AN ADAPTIVE FILTERING ALGORITHM USING MEAN FIELD ANNEALING TECHNIQUES

P. Persson, I. Claesson

Blekinge Institute of Technology
Dept. of Telecommunications
and Signal Processing
Ronneby, Sweden

S. Nordebo

Växjö University
Dept. of Mathematics
and Systems Engineering
Växjö, Sweden

ABSTRACT

We present a new approach to discrete adaptive filtering based on the mean field annealing algorithm. The main idea is to find the discrete filter vector that minimizes the matrix form of the Wiener-Hopf equations in a least-squares sense by a generalized mean field annealing algorithm. It is indicated by simulations that this approach, with complexity $\mathcal{O}(M^2)$ where M is the filter length, finds a solution comparable to the one obtained by the recursive least squares (RLS) algorithm but without the transient behavior of the RLS algorithm.

Further advantages of the proposed algorithm over other methods such as the recursive least-squares algorithm are that the filter coefficients are always limited and that it facilitates fast recovery after an abrupt system change.

1. INTRODUCTION

Consider the system shown in Fig. 1 where $x(n)$ is a sequence of input data, $v(n)$ is white noise, $d(n)$ is the desired output for the adaptive filter $\mathbf{w}$, formed by filtering the signal $x(n)$ through unknown filter $\mathbf{g}$. The difference between the desired signal $d(n)$ and the actual output $y(n)$ from the adaptive filter $\mathbf{w}$ is denoted $e(n)$,

$$e(n) = \mathbf{x}_n^T\mathbf{g} + v(n) - \mathbf{x}_n^T\mathbf{w} = d(n) - \mathbf{x}_n^T\mathbf{w} \qquad (1)$$

Common methods for finding the best estimate of $\mathbf{g}$ include the least-mean-squares (LMS) algorithm and the recursive least-squares (RLS) algorithm [1].

Defining the weighted mean square error $\mathcal{E}(n, \mathbf{w})$ as

$$\mathcal{E}(n,\mathbf{w}) = \sum_{i=0}^{n} \lambda^{n-i}|e(i)|^2 = \sum_{i=0}^{n} \lambda^{n-i}|d(i) - \mathbf{x}_i^T\mathbf{w}|^2 \qquad (2)$$

which by expansion can be written as

$$\mathcal{E}(n,\mathbf{w}) = \sum_{i=0}^{n} \lambda^{n-i}\left(\mathbf{w}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{w} - 2\mathbf{w}^T d(i)\mathbf{x}_i + d^2(i)\right) \qquad (3)$$

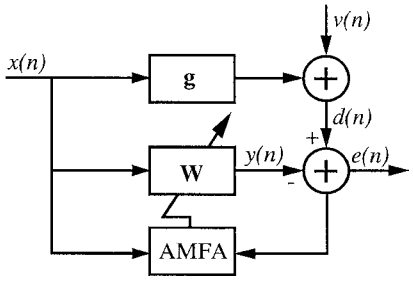

Fig. 1. The adaptive filter system detailed in the text.

or, equivalently, as

$$\mathcal{E}(n,\mathbf{w}) = \mathbf{w}^T\mathbf{R}_{xx}(n)\mathbf{w} - 2\mathbf{w}^T\mathbf{r}_{dx}(n) + D(n) \qquad (4)$$

where

$$\mathbf{x}_i = [x(i), x(i-1), \ldots, x(i-M-1)]^T \qquad (5)$$

and $\mathbf{R}_{xx}$ and $\mathbf{r}_{dx}$ are the autocorrelation matrix and cross-correlation vector estimates built from $x(n)$ and $d(n)$,

$$\mathbf{R}_{xx}(n) = \sum_{i=0}^{n} \lambda^{n-i}\mathbf{x}_i\mathbf{x}_i^T \qquad (6)$$

$$\mathbf{r}_{dx}(n) = \sum_{i=0}^{n} \lambda^{n-i}d(i)\mathbf{x}_i \qquad (7)$$

$$D(n) = \sum_{i=0}^{n} \lambda^{n-i}d^2(i). \qquad (8)$$

The value of $\mathbf{w}$ that minimizes $\mathcal{E}$ in a least squares sense at sample n, denoted $\mathbf{w}_n$, is obtained by solving $\nabla\mathcal{E}(\mathbf{w}) = \mathbf{0}$ which renders the matrix formulation of the Wiener-Hopf equations

$$\mathbf{w}_n = \mathbf{R}_{xx}^{-1}(n)\mathbf{r}_{dx}(n). \qquad (9)$$

The above solution requires a matrix inversion and is unsuitable for real-time implementations but e.g. the recursive

0-7803-7587-4/02/$17.00 ©2002 IEEE

least squares (RLS) algorithm circumvents that problem by recursively constructing an estimate of $\mathbf{R}_{xx}^{-1}(n)$ [1].

Besides the problem of obtaining $\mathbf{R}_{xx}^{-1}(n)$, the RLS algorithm displays large initial transients if not properly initialized [2]. An algorithm with complexity $\mathcal{O}(M^{2.2})$ based on interior point optimization methods, that addresses this problem, has recently been proposed [3].

In this paper we will present the use of a generalized mean field annealing (MFA) algorithm to find a good estimate of g based on a recursively built estimate of $\mathbf{R}_{xx}(n)$. Furthermore, the MFA based algorithm does not display the transient behavior of the RLS.

2. ADAPTIVE MEAN FIELD ANNEALING

We will start by discussing the basis of the MFA algorithm, deriving the MFA equation using conditional expectations and then proceed to a generalized form of the MFA equations suitable for adaptive filtering.

2.1. Mean field annealing

The mean field annealing (MFA) algorithm [4],[5],[6] is an efficient tool to solve combinatorial problems with binary variables related to the Simulated Annealing (SA) algorithm. Both are derived from statistical mechanics but with the fundamental difference that SA is a stochastic algorithm whereas MFA is a *deterministic* algorithm.

The traditional use of MFA for combinatorial optimization requires the problem at hand to be coded such that every solution, a state, can be uniquely described by a sequence of L binary digits. Consider the set $\mathcal{S}$ of 2^L states defined by

$$\mathcal{S} = \{\mathrm{s} \mid s_i \in \{0,1\}, i = 1, \ldots, L\} \tag{10}$$

and the global cost, $J(\mathrm{s})$, associated with each state.

From statistical mechanics it is known [7] that the probability of a particular state s of a system in thermal equilibrium at temperature c is given by the Boltzmann distribution

$$P(\mathrm{s}) = \frac{e^{-J(\mathrm{s})/c}}{\sum_{\mathcal{S}} e^{-J(\mathrm{s})/c}}. \tag{11}$$

Under the condition that the system is kept in thermal equilibrium, it is easy to show that (11) gives

$$\lim_{c \to 0} P(\mathrm{s}^{opt}) = 1 \tag{12}$$

where s^{opt} is a state with $J(\mathrm{s}^{opt}) < J(\mathrm{s} \neq \mathrm{s}^{opt})$ [8]. The complication here is that the sum in (11) is running over *all* states in $\mathcal{S}$ which renders it impossible to compute for most real-world problems.

The fundamental idea behind MFA is to find a way to track the trivial solution at high temperature, through a sequence of lowered temperatures, to an optimal, or near optimal, solution at a temperature close to zero.

An analysis of the basic MFA algorithm is given in [5]. For a more intuitive picture the MFA algorithm could be envisioned as replacing all problem variables but one with their expected mean, thus effectively making the remaining variable independent of the rest, and then finding the expectation value of the free variable. The process is then iterated for the remaining variables.

Alternatively, well known results from statistics can be used to express the expectation value of s_i in terms of conditional expectations

$$\begin{aligned} E[s_i] &= \sum_{\mathrm{s}} s_i P(\mathrm{s}) = \sum_{\mathrm{s}} s_i P(\bar{\mathrm{s}}_i) P(s_i \mid \bar{\mathrm{s}}_i) = \\ &= \sum_{\bar{\mathrm{s}}_i} P(\bar{\mathrm{s}}_i) \sum_{s_i} s_i P(s_i \mid \bar{\mathrm{s}}_i) = \\ &= E_{\bar{\mathrm{s}}_i}\left[\sum_{s_i} s_i P(s_i \mid \bar{\mathrm{s}}_i)\right] \end{aligned} \tag{13}$$

and, letting $v_i \in [0,1]$ denote $E[s_i]$, we have

$$\begin{aligned} v_i &= E_{\bar{\mathrm{s}}_i}\left[\sum_{s_i} s_i P(s_i \mid \bar{\mathrm{s}}_i)\right] \approx \\ &\approx \sum_{s_i} s_i P(s_i \mid E_{\bar{\mathrm{s}}_i}[\bar{\mathrm{s}}_i]) = \\ &= \sum_{s_i} s_i P(s_i \mid \bar{\mathbf{v}}_i) \end{aligned} \tag{14}$$

where $\bar{\mathrm{s}}_i$ (and similarly $\bar{\mathbf{v}}_i$) denotes the vector s with the ith element s_i removed. Writing (14) using the Boltzmann distribution (11) we arrive at the MFA algorithm core

$$v_i = \frac{\sum_{s_i} s_i e^{-\frac{1}{c} J(\mathbf{v})|_{v_i = s_i}}}{\sum_{s_i} e^{-\frac{1}{c} J(\mathbf{v})|_{v_i = s_i}}} \tag{15}$$

which is a fixed point equation with property (12) as well as a sigmoid behavior making $v_i \to s_i$ as $c \to 0$, see Fig. 2.

2.2. Multi-mode mean field annealing

Noting that $s_i \in \{0,1\}$ is not used in the derivation of (15), we are free to let s_i take on any number of discrete levels [9], e.g. the 2^b levels in a digital filter with wordlength b or the set of levels that can be represented by a sum of at most two *signed power-of-two* (SPT) coefficients. The discretization of levels now take place in a n-dimensional space which is hard to picture. The special case of $s_i \in \{-1,0,1\}$ has a direct interpretation in terms of SPT coefficients [10].

2.3. Adaptive mean field annealing algorithm

From the theory presented in the previous sections we now present an adaptive mean field annealing (AMFA) suitable for adaptive filtering.

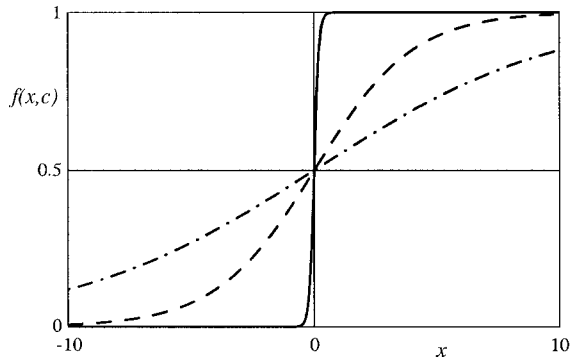

Fig. 2. A sigmoid function $f(x,c)$ plotted for $c = 5$ (dash-dotted), 2 (dashed) and 0.1 (solid). As $c \to 0$, the variable $x = f(x,c)$ is forced to approach $\{0,1\}$.

The notation for the weighted least squares error $\mathcal{E}(n,\mathbf{w})$ is changed to $J_n(\mathbf{w})$ to emphasize that we are now optimizing over $\mathbf{w}$, giving

$$\begin{aligned} J_n(\mathbf{w}) &\equiv \mathcal{E}(n,\mathbf{w}) = \\ &= \mathbf{w}^T\mathbf{R}_{xx}(n)\mathbf{w} - 2\mathbf{w}^T\mathbf{r}_{dx}(n) + D(n). \end{aligned} \quad (16)$$

From (6), (7) and (8) we find the recursive expressions

$$\mathbf{R}_{xx}(n) = \lambda\mathbf{R}_{xx}(n-1) + \mathbf{x}(n)\mathbf{x}^T(n) \quad (17)$$

$$\mathbf{r}_{dx}(n) = \lambda\mathbf{r}_{dx}(n-1) + d(n)\mathbf{x}(n) \quad (18)$$

$$D(n) = \lambda D(n-1) + d^2(n) \quad (19)$$

where the forgetting factor $\lambda = [0,1]$ provides a possibility to give less weight to old data and improve the ability of the algorithm to track a time-varying system.

The generalized MFA equation in this context is derived by, in (15), substituting $\mathbf{w}$ for $\mathbf{v}$ and selecting the set $\mathcal{N}$ from a problem specific set of feasible coefficients, giving

$$w(k) = \frac{\sum_{s\in\mathcal{N}} s \cdot e^{-\frac{1}{c}J_n(\mathbf{w})|_{w(k)=s}}}{\sum_{s\in\mathcal{N}} e^{-\frac{1}{c}J_n(\mathbf{w})|_{w(k)=s}}} \quad (20)$$

and the adaptive MFA algorithm then follows as:

initialize: $\mathbf{w} = \mathbf{0}$, $\mathbf{R}_{xx} = \mathbf{0}$, $\mathbf{r}_{dx} = \mathbf{0}$ and $D = 0$
for $n = 1\ldots$, compute
$\mathbf{R}_{xx}(n) = \lambda\mathbf{R}_{xx}(n-1) + \mathbf{x}(n)\mathbf{x}^T(n)$
$\mathbf{r}_{dx}(n) = \lambda\mathbf{r}_{dx}(n-1) + d(n)\mathbf{x}(n)$
$D(n) = \lambda D(n-1) + d^2(n)$
set initial c
repeat
for $k = 1\ldots M$
compute $w(k)$ from (20)
endfor
decrease c
until $\mathbf{w}$ is stable
endfor

3. IMPLEMENTATION DETAILS

In practice, a number of issues have to be considered. Most important is the complexity which is $\mathcal{O}(M^3)$ in the basic formulation but can be reduced to $\mathcal{O}(M^2)$ given some observations. Stepwise updating of the cost J alleviates us from having to compute the full quadratic form (16) and also provides a way to detect abrupt system changes, letting us reset $\mathbf{R}_{xx}$, $\mathbf{r}_{dx}$, D and J and to zero which proves to be more efficient than normal adaptation under certain conditions. The forgetting factor λ provides a trade-off between adaptation speed and steady state error.

3.1. Incremental updating of cost function

For every new sample the autocorrelation matrix, crosscorrelation and constant term will have to be updated according to (17)-(19). In principle, we could use (16) to compute $J_n(\mathbf{w})$ at all times, but it is computationally more efficient to implement an updating scheme, keeping track of all changes to the cost.

Consider the cost just before updating $\mathbf{w}$ but after updating $\mathbf{R}_{xx}$, $\mathbf{r}_{dx}$ and D, i.e. our current estimate of $\mathbf{w}$ is $\mathbf{w}_{n-1}$ and the cost is

$$\begin{aligned} J_n(\mathbf{w}_{n-1}) = \mathbf{w}_{n-1}^T\mathbf{R}_{xx}(n)\mathbf{w}_{n-1} - \\ -2\mathbf{w}_{n-1}^T\mathbf{r}_{dx}(n) + D(n). \end{aligned} \quad (21)$$

By substituting expressions (17)-(19) for $\mathbf{R}_{xx}(n)$, $\mathbf{r}_{dx}(n)$ and $D(n)$ in (21) we have

$$\begin{aligned} J_n(\mathbf{w}_{n-1}) &= \mathbf{w}_{n-1}^T\left(\lambda\mathbf{R}_{xx}(n-1) + \mathbf{x}_n\mathbf{x}_n^T\right)\mathbf{w}_{n-1} - \\ &- 2\mathbf{w}_{n-1}^T\left(\lambda\mathbf{r}_{dx}(n-1) + d(n)\mathbf{x}_n\right) + \\ &+ \lambda D(n-1) + d^2(n) \end{aligned} \quad (22)$$

and by identifying $J_{n-1}(\mathbf{w}_{n-1})$ we find an expression for $J_n(\mathbf{w}_{n-1})$ in terms of an update to the cost from the previous sample step

$$\begin{aligned} &J_n(\mathbf{w}_{n-1}) = \\ &\lambda\left(\mathbf{w}_{n-1}^T\mathbf{R}_{xx}(n-1)\mathbf{w}_{n-1} - 2\mathbf{w}_{n-1}^T\mathbf{r}_{dx}(n-1) + D(n-1)\right) + \\ &+ \left(\mathbf{w}_{n-1}^T\mathbf{x}_n\right)\left(\mathbf{x}_n^T\mathbf{w}_{n-1}\right) - 2d(n)\mathbf{w}_{n-1}^T\mathbf{x}_n + d^2(n) = \\ &= \lambda J_{n-1}(\mathbf{w}_{n-1}) + \Delta J(n) \end{aligned} \quad (23)$$

where

$$\Delta J(n) = \left(\mathbf{w}_{n-1}^T\mathbf{x}_n\right)\left(\mathbf{x}_n^T\mathbf{w}_{n-1}\right) - 2d(n)\mathbf{w}_{n-1}^T\mathbf{x}_n + d^2(n). \quad (24)$$

Furthermore, by writing (16) explicitly and factoring out component w_k we find that it can be separated into two parts of which one does not depend on w_k,

$$\begin{aligned} J_n(\mathbf{w}) &= \sum_i \sum_j w_i w_j R_{i,j} - 2\sum_i r_i w_i + \sum_i d_i^2 = \\ &= \sum_{i\neq k} \sum_{j\neq k} w_i w_j R_{i,j} - 2\sum_{i\neq k} r_i w_i + \sum_i d_i^2 + \\ &+ 2w_k \sum_i w_i R_{i,k} - w_k^2 R_{k,k} - 2r_k w_k = \\ &= \alpha_k + 2w_k\beta_k + w_k^2 R_{k,k} - 2r_k w_k \end{aligned} \quad (25)$$

where $R_{i,j}$ denotes the elements of $\mathbf{R}_{xx}$, and

$$\alpha_k = \sum_{i\neq k} \sum_{j\neq k} w_i w_j R_{i,j} - 2\sum_{i\neq k} r_i w_i + \sum_i d_i^2 \quad (26)$$

$$\beta_k = \sum_{i\neq k} w_i R_{i,k} \quad (27)$$

which means that when computing the sums in (20) it is not necessary to compute the full expression (16). Note that computing α_k as

$$\alpha_k = J_n(\mathbf{w}) - 2w_k\beta_k - w_k^2 R_{k,k} + 2r_k w_k \quad (28)$$

instead of using (26) directly decreases the complexity of the algorithm from $\mathcal{O}(M^3)$ to $\mathcal{O}(M^2)$. Utilizing the above we can formulate the AMFA

initialize: $\mathbf{w} = \mathbf{0}$, $\mathbf{R}_{xx} = \mathbf{0}$, $\mathbf{r}_{dx} = \mathbf{0}$ and $D = 0$
for $n = 1 \ldots$, compute
  $\mathbf{R}_{xx}(n)$, $\mathbf{r}_{dx}(n)$ and $D(n)$ from (17), (18), and (19)
  $J_n(\mathbf{w})$ from (23), taking $\mathbf{w}_{n-1}$ as initial value for $\mathbf{w}$
  set initial c
  repeat
    for $k = 1 \ldots M$, compute
      β_k and α_k from (27) and (28)
      $w(k)$ from (20), utilizing (25) instead of (16)
      $J_n(\mathbf{w})$ from (25)
    endfor
    decrease c
  until $\mathbf{w}$ is stable
endfor

The cooling schedule is not critical and for most practical purposes, an initial $c = 10^4$ and a minimum limit of $c = 10^{-4}$ is enough if employing a geometrical decrease $c_n = 0.7c_{n-1}$. In practice, this places an upper limit on the number of cooling steps required, of approximately 50, independent of problem.

4. SIMULATIONS

To test the adaptive mean field annealing algorithm a number of simulations comparing it to the RLS algorithm with complexity $\mathcal{O}(M^2)$ as described in [1] were performed. For the sake of comparison, the tests were adopted from [2] and [3]. In all simulations the mean squared error in the filter coefficients, $\epsilon(n) = \|\mathbf{w}_n - \mathbf{g}\|^2$, and the squared estimation error $|e(n)|^2$ were monitored as a measure of performance. The unknown filter $\mathbf{g}$ of length M had its coefficients chosen randomly in the range $[-1, 1]$ for each simulation run. Input data, $x(n)$, was obtained by filtering white noise through the IIR filter

$$H(z) = \frac{1 + 2z^{-1} + 3z^{-2}}{(1 - 1.1314z^{-1} + 0.64z^{-2})(1 + 0.9z^{-1})}. \quad (29)$$

White noise with a signal-to-noise ratio of 10 dB was added to the output from $\mathbf{g}$. For the RLS, the matrix $\mathbf{R}_{xx}$ was initialized to $\delta\mathbf{I}$, with $\delta = 10^{-4}$ for all runs and for the AMFA algorithm the set of feasible coefficients used was $\mathcal{N} = \{l \cdot 2^{-b} \mid -2^b < l < 2^b\}$, where l is an integer and $b = 7$ in all runs. The forgetting factor λ was set to 0.99 in all simulations.

4.1. Initial transient behavior

Comparing the transient behavior of the AMFA to an ordinary RLS algorithm we can conclude that the AMFA performs well. Fig. 3 shows the mean square filter weight error, $\epsilon(n)$, and the estimation error, $|e(n)|^2$, for the first 200 samples, averaged over 100 runs, each time with a new $\mathbf{g}$.

4.2. Abrupt system change

The RLS algorithm has a very fast convergence after the initial transient but cannot track abrupt system changes efficiently. A λ close to one, which is needed for a small steady state error also slows down the adaptation speed. In this respect the RLS and AMFA has essentially identical behavior, as shown in Fig. 4 where, at sample $n = 501$, an abrupt system change is introduced by replacing $\mathbf{g}$.

For the RLS algorithm there are variants such as the *sliding window* RLS, which only uses data from a certain number of previous samples, that improves the ability to track changes. Utilizing the fact that the AMFA produces a $\mathbf{w}$ that is constrained to $w_k \in [-1, 1]$ and does not display the transient behavior of the RLS, recovering after an abrupt system change can be improved by simply re-initializing $\mathbf{w} = \mathbf{0}$, $\mathbf{R}_{xx} = \mathbf{0}$, $\mathbf{r}_{dx} = \mathbf{0}$ and $D = 0$ according to the normal algorithm initialization. In order to detect an abrupt change that calls for a reset of the internal state it possible to monitor either $\Delta J(n)$ from (24) or the error $e(n)$ since both signals increase rapidly from the steady-state level when the

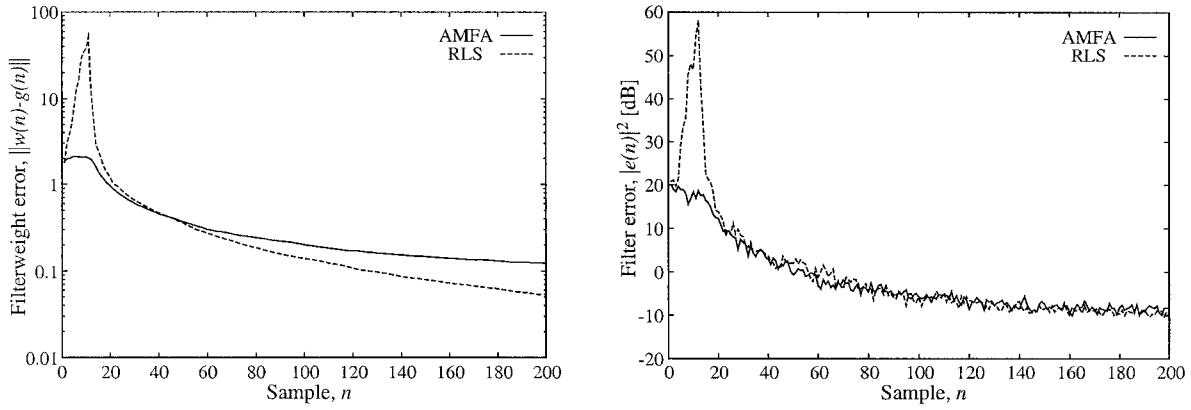

Fig. 3. A comparison of the initial transient behavior of the AMFA (solid) and the RLS (dashed) algorithm.The left plot shows the mean square filter weight error, $\epsilon(n)$ and the plot on the right shows the estimation error $|e(n)|^2$, averaged over 100 simulation runs.

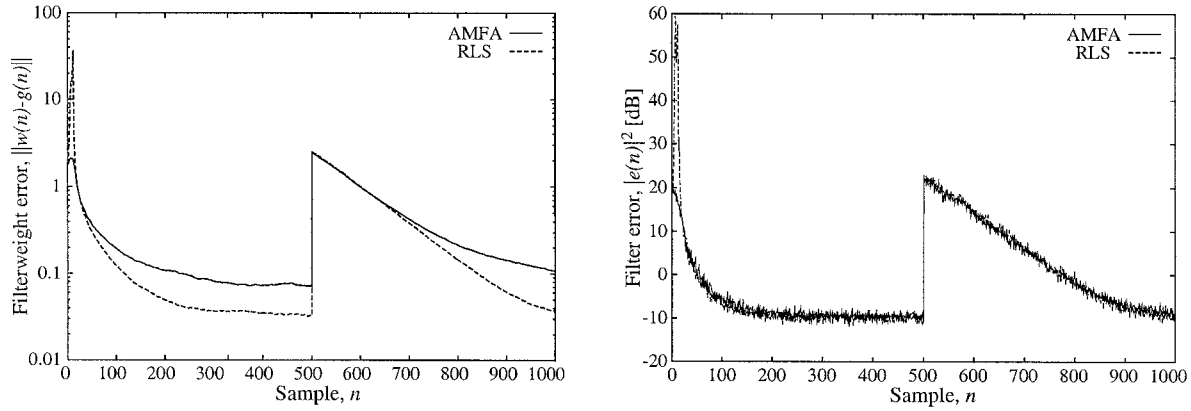

Fig. 4. A comparison of the response of the AMFA (solid) and the RLS (dashed) algorithm to an abrupt system change. The left plot shows the mean square filter weight error, $\epsilon(n)$ and the plot on the right shows the estimation error $|e(n)|^2$, averaged over 100 simulation runs.

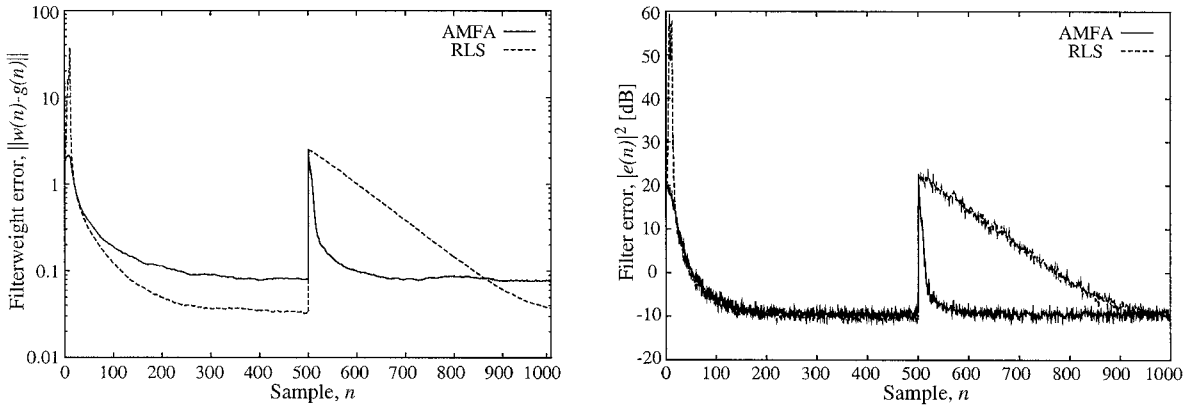

Fig. 5. A comparison of the response of the AMFA (solid) and the RLS (dashed) algorithm to an abrupt system change. The AMFA algorithm in this case re-initializes itself when a large increase in estimation error is detected. The left plot shows the mean square filter weight error, $\epsilon(n)$ and the plot on the right shows the estimation error $|e(n)|^2$, averaged over 100 simulation runs.

system changes. Fig. 5 shows the behavior of an AMFA algorithm that re-initializes itself when a large increase in estimation error is detected, compared to an ordinary RLS algorithm. When the error exceeded a threshold level, the internal state was reset to its initial state and the reset mechanism was inhibited for the following $2M$ samples.

5. CONCLUSIONS

We have presented a new approach to the adaptive filtering problem, based on the mean field annealing optimization algorithm.

It has been shown by simulations that this approach, with complexity $\mathcal{O}(M^2)$ where M is the filter length, finds a solution comparable to the recursive least squares algorithm but without the transient behavior of the latter. The robustness of the adaptive mean field annealing algorithm with respect to initialization was exploited to render a fast recovery after abrupt system changes.

6. REFERENCES

[1] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*, John Wiley & Sons, Inc., 1996.

[2] G. V. Moustakides, "Study of the transient phase of the forgetting factor rls," *IEEE Trans. Signal Processing*, vol. 45, pp. 2468–2476, October 1997.

[3] K. H. Afkhamie, Z. Q. Luo, and K. M. Wong, "Adaptive linear filtering using interior point optimization techniques," *IEEE Trans. Signal Processing*, vol. 58, no. 6, pp. 1637–1648, June 2000.

[4] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[5] C. Peterson and B. Söderberg, "A new method for mapping optimization problems onto neural networks," *International Journal of Neural Systems*, vol. 1, no. 1, pp. 3–22, Jan 1989.

[6] M. Ohlsson, C. Peterson, and B. Söderberg, "Neural networks for optimization problems with inequality constraints - the knapsack problem," *Neural Computation*, vol. 5, no. 2, pp. 331–339, 1993.

[7] G. Parisi, *Statistical Field Theory*, Perseus Books Publishing, L.L.C., 1998.

[8] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons, Inc., 1989.

[9] P. Persson, S. Nordebo, and I. Claesson, "A multimode mean field annealing technique to design recursive digital filters," *IEEE Trans. on Circuits and Systems II*, December 2001.

[10] P. Persson, S. Nordebo, and I. Claesson, "Hardware efficient digital filter design by multi-mode mean field annealing," *IEEE Signal Processing Letters*, vol. 8, no. 7, July 2001.