EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

A low complexity Hopfield neural network turbo equalizer

Hermanus C Myburgh[1*] and Jan C Olivier[2]

Abstract

In this article, it is proposed that a Hopfield neural network (HNN) can be used to jointly equalize and decode information transmitted over a highly dispersive Rayleigh fading multipath channel. It is shown that a HNN MLSE equalizer and a HNN MLSE decoder can be merged in order to realize a low complexity joint equalizer and decoder, or turbo equalizer, without additional computational complexity due to the decoder. The computational complexity of the Hopfield neural network turbo equalizer (HNN-TE) is almost quadratic in the coded data block length and approximately independent of the channel memory length, which makes it an attractive choice for systems with extremely long memory. Results show that the performance of the proposed HNN-TE closely matches that of a conventional turbo equalizer in systems with short channel memory, and achieves near-matched filter performance in systems with extremely large memory.

Keywords: Turbo equalizer, Hopfield neural network, Rayleigh fading, Low complexity

1 Introduction

Turbo equalization has its roots in turbo coding, first proposed in [1] for the iterative decoding of concatenated convolutional codes. In [2,3], the idea of turbo decoding was applied to systems transmitting convolutional coded information through multipath channels, in order to improve the bit-error rate (BER) performance, with great success. Due to the computational complexity of its constituent maximum a posteriori (MAP) equalizer and MAP decoder, the computational complexity of these turbo equalizers are exponentially related to the channel impulse response (CIR) length as well as the encoder constraint length, limiting their effective use in systems where the channel memory and/or the encoder constraint length is large, with the MAP equalizer being the main culprit due to long channel delay spreads.

To mitigate the high computational complexity exhibited by the MAP equalizer, several authors have proposed suboptimal equalizers to replace the optimal MAP equalizer in the Turbo Equalizer structure, with complexity that is linearly related to the channel memory length. In [4,5], it was shown how a minimum mean squared error (MMSE) equalizer is used in a Turbo Equalizer by modifying it to make use of prior information provided in the form of extrinsic information. Various authors have also proposed the use of decision feedback equalizers (DFE) while using extrinsic information as prior information to improve the BER performance after each iteration [6-10]. Also, in [11,12] it was proposed that a soft interference canceler (SIC) be modified to make use of soft information in order to be used as a low complexity equalizer in a turbo equalizer, and in [13] the way in which a SIC incorporates soft information was modified to improve performance. The proposed equalizers inherently suffer from noise enhancement (MMSE) and error propagation (DFE and SIC) which limit their performance, and hence the overall performance of the turbo equalizers in which they are used. Due to the fact that none of the proposed equalizers are able to produce exact MAP estimates of the transmitted coded information, the performance of the Turbo Equalizer in which they are implemented will ultimately be worse than when an optimal MAP equalizer is utilized, due to the performance loss incurred at the output of these suboptimal equalizers. This trade-off always exists: If one gains in terms complexity, one loses in terms of performance.

In this article, we propose to combat the performance loss due to suboptimal (or non-MAP) equalizer output,

*Correspondence: herman.myburgh@up.ac.za
[1] Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria, 0002, South Africa
Full list of author information is available at the end of the article

by combining the equalizer and the decoder into one equalizer/decoder structure, so that all information can be processed as a whole, and not be passed between the equalizer and the decoder. This vision has successfully been implemented and demonstrated by the authors in [14] using a dynamic Bayesian network (DBN) as basis. In this paper, however, we show that using the Hopfield neural network (HNN) [15] as the underlying structure also works well, and has a number of advantages as discussed in [16].

In [16], the authors proposed a maximum likelihood sequence estimation (MLSE) equalizer which is able to equalize M-ary quadrature amplitude modulation (M-QAM) modulated signals in systems with extremely long memory. The complexity of the equalizer proposed in [16] is quadratic in the data block length and approximately independent of the channel memory length. Its superior computational complexity is due to the high parallelism of its underlying neural network structure. It uses the HNN structure which enables fast parallel processing of information between neurons, producing ML sequence estimates at the output. It was shown in [16] that the performance of the HNN MLSE equalizer closely matches that of the Viterbi MLSE equalizer in short channels, and near-optimally recombines the energy spread across the channel in order to achieve near-matched filter performance when the channel is extremely long.

The HNN has also been shown by several authors to be able to decode balanced check codes [17,18]. These codes, together with methods for encoding and decoding, were first proposed in [19], but it was later shown in [17,18] that single codeword decoding can also be performed using the HNN. To date, balanced codes is the only class of codes that can be decoded with the HNN. The ability of the HNN to detect binary patterns allows it to determine the ML codeword from a predefined set of codewords. In this paper it is shown that the HNN ML decoder can be extended to allow for the ML estimation of a sequence of balanced check codes. It is therefore extendable to an MLSE decoder.

In this article, a novel turbo equalizer is developed by combining the HNN MLSE equalizer developed in [16] and a HNN MLSE decoder (used to decode balanced codes, and only balanced codes), resulting in the Hopfield neural network turbo equalizer (HNN-TE), which can be used as replacement for a conventional turbo equalizer (CTE), made up of a equalizer/decoder pair, in systems with extremely long memory, where the coded symbols are interleaved before transmission through the multipath channel. The HNN-TE is able to equalize and decode (balanced codes) in systems with extremely long memory, since the computational complexity is nearly independent of the channel memory length. Like the HNN MLSE equalizer, its superior complexity characteristics are due to the high parallelism of its underlying neural network structure.

This article is structured as follows. Section 2 presents a brief discussion on Turbo Equalization. Section 3 discusses the HNN in general, while the HNN MLSE equalizer and the HNN MLSE decoder are discussed in Section 4, followed by a discussion on the fusion of the two in order to realize the HNN-TE. In Section 5, the results of a computational complexity analysis of the HNN-TE and a CTE are presented, followed by a memory requirements analysis in Section 6. Simulation results are presented in Section 7 and conclusions are drawn in Section 8.

2 Turbo equalization

Turbo equalizers are used in multipath communication systems that make use of encoders, usually convolutional encoders, to encoded the source symbol sequence $\mathbf{s}$ of length N_u (using some generator matrix $\mathbf{G}$) at a rate R_c to produce coded information symbols $\mathbf{c}$ of length $N_c = N_u/R_c$, after which the coded symbols $\mathbf{c}$ are interleaved with a random interleaver before modulation and transmission. The interleaved coded symbols $\acute{\mathbf{c}}$ are transmitted through a multipath channel with a CIR length of L, causing inter-symbol interference among adjacent transmitted symbols at the receiver. At the receiver the received inter-symbol interference (ISI) corrupted coded symbols are matched filtered and used as input to the turbo equalizer. The received symbol sequence is given by

$$\mathbf{r} = \mathbf{H}\acute{\mathbf{c}} + \mathbf{n}, \tag{1}$$

where $\mathbf{n}$ is a vector containing complex Gaussian noise samples and $\acute{\mathbf{c}}$ is the interleaved coded symbols given by

$$\acute{\mathbf{c}} = \mathbf{J}\mathbf{G}^T\mathbf{s}, \tag{2}$$

where $\mathbf{J}$ is an $N_c \times N_c$ interleaver matrix, and $\mathbf{H}$ is the $N_c \times N_c$ channel matrix

$$\mathbf{H} = \begin{bmatrix} h_0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & h_0 & \dots & 0 & 0 & 0 & 0 \\ h_{L-1} & \vdots & \ddots & 0 & 0 & 0 & 0 \\ 0 & h_{L-1} & \ddots & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & h_0 & 0 & 0 \\ 0 & 0 & 0 & h_{L-1} & \dots & h_0 & 0 \\ 0 & 0 & 0 & 0 & h_{L-1} & \dots & h_0 \end{bmatrix}. \tag{3}$$

The turbo equalizer uses two a maximum a posterior (MAP) algorithms, one to equalize the ISI-corrupted received symbols and one to decode the equalized coded symbols, which iteratively exchange information. With each iteration of the system, extrinsic information is exchanged between the two MAP algorithms in order to improve the ability of each algorithm to produce correct

estimates. This principle was first applied to Turbo Coding, where both MAP algorithms were MAP decoders [3], but has since been applied to iterative equalization and decoding (today known as Turbo Equalization) to reduce the BER performance of the coded multipath communication system [2-5].

Figure 1 shows the structure of the Turbo Equalizer. The MAP equalizer takes as input the ISI-corrupted received symbols $\mathbf{r}$ and the extrinsic information $L_e^D(\hat{\mathbf{s}})$ (where $\hat{\mathbf{s}}$ the interleaved coded symbol estimates) and produces a sequence of posterior transmitted symbol log-likelihood ratio (LLR) estimates $L^E(\hat{\mathbf{s}})$ (note that $L_e^D(\hat{\mathbf{s}})$ is zero during the first iteration). Extrinsic information $L_e^E(\hat{\mathbf{s}})$ is determined by

$$L_e^E(\hat{\mathbf{s}}) = L^E(\hat{\mathbf{s}}) - L_e^D(\hat{\mathbf{s}}), \tag{4}$$

which is deinterleaved to produce $L_e^E(\hat{\mathbf{s}}')$, which is used as input to the MAP decoder to produce a sequence of posterior coded symbol LLR estimates $L^D(\hat{\mathbf{s}}')$. $L^D(\hat{\mathbf{s}}')$ is used together with $L_e^E(\hat{\mathbf{s}}')$ to determine the extrinsic information

$$L_e^D(\hat{\mathbf{s}}') = L^D(\hat{\mathbf{s}}') - L_e^E(\hat{\mathbf{s}}'), \tag{5}$$

$L_e^D(\hat{\mathbf{s}}')$ is interleaved to produce $L_e^D(\hat{\mathbf{s}})$. $L_e^D(\hat{\mathbf{s}})$ is used together with the received symbols $\mathbf{r}$ in the MAP equalizer, with $L_e^D(\hat{\mathbf{s}})$ serving to provide prior information on the received symbols. The equalizer again produces posterior information $L^E(\hat{\mathbf{s}})$ of the interleaved coded symbols. This process continues until the outputs of the decoder settle, or until a predefined stop-criterion is met [3]. After termination, the output $L(\hat{\mathbf{u}})$ of the decoder gives an estimate of the source symbols.

The proposed HNN-TE is modeled on one HNN structure, implying that there is no exchange of extrinsic information between its constituent parts. Rather, all information is intrinsically processed in an iterative fashion.

3 The Hopfield neural network

The HNN was first proposed in [15] and it was shown in that the HNN can be used to solve combinatorial optimization problems as well as pattern recognition problems. In [15] Tank and Hopfield derived an energy function and showed how the HNN can be used to minimize this energy function, thus producing near-ML sequence estimates at the output of the neurons. To enable the HNN to solve an optimization problem, the cost function of that problem is mapped to the HNN energy function, where after the HNN iteratively minimizes its energy function and performs near-MLSE. Also, to enable the HNN to solve a binary pattern recognition problem, the autocorrelation matrix of the set of patterns is used as the weights between the HNN neurons, while the noisy pattern to be recognized is used as the input to the HNN. Again, the HNN iteratively performs pattern recognition in order to produce the near-ML patter at the output of the HNN.

3.1 Energy function

The Hopfield energy function can be written as [16]

$$\mathcal{L} = -\frac{1}{2}\mathbf{s}^T\mathbf{X}\mathbf{s} - \mathbf{I}^T\mathbf{T}\mathbf{s}, \tag{6}$$

where $\mathbf{I}$ is a column vector with N elements, $\mathbf{X}$ is an $N \times N$ matrix. Assuming that $\mathbf{s}$, $\mathbf{I}$, and $\mathbf{X}$ contain complex values, these variables can be written as [16]

$$\begin{aligned} \mathbf{s} &= \mathbf{s}_i + j\mathbf{s}_q, \\ \mathbf{I} &= \mathbf{I}_i + j\mathbf{I}_q, \\ \mathbf{X} &= \mathbf{X}_i + j\mathbf{X}_q, \end{aligned} \tag{7}$$

where $\mathbf{s}$ and $\mathbf{I}$ are column vectors of length N, and $\mathbf{X}$ is an $N \times N$ matrix, where subscripts i and q are used to denote the respective in-phase and quadrature components. $\mathbf{X}$ is the cross-correlation matrix of the complex received symbols such that

$$\mathbf{X}^H = \mathbf{X}_i^T - j\mathbf{X}_q^T = \mathbf{X}_i + j\mathbf{X}_q, \tag{8}$$

implying that it is Hermitian. Therefore $\mathbf{X}_i^T = \mathbf{X}_i$ is symmetric and $\mathbf{X}_q^T = -\mathbf{X}_q$ is skew symmetric [16]. By using the symmetric properties of $\mathbf{X}_i$ and $\mathbf{X}_q$, (6) can be expanded and rewritten as

$$\mathcal{L} = -\frac{1}{2}\left[\mathbf{s}_i^T\mathbf{X}_i\mathbf{s}_i + \mathbf{s}_q^T\mathbf{X}_q\mathbf{s}_q + 2\mathbf{s}_q^T\mathbf{X}_q\mathbf{s}_i\right] - \left[\mathbf{s}_i^T\mathbf{I}_i + \mathbf{s}_q^T\mathbf{I}_q\right]$$

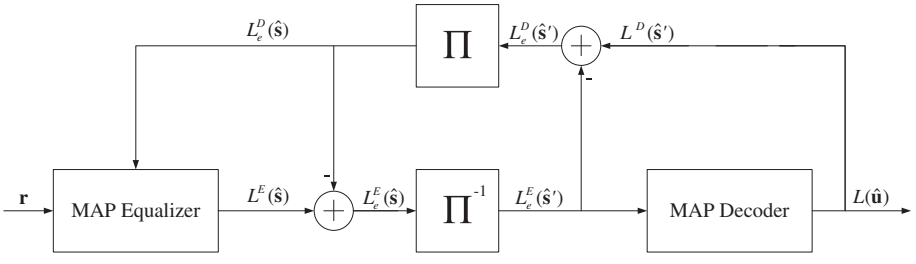

Figure 1 Turbo equalizer. Shows the structure of the turbo equalizer.

which in turn can be rewritten as [16]

$$\mathcal{L} = -\frac{1}{2}\left[\mathbf{s}_i^T|\mathbf{s}_q^T\right]\begin{bmatrix}\mathbf{X}_i & \mathbf{X}_q^T\\ \mathbf{X}_q & \mathbf{X}_i\end{bmatrix}\begin{bmatrix}\mathbf{s}_i\\ \mathbf{s}_q\end{bmatrix} - \left[\mathbf{I}_i^T|\mathbf{I}_q^T\right]\begin{bmatrix}\mathbf{s}_i\\ \mathbf{s}_q\end{bmatrix}. \quad (9)$$

It is clear that (9) is in the form of (6), where the variables in (6) are substituted as follows:

$$\begin{aligned}\mathbf{s}^T &= \left[\mathbf{s}_i^T|\mathbf{s}_q^T\right],\\ \mathbf{I}^T &= \left[\mathbf{I}_i^T|\mathbf{I}_q^T\right],\\ \mathbf{X} &= \begin{bmatrix}\mathbf{X}_i & \mathbf{X}_q^T\\ \mathbf{X}_q & \mathbf{X}_i\end{bmatrix}.\end{aligned} \quad (10)$$

Equation (9) is used to derive the HNN MLSE equalizer, decoder, and eventually the HNN-TE.

3.2 Iterative system

The HNN minimizes the energy function (6) with the following iterative system:

$$\begin{aligned}\mathbf{u}^{(i)} &= \mathbf{T}\mathbf{s}^{(i)} + \mathbf{I}\\ \mathbf{s}^{(i+1)} &= g\left(\beta(i)\mathbf{u}^{(i)}\right),\end{aligned} \quad (11)$$

where $\mathbf{u} = \{u_1, u_2, \ldots, u_N\}^T$ is the internal state of the HNN, $\mathbf{s} = \{s_1, s_2, \ldots, s_N\}^T$ is the vector of estimated symbols, $g(.)$ is the decision function associated with each neuron and i indicates the iteration number. $\beta(.)$ is a function used for optimization as in [14].

The estimated symbol vector $\left[\mathbf{s}_i^T|\mathbf{s}_q^T\right]$ is updated with each iteration. $\left[\mathbf{I}_i^T|\mathbf{I}_q^T\right]$ contains the best blind estimate for $\mathbf{s}$, and is therefore used as input to the network, while $\begin{bmatrix}\mathbf{X}_i & \mathbf{X}_q^T\\ \mathbf{X}_q & \mathbf{X}_i\end{bmatrix}$ contains the cross-correlation information of the received symbols. The system produces the MLSE estimates in $\mathbf{s}$ after Z iterations.

4 The Hopfield neural network turbo equalizer

In this section, the derivation of the HNN-TE is discussed, by first deriving its constituent parts—the HNN MLSE equalizer and the HNN MLSE decoder—and then showing how the HNN-TE is finally realized by combining the two.

4.1 HNN MLSE equalizer

The HNN MLSE equalizer was developed by the authors in [16]. The HNN MLSE equalizer was applied to single-carrier M-QAM modulated system with extremely long memory, where the CIR length was as long as $L = 250$, even though this is not a limit. The ability of the HNN MLSE equalizer to equalize signals in systems with highly dispersive channels is due to the fact that its complexity grows quadratically with an increase in transmitted data block size, and that it is approximately independent of the channel memory length. In the following the HNN MLSE equalizer developed in [16] will be presented, without spending time on the derivation.

It was shown in [16] that the correlation matrices $\mathbf{X}_i$ and $\mathbf{X}_q$ in (10), for a single carrier system transmitting a data block of length N through a multipath channel of length L with the data block initiated and terminated by $L-1$ known tail symbols, with values 1 for BPSK modulation and $\frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}}$ for M-QAM modulation, can be determined by

$$\mathbf{X}_i = -\begin{bmatrix}0 & \alpha_1 & \ldots & \alpha_{L-1} & \ldots & 0\\ \alpha_1 & 0 & \alpha_1 & \ldots & \ddots & \vdots\\ \vdots & \alpha_1 & 0 & \ddots & \vdots & \alpha_{L-1}\\ \alpha_{L-1} & \vdots & \ddots & \ddots & \alpha_1 & \vdots\\ \vdots & \ddots & \ldots & \alpha_1 & 0 & \alpha_1\\ 0 & \ddots & \alpha_{L-1} & \ldots & \alpha_1 & 0\end{bmatrix} \quad (12)$$

and

$$\mathbf{X}_q = -\begin{bmatrix}0 & \gamma_1 & \ldots & \gamma_{L-1} & \ldots & 0\\ \gamma_1 & 0 & \gamma_1 & \ldots & \ddots & \vdots\\ \vdots & \gamma_1 & 0 & \ddots & \vdots & \gamma_{L-1}\\ \gamma_{L-1} & \vdots & \ddots & \ddots & \gamma_1 & \vdots\\ \vdots & \ddots & \ldots & \gamma_1 & 0 & \gamma_1\\ 0 & \ddots & \gamma_{L-1} & \ldots & \gamma_1 & 0\end{bmatrix} \quad (13)$$

where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_{L-1}\}$ and $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{L-1}\}$ are respectively, determined by

$$\alpha_k = \sum_{j=0}^{L-k-1} h_j^{(i)} h_{j+k}^{(i)} + \sum_{j=0}^{L-k-1} h_j^{(q)} h_{j+k}^{(q)}, \quad (14)$$

and

$$\gamma_k = \sum_{j=0}^{L-k-1} h_j^{(q)} h_{j+k}^{(i)} - \sum_{j=0}^{L-k-1} h_j^{(i)} h_{j+k}^{(q)}, \quad (15)$$

where $k = 1, 2, 3, \ldots, L-1$ and i and q denote the in-phase and quadrature components of the CIR coefficients.

Upon inspection it is easy to see from (12) through (15) that $\mathbf{X}_i$ and $\mathbf{X}_q$ can be determined using the respective in-phase and quadrature components of the $N \times N$ channel matrix, with the in-phase and quadrature

components of the CIR, $\mathbf{h}^{(i)} = \{h_0^{(i)}, h_1^{(i)}, \ldots, h_{L-1}^{(i)}\}^T$ and $\mathbf{h}^{(q)} = \{h_0^{(q)}, h_1^{(q)}, \ldots, h_{L-1}^{(q)}\}^T$, on the diagonals such that

$$\mathbf{H}^{(i)} = \begin{bmatrix} h_0^{(i)} & 0 & \ldots & 0 & 0 & 0 & 0 \\ \vdots & h_0^{(i)} & \ldots & 0 & 0 & 0 & 0 \\ h_{L-1}^{(i)} & \vdots & \ddots & 0 & 0 & 0 & 0 \\ 0 & h_{L-1}^{(i)} & \ddots & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & h_0^{(i)} & 0 & 0 \\ 0 & 0 & 0 & h_{L-1}^{(i)} & \ldots & h_0^{(i)} & 0 \\ 0 & 0 & 0 & 0 & h_{L-1}^{(i)} & \ldots & h_0^{(i)} \end{bmatrix} \tag{16}$$

and

$$\mathbf{H}^{(q)} = \begin{bmatrix} h_0^{(q)} & 0 & \ldots & 0 & 0 & 0 & 0 \\ \vdots & h_0^{(q)} & \ldots & 0 & 0 & 0 & 0 \\ h_{L-1}^{(q)} & \vdots & \ddots & 0 & 0 & 0 & 0 \\ 0 & h_{L-1}^{(q)} & \ddots & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & h_0^{(q)} & 0 & 0 \\ 0 & 0 & 0 & h_{L-1}^{(q)} & \ldots & h_0^{(q)} & 0 \\ 0 & 0 & 0 & 0 & h_{L-1}^{(q)} & \ldots & h_0^{(q)} \end{bmatrix}. \tag{17}$$

Using $\mathbf{H}^{(i)}$ and $\mathbf{H}^{(q)}$ the correlation matrices in (12) and (13) can be determined by

$$\mathbf{X}_i = -\left(\mathbf{H}^{(i)T}\mathbf{H}^{(i)} + \mathbf{H}^{(q)T}\mathbf{H}^{(q)}\right) \tag{18}$$

which is simply

$$\mathbf{X}_i = -\mathrm{Re}\{\mathbf{H}^T\mathbf{H}\}. \tag{19}$$

Also

$$\mathbf{X}_q = -\left(\mathbf{H}^{(q)T}\mathbf{H}^{(i)} - \mathbf{H}^{(i)T}\mathbf{H}^{(q)}\right)^T, \tag{20}$$

which is

$$\mathbf{X}_q = -\mathrm{Im}\{\mathbf{H}^T\mathbf{H}\}. \tag{21}$$

$\mathbf{X}_i$ and $\mathbf{X}_q$ are then used to construct the combined correlation matrix in (10).

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_i & \mathbf{X}_q^T \\ \mathbf{X}_q & \mathbf{X}_i \end{bmatrix}. \tag{22}$$

It was also shown in [16] that the input vectors $\mathbf{I}_i$ and $\mathbf{I}_q$ in (10) are determined by

$$\mathbf{I}_i = \begin{bmatrix} \lambda_1 - \rho(\alpha_1 + \gamma_1 + \cdots + \alpha_{L-1} + \gamma_{L-1}) \\ \lambda_2 - \rho(\alpha_2 + \gamma_2 + \cdots + \alpha_{L-1} + \gamma_{L-1}) \\ \lambda_3 - \rho(\alpha_3 + \gamma_3 + \cdots + \alpha_{L-1} + \gamma_{L-1}) \\ \vdots \quad \vdots \quad \vdots \\ \lambda_{L-1} - \rho(\alpha_{L-1} + \gamma_{L-1}) \\ \lambda_L \\ \vdots \quad \vdots \quad \vdots \\ \lambda_{N-L+1} \\ \lambda_{N-L+2} - \rho(\alpha_{L-1} - \gamma_{L-1}) \\ \vdots \quad \vdots \quad \vdots \\ \lambda_{N-2} - \rho(\alpha_3 - \gamma_3 + \cdots + \alpha_{L-1} - \gamma_{L-1}) \\ \lambda_{N-1} - \rho(\alpha_2 - \gamma_2 + \cdots + \alpha_{L-1} - \gamma_{L-1}) \\ \lambda_N - \rho(\alpha_1 - \gamma_1 + \cdots + \alpha_{L-1} - \gamma_{L-1}) \end{bmatrix} \tag{23}$$

and

$$\mathbf{I}_q = \begin{bmatrix} \omega_1 - \rho(\alpha_1 - \gamma_1 + \cdots + \alpha_{L-1} - \gamma_{L-1}) \\ \omega_2 - \rho(\alpha_2 - \gamma_2 + \cdots + \alpha_{L-1} - \gamma_{L-1}) \\ \omega_3 - \rho(\alpha_3 - \gamma_3 + \cdots + \alpha_{L-1} - \gamma_{L-1}) \\ \vdots \quad \vdots \quad \vdots \\ \omega_{L-1} - \rho(\alpha_{L-1} - \gamma_{L-1}) \\ \omega_L \\ \vdots \quad \vdots \quad \vdots \\ \omega_{N-L+1} \\ \omega_{N-L+2} - \rho(\alpha_{L-1} + \gamma_{L-1}) \\ \vdots \quad \vdots \quad \vdots \\ \omega_{N-2} - \rho(\alpha_3 + \gamma_3 + \cdots + \alpha_{L-1} + \gamma_{L-1}) \\ \omega_{N-1} - \rho(\alpha_2 + \gamma_2 + \cdots + \alpha_{L-1} + \gamma_{L-1}) \\ \omega_N - \rho(\alpha_1 + \gamma_1 + \cdots + \alpha_{L-1} + \gamma_{L-1}) \end{bmatrix}, \tag{24}$$

where $\rho = 1/\sqrt{2}$ for M-QAM modulation, $\rho = 1$ in $\mathbf{I}_i$ and $\rho = 0$ in $\mathbf{I}_q$ for BPSK modulation, and $\mathbf{\Lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ is determined by

$$\lambda_k = \sum_{j=0}^{L-1} r_{j+k}^{(i)} h_j^{(i)} + \sum_{j=0}^{L-1} r_{j+k}^{(q)} h_j^{(q)}, \tag{25}$$

and $\mathbf{\Omega} = \{\omega_1, \omega_2, \ldots, \omega_N\}$ is determined by

$$\omega_k = \sum_{j=0}^{L-1} r_{j+k}^{(q)} h_j^{(i)} - \sum_{j=0}^{L-1} r_{j+k}^{(i)} h_j^{(q)}, \tag{26}$$

where $k = 1, 2, 3, \ldots, N$ with i and q again denoting the in-phase and quadrature components of the respective

elements. The combined input vector in (10) is therefore constructed as

$$\mathbf{I} = \left[\frac{\mathbf{I}_i}{\mathbf{I}_q}\right]. \tag{27}$$

Note that $\mathbf{\Lambda}$ and $\mathbf{\Omega}$ can easily be determined by

$$\mathbf{\Lambda} = \mathbf{H}^{(i)T}\mathbf{r}^{(i)} + \mathbf{H}^{(q)T}\mathbf{r}^{(q)}, \tag{28}$$

and

$$\mathbf{\Omega} = \mathbf{H}^{(i)T}\mathbf{r}^{(q)} - \mathbf{H}^{(q)T}\mathbf{r}^{(i)}, \tag{29}$$

where $\mathbf{r}^{(i)}$ and $\mathbf{r}^{(q)}$ are the respective in-phase and quadrature components of the received symbols $\mathbf{r} = \{r_1, r_2, \ldots, r_{N+L-1}\}^T$.

By deriving the cross-correlation matrix $\mathbf{X}$ and the input vector $\mathbf{I}$ in (10), the model in (9) is complete, and the iterative system in (11) can be used to equalize M-QAM modulated symbols transmitted through a channel with large CIR lengths. The HNN MLSE equalizer was evaluated in [16] for BPSK and 16-QAM with performance reaching the matched-filter bound in extremely long channels.

4.2 HNN MLSE decoder

The HNN has been shown to be able to decode balanced codes [17,18]. A binary word of length m is said to be balanced if it contains exactly $m/2$ ones and $m/2$ zeros [19]. In addition, balanced codes have the property that no codeword is contained in another word, which simply means that positions of ones in one codeword will never be a subset of the positions of ones in another codeword [19].

The encoding process is described in [19] where the first k bits of the uncoded word is flipped in order to ensure the resulting codedword is "balanced," whereafter the position k is appended to the balanced codeword before transmission. This encoding process is not followed here, as the set of $m = 2^n$ balanced codewords are determined before hand, after which encoding is performed by mapping a set of n bits to 2^n balanced binary phase-shift keying (BPSK) symbols of length 2^n, or by mapping a set of $2n$ bits to 2^n balanced quaternary quadrature amplitude modulation (4-QAM) symbols of length 2^n.

The HNN decoder developed here uses the set of predetermined codewords to determined the connection weights describing the level of connection between the neurons. It has previously been shown how a HNN can be used to decoded one balanced code at a time, but the HNN MLSE decoder we derive here is able to simultaneously decode any number of concatenated codewords in order to provide the ML transmitted sequence of codewords. After the HNN MLSE decoding, the ML BPSK or 4-QAM codewords of length 2^n are demapped to n bits (or $2n$ bits for 4-QAM), which completes the decoding process.

4.2.1 Codeword selection

The authors have found that Walsh-Hadamard codes, widely used in code division multiple access (CDMA) systems [20], are desirable codes for this application, due to their seeming balance and orthogonality characteristics. Walsh-Hadamard codes are linear codes that map n bits to 2^n codewords, where each set of codewords have a Hamming distance of 2^{n-1} and a Hamming weight of 2^{n-1}.

Walsh-Hadamard codes are not "balanced" as described above. The first codeword is always all-ones, while subsets of some codewords are contained in others, violating both restrictions for balance. Instead of using the complete set of Walsh-Hadamard codes to map n bits to 2^n codewords, a subset of codes in the Walsh-Hadamard matrix is selected, duplicated and modified so as to construct a new set of 2^n codewords of length 2^n. Consider the set of length $2^n = 8$ Walsh-Hadamard codes

$$\mathbf{H}_8 = \begin{bmatrix} 1&1&1&1&1&1&1&1 \\ 1&0&1&0&1&0&1&0 \\ 1&1&0&0&1&1&0&0 \\ 1&0&0&1&1&0&0&1 \\ 1&1&1&1&0&0&0&0 \\ 1&0&1&0&0&1&0&1 \\ 1&1&0&0&0&0&1&1 \\ 1&0&0&1&0&1&1&0 \end{bmatrix}. \tag{30}$$

To construct a set of balanced codewords from $\mathbf{H}_8$, a subset of 2^{n-1} codewords is selected, which is used as the first 2^{n-1} codewords in the new set of codewords. The second set of 2^{n-1} codewords are constructed as follows:

1. Reverse the order in which the first 2^{n-1} codewords appear in the new set.
2. Flip the bits of the reversed set of 2^{n-1} codewords.

Assuming the subset selected from $\mathbf{H}_8$ above is the set $\mathbf{H}_{8,4:7}$ (implying that codewords in rows 4 through 7 are selected), the resulting set of 2^n balanced codewords is

$$\mathbf{C}_8 = \begin{bmatrix} 1&0&0&1&1&0&0&1 \\ 1&1&1&1&0&0&0&0 \\ 1&0&1&0&0&1&0&1 \\ 1&1&0&0&0&0&1&1 \\ 0&0&1&1&1&1&0&0 \\ 0&1&0&1&1&0&1&0 \\ 0&0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1&0 \end{bmatrix}. \tag{31}$$

It is clear that $\mathbf{C}_8$ is balanced in the sense that the rows (codewords) as well as the columns are balanced. It has been found that the HNN decoder performs better if the rows as well as the columns are balanced. The Hamming weight of $\mathbf{C}_8$ is still $2^{n-1} = 2^2$, while the Hamming distance increases slightly larger than $2^{n-1} = 2^2$.

By following the steps described above, any set of Walsh-Hadamard codes of length 2^n can be used to create a new set of 2^n balanced codes of length $m = 2^n$.

4.2.2 Encoding

Encoding is performed by mapping a group of n bits to 2^n BPSK symbols, or a group of $2n$ bits to 2^n 4-QAM symbols. Before encoding, the set of codewords $\mathbf{C}_{2^n}$ derived from the set of Walsh-Hadamard codes $\mathbf{H}_{2^n}$ is made bipolar by converting the 0's to -1.

BPSK encoding When BPSK modulation is used, n bits are mapped to 2^n BPSK symbols. The n bits are used to determine an index k in the range 1–2^n, which is then used to select a codeword from the set of codewords in $\mathbf{C}_{2^n}$ such that the selected codeword $\mathbf{c} = \mathbf{C}_{2^n}(k)$. Table 1 shows the number of uncoded bits, codeword length, uncoded bit to coded symbol rate R_s and the uncoded bit to coded bit rate R_c (code rate) for different n.

4-QAM encoding When 4-QAM modulation is used, $2n$ bits are mapped to 2^n 4-QAM symbols. The first and second groups of n bits (out of $2n$ bits) are used to determine two indices, $k^{(i)}$ and $k^{(q)}$, in the range 1–2^n, one for the in-phase part, and the other for the quaternary part of the codeword. The first index $k^{(i)}$ selects a codeword from $\mathbf{C}^{(i)}_{2^n}$, where $\mathbf{C}^{(i)}_{2^n}$ is derived as before, and the second index $k^{(q)}$ selects a codeword from $\mathbf{C}^{(q)}_{2^n}$, which can be equal to $\mathbf{C}^{(i)}_{2^n}$ or can be uniquely determined as explained earlier. The 4-QAM "codeword" is then calculated as $\mathbf{c} = \mathbf{C}^{(i)}_{2^n}(k^{(i)}) + j\mathbf{C}^{(q)}_{2^n}(k^{(q)})$, which is much like the result of coded modulation where groups of coded bits (in this case uncoded bits) are mapped to signal constellation points to improve spectral efficiency [20]. Table 2 shows the number of uncoded bits, codeword length, the uncoded bit to coded symbol rate R_s and code rate R_c for different $2n$. Even though the code rate remains the same as with BPSK modulation, the throughput doubles as expected.

4.2.3 Decoder

The HNN is known to be able to recognize input patterns from a set of stored patterns [15,21]. In the context of the HNN decoder, the patterns are the balanced codewords, and the HNN is able to determine the ML codeword from a set of codewords. This has been demonstrated before but only for one codeword at a time [17]. Therefore, if a received data block contains P codewords, the HNN will have to be applied P times in order to determine P ML codewords. However, the HNN MLSE decoder developed here is able to determine the most likely sequence of codewords using a single HNN. The HNN MLSE decoder is therefore applied once to a received data block containing any number of codewords.

Table 1 Input-output relationship for BPSK encoder

n	2^n	Rs	Rc
1	2	1/2	1/2
2	4	1/2	1/2
3	8	3/8	3/8
4	16	1/4	1/4

Table 2 Input-output relationship for 4-QAM encoder

$2n$	2^n	Rs	Rc
2	2	1	1/2
4	4	1	1/2
6	8	3/4	3/8
8	16	1/2	1/4

After the HNN MLSE decoder has determined the sequence of most likely transmitted codewords, the codewords are demapped by calculating the Euclidean distance between each ML codeword and each codeword in $\mathbf{C}_{2^n}$ for BPSK modulation, and each codeword in $\mathbf{C}^{(i)}_{2^n} + j\mathbf{C}^{(q)}_{2^n}$ for 4-QAM modulation. The indices(s) corresponding to the codeword(s) that have the lowest Euclidean distance/distances is/are converted to bits, which completes the decoding phase.

The derivation of the HNN MLSE decoder entails the calculation of the cross-correlation matrices $\mathbf{X}_i$ and $\mathbf{X}_q$, and the input vectors $\mathbf{I}_i$ and $\mathbf{I}_q$ in (10). The HNN MLSE decoder is first derived for the decoding of a single codeword, after which it will be extended to enable the decoding of any number of codewords simultaneously. Derivations are performed for 4-QAM only, since the BPSK HNN MLSE decoder is a simplification of its 4-QAM counterpart.

Single codeword decoding To enable the HNN to store a set of codewords, the average correlation between all pattern must be stored in the weights between the neurons. According to Hebb's rule of auto-associative memory [22], the connection weight matrix, or correlation matrix, is calculated by taking the cross-correlation of the patterns to be stored. Since we are working with complex symbols, there are two weight matrices to be calculated. The cross-correlation matrices in (9) are calculated as

$$\begin{aligned}\mathbf{X}_i &= \mathrm{Re}\{\mathbf{C}^T\mathbf{C}\} \\ &= \mathbf{C}^{(i)T}_{2^n}\mathbf{C}^{(i)}_{2^n} + \mathbf{C}^{(q)T}_{2^n}\mathbf{C}^{(q)}_{2^n}\end{aligned} \tag{32}$$

and

$$\begin{aligned}\mathbf{X}_q &= \mathrm{Im}\{\mathbf{C}^T\mathbf{C}\} \\ &= \mathbf{C}^{(q)T}_{2^n}\mathbf{C}^{(i)}_{2^n} - \mathbf{C}^{(i)T}_{2^n}\mathbf{C}^{(q)}_{2^n},\end{aligned} \tag{33}$$

where $\mathbf{C} = \mathbf{C}_{2^n}^{(i)} + j\mathbf{C}_{2^n}^{(q)}$, and $\mathbf{C}_{2^n}^{(i)}$ and $\mathbf{C}_{2^n}^{(q)}$ are the matrices containing the generated codewords as before, respectively, used for the in-phase and quadrature components of the codeword. Note the similarities between the correlation matrices in (32) and (33) and those in (18) and (20). Also, the two input vectors are simply the real and imaginary components of the noise-corrupted received codeword, such that

$$\mathbf{I}_i = \mathrm{Re}\{\mathbf{c}\} + \mathrm{Re}\{\mathbf{n}\} \tag{34}$$

and

$$\mathbf{I}_q = \mathrm{Im}\{\mathbf{c}\} + \mathrm{Im}\{\mathbf{n}\} \tag{35}$$

where $\mathbf{c}$ is of length 2^n and $\mathbf{n}$ is a vector containing complex samples from the distribution $\mathcal{N}(\mu, \sigma^2)$, where $\mu = 0$ and σ is the noise standard deviation. After the ML codeword is detected, each detected codeword (of length 2^n) can be mapped back to n bits for BPSK modulation and $2n$ bits for 4-QAM modulation.

Multiple codeword decoding It was shown how the HNN can be used to decode single codewords, but the HNN decoder can be extended in order to detect ML transmitted sequences of codewords. This step is crucial in our quest of merging the HNN decoder with the HNN MLSE equalizer, since the HNN MLSE equalizer detects ML sequences of transmitted symbols. If the transmitted information is encoded, these sequences contain multiple codewords, and hence the HNN decoder must be extended to detect not only single codewords, but codeword sequences.

This extension is easily achieved by using the HNN parameters already derived in (32) through (35). Consider a system transmitting a sequence of P balanced codewords of length 2^n, where n is the length of the uncoded bit-words. The new correlation matrix is constructed by copying $\mathbf{X}$ in (10) along the diagonal according to the number of transmitted codewords P, such that

$$\mathbf{X}^{(P)} = \begin{bmatrix} \mathbf{X}_i & \mathbf{X}_q^T & & & & \\ \mathbf{X}_q & \mathbf{X}_i & & & & \\ & & \mathbf{X}_i & \mathbf{X}_q^T & & \\ & & \mathbf{X}_q & \mathbf{X}_i & & \\ & & & & \ddots & \ddots \\ & \emptyset & & & \ddots & \ddots \\ & & & & & \mathbf{X}_i \quad \mathbf{X}_q^T \\ & & & & & \mathbf{X}_q \quad \mathbf{X}_i \end{bmatrix}, \tag{36}$$

where $\mathbf{X} = \begin{bmatrix} \mathbf{X}_i & \mathbf{X}_q^T \\ \mathbf{X}_q & \mathbf{X}_i \end{bmatrix}$ is repeated on the diagonal P times and $\emptyset$ implies that the rest of $\mathbf{X}^{(P)}$ is empty, containing only 0's.

Also the input vector $\mathbf{I}$ in (10), consisting of $\mathbf{I}_i$ and $\mathbf{I}_q$, is also extended according to the number of transmitted codewords P such that

$$\mathbf{I}^{(P)} = \begin{bmatrix} \mathbf{I}_i^{(P)} \\ \hline \mathbf{I}_q^{(P)} \end{bmatrix}, \tag{37}$$

where

$$\mathbf{I}_i^{(P)} = \left[\mathrm{Re}\{\mathbf{c}_1\}, \mathrm{Re}\{\mathbf{c}_2\}, \ldots, \mathrm{Re}\{\mathbf{c}_p\}\right]^T + \mathrm{Re}\{\mathbf{n}\} \tag{38}$$

and

$$\mathbf{I}_q^{(P)} = \left[\mathrm{Im}\{\mathbf{c}_1\}, \mathrm{Im}\{\mathbf{c}_2\}, \ldots, \mathrm{Im}\{\mathbf{c}_p\}\right]^T + \mathrm{Im}\{\mathbf{n}\}, \tag{39}$$

where $\mathbf{c}_p$ is the pth codeword of length 2^n, where $p = 1, 2, \ldots, P$, and $\mathbf{n}$ is of length $2^n P$ and contains complex samples from the distribution $\mathcal{N}(\mu, \sigma^2)$, where $\mu = 0$ and σ is the noise standard deviation.

The extended cross-correlation matrix and input vector in (36) and (37) can now be used to estimate the ML sequence of transmitted codewords, after which each detected codeword (of length 2^n) can be mapped back to n bits for BPSK modulation and $2n$ bits for 4-QAM modulation.

4.3 HNN turbo equalizer

The HNN-TE is an amalgamation of the HNN MLSE equalizer and the HNN MLSE decoder, which were discussed in the previous sections. In this section it is explained how the HNN MLSE equalizer and the HNN MLSE decoder are combined in order to perform iterative joint equalization and decoding (turbo equalization) using a single HNN structure. The HNN-TE is able to jointly equalize and decode BPSK and 4-QAM coded modulated signals in systems with highly dispersive multipath channels, with extremely low computational complexity compared to traditional turbo equalizers which employ a MAP equalizer/decoder pair.

4.3.1 System model

Since we already have complete models for the HNN MLSE equalizer and decoder, the combination of the two is fairly straight-forward. In order to distinguish between equalizer and decoder parameters a number of redefinitions are in order. For the HNN MLSE equalizer the correlation matrix and input vector relating to (10), as derived in (22) and (27), are now $\mathbf{X}_E$ and $\mathbf{I}_E$, respectively, and will henceforth be referred to as "equalizer correlation matrix" and "equalizer input vector". Similarly the HNN MLSE decoder correlation matrix and input vector relating to (10), as derived in (36) and (37), are now $\mathbf{X}_D$ and $\mathbf{I}_D$, respectively, and will henceforth be referred to as "decoder correlation matrix" and "decoder input vector".

When a coded data block of length N_c is transmitted through a multipath channel, $\mathbf{X}_E$ and $\mathbf{X}_D$ are determined according to (22) and (36), where both matrices are of

size $N_c \times N_c$. Since the function of the equalizer and the decoder has to be merged, it makes sense to somehow combine $\mathbf{X}_E$ and $\mathbf{X}_D$ to enable the equalizer to perform decoding, or to enable the decoder to perform equalization. This combination is performed by first normalizing $\mathbf{X}_D$ with respect to $\mathbf{X}_E$, because of varying energy in a multipath fading channel between received data blocks. $\mathbf{X}_D$ is therefore normalized with respect to $\mathbf{X}_E$ such that

$$\mathbf{X}_D^{(\text{norm})} = \left(\frac{\|\mathbf{X}_E\|}{\|\mathbf{X}_D\|}\right)\mathbf{X}_D. \tag{40}$$

Next the new correlation matrix is determined as

$$\mathbf{X}_{\text{TE}} = \mathbf{X}_E + \mathbf{X}_D^{(\text{norm})}. \tag{41}$$

The rationale behind the addition of the equalizer correlation matrix and the normalized decoder correlation matrix is that the connection weights in the decoder correlation matrix should bias those of the equalizer correlation matrix. Since $\mathbf{X}_{\text{TE}}$ contains $\mathbf{X}_E$ offset by $\mathbf{X}_D^{(\text{norm})}$, joint equalization and decoding is made possible.

The new input vector also needs to be calculated. $\mathbf{I_D}$ contains the noise-corrupted coded symbols, while $\mathbf{I_E}$ contains not only received coded symbol information, but also the ISI information. Note that when there is no multipath or fading ($L = 1$ and $h_0 = 1$), $\mathbf{I}_E$ reduces to $\mathbf{I}_D$. The new input vector used in the HNN-TE is therefore simply

$$\mathbf{I}_{\text{TE}} = \mathbf{I}_E. \tag{42}$$

With the new correlation matrix $\mathbf{X}_{\text{TE}}$ and input vector $\mathbf{I}_{\text{TE}}$, the HNN-TE model is complete, and the iterative system in (11) can be used to jointly equalize and decode (turbo equalize) the transmitted coded information.

4.3.2 Transformation

Upon reception the received symbol vector has to be deinterleaved to restore the one-to-one relationship between each element in $\mathbf{r}$ and $\mathbf{c}$ with respect to the first coefficient h_0 of the CIR $\mathbf{h} = \{h_0, h_1, \ldots, h_{L-1}\}^T$. Deinterleaving $\mathbf{r}$ transforms the transmission model in (1). Substituting (2) in (1) and applying the deinterleaver, which is simply the Hermitian transpose of the interleaver matrix $\mathbf{J}$, gives

$$\mathbf{J}^H\mathbf{r} = \mathbf{J}^H\mathbf{H}\mathbf{H}\mathbf{G}^H\mathbf{s} + \mathbf{J}^H\mathbf{n}, \tag{43}$$

which is equivalent to transmitting the coded symbol sequence $\mathbf{c} = \mathbf{G}^T\mathbf{s}$ through a channel

$$\mathbf{Q} = \mathbf{J}^H\mathbf{H}\mathbf{H}. \tag{44}$$

Therefore (43) can be written as

$$\mathbf{J}^H\mathbf{r} = \mathbf{Q}\mathbf{G}^H\mathbf{s} + \mathbf{J}^H\mathbf{n}. \tag{45}$$

Consequently the new channel matrix $\mathbf{Q}$, rather than the conventional channel matrix $\mathbf{H}$ in (3), is used in the calculation of the equalizer correlation matrix $\mathbf{X}_E$ derived in (22). Due to the above transformation, $\mathbf{Q}$ does not contain the CIR $\mathbf{h}$ on the diagonal as in $\mathbf{H}$. Rather, each column in $\mathbf{Q}$ (of length N_c) contains a unique random combination of all CIR coefficients (where the rest of the $N_c - L$ elements in a column are equal to 0), dictated by the randomization effect exhibited in $\mathbf{Q}$ due to the random interleaver. This randomization effect results from first multiplying the channel $\mathbf{H}$ with the interleaving matrix $\mathbf{J}$ and then deinterleaving by multiplying the result with $\mathbf{J}^T$ (see (44)). Deinterleaving places the first CIR coefficient (h_0) on the diagonal of $\mathbf{Q}$, restoring the one-to-one relationship between each element in $\mathbf{r}$ and each corresponding coded transmitted symbol in $\mathbf{c}$.

To illustrate this concept, consider the three-dimensional representations of $|\mathbf{HJ}|$ and $|\mathbf{Q}|$ in Figures 2a, b, 3a,b, 4a,b, and 5a,b, for a hypothetical system transmitting coded information through a multipath channel with CIR lengths of $L = 1$, $L = 5$, $L = 10$, and $L = 20$, respectively, with a block length $N_c = 80$. Figure 2a,b show $|\mathbf{HJ}|$ and $|\mathbf{Q}|$ for channels of length $L = 1$, where Figure 2a is clearly interleaved. It is also clear that the new channel $\mathbf{Q}$ in Figure 2b is deinterleaved, since the first coefficient h_0 of the CIR has been restored to the diagonal of $\mathbf{Q}$. Figure 3a and 5a show the interleaved channels for $L = 5$, $L = 10$, and $L = 20$, where Figure 3b and 5b show the new channels $\mathbf{Q}$, again with the first CIR coefficient h_0 restored to the diagonal. Even though h_0 is restored to the diagonal of $\mathbf{Q}$, it is clear that the rest of the CIR coefficients $h_1, h_2, \ldots, h_{L-1}$ are scattered throughout $\mathbf{Q}$. As stated before, each column in $\mathbf{Q}$ contains a unique random combination of all CIR coefficients (with h_0 on the diagonal for each column), dictated by the randomization effect exhibited in $\mathbf{Q}$, where the rest of the $N_c - L$ elements in each column are equal to 0.

5 Computational complexity analysis

The computational complexity of the HNN-TE is compared to that of the CTE by calculating the number of computations performed for each received data block, for a fixed set of system parameters. The number of computations are normalized by the coded data block length so as to factor out the effect of the length of the transmitted data block, which allows us to present the computational complexity in terms of the number of computations required per received coded symbol. The complexity of the HNN-TE is quadratically related to the coded data block length, so a change in N_c will still have an effect on the normalized computational complexity.

The computational complexity of the HNN-TE was calculated as

$$\begin{aligned} CC_{\text{HNN-TE}} = {} & 2N_c^{2.376} + 8(N_c + L - 1) + Z_{\text{HNN-TE}}((N_cM/2)^2 \\ & + (N_cM/2)) + 4N_ck^2 + 2(Nc + L - 1)^{2.376}, \end{aligned} \tag{46}$$

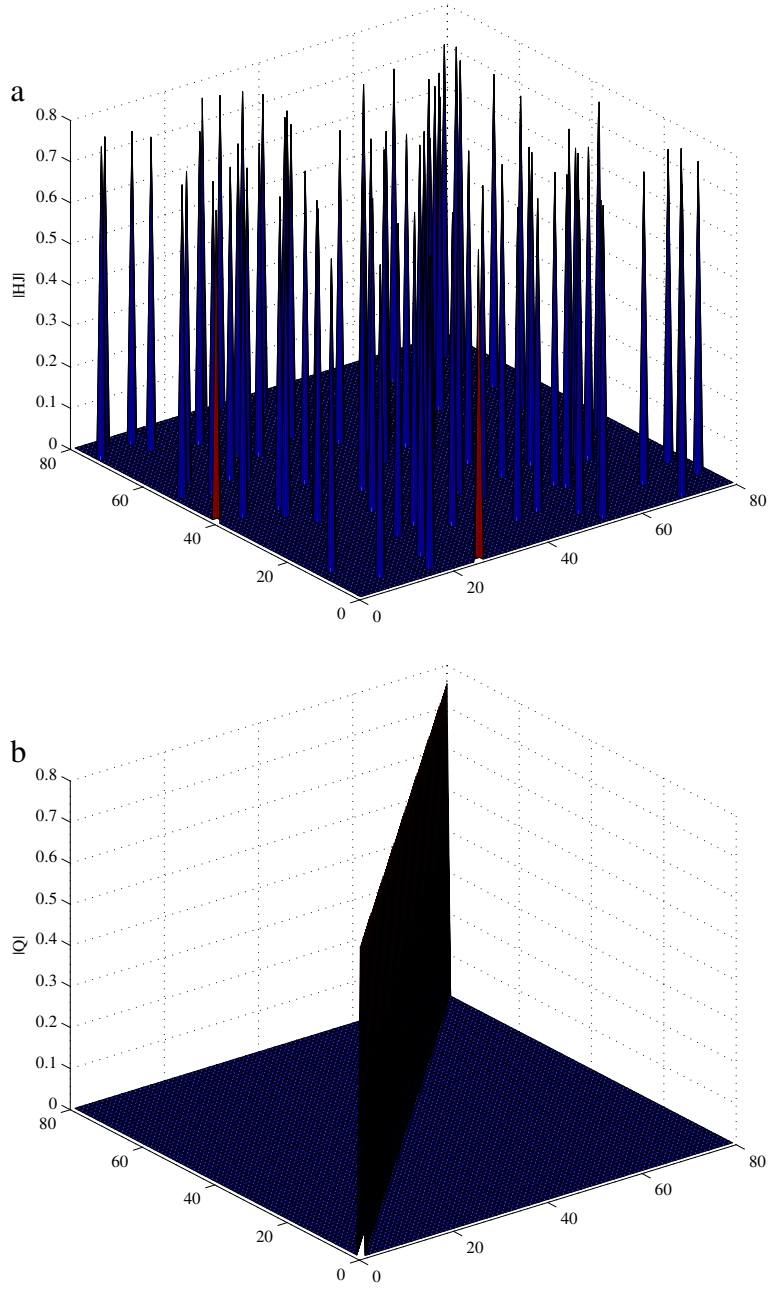

Figure 2 |HJ| and |Q| for systems with $L = 1$ CIR coefficients. (**a**) |HJ| (**b**) |Q|.

where N_c is the coded data block length, L is the CIR length, M is the modulation constellation alphabet size (2 for BPSK and 4 for 4-QAM), $Z_{\mathrm{HNN-TE}}$ is the number of iterations and k is the codeword length, which was chosen as $k = 8$ for a code rate of $R_c = 3/8$. The first term in (46) is associated with the calculation of $\mathbf{X}_i$ in (19) and $\mathbf{X}_q$ in (21). The second term is associated with the calculation of $\mathbf{\Lambda}$ in (28) and $\mathbf{\Omega}$ in (29). The third term is for the iterative calculation of the ML coded symbols in (11) while the second to last term in (46) is for the trivial ML detection of codewords after joint iterative MLSE equalization and decoding. The last term is due to the transformation in (43) through (45). Note that in the first and last terms of (46) the exponent is 2.376. It has been shown in [23] that the complexity of multiplication of two $N \times N$ matrices can be reduced from $O(N^3)$ to $O(N^{2.376})$. However, due to the fact that cubic complexity matrix multiplication is still preferred in practical applications due to ease of implementation, (46) serves as a lower bound on the HNN-TE computational complexity.

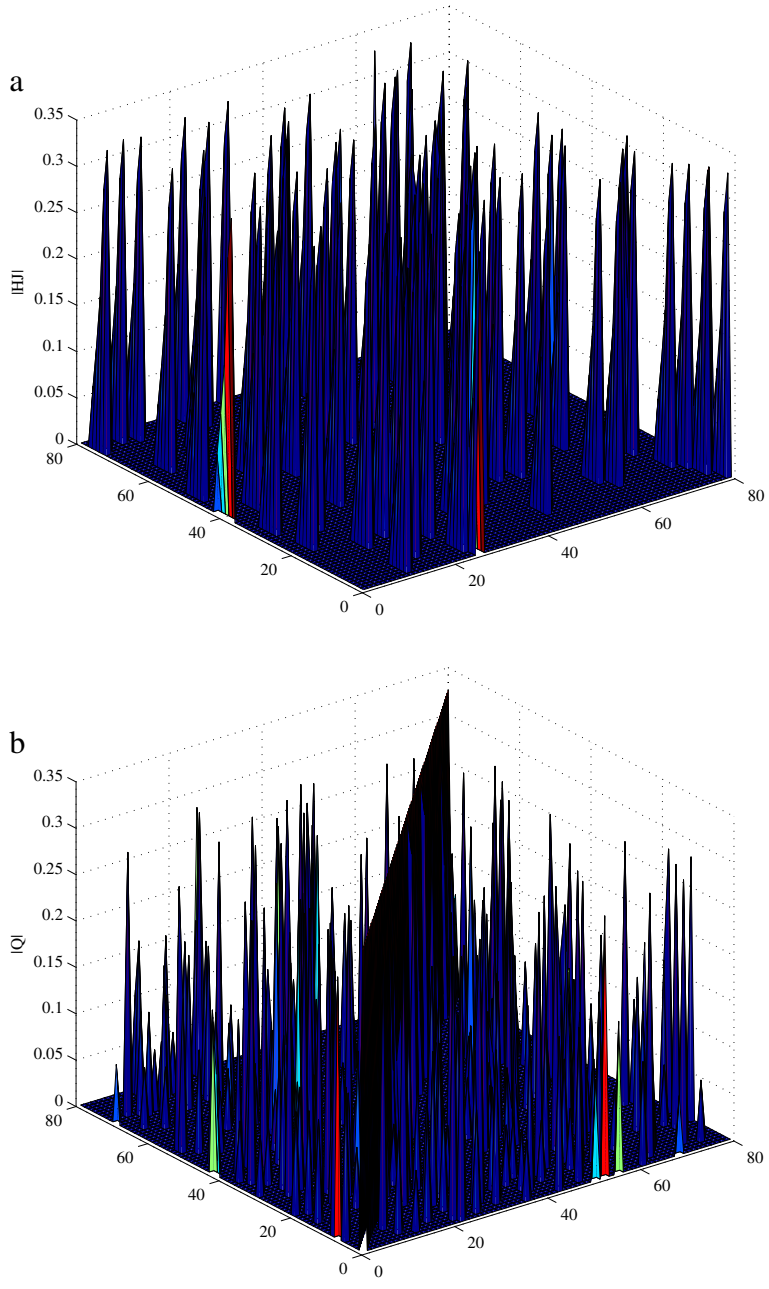

Figure 3 |HJ| and |Q| for systems with $L = 5$ CIR coefficients. (**a**) |HJ| (**b**) |Q|.

Therefore, the computational complexity of the HNN-TE is approximately quadratic at best, or more realistically cubic in the coded data block length (N_c), quadratic in the modulation constellation alphabet size (M), quadratic in the codeword length k, and approximately independent of the channel memory length (L).

The complexity of the CTE was determined as

$$CC_{\mathrm{CTE}} = Z_{\mathrm{CTE}} \left(4N_c LQ + 4N_c k^2\right), \tag{47}$$

where Z_{CTE} is the number of iterations and Q is the number of equalizer states, determined by 2^{L-1} for BPSK modulation and 4^{L-1} for 4-QAM. The first term in (47) is associated with the equalizer while the second term is associated with MAP decoding. The computational complexity of the CTE is therefore linear in the coded data block length (N_c), exponential in the channel memory length (L) and quadratic in the codeword length (k).

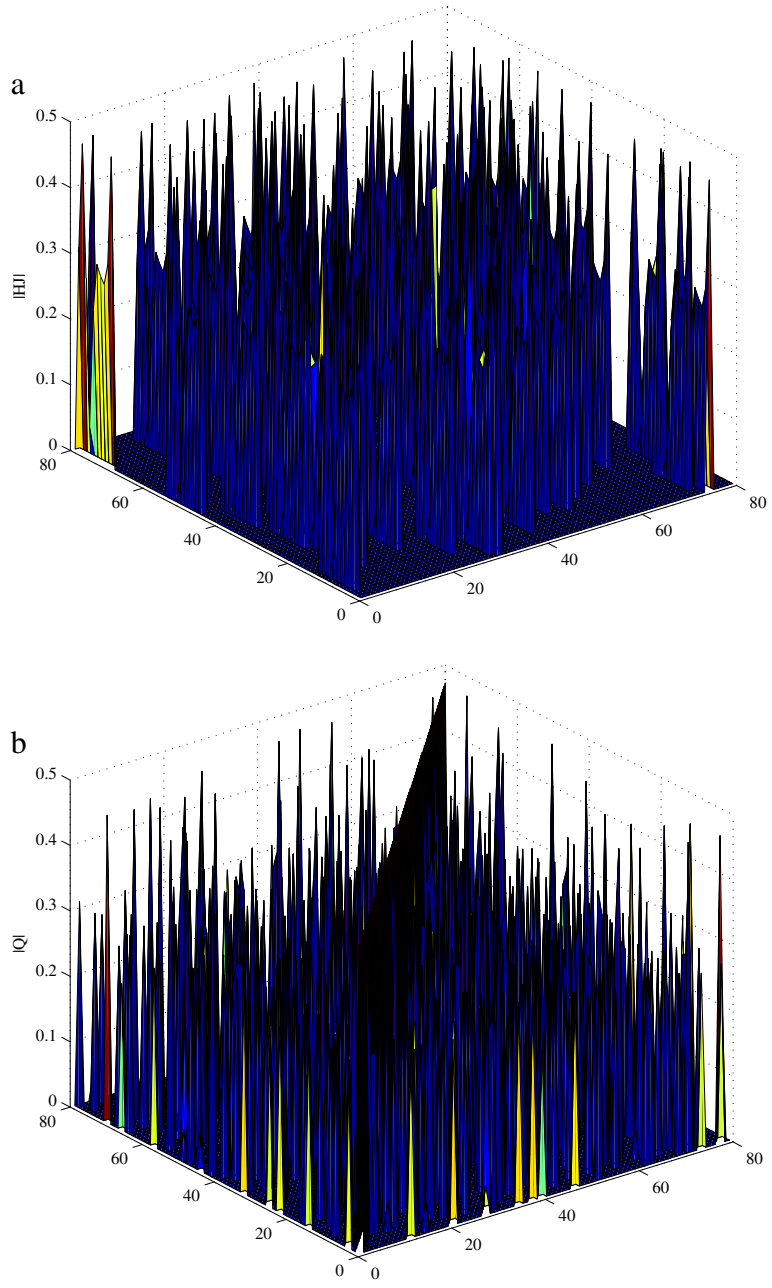

Figure 4 |HJ| and |Q| for systems with $L = 10$ CIR coefficients. (**a**) |HJ| (**b**) |Q|.

Figure 6 and shows the normalized computational complexity of the HNN-TE and the CTE for coded data block lengths of $N_c = 80$, $N_c = 160$, $N_c = 320$, $N_c = 640$, $N_c = 1280$, and $N_c = 2560$, where $Z_{\text{HNN-TE}} = 25$ and $Z_{\text{CTE}} = 5$, for BPSK and 4-QAM modulation when $O(N^{2.376})$ matrix multiplication complexity is considered. Figure 7 shows the same information as Figure 6, but with $O(N^3)$ matrix multiplication complexity. It is clear that the computational complexity of the HNN-TE increases with an increase in coded data block length, but for realistic data block lengths the complexity of the HNN-TE is superior to that of the CTE for channels with long memory. The HNN-TE is computationally less complex for BSPK modulation than for 4-QAM, but only slightly so. On the other hand, the complexity of the CTE grows exponentially with and increase in modulation order. From Figure 6 it is clear that the complexity of the HNN-TE is almost quadratically related to the coded data block length and approximately independent of the channel memory length, which is more evident when L is increased. The

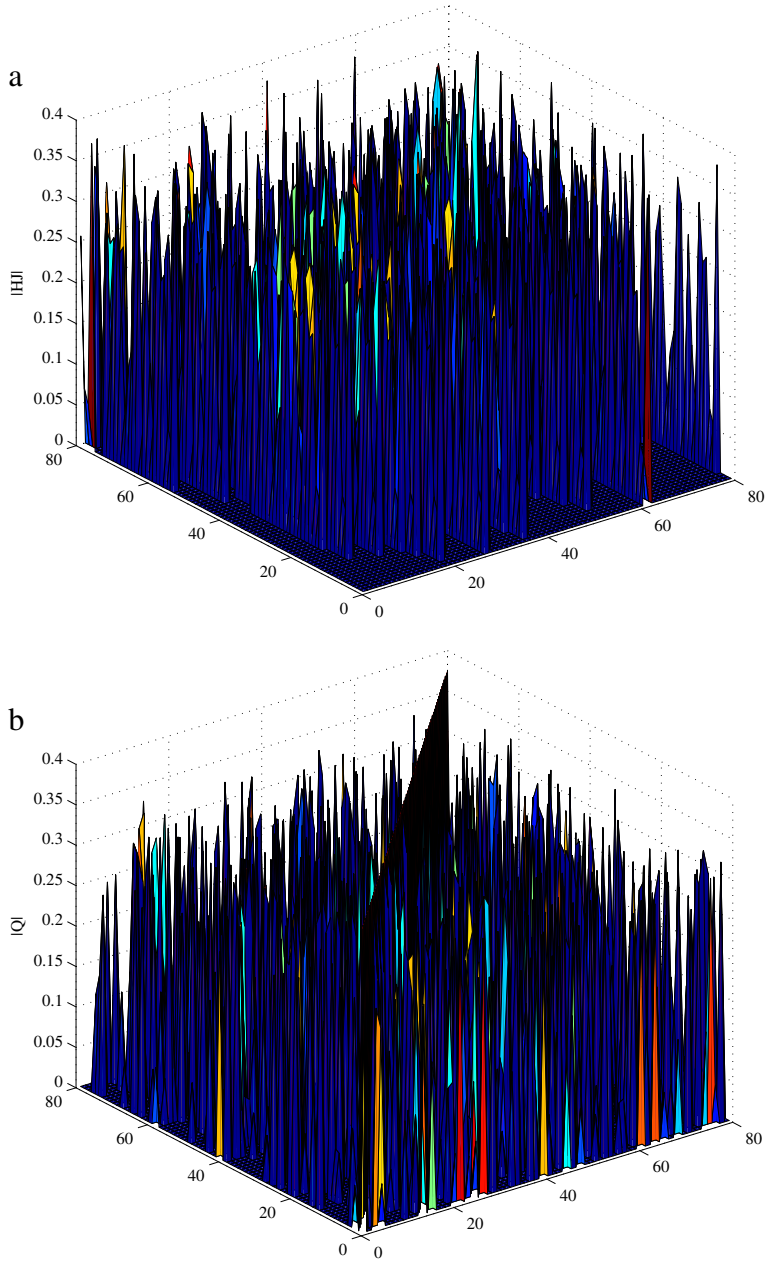

Figure 5 |HJ| and |Q| for systems with $L = 20$ CIR coefficients. (**a**) |HJ| (**b**) |Q|.

normalized computational complexity of the HNN-TE and the CTE (for $O(N^{2.376})$ and $O(N^3)$ matrix multiplication complexity) for $N_c = 1280$ using BPSK and 4-QAM for extremely long channels is shown in Figure 8, where there is no comparison between the complexity of the HNN-TE and that of the CTE, for both BSPK and 4-QAM modulation.

6 Memory requirements analysis

The memory requirements of the HNN-TE and the CTE are closely related to their respective computational complexities due to the structures employed by these algorithms. Table 3 describes the memory requirements of the HNN-TE for each received data block. The total memory requirement for the HNN-TE is $2N_c^2 + 6N_c + N_c + L - 1 + 2(N_c + L - 1)^2$ where each variable is of type float, which uses 32 bits. The memory requirements of the CTE per data block is shows in Table 4. The total memory requirement of the CTE is $N_c M^{L-1} + 4N_c + L$. Figure 9 shows the memory requirement of the HNN-TE and the CTE in bytes (32 bits = 8 bytes) for coded data block sizes of $N_c = 160$, $N_c = 640$, and $N_c = 2560$

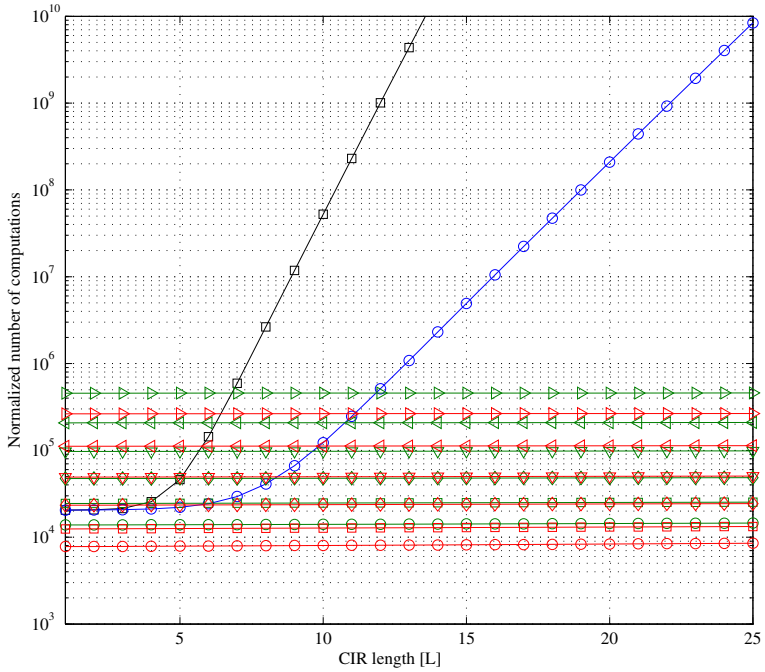

Figure 6 HNN-TE and CTE normalized computational complexity for short channels and varying coded block length assuming $O(N_c^{2.376})$ matrix multiplication complexity. Blue circle: CTE (BPSK); Black square: CTE (4-QAM); Red circle: HNN-TE - $N_c = 80$ (BPSK); Red square: HNN-TE - $N_c = 160$ (BPSK); Red diamond: HNN-TE - $N_c = 320$ (BPSK); Red down triangle: HNN-TE - $N_c = 640$ (BPSK); Red left triangle: HNN-TE - $N_c = 1280$ (BPSK); Red right triangle: HNN-TE - $N_c = 2560$ (BPSK); Green circle: HNN-TE - $N_c = 80$ (4-QAM); Green square: HNN-TE - $N_c = 160$ (4-QAM); Green diamond: HNN-TE - $N_c = 320$ (4-QAM); Green down triangle: HNN-TE - $N_c = 640$ (4-QAM); Green left triangle: HNN-TE - $N_c = 1280$ (4-QAM); Green right triangle: HNN-TE - $N_c = 2560$ (4-QAM).

and CIR lengths increasing from $L = 1$ to $L = 25$. From Figure 9 it is clear that the memory requirement of the HNN-TE remains constant over all channel lengths and modulation alphabet sizes, with less than 1 MB of memory required for $N_c = 160$, 6.6 MB for $N_c = 640$ and 100 MB for $N_c = 2560$. The memory requirements of the CTE, however, grows exponentially with the channel memory length, since the size of the trellis structure used in the MAP equalizer grows according to the same measure. The break-even point between the BPSK CTE and the HNN-TE (for both BPSK and 4-QAM) is $L = 10.40$ for $N_c = 160$, $L = 12.35$ for $N_c = 640$ and $L = 14.30$ for $N_c = 2560$, beyond which the HNN-TE require less memory than the CTE. Also, the break-even point between the 4-QAM CTE and the HNN-TE is $L = 5.68$ for $N_c = 160$, $L = 6.66$ for $N_c = 640$ and $L = 7.66$ for $N_c = 2560$. The memory requirements of the HNN-TE are therefore more favorable when higher order modulation alphabets are employed.

7 Simulation results

The proposed HNN-TE was evaluated in a mobile fading environment for BPSK and 4-QAM modulation at a code rate of $R_c = n/k = 3/8$. To simulated the fading effect of mobile channels, the Rayleigh fading simulator proposed in [24] was used to generate uncorrelated fading vectors. When imperfect channel state information (CSI) was assumed, least squares channel estimation was used using various amounts of training symbols in the transmitted data block. On the other hand, when perfect CSI was assumed, the CIR coefficients were "estimated" by taking the mean of the uncorrelated fading vectors. Simulations were performed for short and long channels at various mobile speeds. Simulations were also performed to compare the performance of the HNN-TE and a CTE in short mobile fading channels for BPSK modulation. For all simulations the uncoded data block length was $N_u = 480$ and the coded data block length was $N_c = 1280$. In all simulations the frequency was hopped four times during each data block in order to further reduce the BER. For the CTE the number of iterations were $Z = 5$, and instead of using a fixed number of iterations for the HNN-TE, we use the function $Z(E_b/N_0) = 2(5^{(E_b/N_0)/5})$ (which produces $Z(E_b/N_0) = \{2, 4, 8, 10, 22, 55\}$ for $E_b/N_0 = \{0, 2.5, 5, 7.5, 10\}$) to determine the number of iterations to be used given E_b/N_0.

Figure 10 show the performance of the HNN-TE and the CTE for channel lengths of $L = 4$, $L = 6$, and $L = 8$ at a fixed mobile speed of 20 km/h, assuming perfect CSI. The

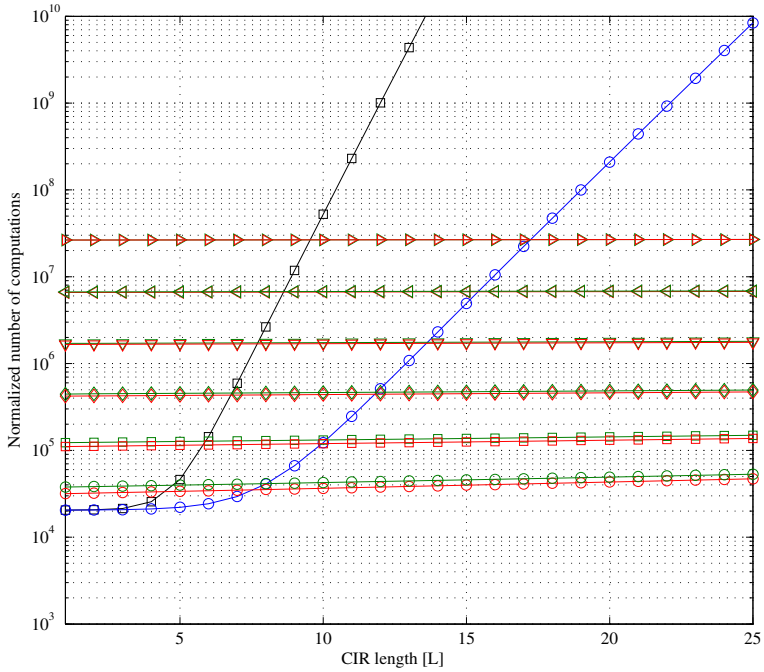

Figure 7 HNN-TE and CTE normalized computational complexity for short channels and varying coded block length assuming $O(N_c^3)$ HNN-TE matrix multiplication complexity. Blue circle: CTE (BPSK); Black square: CTE (4-QAM); Red circle: HNN-TE - $N_c = 80$ (BPSK); Red square: HNN-TE - $N_c = 160$ (BPSK); Red diamond: HNN-TE - $N_c = 320$ (BPSK); Red down triangle: HNN-TE - $N_c = 640$ (BPSK); Red left triangle: HNN-TE - $N_c = 1280$ (BPSK); Red right triangle: HNN-TE - $N_c = 2560$ (BPSK); Green circle: HNN-TE - $N_c = 80$ (4-QAM); Green square: HNN-TE - $N_c = 160$ (4-QAM); Green diamond: HNN-TE - $N_c = 320$ (4-QAM); Green down triangle: HNN-TE - $N_c = 640$ (4-QAM); Green left triangle: HNN-TE - $N_c = 1280$ (4-QAM); Green right triangle: HNN-TE - $N_c = 2560$ (4-QAM).

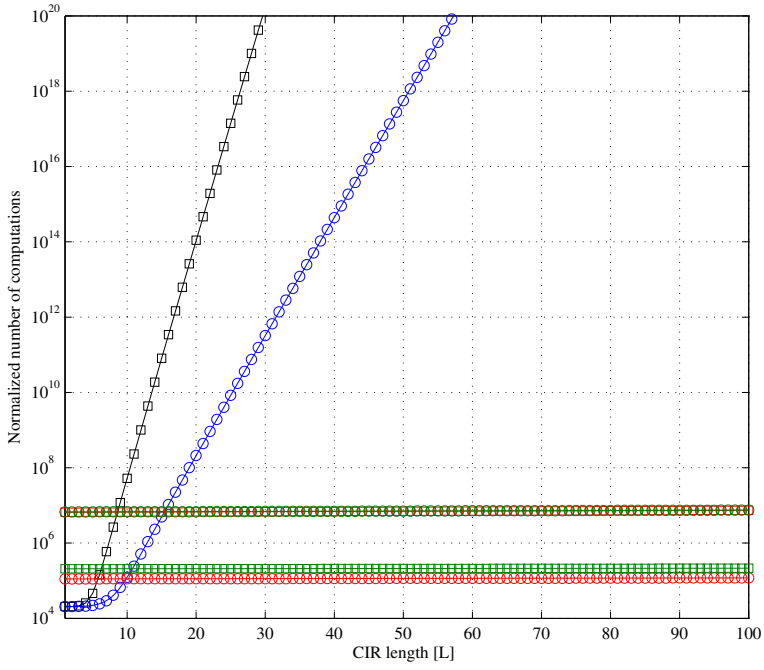

Figure 8 HNN-TE and CTE normalized computational complexity for long channels and $N_{c=1280}$ for both $O(N_c^{2.376})$ and $O(N_c^3)$ HNN-TE matrix multiplication complexity. Blue circle: CTE - BPSK; Black square: CTE - 4-QAM; Red circle: HNN-TE - BPSK ($O(N_c^{2.376})$); Green square: HNN-TE - 4-QAM ($O(N_c^{2.376})$); Red square: HNN-TE - BPSK ($O(N_c^3)$); Green circle: HNN-TE - 4-QAM ($O(N_c^3)$).

Table 3 HNN-TE memory requirements

Description	Size (float)
Correlation matrices $\mathbf{X}_i$, $\mathbf{X}_q$	$2N_c^2$
Input vectors $\mathbf{I}_i$, $\mathbf{I}_q$	$2N_c$
HNN internal state $\mathbf{u}$	$2N_c$
HNN output $\mathbf{s}$	$2N_c$
Channel matrix $\mathbf{HJ}$	$(N_c + L - 1)^2$
Interleaver matrix $\mathbf{J}$	$(N_c + L - 1)^2$
Received vector $\mathbf{r}$	$N_c + L - 1$

Table 4 CTE memory requirements

Description	Size (float)
Equalizer forward and backward messages	$N_c M^{L-1}$
Decoder forward and backward messages	$(\frac{N_c}{k})k = N_c$
Equalizer output $L^E(\hat{\mathbf{s}})$	N_c
Decoder output $L^D(\hat{\mathbf{s}}')$	N_c
Channel impulse response $\mathbf{h}$	L
Received vector $\mathbf{r}$	N_c

performance of the HNN-TE is slightly better than that of the CTE for high SNR levels.

Figure 11 shows the performance of the HNN-TE and the CTE for a channel of length $L = 6$ at mobile speeds of 3 km/h, 50 km/h, 80 km/h, 140 km/h, and 200 km/h, assuming perfect CSI. It is clear that the HNN-TE outperforms the CTE at mobile speeds greater than 20 km/h, with the advantage of performance increasing with an increase in mobile speeds. It seems that the HNN-TE is less affected by increasing mobile speeds, which suggests that the HNN-TE is able to perform well in fast-fading mobile environments.

Figure 12 shows the performance of the HNN-TE and the CTE for a channel of length $L = 6$ at a mobile speed of 20 km/h, assuming imperfect CSI. To estimate the channel training sequences of length $4L$, $6L$, $8L$, and $10L$ were used. From Figure 12 it is clear that the HNN-TE is superior to the CTE at high SNR levels when perfect CSI is not available. The HNN-TE seems to be less sensitive to channel estimation errors.

It is clear from Figures 10, 11, and 12 that the performance of the HNN-TE is superior to that of a CTE in short channels at varying mobile speeds, for both perfect and imperfect CSI. The HNN-TE outperforms the CTE in short channels, but with higher computational complexity. Figure 6 shows that the HNN-TE is more computationally complex than the CTE for short channels ($L < 10$), when the coded data block length is relatively small ($N_u < 1280$). However, the complexity of the HNN-TE is vastly superior to that of the CTE for long channels. It might be argued that the HNN-TE will perform better than the CTE since more iterations are used, but that is

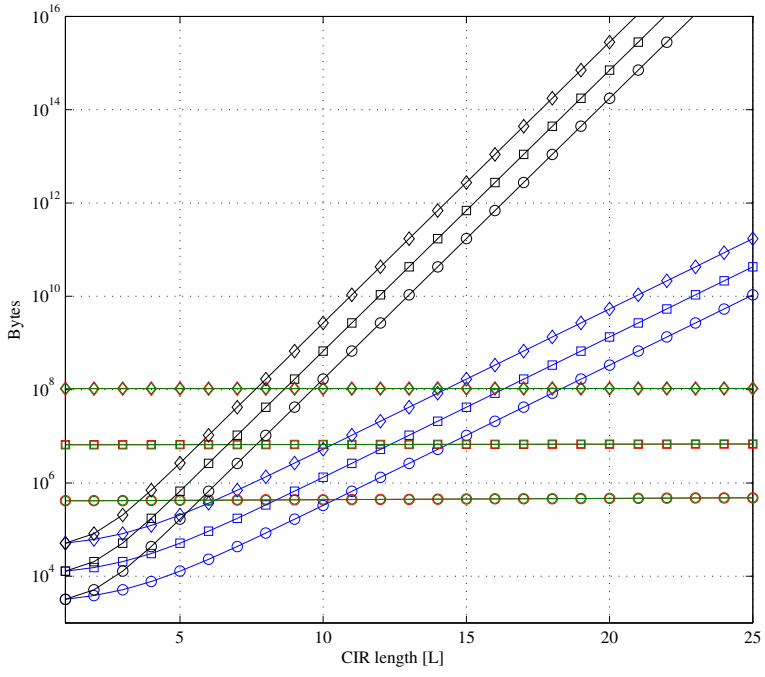

Figure 9 HNN-TE and CTE memory requirements per coded data block in bytes. Blue circle: CTE - $N_c = 160$ (BPSK); Blue square: CTE - $N_c = 640$ (BPSK); Blue diamond: CTE - $N_c = 2560$ (BPSK); Blue circle: CTE - $N_c = 160$ (4-QAM); Blue square: CTE - $N_c = 640$ (4-QAM); Blue diamond: CTE - $N_c = 2560$ (4-QAM); Red circle: HNN-TE - $N_c = 160$ (BPSK); Red square: HNN-TE - $N_c = 640$ (BPSK); Red diamond: HNN-TE - $N_c = 2560$ (BPSK); Green circle: HNN-TE - $N_c = 160$ (4-QAM); Green square: HNN-TE - $N_c = 640$ (4-QAM); Green diamond: HNN-TE - $N_c = 2560$ (4-QAM).

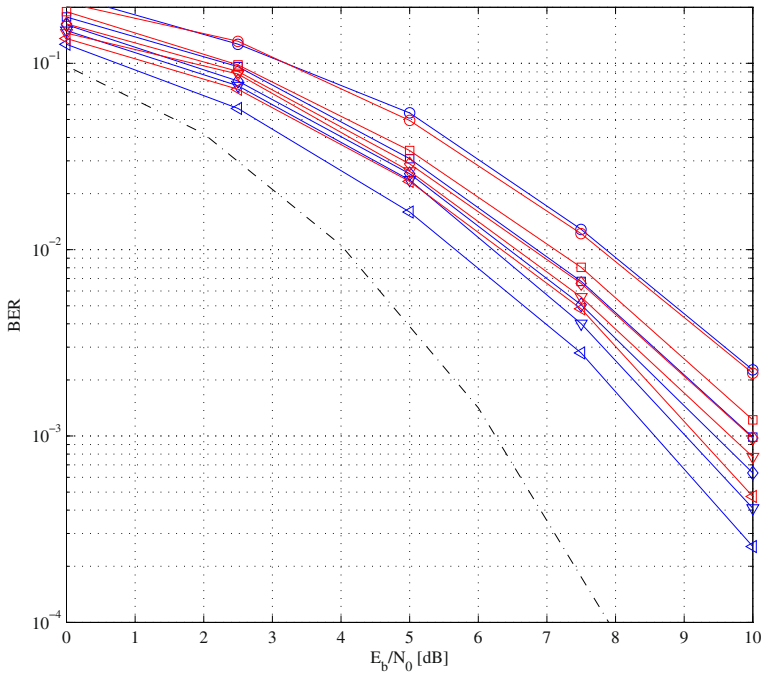

Figure 10 HNN-TE and CTE BPSK performance in short channels at a fixed mobile speed assuming perfect CSI. Shows the HNN-TE and CTE performance in systems with CIR lengths of $L = 4$, $L = 6$, and $L = 8$ at a mobile speed of 20 km/h. Red diamond: CTE - $L = 4$; Red square: CTE - $L = 6$; Red circle: CTE - $L = 8$; Blue diamond: HNN-TE - $L = 4$; Blue square: HNN-TE - $L = 6$; Blue circle: HNN-TE - $L = 8$; Black dashed: coded AWGN bound.

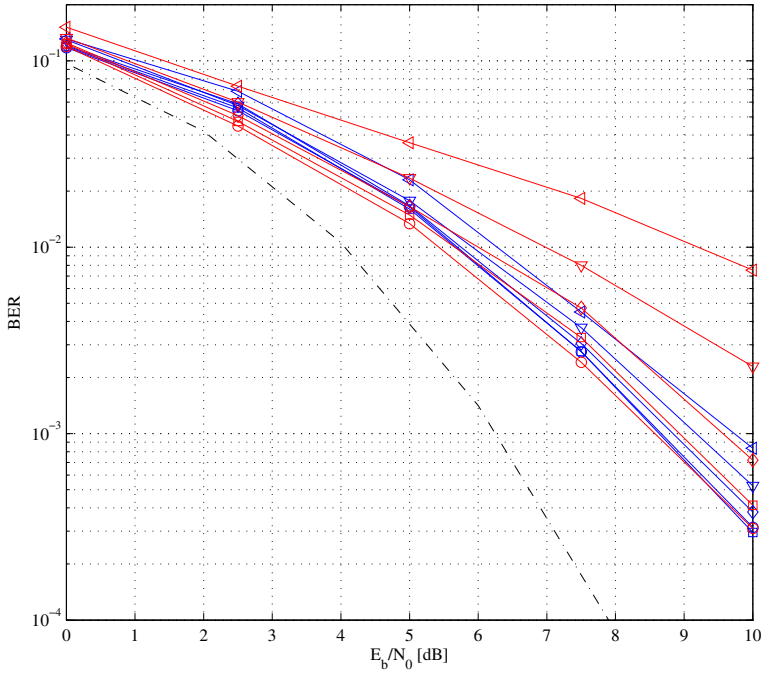

Figure 11 HNN-TE and CTE BPSK performance in a short channel at various mobile speeds assuming perfect CSI. Shows the HNN-TE and CTE performance in a system with CIR length $L = 6$ at mobile speeds of 3 km/h, 20 km/h, 50 km/h, 80 km/h, and 110 km/h. Red circle: CTE - $v = 3$ km/h; Red square: CTE - $v = 50$ km/h; Red diamond: CTE - $v = 80$ km/h; Red down triangle: $v = 140$ km/h; Red left triangle: CTE - $v = 200$ km/h; Blue circle: HNN-TE - $v = 3$ km/h; Blue square: HNN-TE - $v = 50$ km/h; Blue diamond: HNN-TE - $v = 80$ km/h; Blue down triangle: HNN-TE - $v = 140$ km/h; Blue left triangle: HNN-TE - $v = 200$ km/h; Black dashed: coded AWGN bound.

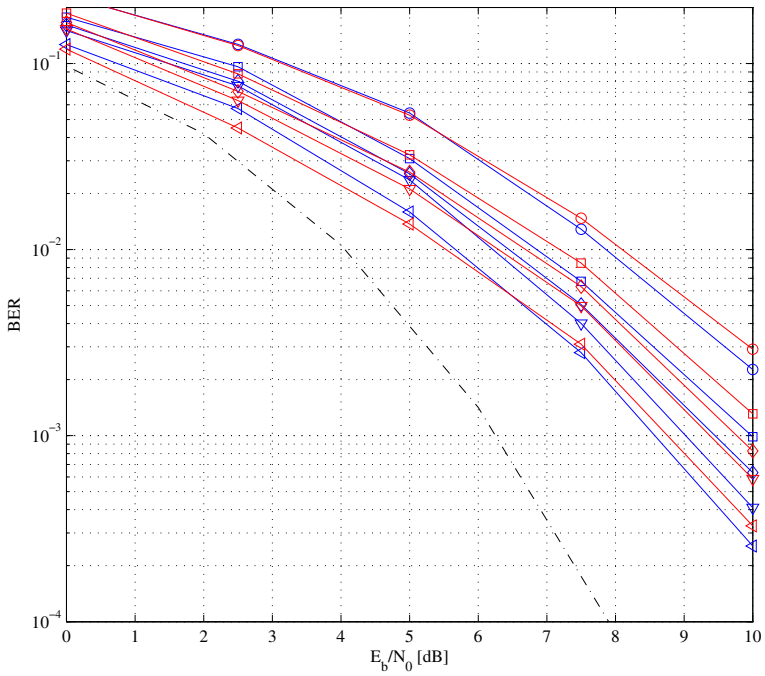

Figure 12 HNN-TE and CTE BPSK performance in a short channel at a fixed mobile speed for various amounts of training symbols for channel estimation. Shows the HNN-TE and CTE performance in systems with CIR length $L = 6$ at a fixed mobile 20 km/h using $4L$, $6L$, $8L$, and $10L$ symbols for channel estimation. Red circle: CTE - $4L$ pilots; Red square: CTE - $6L$ pilots; Red diamond: CTE - $8L$ pilots; Red down triangle: CTE - $10L$ pilots; Red left triangle: CTE - Perfect CSI; Blue circle: HNN-TE - $4L$ pilots; Blue square: HNN-TE - $6L$ pilots; Blue diamond: HNN-TE - $8L$ pilots; Blue down triangle: HNN-TE - $10L$ pilots; Blue left triangle: HNN-TE - Perfect CSI; Black dashed: coded AWGN bound.

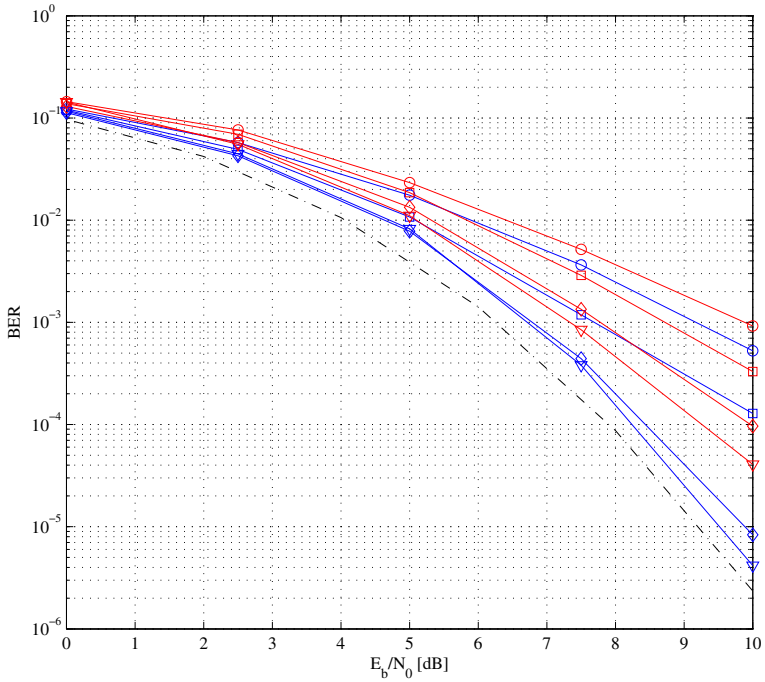

Figure 13 HNN-TE BPSK and 4-QAM performance in a long channel at a fixed speed assuming perfect CSI. Shows the HNN-TE BPSK and 4-QAM performance in systems with CIR lengths of $L = 10$, $L = 20$, $L = 50$, and $L = 100$ at a mobile speed of 20 km/h. Blue circle: BPSK HNN-TE - $L = 10$; Blue square: BPSK HNN-TE - $L = 20$; Blue diamond: BPSK HNN-TE - $L = 50$; Blue down triangle: BPSK HNN-TE - $L = 100$; Red circle: 4-QAM HNN-TE - $L = 10$; Red square: 4-QAM HNN-TE - $L = 20$; Red diamond: 4-QAM HNN-TE - $L = 50$; Red down triangle: 4-QAM HNN-TE - $L = 100$; Black dashed: coded AWGN bound.

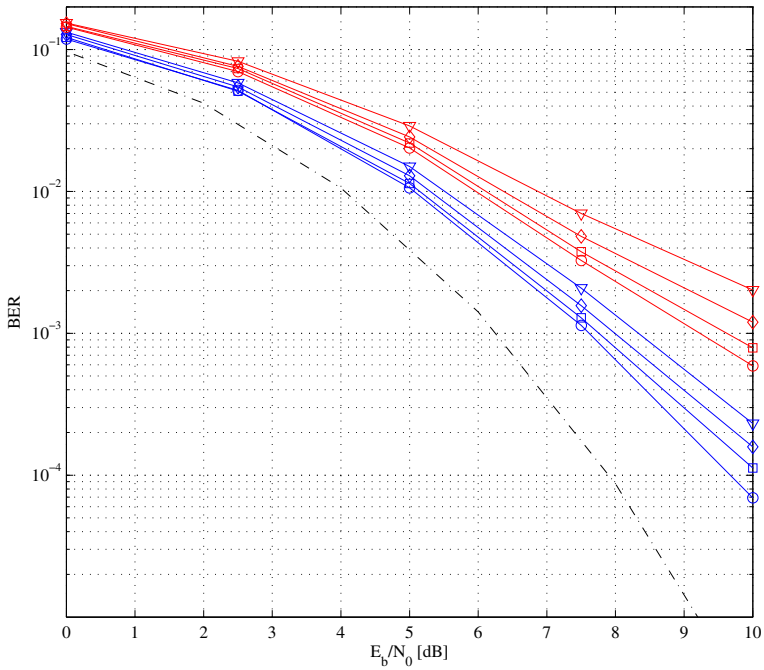

Figure 14 HNN-TE BPSK and 4-QAM performance in a long channel at various mobile speeds assuming perfect CSI. Shows the HNN-TE BPSK and 4-QAM performance in a system with CIR length $L = 50$ at mobile speeds of 20 km/h, 80 km/h, 140 km/h, and 200 km/h. Blue circle: BPSK HNN-TE - $v = 20$ km/h; Blue square: BPSK HNN-TE - $v = 80$ km/h; Blue diamond: BPSK HNN-TE - $v = 140$ km/h; Blue down triangle: BPSK HNN-TE - $v = 200$ km/h; Red circle: 4-QAM HNN-TE - $v = 20$ km/h; Red square: 4-QAM HNN-TE - $v = 80$ km/h; Red diamond: 4-QAM HNN-TE - $v = 140$ km/h; Red down triangle: 4-QAM HNN-TE - $v = 200$ km/h; Black dashed: coded AWGN bound.

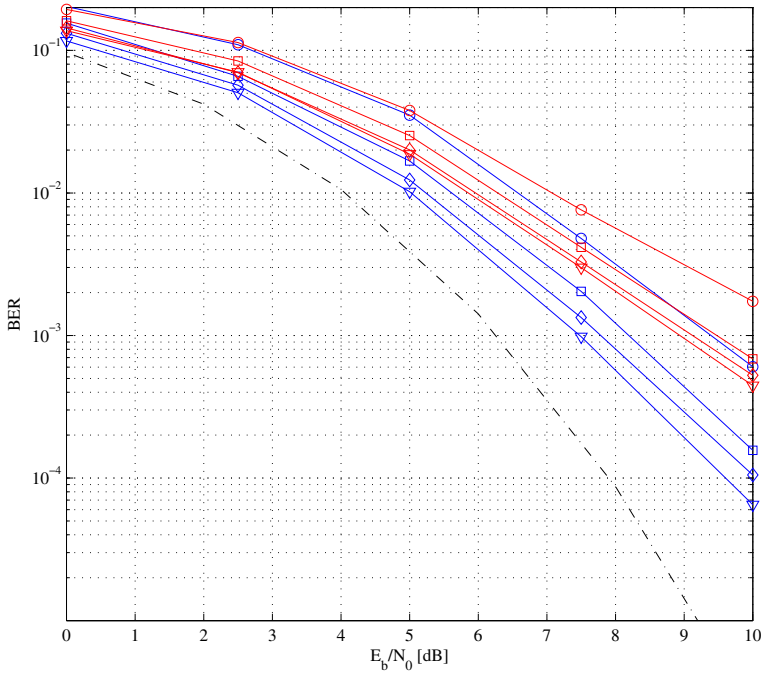

Figure 15 HNN-TE BPSK and 4-QAM performance in a long channel at a fixed speed for various amounts of training symbols for channel estimation. Shows the HNN-TE BPSK and 4-QAM performance in systems with CIR length $L = 50$ at a fixed mobile 20 km/h using $4L$, $6L$, $8L$, and $10L$ symbols for channel estimation. Blue circle: BPSK HNN-TE - $10L$; Blue square: BPSK HNN-TE - $8L$; Blue diamond: BPSK HNN-TE - $6L$; Blue down triangle: BPSK HNN-TE - $4L$; Red circle: 4-QAM HNN-TE - $10L$; Red square: 4-QAM HNN-TE - $8L$; Red diamond: 4-QAM HNN-TE - $6L$; Red down triangle: 4-QAM HNN-TE - $4L$; Black dashed: coded AWGN bound.

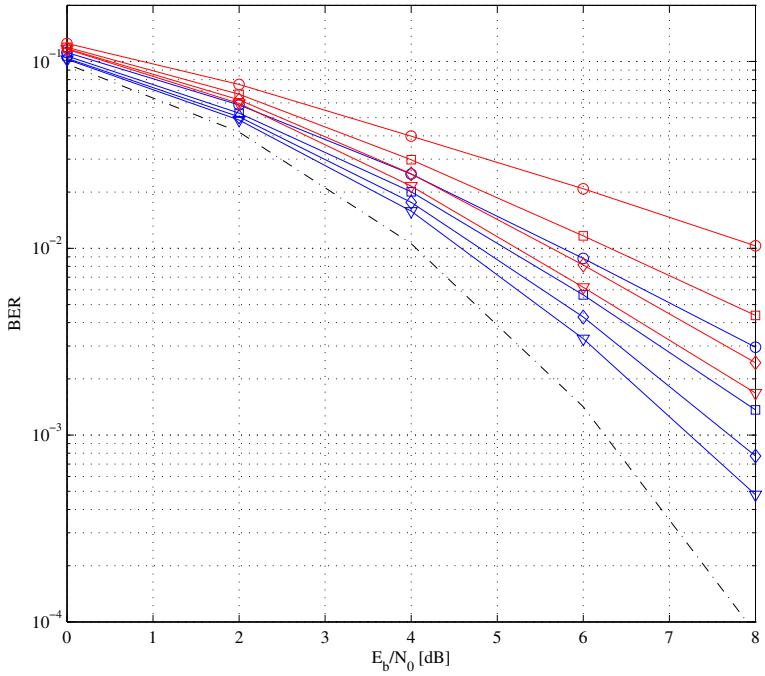

Figure 16 HNN-TE BPSK and 4-QAM performance in a long channel at a fixed speed for various numbers iterations. Shows the HNN-TE BPSK and 4-QAM performance in systems with CIR length $L = 50$ at a fixed mobile 20 km/h using $Z = 5$, $Z = 10$, $Z = 20$, and $Z = 50$ iterations.

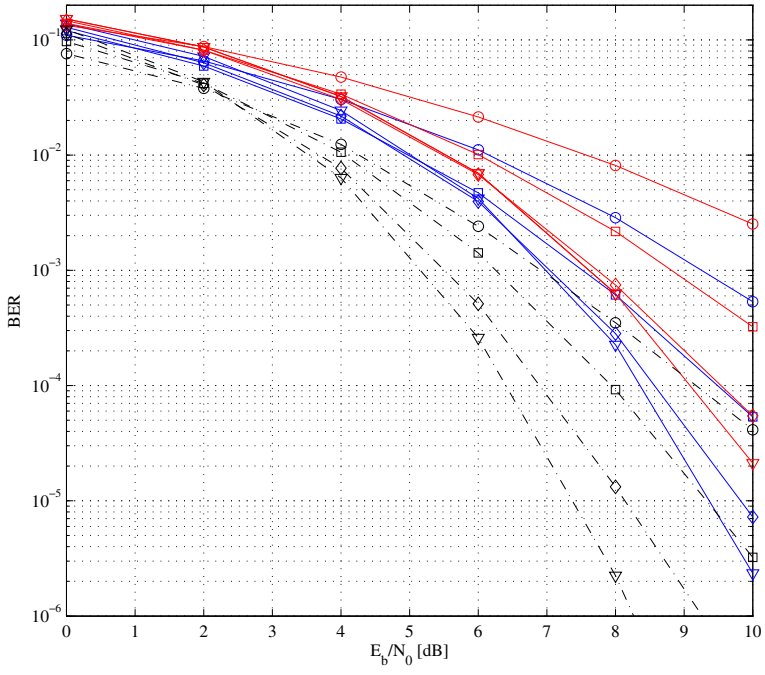

Figure 17 HNN-TE BPSK and 4-QAM performance in a long channel for different code rates, at a fixed speed assuming perfect CSI. Shows the HNN-TE BPSK and 4-QAM performance in systems with CIR length $L = 25$ at a fixed mobile 20 km/h for code rates of $R_c = 2/4$, $R_c = 3/8$, $R_c = 4/16$, and $R_c = 5/32$. Blue circle: BPSK HNN-TE - $R_c = 2/4$; Blue square: BPSK HNN-TE - $R_c = 3/8$; Blue diamond: BPSK HNN-TE - $R_c = 4/16$; Blue down triangle: BPSK HNN-TE - $R_c = 5/32$; Red circle: 4-QAM HNN-TE - $R_c = 2/4$; Red square: 4-QAM HNN-TE - $R_c = 3/8$; Red diamond: 4-QAM HNN-TE - $R_c = 4/16$; Red down triangle: 4-QAM HNN-TE - $R_c = 5/32$; Black dashed circle: coded AWGN bound - $R_c = 2/4$; Black dashed square: coded AWGN bound - $R_c = 3/8$; Black dashed diamond: coded AWGN bound - $R_c = 4/16$; Black dashed down triangle: coded AWGN bound - $R_c = 5/32$.

not true. It is stated in [3] that the performance of the CTE cannot be improved significantly beyond $Z = 3$ iterations in Rayleigh fading channels, so the performance gain of the HNN-TE compared to the CTE is probably due to the fact that HNN-TE is able to process all the available information internally as a whole, without having to exchange information between the equalizer and the decoder, as is the case in a CTE.

Figure 13 shows the performance of the HNN-TE for channels of length $L = 10$, $L = 20$, $L = 50$, $L = 100$ at a fixed mobile speed of 20 km/h for BPSK and 4-QAM modulation, assuming perfect CSI. It is clear that the performance for BPSK modulation is better than the performance for 4-QAM, which is due to the fact that Gray coding cannot be applied in the encoding process described in Section 4.2.2. The performance loss is therefore warranted.

Figure 14 shows the performance of the HNN-TE for a channel of length $L = 50$ at mobile speeds of 20 km/h, 80 km/h, 140 km/h, and 200 km/h for BPSK and 4-QAM modulation, assuming perfect CSI. It is clear that an increase in mobile speed leads to a performance degradation, although not as much as expected. Again BPSK modulation performs better than 4-QAM modulation.

Figure 15 shows the performance of the HNN-TE for a channel of length $L = 50$ at a mobile speed of 20 km/h for BPSK and 4-QAM modulation, assuming imperfect CSI. To estimate the channel, training sequences of length $4L$, $6L$, $8L$, and $10L$ were used. As expected, a performance loss is incurred with a decrease in the number of training symbols. Again BPSK modulation outperforms 4-QAM modulation.

Figure 16 shows the performance of the HNN-TE for a channel of length $L = 25$ at a mobile speed of 20 km/h for BPSK and 4-QAM modulation, assuming perfect CSI, for different numbers of iterations. The number of iterations were chosen to be $Z = 5$, $Z = 10$, $Z = 20$, and $Z = 50$. The BER performance increases with an increase in the number of iterations. Since the performance degradation due to a decrease in the number of iterations is low at low signal levels, we adopt an iteration schedule that is dependent on the signal level. As stated before, we use the following function to determine the number of iterations: $Z(E_b/N_0) = 2(5^{(E_b/N_0)/5})$.

Figure 17 shows the performance of the HNN-TE for a channel of length $L = 50$ at a mobile speed of 20 km/h for BPSK and 4-QAM modulation, assuming perfect CSI, for different code rates. The code rates were $R_c = 1/2$ (2/4), $R_c = 3/8$, $R_c = 1/4$ (4/16), and $R_c = 5/32$. From Figure 17 it is clear that the performance of the HNN-TE increases with a decrease in the code rate, with 4-QAM modulation performing worse than BPSK modulation.

From Figures 13, 14, 15, 16 and 17 it is clear that the HNN-TE is able to jointly equalize and decode BPSK and 4-QAM modulated signals, transmitted trough extremely long mobile fading channels. While the data rate using 4-QAM modulation is twice that using BPSK modulation, the performance is worse for 4-QAM modulation, due to the fact that Gray coding cannot be applied during coded modulation.

8 Conclusion

In this article, a low complexity turbo equalizer was developed which is able to jointly equalize and decode BPSK and 4-QAM coded-modulated signals in systems transmitting interleaved information through a multipath fading channels. It uses the Hopfield neural network as framework and hence it was fittingly named the Hopfield Neural Network Turbo Equalizer, or HNN-TE. The HNN-TE is able to turbo equalize coded modulated BPSK and 4-QAM signals in short as well as long multipath channels, slightly outperforming the CTE for short channels, although at higher computational cost. However, the HNN-TE computational complexity in long channels is vastly superior to that of CTE. The computational complexity of the HNN-TE is almost quadratically related to the coded data block length, while being approximately independent of the CIR length. This enables it to turbo equalize signals in systems with multiple hundreds of multipath elements. It was also demonstrated that the HNN-TE is less susceptible than the CTE to channel estimation errors, and it also outperforms the CTE in fast fading channels. The performance of the HNN-TE for BPSK modulation is better than for 4-QAM modulation, since Gray coding cannot be employed due to the coded modulation explained in this paper, while the complexity for 4-QAM is slightly higher.

Competing interests
Both authors declare that they have no competing interests.

Author details
[1]Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria, 0002, South Africa. [2]School of Engineering, University of Tasmania, Hobart, 7001, Australia.

Received: 11 June 2012 Accepted: 15 January 2013
Published: 8 February 2013

References

1. C Berrou, A Glavieux, P Thitimajshima, in *Int. Conf. Commun*, vol. 1. Near Shannon limit error-correction and decoding: Turbo-Codes, (1993), pp. 1064–1070
2. C Douillard, M Jezequel, C Berrou, A Picart, P Didier, A Glavieux, Iterative correction of intersymbol intereference: turbo-equalization. Europ. Trans. Telecommun. **6**, 507–511 (1995)
3. G Bauch, H Khorram, J Hagenauer, in *Proceedings of European Personal Mobile Communications Conference (EPMCC)*. Iterative equalization and decoding in mobile communication systems, (1997), pp. 307–312
4. R Koetter, M Tuchler, AC Singer, Turbo equalization. IEEE Signal Process. Mag. **21**(1), 67–80 (2004)
5. R Koetter, M Tuchler, AC Singer, Turbo equalization: principles and new results. IEEE Trans. Commun. **50**(5), 754–767 (2002)
6. RR Lopes, JR Barry, The soft feedback equalizer for turbo equalization of highly dispersive channels. IEEE Trans. Commun. **54**(5), 783–788 (2006)

7. A Dual-Hallen, C Hegaard, Delayed decision feedback sequence estimation. IEEE Trans. Commun. **37**(5), 428–436 (1989)
8. MV Eyuboglu, SU Qureshi, Reduced-state sequence estimation with set partitioning and decision feedback. IEEE Trans. Commun. **36**(1), 13–20 (1988)
9. J Wu, S Leong, K Lee, C Xiao, JC Olivier, Improved BDFE using a priori information for turbo equalization. IEEE Trans. Wirel. Commun. **7**(1), 233–240 (2008)
10. H Lou, C Xiao, Soft-decision feedback turbo equalization for multilevel modulations. IEEE Trans. Signal Process. **59**(1), 186–195 (2011)
11. I Fijalkow, D Pirez, A Roumy, S Ronger, P Vila, in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1. Improved interference cancellation for turbo-equalization, (2000), pp. 416–419
12. X Wang, HV Poor, Iterative (turbo) soft interference cancellation and decoding for coded CDMA. IEEE Trans. Commun. **47**(7), 1046–1061 (1999)
13. D Ampeliotis, K Berberidis, Low complexity turbo equalization for high data rate. EURASIP J. Commun. Network. **2006**(ID 25686), 1–12 (2006)
14. HC Myburgh, JC Olivier, Reduced complexity turbo equalization using a dynamic Bayesian network. EURASIP J. Adv. Signal Process (2012). (Submitted for Publication)
15. JJ Hopfield, DW Tank, Neural computations of decisions in optimization problems. Biol. Cybern. **52**, 1–25 (1985)
16. HC Myburgh, JC Olivier, Low complexity MLSE equalization in highly dispersive Rayleigh fading channels. EURASIP J. Adv. Signal Process. **2010**(ID 874874) (2010). http://asp.eurasipjournals.com/content/2010/1/874874
17. N Wiberg, in *Proceedings of the IEEE-SP Workshop on Neural Networks for Signal Processing*. A class of Hopfield decodable codes, (1993), pp. 88–97
18. Q Wang, VK Bhargava, in *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*. An error correcting neural network, (1989), pp. 530–533
19. D Knuth, Efficient balanced codes. IEEE Trans. Inf. Theory. **IT-32**(1), 530–533 (1986)
20. JG Proakis, *Digital Communications*, 4th edn. (McGraw-Hill, International Edition, New York, 2001)
21. JJ Hopfield, Artificial neural networks. IEEE Circ. Dev. Mag. **4**(5), 3–10 (1988)
22. DO Hebb, *The Organization of Behavior*, 4th edn. (Wiley, New York, 1949)
23. S Winograd, D Coppersmith, Matrix multiplication via arithmetic progressions. J. Symbolic Comput. **9**(3), 251–280 (1990)
24. YR Zheng, C Xiao, Improved models for the generation of multiple uncorrelated Rayleigh fading waveforms. IEEE Commun. Lett. **6**, 256–258 (2002)

doi:10.1186/1687-6180-2013-15
Cite this article as: Myburgh and Olivier: **A low complexity Hopfield neural network turbo equalizer.** *EURASIP Journal on Advances in Signal Processing* 2013 **2013**:15.