CONTRIBUTED ARTICLE

A Neural Approach to the Underdetermined-Order Recursive Least-Squares Adaptive Filtering

BUYURMAN BAYKAL and ANTHONY G. CONSTANTINIDES

Imperial College of Science, Technology and Medicine, London

(Received 30 May 1996; accepted 8 April 1997)

Abstract—The incorporation of the neural architectures in adaptive filtering applications has been addressed in detail. In particular, the Underdetermined-Order Recursive Least-Squares (URLS) algorithm, which lies between the well-known Normalized Least Mean Square and Recursive Least Squares algorithms, is reformulated via a neural architecture. The response of the neural network is seen to be identical to that of the algorithmic approach. Together with the advantage of simple circuit realization, this neural network avoids the drawbacks of digital computation such as error propagation and matrix inversion, which is ill-conditioned in most cases. It is numerically attractive because the quadratic optimization problem performs an implicit matrix inversion. Also, the neural network offers the flexibility of easy alteration of the prediction order of the URLS algorithm which may be crucial in some applications. It is rather difficult to achieve in the digital implementation, as one would have to use Levinson recursions. The neural network can easily be integrated into a digital system through appropriate digital-to-analog and analog-to-digital converters. © 1997 Elsevier Science Ltd.

Keywords—Adaptive Filtering, Underdetermined Recursive Least Squares, Neural Networks, Analog Adaptive Filter.

1. INTRODUCTION

The last decade has witnessed the growth of parallel computational structures. Extensive research has been done and published (Baykal, 1992; Sturges, 1988; Cichocki & Unbehauen, 1992a, 1992b, 1994; Kennedy & Chua, 1988; Tank & Hopfield, 1986; Zhang & Constantinides, 1992). Among the many structures now available, the most appealing ones are neural network configurations which can essentially store and retrieve information. The highly parallel structure of neural networks make their analysis and synthesis difficult. Consequently, a well-developed design theory has not been established as yet. Despite the above difficulties, some results have also been obtained (Zhang & Constantinides, 1992) in which a class of neural architectures is described, namely the Lagrange Programming Neural Network (LPNN). As the name implies the LPNNs exploit the well-known Lagrange multiplier theory for nonlinear programming in constrained optimization, to search for the optimum of a cost function subject to various constraints. The LPNN architecture, similar to other neural architectures, is a highly nonlinear and coupled analog computational circuit. The use of analog components is advantageous in terms of computational flexibility, convergence and wide applicability. In fact, the computational power of analog circuits was known well before the overwhelming advance of digital computers (Dennis, 1959); but the topic has not been investigated in detail, because of the relatively high computational power offered by digital computers. Since extensive research on fast implementations of algorithmic approaches cannot further reduce the computational complexity and since computational improvements may worsen the numerical conditioning of an algorithm, analog neural networks seem to be an attractive and reliable way to tackle the high computational complexity and numerically ill-conditioned problems.

Neural network applications in signal processing are emerging [see Cichocki & Unbehauen (1993); and the references therein], as well as their realization via analog circuitry. The reformulation of more conventional digital signal processing applications in terms of neural networks also receives attention, e.g. Baykal (1992) and Culhane et al. (1989), which compute the Karhunen-Loeve Transform, fixed kernel transforms such as...
Discrete Fourier Transform or Discrete Cosine Transform, and some spectral estimation applications via LPNNs. The LPNN realization of such applications turns out to be very efficient, as for example the Karhunen–Loeve Transform, which is almost impossible to compute in real-time by using the classical algorithmic approach, can be formulated as an optimization problem and a LPNN can be associated with it. Such structures are advantageous in terms of computational complexity and numerical conditioning. This paper is focused on LPNNs, for the implementation of the Underdetermined-Order Recursive Least-Squares (URLS) adaptive algorithm (Baykal & Constantinides, 1997) so that we can avoid any drawbacks which might arise from a digital implementation. It would appear that this is one of the few studies on the use of dynamic neural networks in adaptive signal processing.

The formulation of the digital signal processing applications as optimization problems and solving them using LPNNs defines a class of parallel digital signal processing architectures in the genre of systolic arrays (Kung, 1982; Kung & Leiserson, 1978) and wavefront arrays (Kalouptsidis & Theodoridis, 1993). The fundamental difference between LPNNs and systolic or wavefront arrays is that the latter two are digital architectures, hence they inherit the drawbacks of the digital computation such as error propagation in time. The design of such analog circuits has become challenging because of the availability of Very Large Scale Integration (VLSI) techniques for dense interconnection of analog computational elements. Not surprisingly, as analog VLSI becomes more sophisticated, the implementation of neural structures becomes more tractable.

This paper is organized as follows: In Section 2 the URLs algorithm is elaborated. The reformulation of the URLs algorithm via neural architectures is described in Section 3. Sections 4 and 5 address the practical issues in the design of the neural network. In Section 6 we conclude with a discussion on the achievements through the neural network approach.

1.1. LPNNs for Digital Signal Processing

Many digital signal processing problems may be cast as an iterative problem of the form

$$ w_k = f(w_{k-1}, ..., w_{k-n}, u_k, ..., u_{k-m}), $$

where $w_k$ is the desired quantity at discrete-time $k$, $w_{k-i}, i = 1, ..., n$ are its computed values at previous time instants $k - i$ and $u_{k-i}, i = 0, ..., m$ are the external inputs to the system at time instants $k - i$. The algorithmic approach to the problem in (1) is to solve for $w_k$, i.e., to implement the function $f$ in digital computational form. The alternative approach presented herein exploits the advantages of the analog computation. A straightforward approach to solve (1) would be to use the neural network that corresponds to the following optimization problem,

$$ \min \|w_k - f(w_{k-1}, ..., w_{k-n}, u_k, ..., u_{k-m})\| $$

with respect to $w_k$, with the initial condition $w_1 = w_{k-1}$.

This obvious generalization has the simplicity of transforming the digital problem into an analog domain.

We may further extend our approach to (1) and completely reformulate the problem as a genuine optimization problem rather than oversimplifying the approach as in (2). This is application dependent and requires some insight to the problem. Such reformulations have appeared in the literature in the computation of the Karhunen–Loeve Transform, Fourier Transform and other fixed kernel transforms and spectral estimation (Baykal, 1992). The URLs algorithm is formulated below as the solution of an optimization problem. Hence, apart from other digital signal processing applications mentioned above, this is the first time neural architectures are introduced to the adaptive filtering framework.

2. DEFINITION OF THE UNDERDETERMINED RECURSIVE LEAST SQUARES ADAPTIVE ALGORITHM

In this section, a general framework is introduced in which some of the parameter estimation algorithms can be derived as a result of a constrained minimization problem. The minimization problem is formulated in the light of Widrow’s principle of minimal disturbance (Widrow & Lehr, 1990). An example of this principle has appeared in Goodwin and Sin (1984) in the derivation of the Normalized Least Mean Square (NLMS) algorithm. Here, it is shown that the optimization problem can be extended to cover other parameter estimation algorithms and in particular the URLs algorithm is considered in detail. We also show that the URLs algorithm iteratively minimizes an error criterion, the solution of which does not exist in a well-defined sense and requires the application of the pseudo-inverse (Moore–Penrose inverse) of a rectangular matrix. This is due to the underlying underdetermined set of equations for the unknown parameters. The motivation behind the URLs algorithm is to obtain a time recursive algorithm where the pseudo-inverse must be updated at each time instant as new data arrive. The original URLs algorithm was proposed as early as 1984 by Ozeki and Umeda (1984) where they named it the Affine Projection algorithm (AP). The concept of AP will become clear as we consider the optimization problem later in this section.

The principle of minimal disturbance states that, in the light of new input data, the parameters of an adaptive system should only be disturbed in a minimal fashion (Widrow & Lehr, 1990). In fact, this principle is a manifestation of the optimality of least-squares estimates, but more informative in an adaptive sense as
true least-squares estimates are non-adaptive. Based on this argument, the following optimization problem can be established.

Determine the tap-weight vector of a multichannel filter of length \( N_p \) at time \( k \), \( w_k \), given the tap-input vectors \( x_1, x_{i-1}, \ldots \), and the desired responses \( d_k, d_{k-1}, \ldots \), so as to minimize the squared Euclidean norm of the change in the tap-weight vector \( w_k \), i.e., minimize

\[
\delta \| w_k \|^2 = \| w_k - w_{k-1} \|^2
\]

subject to the constraints

\[
w_k^H x_{i} = d_i
\]

(3a)

Note the definitions

\[
w_k = [w_k^{(1)}, w_k^{(2)}, \ldots, w_k^{(p)}]^T,
\]

\[
x_k^H = [x_k^{(1)} H, x_k^{(2)} H, \ldots, x_k^{(p)} H]^T,
\]

(3b)

where the superscript \((i)\) denotes the contribution from the \( i \)-th channel and \( N_p \) is the length of the adaptive filter in the \( i \)-th channel.

To solve this constrained optimization problem, the method of Lagrange multipliers can be used, which yields

\[
w_k = w_{k-1} + \mu e_k^H (X_k^H X_k)^{-1} X_k^H d_k
\]

(4)

where the error vector \( e_k \) and the \( N \times m \) matrix (vector assembly of \( x_k \)) \( X_k \) are defined as

\[
X_k = [x_k x_{k-1} \cdots x_{k-m+1}].
\]

(5)

\[
e_k^H = d_k^H - w_{k-1}^H x_k,
\]

(6)

\[
d_k^H = [d_k d_{k-1} \cdots d_{k-m+1}].
\]

(7)

Equation (4) defines the \( m \)-th order URLS algorithm. It can be made more general in the adaptive sense if a step-size, \( \mu \) is introduced to control the convergence of the algorithm, which in turn, together with the definition

\[
U_k = X_k^H X_k,\] leads to the update equation

\[
w_k = w_{k-1} + \mu e_k^H U_k^{-1} X_k^H d_k.
\]

(8)

If the filter is single channel the matrix reduces to the covariance matrix of the input signal so that the matrix inversion lemma can be employed to reduce the computational load. Hence, the complexity of the algorithm becomes \( O(mN) \) and, in this sense as well as in performance, the URLS algorithm lies between the NLMS and Recursive Least Squares (RLS) algorithms. When \( m = 1 \), (8) reduces to NLMS algorithm and if \( m = N \), (8) becomes the sliding window RLS algorithm in which the window length equals the adaptive filter length.

When the filter is multichannel a fast update rule cannot be found for the \( U \) matrix because the update \( U_k \rightarrow U_{k+1} \) includes a rank modification, not a rank 1 modification, e.g., as in the case of a single channel. Therefore matrix inversion is performed at each instant via Gaussian elimination with complexity \( O(m^3) \).

Strictly speaking, the algorithm in (8) is not a least squares type algorithm unless \( \mu = 1 \), but a combination of recursive least squares type and gradient search type algorithms. It can be observed that the input vector at each time instant is projected onto an affine subspace which is spanned by previous input vectors (Ozeki & Umeda, 1984). An affine subspace is not a proper linear subspace of a higher dimensional space but a hyperplane which may or may not contain the origin.

The algorithm updates the minimum-norm solution to an RLS-like least-squares error criterion which is under-determined: the number of equations is less than the number of unknowns. This is not a well-defined problem and it is necessary to resort to finding the minimum-norm solution among several candidates and update it by projecting the last solution \( w_{k-1} \) onto the orthogonal complement of the subspace spanned by the \( m \) input vectors. Thus, the name Underdetermined Recursive Least-Squares follows (Baykal & Constantinides, 1997). When (8) is carefully examined some adaptive algorithms which are approximations of the URLS algorithm, may be identified. Some of these algorithms are described elsewhere (Baykal & Constantinides, 1997).

3. ANALOG APPROACH TO THE UNDERDETERMINED RECURSIVE LEAST SQUARES ALGORITHM

The solution of the problem in (3) defines one iteration of the URLS algorithm from discrete-time index \( k - 1 \) to \( k \). To obtain an algorithm which spans all discrete-time indices \( k = 0, \ldots, \infty \), the optimization problem of (3) must be solved at each discrete-time index, i.e., the solution obtained at each discrete-time index becomes the coefficients of the adaptive filter as if they were computed by an ordinary sequential computation method.

3.1. Definition of the Neural Network

In previous sections the LPNN has been introduced as an analog neural architecture which is capable of solving constrained optimization problems. When the problem defined in (3) is desired to be solved using analog signal processing, it is amenable to an interpretation through the LPNN architecture. The optimization problem has a quadratic cost function which is subject to linear constraints. This is a well-defined problem. Moreover, the architecture does not have any nonlinear elements that may deteriorate the conditioning of the problem.
Extensive research in signal processing suggests that dealing with the input signal directly rather than with its correlation matrices in least-squares problems, significantly enhances the numerical properties of computations whether they are analog or digital\(^1\) (Cichocki & Unbehauen, 1994; Golub & Loan, 1989). It is observed that in (3) we also deal with the input signal directly rather than its correlation or pseudo-inverse matrices and hence there is strong motivation to pursue the analog implementation of the URLS algorithm further. It is expected therefore that, the LPNN would offer good numerical conditioning together with simple realizations. The neural network corresponding to the URLS algorithm is defined next as follows.

Let us redefine the cost function to simplify derivative expressions as,

\[
\text{Minimize } \frac{1}{2} \| w_k - w_{k-1} \|^2, \quad (9a)
\]

and the constraints as

\[
h_i(w_k) = w_k x_i - d_i = 0, \quad i = 1, \ldots, N \\
\]

\[
h_m(w_k) = w_k x_m - d_m - d_{m+1} = 0.
\]

The Lagrangian function for this problem is

\[
L(w_k, x) = \frac{1}{2} \sum_{j=1}^{N} \| w_{j,k} - w_{j,k-1} \|^2 + \sum_{j=1}^{m} s_j h_j(w_k). \quad (10)
\]

A penalty function may also be imposed in the Lagrangian function to improve convergence properties of the network as suggested by experimental evidence. Thus we can write,

\[
L(w_k, x) = \frac{1}{2} \sum_{j=1}^{N} \| w_{j,k} - w_{j,k-1} \|^2 + \sum_{j=1}^{m} s_j h_j(w_k) + \frac{1}{2} \sum_{j=1}^{N} | h_j(w_k) |^2. \quad (11)
\]

The time-domain differential equations that govern the dynamics of the augmented neural network can be derived by applying the method in Zhang and Constantinides (1992). Hence, for a single channel filter we obtain

\[
\frac{dw_{i,k}(t)}{dt} = -[w_{i,k}(t) - w_{i,k-1} + \sum_{j=1}^{m} s_j h_j(t) x_{i-j+2}, \quad i = 1, \ldots, N.
\]

\[
\frac{ds_{i,k}(t)}{dt} = h_i(w_k(t)), \quad i = 1, \ldots, m. \quad (12b)
\]

The system of differential equations in (12) defines the URLS algorithm. When the equilibrium is reached at any \( k \), the URLS algorithm will have completed one iteration. The dynamics of the neural network in (12) can be put in words as follows: Assume that, without loss of generality, the algorithm is switched on at the discrete-time index \( k \). The scalars and vectors that explicitly depend on \( k \) remain constant between the time interval \( k \), \( k+1 \), and (12) is released to converge. A little time before \( k+1 \), which depends on the structure of the network (the components used in the analog circuit affect the convergence behaviour significantly) the resulting states \( w_{i,k}, i = 1, \ldots, N \) are sampled to obtain the result of the iteration at \( k \). Then, they are used in the next iteration \( k+1 \) for \( w_{i,k+1}, i = 1, \ldots, N \) values. Also, the variables that depend on \( k \) are replaced with their updated values at \( k+1 \). The same process is repeated and another set \( w_{i,k+1}, i = 1, \ldots, N \) is obtained. When this process is continued indefinitely, the URLS algorithm is realized with \( w_{i,k+1} = 1, \ldots, N \) being the results of each iteration.

A structure of the LPNN implementation of the single channel URLS algorithm is presented in Figure 1 for \( N = 3 \) and \( m = 2 \). The components in this structure are also described in Figure 2. In Figure 1 the step-size parameter \( \mu \) is also included. The incorporation of a step-size is useful in circumstances where a high observation noise is present on the desired signal \( d_k \). This is achieved by computing the update term \( s_i U_1^{-1} X^T \) via the neural network, scaling it by \( \mu \) and adding the result to \( w_{i,k-1} \) to find the updated vector \( w_i \).

In the multichannel URLS algorithm some additional constraints can be employed to enhance the convergence properties (Benesty et al., 1996). For example, the constraint

\[
\partial w^{(0)}_{i,k} = 0 \quad (13)
\]

forces the algorithm to orthogonalize the signal in \( i \)-th channel and the difference between filter coefficients of channel \( j \). Then, the Lagrangian function is modified, so is the network. Such constraints are easy to implement in the neural network by additional circuitry and can be simply turned off by disconnecting the output of the associated integrator.

It is also possible to find neural networks for other adaptive applications which have their roots in the minimal disturbance principle. Hence, a general optimization problem in the form of (3) constitutes a framework for the neural applications in adaptive filtering.

3.2. Simulation Example

The analog network of Figure 1 is simulated on a digital computer, i.e., the stationary points of the system of differential equations in (12) are searched by numerical integration methods. In particular, we have used the fourth order Runge–Kutta method to obtain the solution (Atkinson, 1989). The instants at which the discrete-time
index \( k \) is advanced are chosen to coincide with integers across the time axis so that a normalized sampling period is obtained for the ease of presentation and comparison purposes. In real-time applications, of course, the discrete time-index \( k \) is advanced at the sampling frequency, which implies that the system of differential equations (12a) and (12b) must converge within a sampling period.

Various experiments were performed to investigate the behaviour of the neural network. The values of the critical parameters, namely, the penalty parameter \( C \) and step-size \( \mu \) are altered to gain insight to the internal dynamics. Usually the choice \( C = 10 \) is sufficient for satisfactory performance. We have also designed setups for various external conditions such as different SNR levels and a time-varying unknown system, and tested the response of the LPNN. Figure 3 shows typical system mismatch curves in dB obtained from digital and analog simulations of the URLS algorithm for a system identification experiment. The system mismatch is defined as

\[
\text{s}_k = \frac{||w_k - w_{opt}||}{||w_{opt}||},
\]

(14)

where \( w_{opt} \) denotes the optimal system. We can observe that both approaches yield close results. It is important to note that the analog simulations are made in the idealized domain where interferers such as parasitic capacitance and leakage are not present. In practice, we would expect the performance degrade more (Denyer et al., 1983). Also, the signals are contaminated by noise in analog networks, as for example, the effect of thermal noise may be significant. Nevertheless, the immunity to noise can be improved by employing feedback loops. The digital implementation also suffers from non-ideal conditions such as shorter wordlengths, which causes deviations from true values.

Figure 4 shows a possible use of the analog implementation of the URLS algorithm in digital environment. The digital quantities must be converted to analog quantities before they are fed into the LPNN. For example, if the LPNN is desired to perform the echo cancellation activity in hands-free telephone devices (Tanrikulu et al., 1997) connected to a digital network such as GSM, the hybrid analog-digital implementation of the Figure 4 is a simplified model of the acoustical echo cancellation setup.
4. PRACTICAL CONSIDERATIONS

From the implementation point of view, there are two alternative VLSI implementations of neural networks. The first approach employs continuous-time computing units which simulates the differential equations dictated by the neural network. In the second approach, the differential equations are discretized difference equations that may be solved by discrete-time computational units. For example, by applying the Euler integration rule, the following neural network

\[ \frac{d w_i}{dt} = f_i(w) \]

\[ \frac{d w_n}{dt} = f_n(w) \]

can be discretized and converted to the difference equations

\[ w_i^{(T+1)} = f_i(w_i^{(T)}) \]

\[ w_n^{(T+1)} = f_n(w_n^{(T)}) \]

with \( w_i^{(0)} \), for \( i = 1, \ldots, n \) and \( T = 0, 1, \ldots, \infty \), where \( w_i(T) = w_i(T^r) \) and \( r \) is the integration step. The above system of difference equations can be implemented by various digital or indeed mixed signal methods. For instance, switched-capacitor techniques are very popular and suitable for the system (16) (Rodriguezvazquez et al., 1990). The neural network for the URLS algorithm contains easy-to-implement components such as integrators and multipliers. The audio applications that have a sampling frequency of 8 KHz are well below the maximum speed that can be achieved in analog VLSI. For high frequency applications such as mobile communications, analog VLSI can maintain the speed as well (Choi et al., 1993). Therefore, in terms of the implementation of the URLS algorithm in VLSI, there are no major problems and the architecture easily lends itself to practical implementation.

5. COMPARISON OF DIGITAL AND ANALOG APPROACHES

By the nature of the underdetermined recursive least-squares problem spawned by the minimal disturbance principle, the resulting algorithm is amenable to neural implementations. In neural applications it is important to understand the underlying optimization interpretation. The URLS algorithm could have been formulated as a generalization of the NLMS algorithm as a result of which, a neural architecture cannot be easily found. Various advantages and
disadvantages of both implementations are discussed in Table 1.

6. CONCLUDING REMARKS
The work in this paper reveals that there is significant potential behind analog neural network implementations of DSP applications. We have introduced the analog neural networks into the framework of adaptive filtering. In terms of numerical properties, the neural network is attractive because the quadratic optimization problem performs an implicit matrix inversion, which is poorly conditioned if implemented in digital form. Another good feature is the ease of implementation of the analog circuitry. The parallelism of neural architectures allows to tackle high computational
TABLE 1
A comparison of digital and analog implementations of the URLS algorithm

<table>
<thead>
<tr>
<th>Digital URLs</th>
<th>Analogue URLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost of implementation is low. A DSP chip is required.</td>
<td>Cost of implementation may be higher. Custom design in analog VLSI is required.</td>
</tr>
<tr>
<td>If single channel, fast implementations are possible to reduce the complexity.</td>
<td>Analog circuitry cannot be simplified.</td>
</tr>
<tr>
<td>No extra variables required for convergence. Noise level is determined by the number of bits used to represent the numbers. Usually low.</td>
<td>Requires extra penalty function circuitry for convergence.</td>
</tr>
<tr>
<td>Suitable high speed applications.</td>
<td>May not be suitable for high speed applications.</td>
</tr>
<tr>
<td>May be numerically bad conditioned due to matrix computations.</td>
<td>Dynamical search over all values, limited dynamic range.</td>
</tr>
<tr>
<td>Inflexible implementation.</td>
<td>No explicit computation of badly conditioned matrices.</td>
</tr>
<tr>
<td>Arbitrary control of constraints is not possible.</td>
<td>Flexible implementation, it is easy to add constraints.</td>
</tr>
<tr>
<td>More flexible implementation renders the neural architecture more appropriate for time-varying systems and/or non-stationary signals.</td>
<td>Easy to alter the prediction order by the ease of constraint control.</td>
</tr>
<tr>
<td>May not be suitable for time-varying systems and/or non-stationary signals due to inflexible implementation.</td>
<td></td>
</tr>
</tbody>
</table>

complexity problems. The neural network implementation of the URLS algorithm has the advantage that it is easy to alter the prediction order of the algorithm by simply turning on and off the associated constraints in the network, which would be problematic in digital approach. This flexible feature of the neural network may be desirable if the input to the adaptive filter is highly non-stationary and/or the identified system is time-varying.

REFERENCES
