An Approximate Minimum BER Approach to Channel Equalisation Using Recurrent Neural Networks

Rodrigo C. de Lamare and Raimundo Sampaio-Neto
CETUC - PUC-RIO, 22453-900, Rio de Janeiro - Brazil
e-mails: delamare@infolink.com.br, raimundo@cetuc.puc-rio.br

Abstract—
In this paper we investigate the use of an approximate minimum bit error rate (MBER) approach to channel equalisation using recurrent neural networks (RNN). We examine a stochastic gradient adaptive algorithm for approximating the MBER from training data using RNN structures. A comparative analysis of linear equalisers and neural equalisers, employing minimum mean squared error (MMSE) and approximate MBER (AMBER) adaptive algorithms is carried out. Computer simulation experiments show that the neural equaliser operating with a criterion similar to the AMBER algorithm outperforms neural receivers using the MMSE criterion via gradient-type algorithms and linear receivers with MMSE and MBER techniques.

I. Introduction

Neural networks have recently been used in communication channel equalisation applications [1-6]. The use of non-linear structures can combat more effectively intersymbol interference (ISI) [5], which arise due to the multipath effect of radio signals. In the last few years, different artificial neural networks structures have been used in the design of channel equalisers: multilayer perceptrons (MLP) [3], radial-basis functions (RBF) [4], and recurrent neural networks (RNN) [5,6]. These neural systems make use of non-linear functions to create decision boundaries in order to detect transmitted symbols, whilst conventional equalisers employ linear functions to form such decision regions. Indeed, neural equalisers employing the minimum mean square error (MMSE) [1-6] criterion have become rather successful, since they usually show good performance and have simple adaptive implementation [7]. However, it is well known that the MSE cost function is not optimal in digital communications applications, and the most appropriate cost function is the bit error rate (BER) [8]. The approximate minimum bit error rate (AMBER) [8] is one of the most successful and suitable algorithms for adaptive implementation using linear receiver structures, provided the application can handle a long training sequence. In this work, we investigate the convergence and BER performance of an RNN equaliser operating with a criterion similar to the AMBER. An extension of the real time recurrent learning (RTRL) algorithm, called RTRL-AMBER, is introduced and used to train the RNN equaliser. We perform a comparative analysis of the linear equaliser using the LMS [1] and the AMBER [8] techniques with the neural equaliser employing the RTRL [2] and the RTRL-AMBER algorithms.

This paper is organised as follows. Section II briefly describes the communication system model and the adaptive equalisation problem. The RNN receiver is presented in Section III. Section IV is dedicated to the AMBER and the RTRL-AMBER algorithms. Section V presents the simulation results and Section VI the conclusions of this work.

II. System Model

We assume a BPSK communication system that transmits modulated symbols through a radio-type communication channel, which is followed by an adaptive equaliser and a symbol detector, as shown in Fig. 1. Consider a discrete time communication channel, where $x(k)$ is the ± 1 binary transmitted symbol, $h(k)$ is the channel impulse response with memory M and $n(k)$ is additive white gaussian noise (AWGN) with power spectrum density σ^2. The output signal $r(k)$ of the channel is expressed by:

$$r(k) = s(k) + n(k) = \sum_{i=0}^{M} h(i)x(k-i) + n(k) \qquad (1)$$

where $s(k)$ is the channel output without noise.

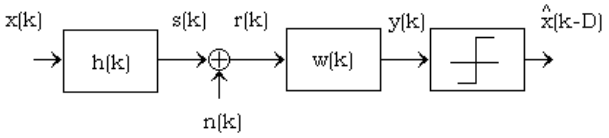

Fig. 1. Communication channel and receiver.

The channel output vector $\mathbf{r}(k) = \left[r(k) \dots r(k-N)\right]^T$ is given by:

$$\mathbf{r}(k) = \mathbf{s}(k) + \mathbf{n}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{n}(k) \qquad (2)$$

where $\mathbf{x}(k) = \left[x(k) \dots x(k-M-N)\right]^T$ is the vector with the channel inputs, $\mathbf{n}(k) = \left[n(k) \dots n(k-N)\right]^T$ is the noise sample vector, $\mathbf{s}(k) = \left[s(k) \dots s(k-N)\right]^T$ is the vector without noise and $\mathbf{H}$ is a $(N+1) \times (M+N+1)$ Toeplitz convolution matrix expressed by:

$$\mathbf{H} = \begin{pmatrix} h(0) & \dots & h(M) & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & h(0) & \dots & h(M) \end{pmatrix} \qquad (3)$$

The adaptive channel equalisation problem involves the application of a receiving filter, that adjusts its coefficients in order to minimise a given objective function [7]. The equaliser can be implemented by a linear structure such

as an FIR filter or by a non-linear system based on an artificial neural network. The equaliser must be adaptive in order to track the signal variations imposed by the channel, however, it requires a desired signal taken from a training sequence to adjust its parameters, as shown in Fig. 2.

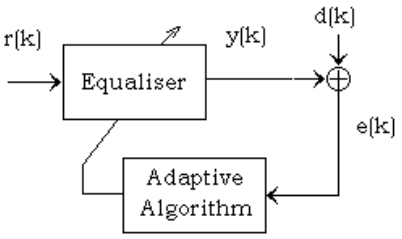

Fig. 2. Block diagram of an adaptive equaliser.

The linear transversal equaliser, depicted in Fig. 2, consists of a linear filter with $N+1$ taps described by the vector $\mathbf{w} = \begin{bmatrix} w_0 \ ... \ w_N \end{bmatrix}^T$. The equaliser output is given by:

$$y(k) = \mathbf{w}^T \mathbf{r}(k) \tag{4}$$

where $\mathbf{r}(k) = \begin{bmatrix} r(k) \ ... \ r(k-N) \end{bmatrix}^T$ is the observed output signal vector of the channel.

The decision $\hat{x}(k-D)$ on the transmitted symbol $x(k-D)$ is determined by the equaliser output signal, $\hat{x}(k-D) = sgn(y(k))$, where D corresponds to the delay imposed by the channel and the equaliser.

The adaptive equalisation solution via the LMS algorithm [1] is based upon the MMSE error criterion formed by the error signal $e(k) = d(k) - y(k)$, and is expressed by:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k) \mathbf{r}(k) \tag{5}$$

where $d(k) = x(k-D)$ is the desired signal taken from the training sequence and μ is the algorithm step size.

III. Recurrent Neural Networks

Recurrent neural networks (RNN) have one or more feedback connections, where each artificial neuron is connected to the others, as shown in Fig. 3. RNN structures are suitable to channel equalisation applications, since they are able to cope with channel transfer functions that exhibit deep spectral nulls, forming optimal decision boundaries and are less computationally demanding than MLP networks [5]. To describe RNN systems we use a state-space approach, where the $(N+1) \times 1$ vector $\mathbf{u}(k)$ corresponds to the $(N+1)$ states of the artificial network, the $(N+1) \times 1$ vector $\mathbf{r}(k)$ to the channel $(N+1)$ symbols output observation vector and the output of the neural equaliser $y(k)$ is given by:

$$\xi(k) = \begin{bmatrix} \mathbf{u}^T(k-1) \ \mathbf{r}^T(k) \end{bmatrix}^T \tag{6}$$

$$\mathbf{u}(k) = tanh(\mathbf{w'}^T(k) \xi(k)) \tag{7}$$

$$y(k) = \mathbf{C}\mathbf{u}(k) \tag{8}$$

where the $2(N+1) \times (N+1)$ matrix $\mathbf{w'}(k)$ contains the coefficients of the RNN receiver, $\mathbf{C} = \begin{bmatrix} 1 \ 0 \ ... \ 0 \end{bmatrix}$ is the $1 \times (N+1)$ matrix that defines the number of outputs of the network. Note that, in this work, we have only one output $y(k)$ per symbol, which corresponds to the equalised symbol.

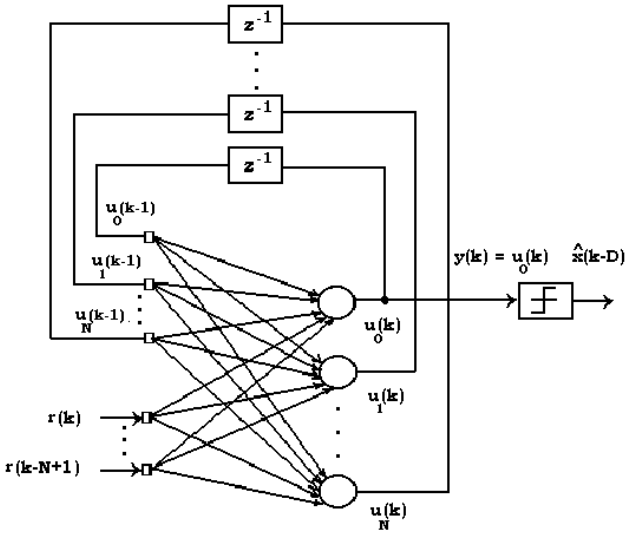

Fig. 3. Adaptive equaliser structure based on a recurrent neural network.

To train the equaliser parameters, we employ a stochastic gradient based adaptive technique called real time recurrent learning (RTRL) [2,5] algorithm. The RTRL algorithm employs the minimum mean square error criterion, where the error signal is formed by $e(k) = d(k) - y(k)$, and is expressed by:

$$\mathbf{\Phi}(k) = diag\big(sech^2(\mathbf{w'}^T(k) \xi(k))\big) \tag{9}$$

$$\mathbf{U}_j(k) = \begin{bmatrix} \mathbf{0}_u^T \ \xi^T(k) \ \mathbf{0}_l^T \end{bmatrix} \tag{10}$$

$$\mathbf{\Lambda}_j(k+1) = \mathbf{\Phi}(k) \big[\mathbf{w'}_p(k) \mathbf{\Lambda}_j(k) + \mathbf{U}_j(k) \big] \tag{11}$$

$$\triangle \mathbf{w'}_j(k) = \mu \mathbf{\Lambda}_j^T(k) \mathbf{C}^T e(k) \tag{12}$$

$$\mathbf{w'}(k+1) = \mathbf{w'}(k) + \triangle \mathbf{w'}(k) \tag{13}$$

where the index j varies from 1 to $N+1$, the state dimensionality of the neural structure. The matrices $\mathbf{\Phi}(k)$, $\mathbf{U}_j(k)$, $\mathbf{\Lambda}_j(k)$, $\mathbf{w'}_p(k)$ (a partition of $\mathbf{w'}(k)$ as described in [2]) and $\triangle \mathbf{w'}_j(k)$ have dimensions $(N+1) \times (N+1)$, $(N+1) \times 2(N+1)$, $(N+1) \times 2(N+1)$, $(N+1) \times (N+1)$ and $2(N+1) \times 1$, respectively. Note that $\mathbf{0}_u$ and $\mathbf{0}_l$ are zero valued matrices with variable size that depend on j, as detailed in [2], and whose dimensions are $(j-1) \times 2(N+1)$ and $(N+1-j) \times 2(N+1)$, respectively. The decision $\hat{x}(k-D)$ on the transmitted symbol $x(k-D)$ is determined by the neural equaliser output signal, $\hat{x}(k-D) = sgn(y(k))$, where D corresponds to the delay imposed by the channel and the equaliser.

IV. Approximate Minimum BER algorithms

In this section, we describe the approximate minimum bit error rate (AMBER) algorithm [8], and propose an extension of the real time recurrent learning (RTRL) algorithm [2] to train recurrent neural networks. The new approach, denoted RTRL-AMBER, is based on a modification of the RTRL that incorporates the AMBER update rule.

Given a transmitted training sequence $\mathbf{d}$, the bit error probability $P(\epsilon|\mathbf{d})$, for linear and neural receivers, is expressed by:

$$P\big(\epsilon|\mathbf{d}\big) = P\big(d(k)sgn(y(k)) = -1\big)$$
$$P\big(\epsilon|\mathbf{d}\big) = P\big(sgn(d(k)y(k)) = -1\big) = P\big(d(k)y(k) < 0\big) \tag{14}$$

where $y(k)$ is given by (4), for the linear equaliser, and expressed by (8), in the case of the neural equaliser and $d(k)$ is the desired symbol taken from the training sequence.

A. The AMBER algorithm

In the AMBER approach, the vector function $g(\mathbf{w}(k))$ [3] is used to approximate an expression for a coefficient vector $\mathbf{w}(k)$ that achieves a MBER performance with linear receiver structures, as described by:

$$g(\mathbf{w}(k)) = E\left[Q\left(\frac{d(k)\mathbf{w}^T(k)\mathbf{s}(k)}{\parallel \mathbf{w}(k) \parallel \sigma}\right)d(k)\mathbf{s}(k)\right] \tag{15}$$

where $d(k)$ is the desired transmitted symbol taken from the training sequence and $Q(.)$ is the Gaussian error function. A simple stochastic solution for $\mathbf{w}$ can be derived by using $g(\mathbf{w}(k))$ and adjusting the receiver weights by:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu g(\mathbf{w}(k)) \tag{16}$$

Note that for linear receiver structures the quantity $Q\Big(\frac{d(k)\mathbf{w}^T(k)\mathbf{s}(k)}{\|\mathbf{w}(k)\|\sigma}\Big)$ inside the expected value operator in (15) corresponds to the conditional bit error probability given the product $d(k)\mathbf{s}(k)$. This quantity can be replaced in (15) by an error indicator function $i_d(k)$ given by:

$$i_d(k) = \frac{1}{2}(1 - sgn(d(k)y(k))) \tag{17}$$

where $y(k)$ is the estimated symbol and $d(k)$ is the desired signal provided by the training sequence. Following this approach, the AMBER algorithm, as devised for linear equalisers [8], is described by the following equalities:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu E\left[Q\left(\frac{d(k)\mathbf{w}^T(k)\mathbf{s}(k)}{\parallel \mathbf{w}(k) \parallel \sigma}\right)d(k)\mathbf{s}(k)\right]$$
$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu E\Big[E\big[i_d(k) \mid d(k)\mathbf{s}(k)\big]d(k)\mathbf{s}(k)\Big]$$
$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu E\big[i_d(k)d(k)\mathbf{s}(k)\big]$$

Since $\mathbf{s}(k) = \mathbf{r}(k) - \mathbf{n}(k)$, and $i_d(k)$ and $d(k)$ are statistically independent, we have $E[i_d(k)d(k)\mathbf{n}(k)] = E[d(k)]E[i_d(k)\mathbf{n}(k)] = 0$ and thus:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu E\Big[i_d(k)d(k)\mathbf{r}(k)\Big] \tag{18}$$

The AMBER stochastic gradient update equation is given by:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu i_d(k)d(k)\mathbf{r}(k) \tag{19}$$

Note that the expression in (5) equals (19) if we replace $e(k)$ by $i_d(k)d(k)$. In practice, a modified error indicator function $i_d(k) = \frac{1}{2}(1 - sgn(d(k)y(k) - \tau))$ is employed, where the threshold τ is responsible for increasing the algorithm rate of convergence. This algorithm updates when an error is made and also when an error is almost made, becoming a smarter choice for updating the filter coefficients.

B. The RTRL-AMBER algorithm

In this section, we use a strategy that combines the strengths of neural networks, in dealing with deep spectral nulls and creating optimal decision boundaries, and the AMBER algorithm, that only updates the weights of the receiver when an error is made or almost made. Even though the AMBER algorithm was devised for linear equalisers, the principle of its update rule can be extended to neural equalisers. We adopt a criterion similar to the AMBER algorithm to be used in RNN structures, by modifying the weight update rule of the RTRL technique, described in (12). This strategy is based on the substitution of the error $e(k)$ by $i_d(k)d(k)$. The RTRL-AMBER algorithm is described by the expression:

$$\triangle \mathbf{w'}_j(k) = \mu \mathbf{\Lambda}_j^T(k) C^T i_d(k) d(k) \tag{19}$$

Note that the RTRL-AMBER algorithm employs the updating principles of the AMBER algorithm. Although it does not exactly minimise the BER nor was it devised taking in consideration the non-linear nature of the RNN structure, the simulations verify its superior convergence and BER performances.

V. Simulations

In this section, we conduct simulation experiments to assess the convergence and the BER performance of the equalisers operating with the algorithms described and perform a comparative analysis of them. To evaluate the receivers, we have designed transversal equalisers and simulated their operation under different radio-type communication channels.

A. Convergence Analysis

The simulation experiments, conducted to assess the convergence of the different structures and algorithms, employed 1000 training data bits averaged over 100 independent experiments. Furthermore, we use a small fixed threshold $\tau = 0.1$ for the AMBER algorithm in order to increase its convergence rate. In the first situation, the

equalisers operate with the receiver observing 6 symbols at each instant of time, with a step size $\mu = 0.005$. We consider a linear channel with transfer function $H(z) = 1-0.25z^{-1}+0.4z^{-2}$. The bit error rate (BER) is measured at each received symbol for the different receivers. Fig. 4 shows the convergence performance of the algorithms to adjust the filter parameters.

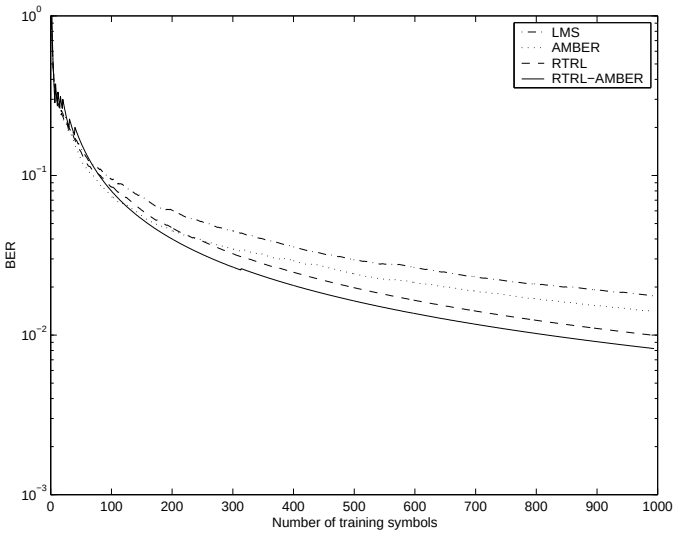

Fig. 4. Convergence performance for channel $H(z) = 1 - 0.25z^{-1} + 0.4z^{-2}$ with $\frac{E_b}{N_0} = 15dB$.

In the second experiment, the equalisers operate with the receiver observing 8 symbols at each instant of time using a step size $\mu = 0.0075$. We consider a linear channel with transfer function $H(z) = 1.1+1.2z^{-1}-0.2z^{-2}$. As occurs in the first experiment the BER is measured at each received symbol for the different receivers, as shown in Fig. 5.

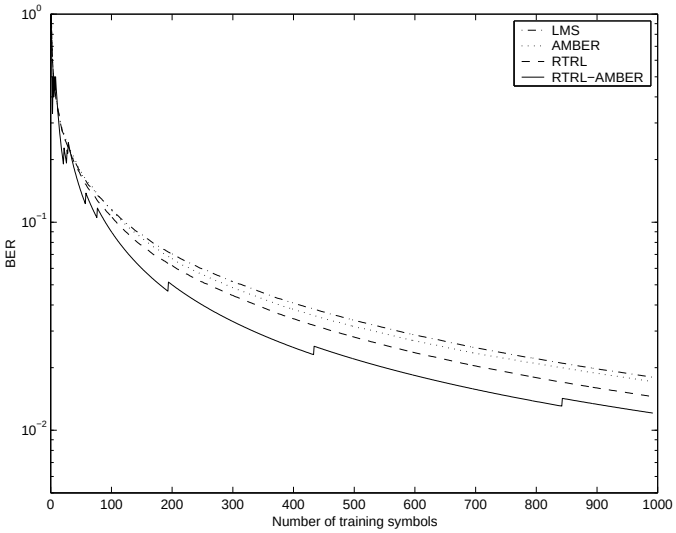

Fig. 5. Convergence performance for channel $H(z) = 1.1 + 1.2z^{-1} - 0.2z^{-2}$ with $\frac{E_b}{N_0} = 18dB$.

According to the curves plotted in Figs. 4 and 5, the linear equaliser operating with the AMBER algorithm is superior to the linear equaliser updated via the LMS. The neural receiver using the RTRL algorithm has a better convergence performance than the linear structures, whereas the RTRL-AMBER neural receiver achieves the best convergence performance.

B. BER Performance

The BER simulation results were obtained with 1000 training data bits and 10^4 data bits averaged over 100 independent experiments. All equalisers operate with a step size μ during training and no adaptation occurs in data mode. In addition, we use a small fixed threshold $\tau = 0.1$ for the AMBER algorithm to increase its convergence rate.

In the first experiment, the equalisers operate with the receiver observing 6 symbols at each instant of time, which corresponds to 6 taps in the case of the linear equalisers, with a step size $\mu = 0.005$. We consider a linear channel with transfer function $H(z) = 1 - 0.25z^{-1} + 0.4z^{-2}$. Fig. 6 shows the BER performance of the receivers.

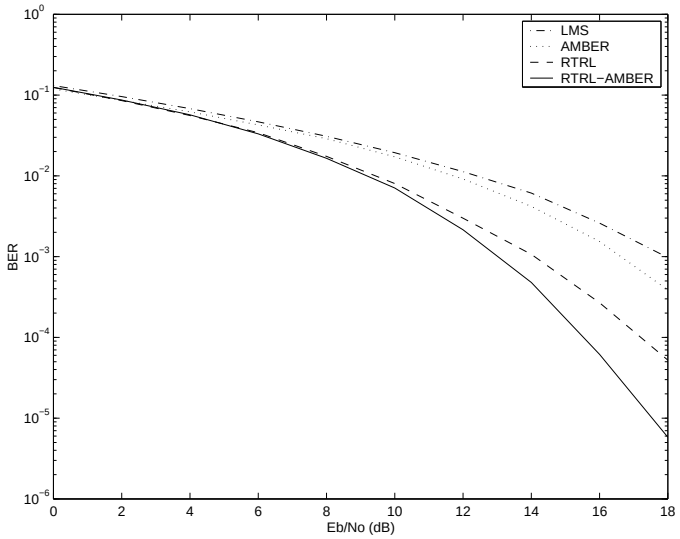

Fig. 6. BER performance for channel $H(z) = 1 - 0.25z^{-1} + 0.4z^{-2}$.

In the second experiment, the equalisers operate with the receiver observing 8 symbols at each instant of time, which corresponds to 8 taps in the case of the linear equalisers, with a step size $\mu = 0.0075$. We consider a linear channel with transfer function $H(z) = 1.1 + 1.2z^{-1} - 0.2z^{-2}$. Fig. 7 shows the BER performance of the receivers.

According to Figs. 6 and 7 the neural equaliser operating with the RTRL algorithm is superior to the linear equaliser updated via the AMBER and the LMS algorithms, at the expense of a higher computational complexity. The neural receiver using the RTRL-AMBER algorithm achieves the best BER and convergence performance, amongst the examined systems. It can save up to 2 dB in comparison with the RTRL approach, for the same BER performance. The proposed RTRL-AMBER algorithm has shown convergence and BER performances superior to the ones achieved by the RTRL technique, whilst requiring a lower compu-

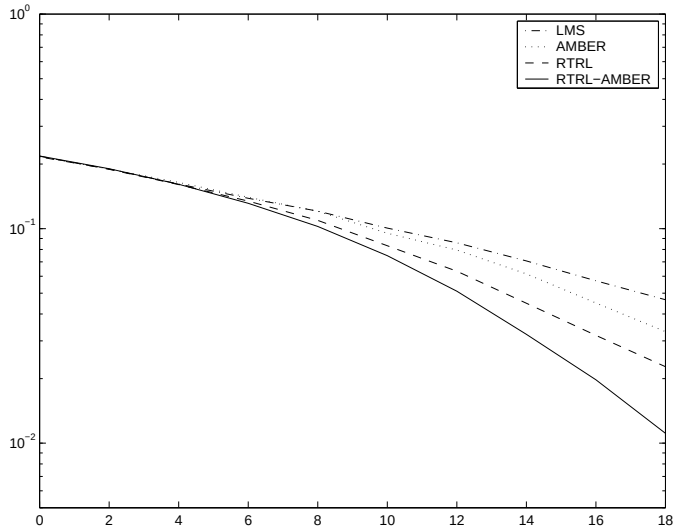

Fig. 7. BER performance for channel $H(z) = 1.1 + 1.2z^{-1} - 0.2z^{-2}$.

tational complexity, since due to the presence of the error indicator function weight updating occurs less frequently.

VI. Conclusions

We have examined the use of an approximate minimum bit error rate (MBER) approach to channel equalisation using recurrent neural networks. We carried out a comparative analysis of linear transversal equalisers, employing the LMS and the AMBER algorithms, and neural equalisers, employing the RTRL and the proposed RTRL-AMBER algorithms. Computer simulation experiments have demonstrated that a neural equaliser operating with the proposed RTRL-AMBER algorithm is superior to the neural receiver with the RTRL technique and to the linear receivers adjusted via the LMS and the AMBER algorithms. In comparison with the RTRL, the proposed RTRL-AMBER can save up to 2 dB, for the same BER performance, requiring a lower computational complexity than the RTRL method.

References

[1] S. Haykin, *Adaptive Filter Theory*, 3rd edition, Prentice-Hall, Englewood Cliffs, NJ, 1996.

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

[3] G. J. Gibson, S. Siu, and C. F. Cowan, "The application of nonlinear structures to the reconstruction of binary signals", *IEEE Transactions on Signal Processing*, vol. 39, no. 8, pp. 1877-1884, August de 1991.

[4] I. Cha e S. A. Kassam, "Channel equalization using adaptive complex radial basis function networks", *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, oo. 122-131, January de 1995.

[5] G. Kechriotis, E. Zervas e E. S. Manolakos, "Using Recurrent Neural Networks for Adaptive Communication Channel Equalization", *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 267-278, March 1994.

[6] M. J. Bradley and P. Mars, "Application of Recurrent Neural Networks to Communication Channel Equalization", *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1995.

[7] S. U. H. Qureshi, "Adaptive equalization", *Proceedings of IEEE*, vol. 73, pp. 1349-1387, 1985.

[8] C. Yeh and J. R. Barry, "Approximate minimum bit-error rate equalization for binary signaling", *Proc IEEE International Conference on Communications*, vol. 1, 1998, pp. 16-20.