data, extending back to the instant of time when the algorithm is initiated. The resulting convergence is therefore typically an order of magnitude faster than for the ordinary LMS algorithm.

In this algorithm the mean squared value of the error signal is directly minimized by a matrix inversion. Consider the FIR filter output

$$y(n) = \boldsymbol{w}^T\boldsymbol{x}(n) \tag{67}$$

where $\boldsymbol{x}(n)$ is the input vector given by $\boldsymbol{x}(n) = [x(n), x(n-1, \ldots, x(n-M+1)]^T$ and $\boldsymbol{w}$ is the weight vector. The optimum weight vector is computed in such a way that the mean squared error, $E[e^2(n)]$ is minimized, where

$$e(n) = d(n) - y(n) = d(n) - \boldsymbol{w}^T\boldsymbol{x}(n) \tag{68}$$

$$E[e^2(n)] = E[\{d(n) - \boldsymbol{w}^T\boldsymbol{x}(n)\}^2] \tag{69}$$

To minimize $E[e^2(n)]$, we can use the orthogonality principle in the estimation of the minimum. That is, we select the weight vector in such a way that the output error is orthogonal to the input vector. Then from Eqs. (67) and (68), we obtain

$$E[\boldsymbol{x}(n)\{d(n)-\boldsymbol{x}^{\mathrm{T}}(n)\boldsymbol{w}\}]=0 \tag{70}$$

Then

$$E[\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)\boldsymbol{w}]=E[d(n)\boldsymbol{x}(n)] \tag{71}$$

Assuming that the weight vector is not correlated with the input vector, we obtain

$$E[\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)]\boldsymbol{w}=E[d(n)\boldsymbol{x}(n)] \tag{72}$$

which can be rewritten as

$$R\boldsymbol{w}=\boldsymbol{p} \tag{73}$$

where R and $\boldsymbol{p}$ are the autocorrelation matrix of the input signal and the correlation vector between the reference signal $d(n)$ and input signal $\boldsymbol{x}(n)$, respectively. Next, assuming ergodicity, $\boldsymbol{p}$ can be estimated in real time as

$$\boldsymbol{p}(n)=\sum_{k=0}^{n}\lambda^{n-k}d(k)\boldsymbol{x}(k) \tag{74}$$

$$\begin{aligned}\boldsymbol{p}(n)&=\sum_{k=0}^{n-1}\lambda^{n-k}d(k)\boldsymbol{x}(k)+d(n)\boldsymbol{x}(n)\\&=\lambda\sum_{k=0}^{n-1}\lambda^{n-k-1}d(k)\boldsymbol{x}(k)+d(n)\boldsymbol{x}(n)\end{aligned} \tag{75}$$

$$\boldsymbol{p}(n)=\lambda\boldsymbol{p}(n-1)+d(n)\boldsymbol{x}(n) \tag{76}$$

where λ is the forgetting factor. In a similar way, we can obtain

$$R(n)=\lambda R(n-1)+\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n) \tag{77}$$

Then, multiplying Eq. (73) by R^{-1} and substituting Eq. (76) and Eq. (77), we get

$$\boldsymbol{w}=[\lambda R(n-1)+\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)]^{-1}[\lambda\boldsymbol{p}(n-1)+d(n)\boldsymbol{x}(n)] \tag{78}$$

Next, according to the *matrix inversion lemma*

$$(A+BCD)^{-1}=A^{-1}-A^{-1}B(DA^{-1}B+C^{-1})^{-1}DA^{-1} \tag{79}$$

with $A=\lambda R(n-1)$, $B=\boldsymbol{x}(n)$, $C=1$, and $D=\boldsymbol{x}^{\mathrm{T}}(n)$, we obtain

$$\begin{aligned}\boldsymbol{w}(n)=&\left[\frac{1}{\lambda}R^{-1}(n-1)-\left(\frac{1}{\lambda}R^{-1}(n-1)\boldsymbol{x}(n)\right)\right.\\&\left.\times\left(\frac{1}{\lambda}\boldsymbol{x}^{\mathrm{T}}(n)R^{-1}(n-1)\boldsymbol{x}(n)+1\right)^{-1}\frac{1}{\lambda}\boldsymbol{x}^{\mathrm{T}}(n)R^{-1}(n-1)\right]\\&\times[\lambda\boldsymbol{p}(n-1)+d(n)\boldsymbol{x}(n)]\end{aligned} \tag{80}$$

$$\begin{aligned}\boldsymbol{w}(n)=&\frac{1}{\lambda}\left(R^{-1}(n-1)-\frac{R^{-1}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)R^{-1}(n-1)}{[\lambda+\boldsymbol{x}^{\mathrm{T}}(n)R^{-1}(n-1)\boldsymbol{x}(n)]}\right)\\&\times[\lambda\boldsymbol{p}(n-1)+d(n)\boldsymbol{x}(n)]\end{aligned} \tag{81}$$

Next, for convenience of computation, let

$$Q(n)=R^{-1}(n) \tag{82}$$

and

$$K(n)=\frac{R^{-1}(n-1)\boldsymbol{x}(n)}{\lambda+\boldsymbol{x}^{\mathrm{T}}(n)R^{-1}(n-1)\boldsymbol{x}(n)} \tag{83}$$

Then from Eq. (81) we have

$$\begin{aligned}\boldsymbol{w}(n)=&\frac{1}{\lambda}[Q(n-1)-K(n)\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)]\\&[\lambda\boldsymbol{p}(n-1)+d(n)\boldsymbol{x}(n)]\end{aligned} \tag{84}$$

$$\begin{aligned}\boldsymbol{w}(n)=&Q(n-1)\boldsymbol{p}(n-1)+\frac{1}{\lambda}d(n)Q(n-1)\boldsymbol{x}(n)\\&-K(n)\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{p}(n-1)\\&-\frac{1}{\lambda}d(n)K(n)\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)\end{aligned} \tag{85}$$

$$\begin{aligned}\boldsymbol{w}(n)=&\boldsymbol{w}(n-1)+\frac{1}{\lambda}d(n)Q(n-1)\boldsymbol{x}(n)\\&-\frac{Q(n-1)\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)\boldsymbol{w}(n-1)}{\lambda+\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)}\\&-\frac{1}{\lambda}\frac{d(n)Q(n-1)\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)}{\lambda+\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)}\end{aligned} \tag{86}$$

$$\begin{aligned}\boldsymbol{w}(n)=&\boldsymbol{w}(n-1)+\frac{1}{\lambda}\frac{Q(n-1)\boldsymbol{x}(n)}{\lambda+\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)}\\&\times[\lambda d(n)+d(n)\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)\\&-\lambda\boldsymbol{x}^{\mathrm{T}}(n)\boldsymbol{w}(n-1)-d(n)\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)]\end{aligned} \tag{87}$$

$$\begin{aligned}\boldsymbol{w}(n)=&\boldsymbol{w}(n-1)+\frac{1}{\lambda}\frac{Q(n-1)\boldsymbol{x}(n)}{\lambda+\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)}\\&\times\lambda[d(n)-\boldsymbol{x}^{\mathrm{T}}(n)\boldsymbol{w}(n-1)]\end{aligned} \tag{88}$$

Finally, we have

$$\boldsymbol{w}(n)=\boldsymbol{w}(n-1)+K(n)\epsilon(n) \tag{89}$$

where

$$K(n)=\frac{Q(n-1)\boldsymbol{x}(n)}{\lambda+\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)} \tag{90}$$

and $\epsilon(n)$ is the *a priori* estimation error, based on the old least-square estimate of the weights vector that was made at time $n-1$, and defined by

$$\epsilon(n)=d(n)-\boldsymbol{w}^{\mathrm{T}}(n-1)\boldsymbol{x}(n) \tag{91}$$

Then Eq. (89) can be written as

$$\boldsymbol{w}(n)=\boldsymbol{w}(n-1)+Q(n)\epsilon(n)\boldsymbol{x}(n) \tag{92}$$

where $Q(n)$ is given by

$$Q(n)=\frac{1}{\lambda}\left(Q(n-1)-\frac{Q(n-1)\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)}{\lambda+\boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)}\right) \tag{93}$$

The applicability of the RLS algorithm requires that it initialize the recursion of $Q(n)$ by choosing a starting value $Q(0)$ to ensure the nonsingularity of the correlation matrix $R(n)$ (3).

RLS Algorithm

```
Initialization:   Set $Q(0)$
                  $\boldsymbol{w}(0) = 0$
Computation:      For $n = 1, 2, \ldots$, compute
                  $K(n) = \dfrac{Q(n-1)\boldsymbol{x}(n)}{\lambda + \boldsymbol{x}^{\mathrm{T}}(n)Q(n-1)\boldsymbol{x}(n)}$
                  $\epsilon(n) = d(n) - \boldsymbol{w}^{\mathrm{T}}(n-1)\boldsymbol{x}(n)$
                  $\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + K(n)\epsilon(n)$
                  $\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \beta e(n)\boldsymbol{x}(n)$
                  $Q(n) = R^{-1}(n)$
```

IMPLEMENTATIONS OF ADAPTIVE FILTERS

In the last few years many adaptive filter architectures have been proposed, for reducing the convergence rate without increasing the computational cost significantly. The digital implementations of adaptive filters are the most widely used. They yield good performance in terms of adaptivity, but consume considerable area and power. Several implementations achieve power reduction by dynamically minimizing the order of the digital filter (15) or employing parallelism and pipelining (16). On the other hand, high-speed and low-power applications require both parallelism and reduced complexity (17).

Is well known that analog filters offer advantages of small area, low power, and higher-frequency operation over their digital counterparts, because analog signal-processing operations are normally much more efficient than digital ones. Moreover, since continuous-time adaptive filters do not need analog-to-digital conversion, it is possible to prevent quantization-related problems.

Gradient descent adaptive learning algorithms are commonly used for analog adaptive learning circuits because of their simplicity of implementation. The LMS algorithm is often used to implement adaptive circuits. The basic elements used for implementing the LMS algorithm are delay elements (which are implemented with all-pass first-order sections), multipliers (based on a square law), and integrators. The techniques utilized to implement these circuits are discrete-time approaches, as discussed in Refs. 18 to 21, and continuous-time implementations (22,23,24).

Several proposed techniques involve the implementation of the RLS algorithm, which is known to have very low sensitivity to additive noise. However, a direct analog implementation of the RLS algorithm would require a considerable effort. To overcome this problem, several techniques have been proposed, such as structures based on Hopfield neural networks (23,25,26,27).

BIBLIOGRAPHY

1. S. U. H. Qureshi, Adaptive equalization, *Proc. IEEE,* **73**: 1349–1387, 1985.
2. J. Makhoul, Linear prediction: A tutorial review, *Proc. IEEE,* **63**: 561–580, 1975.
3. S. Haykin, *Adaptive Filter Theory,* 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 1996.
4. B. Friedlander, Lattice filters for adaptive processing, *Proc. IEEE,* **70**: 829–867, 1982.
5. J. J. Shynk, Adaptive IIR filtering, *IEEE ASSP Mag.,* **6** (2): 4–21, 1989.
6. P. Hughes, S. F. A. Ip, and J. Cook, Adaptive filters—a review of techniques, *BT Technol. J.,* **10** (1): 28–48, 1992.
7. B. Widrow and S. Stern, *Adaptive Signal Processing,* Englewood Cliffs, NJ: Prentice-Hall, 1985.
8. B. Widrow and M. E. Hoff, Jr., Adaptive switching circuits, *IRE WESCON Conv. Rec.,* part 4, 1960, pp. 96–104.
9. J. Nagumo and A. Noda, A learning method for system identification, *IEEE Trans. Autom. Control,* **AC-12**: 282–287, 1967.
10. R. H. Kwong and E. W. Johnston, A variable step size LMS algorithm, *IEEE Trans. Signal Process.,* **40**: 1633–1642, 1992.
11. I. Nakanishi and Y. Fukui, A new adaptive convergence factor with constant damping parameter, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.,* **E78-A** (6): 649–655, 1995.
12. T. Aboulnasr and K. Mayas, A robust variable step size LMS-type algorithm: Analysis and simulations, *IEEE Trans. Signal Process.,* **45**: 631–639, 1997.
13. F. Casco et al., A variable step size (VSS-CC) NLMS algorithm, *IEICE Trans. Fundam.,* **E78-A** (8): 1004–1009, 1995.
14. M. Nakano et al., A time varying step size normalized LMS algorithm for adaptive echo canceler structures, *IEICE Trans. Fundam.,* **E78-A** (2): 254–258, 1995.
15. J. T. Ludwig, S. H. Nawab, and A. P. Chandrakasan, Low-power digital filtering using approximate processing, *IEEE J. Solid State Circuits,* **31**: 395–400, 1996.
16. C. S. H. Wong et al., A 50 MHz eight-tap adaptive equalizer for partial-response channels, *IEEE J. Solid State Circuits,* **30**: 228–234, 1995.
17. R. A. Hawley et al., Design techniques for silicon compiler implementations of high-speed FIR digital filters, *IEEE J. Solid State Circuits,* **31**: 656–667, 1996.
18. M. H. White et al., Charge-coupled device (CCD) adaptive discrete analog signal processing, *IEEE J. Solid State Circuits,* **14**: 132–147, 1979.
19. T. Enomoto et al., Monolithic analog adaptive equalizer integrated circuit for wide-band digital communications networks, *IEEE J. Solid State Circuits,* **17**: 1045–1054, 1982.
20. F. J. Kub and E. W. Justh, Analog CMOS implementation of high frequency least-mean square error learning circuit, *IEEE J. Solid State Circuits,* **30**: 1391–1398, 1995.
21. Y. L. Cheung and A. Buchwald, A sampled-data switched-current analog 16-tap FIR filter with digitally programmable coefficients in 0.8 μm CMOS, *Int. Solid-State Circuits Conf.,* February 1997.
22. J. Ramirez-Angulo and A. Díaz-Sanchez, Low voltage programmable FIR filters using voltage follower and analog multipliers, *Proc. IEEE Int. Symp. Circuits Syst.,* Chicago, May 1993.
23. G. Espinosa F.-V. et al., Ecualizador adaptivo BiCMOS de tiempo continuo, utilizando una red neuronal de Hopfield, *CONIELECOMP'97,* UDLA, Puebla, México, 1997.
24. L. Ortíz-Balbuena et al., A continuous time adaptive filter structure, *IEEE Int. Conf. Acoust., Speech Signal Process.,* Detroit, 1995, pp. 1061–1064.
25. M. Nakano et al., A continuous time equalizer structure using Hopfield neural networks, *Proc. IASTED Int. Conf. Signal Image Process.,* Orlando, FL, November 1996, pp. 168–172.
26. G. Espinosa F.-V., A. Díaz-Méndez, and F. Maloberti, A 3.3 V CMOS equalizer using Hopfield neural network, *4th IEEE Int. Conf. Electron., Circuits, Syst., ICECS97,* Cairo, 1997.

27. M. Nakano-Miyatake and H. Perez-Meana, Analog adaptive filtering based on a modified Hopfield network, *IEICE Trans. Fundam.,* **E80-A**: 2245–2252, 1997.

Reading List

M. L. Honig and D. G. Messerschmitt, *Adaptive Filters: Structures, Algorithms, and Applications,* Norwell, MA: Kluwer, 1988.

B. Mulgrew and C. F. N. Cowan, *Adaptive Filters and Equalisers,* Norwell, MA: Kluwer, 1988.

S. Proakis et al., *Advanced Signal Processing,* Singapore: Macmillan.

GUILLERMO ESPINOSA FLORES VERDAD
JOSÉ ALEJANDRO DÍAZ MÉNDEZ
National Institute for Research in Astrophysics, Optics and Electronics