A Continuous-Time Recurrent Neural Network for Joint Equalization and Decoding – Analog Hardware Implementation Aspects

Mohamad Mostafa, Giuseppe Oliveri,
Werner G. Teich and Jürgen Lindner

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/63387

Abstract

Equalization and channel decoding are "traditionally" two cascade processes at the receiver side of a digital transmission. They aim to achieve a reliable and efficient transmission. For high data rates, the energy consumption of their corresponding algorithms is expected to become a limiting factor. For mobile devices with limited battery's size, the energy consumption, mirrored in the lifetime of the battery, becomes even more crucial. Therefore, an energy-efficient implementation of equalization and decoding algorithms is desirable. The prevailing way is by increasing the energy efficiency of the underlying digital circuits. However, we address here promising alternatives offered by mixed (analog/digital) circuits. We are concerned with modeling joint equalization and decoding *as a whole* in a continuous-time framework. In doing so, continuous-time recurrent neural networks play an essential role because of their nonlinear characteristic and special suitability for analog very-large-scale integration (VLSI). Based on the proposed model, we show that the superiority of joint equalization and decoding (a well-known fact from the discrete-time case) preserves in analog. Additionally, analog circuit design related aspects such as adaptivity, connectivity and accuracy are discussed and linked to theoretical aspects of recurrent neural networks such as Lyapunov stability and simulated annealing.

Keywords: continuous-time recurrent neural networks, analog hardware neural networks, belief propagation, vector equalization, joint equalization and decoding

1. Introduction

Energy efficiency has been increasingly attracting more interest due to economical and environmental reasons. Mobile communications sector has currently a share of 0.2% in global carbon emissions. This share is expected to double between 2007 and 2020 due to the ever-increasing demand for wireless devices [1, 2]. The sustained interest in higher data rate transmission is strengthening this impact. While major resources are being invested in increasing the energy efficiency of digital circuits, there is, on the other hand, a growing interest pointing at alternatives to the digital realization [3], including a mixed (analog/digital) approach. In such an approach, specific energy consuming (sub)tasks are implemented in analog instead of a "conventional" digital realization. The analog implementation possesses a high potential to significantly improve the energy efficiency [4] because of the inherent parallel processing of signals that are continuous in both time and amplitude. This has been shown in the field of error correction coding with a focus on decoding of low-density parity-check (LDPC) codes. Our ongoing research on equalization reveals similar results. We do not intend "analog" for linear signal processing with all its disadvantages like component inaccuracies and susceptibility to noise and temperature dependency [5] but for *nonlinear processing* instead. The work of Mead [6] and others on *Neuromorphic analog very-large-scale integration (VLSI)* has shown that "*analog signal processing systems can be built that share the robustness of digital systems but outperform digital systems by several orders of magnitude in terms of speed and/or power consumption*" [5].

The nonlinearity makes the analog implementation of an algorithm as robust as its digital counterpart [3, 5]. This profits from the match between the needed nonlinear operations for the algorithm and the physical properties of analog devices [7].

The capability of artificial neural networks (in the following neural networks) to successfully solve many scientific and engineering tasks has been shown oftentimes. Moreover, mapping algorithms to neural network structures can simplify the circuit design because of the regular (and repetitive) structure of neural networks and their limited number of *well-defined* arithmetic operations. Digital implementations can be considered precise (reproducibility of results under similar circumstances) but accurate (closeness of a result to the "true" value) *only* to the extent to which they have enough digits to represent [8]. This means, accuracy in digital implementations is achieved at the cost of efficiency (e.g., relatively larger chip area and more power consumption) [9]. An analog implementation is usually efficient in terms of chip area and processing speed [9], however, at the price of an inherent lack of the reproducibility of results [8] (because of a limited accuracy of the network components as an example [9]). However, by exploiting the distributed nature of neural structures the precision of the analog implementation can be improved despite inaccurate components and subsystems [8][1]. In other words, it is the distributed massively parallel nonlinear collective behavior of an analog implementation (of neural networks) which offers the possibility to make it as robust as its digital counterpart but more energy efficient[2] (additionally to smaller chip area). Particularly for *recurrent* neural networks (the class we focus on when considered as nonlinear dynamical systems), the robustness can be additionally achieved by exploiting "attracting" equilibrium points. In the light of this discussion, we map in this chapter a joint equalization and decoding algorithm into a novel *continuous-time* recurrent neural network structure. This class of neural networks has been attracting a lot of interest because of their widespread applications. They can be either trained for system identification [10], or they can be considered as dynamical

[1]For a clear distinction between *accuracy* and *precision* when used in hardware implementation context, we refer to [8].

[2]Energy efficiency is defined later as appropriate.

systems (dynamical solver). In the latter case, there is *no need* for a computationally complex and time-consuming training phase. This relies on the ability of these networks (under specific conditions) to be Lyapunov stable.

Equalization and channel decoding (together, in the following detection) are processes at the receiver side of a digital transmission. They aim to provide a reliable and efficient transmission. Equalization is needed to cope with the interference caused by multipath propagation, multiusers, multisubchannels, multiantennas and combinations thereof [11]. Channel (de)coding is applied for further improving the power efficiency. Equalization and decoding are nonlinear discrete optimization problems. The optimum solutions, in general, are computationally very demanding. Therefore, suboptimum solutions are applied, often soft-valued iterative schemes because of their good complexity-performance trade-off.

For high data rates, the energy consumption of equalization and decoding algorithms is expected to become a limiting factor. The need for floating-point computation and the nonlinear and iterative nature of (some of) these algorithms revive the option of an analog electronic implementation [12, 13], embedded in an essentially digital receiver. This option has been strengthened since the emergence of the "soft-valued" computation in this context [4] since soft-values are a natural property of analog signals. In contrast to analog decoding, analog equalization did not attract that amount of attention.

Furthermore, *joint* equalization and decoding (a technique where equalizer and decoder exchange their *local* available knowledge) further improves the efficiency of the transmission as an example in terms of lower bit error rates, however, at the cost of more computational complexity [14]. Most of the work related to joint equalization and decoding is limited to the discrete-time realization. One of the very few contributions focusing on continuous-time joint equalization and decoding is given in [13]. The consideration in [13] is not "neural networks-based". Stability and convergence are observed but not "deeply" considered.

We introduce in this chapter a novel continuous-time joint equalization and decoding structure. For this purpose, continuous-time single-layer recurrent neural networks play an essential role because of their nonlinear and recursive characteristic, special suitability for analog VLSI and since they serve as promising computational models for analog hardware implementation [15]. Both, equalizer and decoder are modeled as continuous-time recurrent neural networks. An additional proper feedback between equalizer and decoder is established for joint equalization and decoding. We also review individually, both continuous-time equalization and continuous-time decoding based on recurrent neural network structures. No training is needed since the recurrent neural network is serving as a dynamical solver or a computational model [15, 16]. This means, transmission properties are used to define the recurrent neural network (number of neurons, weight coefficients, activation functions, etc.) such that no training is needed. In addition, we highlight challenges emerging from the analog hardware implementation such as adaptivity, connectivity and accuracy. We also introduce our developed circuit for analog equalization based on continuous-time recurrent neural networks [3]. Characteristic properties of recurrent neural networks such as stability and convergence are addressed too. Based on the introduced model, we show by simulations that the superiority of joint equalization and decoding can be preserved in the analog "domain".

The main motivation for performing joint equalization and decoding in analog instead of using conventional digital circuits is to improve the energy efficiency and to minimize the area consumption in the VLSI chips [17]. The proposed continuous-time recurrent neural network serves as a promising computational model for analog hardware implementation.

The remainder of this chapter is organized as follows: In Section 2, we describe the block transmission model. Sections 3 and 4 are dedicated to the equalization process, the application of continuous-time recurrent neural networks and the analog circuit design and its corresponding performance and energy efficiency. Sections 5 and 6 are devoted to the channel decoding and the application of continuous-time recurrent neural networks for belief propagation (a decoding algorithm for LDPC codes). For both equalization and decoding cases, analog hardware design aspects and challenges and the behavior of the continuous-time recurrent neural network as a dynamical system are discussed. The continuous-time joint equalization and decoding based on recurrent neural networks is presented in Sections 7 and 8. Simulation results are shown in Section 9. We finish this chapter with a conclusion in Section 10.

Throughout this chapter, bold small and bold capital letters designate vectors (or finite discrete sets) and matrices, respectively.[3] All nonbold letters are scalars. $\text{diag}_m\{\boldsymbol{B}\}$ returns the matrix $\boldsymbol{B}$ where the nondiagonal elements are set to zeros. $\text{diag}_v\{\boldsymbol{b}\}$ returns a matrix where the vector $\boldsymbol{b}$ is put on the diagonal. $\boldsymbol{0}_N$ is the all-zero vector of length N. $\boldsymbol{0}, \boldsymbol{1}$ and $\boldsymbol{I}$ represent the all-zero, all-one and the identity matrix of suitable size, respectively. We consider column vectors. $(\cdot)^H$ represents the conjugate transpose of a vector or a matrix, whereas $(\cdot)^T$ represents the transpose. $z_r = \Re(z)$, $z_i = \Im(z)$ returns the real and imaginary part of the complex-valued argument $z = z_r + \iota z_i$, respectively. $\iota = \sqrt{-1}$. t and l are designated to the continuous-time variable and the discrete-time index, respectively.

2. Block transmission model

The block transmission model for linear modulation schemes is shown in **Figure 1**. For details, see [18]:

- **SRC** (**SNK**) represents the digital source (sink). **SRC** repeatedly generates successive streams of k bits, i.e., $\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_M$.
- $\boldsymbol{q}\ (\hat{\boldsymbol{q}}) \in \{0,1\}^k$ is the vector of source (detected) bits of length k.
- $\boldsymbol{q}_c \in \{0,1\}^n$ is the vector of encoded source bits of length $n > k$. For an *uncoded* transmission $\boldsymbol{q}_c = \boldsymbol{q}$ (and thus $k = n$).
- COD performs a bijective map from $\boldsymbol{q}$ to $\boldsymbol{q}_c$ where $n > k$ (adding redundancy). We consider in this chapter binary LDPC codes. Only 2^k combinations of n bits out of overall 2^n

[3]Except for $\boldsymbol{L}$, $\check{\boldsymbol{L}}$ and $\boldsymbol{L}_{ch}$ which are vectors.

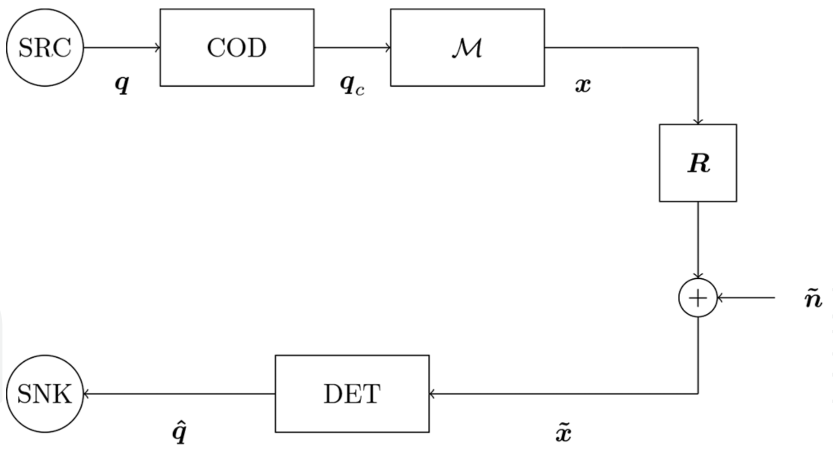


Figure 1. Block transmission model for linear modulation schemes. **SRC** (**SNK**) represents the digital source (sink). DET is the detector. COD performs the encoding process (adding redundancy). $\mathcal{M}$ maps encoded bits to complex-valued symbols. $\boldsymbol{R}$ is the block transmit matrix.

combinations are used. The set of the 2^k combinations represent the code book $\mathcal{C}$. $r_c = k/n$ is the code rate.

- $\boldsymbol{x} \in \boldsymbol{\psi}^N$ is the transmit vector of length N.
- N is the block size. Successive transmit vectors are separated by a guard time to avoid interference between different blocks. Thus, **Figure 1** describes the transmission for a single block and stays valid for the next block (possibly with a different $\boldsymbol{R}$).
- $\boldsymbol{\psi} = \{\psi_1, \psi_2, \ldots, \psi_{2^m}\}$, $m \in \mathbb{N}/\{0\}$ is the symbol alphabet. There exist $2^{m \cdot N}$ possible transmit vectors. The set of all possible transmit vectors is $\boldsymbol{\chi}$. The mapping from $\boldsymbol{q}_c$ to $\boldsymbol{x}$ is performed by $\mathcal{M}$. Each symbol ψ represents m bits. A special class of symbol alphabets are the so-called *separable* symbol alphabet $\boldsymbol{\psi}^{(s)}$ [19, 20].
- $\tilde{\boldsymbol{x}}$ is the receive vector of length N. In general $\tilde{\boldsymbol{x}} \in \mathbb{C}^N$.
- We distinguish:
  - For an uncoded transmission $M \times k = m \times N$.
  - For a coded transmission and $N < n/m$: One codeword lasts over many transmit blocks.
  - For a coded transmission and $N = n/m$: One codeword lasts exactly over a single transmit block.
  - For a coded transmission and $N = M \times n/m$: M codewords are contained in a single transmit block.
- $\boldsymbol{R} = \{r_{ij} : i, j \in \{1, 2, \cdots, N\}\}$ is the block transmit matrix of size $N \times N$. $\boldsymbol{R}$ is hermitian and positive semidefinite. The block transmit matrix $\boldsymbol{R}$ contains the whole knowledge about the transmission scheme (transmit and receive filters) and the physical propagation channel between transmitter(s) and receiver(s) [18].

- $\tilde{\boldsymbol{n}}$ is a sample function of an additive Gaussian noise vector process of length N with zero mean and covariance matrix $\boldsymbol{\Phi}_{\tilde{\boldsymbol{n}}\tilde{\boldsymbol{n}}} = \frac{N_0}{2} \cdot \boldsymbol{R}$ where $\frac{N_0}{2}$ is the double-sided noise power spectral density.
- DET is the detector including equalization and decoding.

The model in **Figure 1** is a general model and fits to different transmission schemes like orthogonal frequency division multiplexing (OFDM), code division multiple access (CDMA), multicarrier CDMA (MC-CDMA) and multiple-input multiple-output (MIMO). The relation with the original continuous-time (physical) model can be found in [11, 18]. The model in **Figure 1** can be described mathematically as follows [11]:

$$\tilde{\boldsymbol{x}} = \boldsymbol{R} \cdot \boldsymbol{x} + \tilde{\boldsymbol{n}}. \tag{1}$$

By decomposing $\boldsymbol{R}$ into a diagonal part $\boldsymbol{R}_d$ = diag$_\mathrm{m}\{\boldsymbol{R}\}$ and a nondiagonal part $\boldsymbol{R}_{\setminus d}$ = $\boldsymbol{R}$–$\boldsymbol{R}_d$, Eq. (1) can be rewritten as:

$$\tilde{\boldsymbol{x}} = \underbrace{\boldsymbol{R}_d \cdot x}_{\text{signal}} + \underbrace{\boldsymbol{R}_{\setminus d} \cdot \boldsymbol{x}}_{\text{interference}} + \underbrace{\tilde{\boldsymbol{n}}}_{\text{additive noise}} . \tag{2}$$

For the j-th element of the receive vector $j \in \{1, 2, \cdots, N\}$ Eq. (2) can be expressed as

$$\tilde{x}_j = r_{jj} \cdot x_j + \sum_{\substack{m=1 \\ m \neq j}}^{N} r_{jm} \cdot x_m + \tilde{n}_j. \tag{3}$$

We notice from Eqs. (2), (3) that the nondiagonal elements of $\boldsymbol{R}$ describe the interference between the elements of the transmit vector at the receiver side. For interference-free transmission $\boldsymbol{R}_{\setminus d}$ = $\boldsymbol{0}$. For an interference-free transmission over an additive white Gaussian noise (AWGN) channel $\boldsymbol{R} = \boldsymbol{I}$.

Figure 2 shows the channel matrix for a MIMO transmission scheme for different number of transmit/receive antennas. **Figure 3** shows the channel matrix for OFDM with/without spreading. **Figure 4** shows the channel matrix for MIMO-OFDM. In **Figures 2–4**, the darker the elements, the larger the absolute values of the entries of the corresponding matrix $\boldsymbol{R}$, and hence larger the interference [21].

Remark 1. For a clear distinction between *channel matrix* and *block transmit matrix*, we refer to [11, 18]. Generally speaking, the block transmit matrix $\boldsymbol{R}$ is a block diagonal matrix of "many" channel matrices.

The detector DET in **Figure 1** has to deliver a vector $\hat{\boldsymbol{q}}$ with a minimum bit error rate compared to $\boldsymbol{q}$ (conditional to the available computational power) given that COD, $\mathcal{M}$ and $\boldsymbol{R}$ are known at the receiver side. The optimum detection (maximum likelihood detection) for realistic cases is often infeasible. Therefore, suboptimum schemes are used, mainly based on separating the detection into an equalization EQ (to cope with interference caused by $\boldsymbol{R}_{\setminus d}$) and a decoding DEC (to utilize the redundancy added by COD). In this case, we distinguish between separate

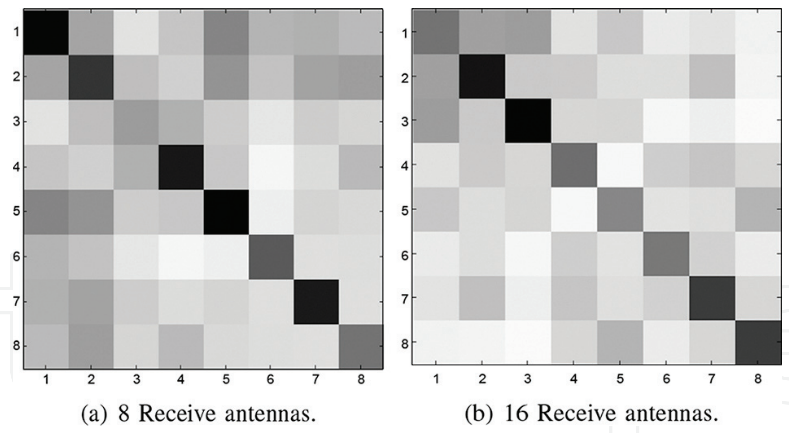


Figure 2. Visualization of the channel matrix for a MIMO transmission scheme with eight transmit antennas and different receive antennas.

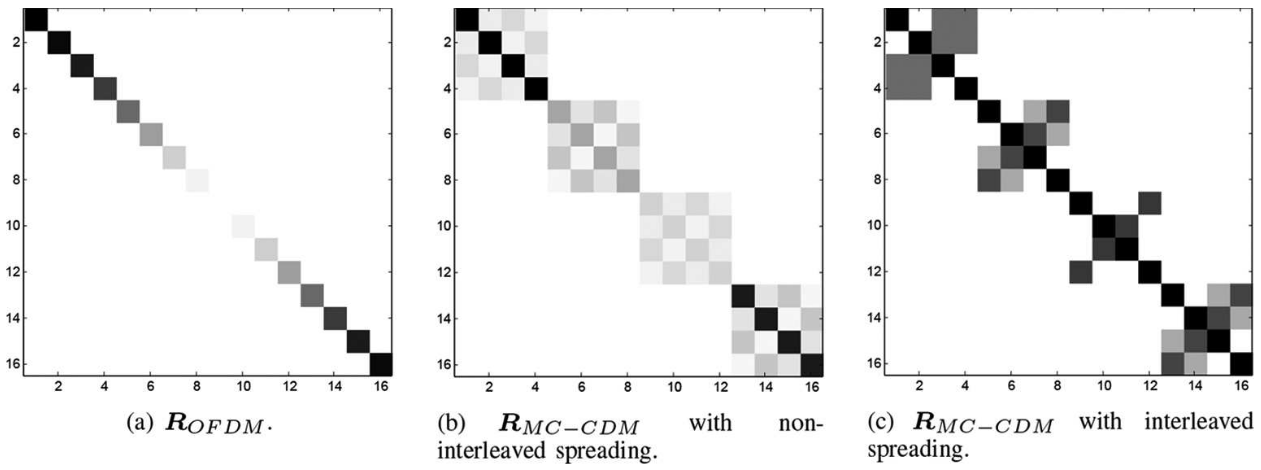


Figure 3. Visualization of the channel matrix for OFDM with 16 subcarriers and spreading over four subcarriers with/without interleaving.

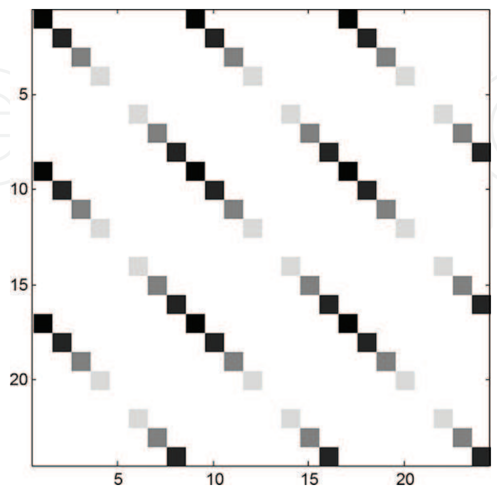


Figure 4. Visualization of the channel matrix for a MIMO-OFDM transmission scheme with eight subcarriers and three transmit antennas.

and joint equalization and decoding, cf. **Figure 5**. The superiority of the latter one is widely accepted: The separate equalization and decoding as in **Figure 5(a)** in general leads to a performance loss since the equalizer does not utilize the knowledge available at the decoder [14]. Each of the components DET, EQ and DEC can be seen as a pattern classifier. By separating the detection into equalization and decoding, an optimum detection in general cannot be achieved anymore (even if optimum equalization and optimum decoding are individually applied). Nevertheless, this is a common practice.

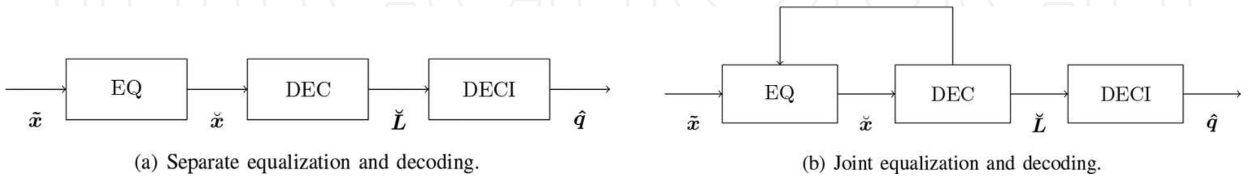


Figure 5. Detection: EQ is the equalizer, DEC is the decoder, DECI is a hard decision function. Notice the feedback from the decoder to the equalizer in (b), i.e., the turbo principle.

DECI in **Figure 5** is a hard decision function. For a coded transmission, DECI is a unit step function. For an uncoded transmission, COD and DEC are removed from **Figure 1** and **Figure 5**, respectively. DECI in this case is a stepwise function depending on the symbol alphabet $\boldsymbol{\psi}$ which maps the (in general complex-valued) elements of the equalized vector $\breve{x}$ to the vector of detected symbols $\hat{x} \in \boldsymbol{\psi}^N$ cf. **Figure 8**. The map from $\hat{x}$ to $\hat{q}$ is then straightforward. In summary

- For an uncoded transmission DECI: $\mathbb{C}^N \rightarrow \boldsymbol{\psi}^N$.
- For a coded transmission DECI: $\mathbb{R}^k \rightarrow \{0,1\}^k$.

3. Vector equalization

For an uncoded transmission, the detection DET reduces to a vector equalization EQ as shown in **Figure 6**.

The optimum vector equalization rule (the maximum likelihood one) is based on the minimum Mahalanobis distance and is given as [21]

$$\hat{x}_{\mathrm{ML}} = \underset{\xi \in \chi}{\mathbf{arg\ min}} \left\{ \frac{1}{2} \cdot \xi^H \cdot \boldsymbol{R} \cdot \xi - \Re\{\xi^H \cdot \tilde{x}\} \right\}. \tag{4}$$

For each receive vector $\tilde{x}$, the optimum vector equalizer calculates the Mahalanobis distance Eq. (4) to *all* possible transmit vectors χ of cardinality $2^{m \cdot N}$ and decides in favor of that possible transmit vector $\hat{x}_{\mathrm{ML}}$ with the minimum Mahalanobis distance to the receive vector $\tilde{x}$, i.e., exhaustive search is required in general. This can be performed for small $2^{m \cdot N}$ which is usually

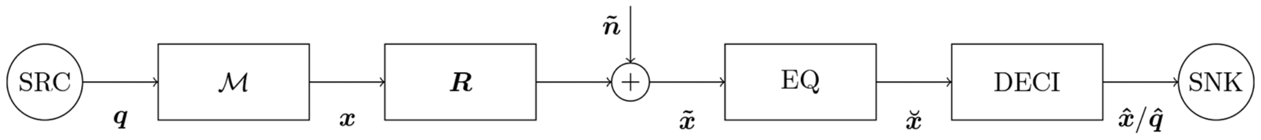


Figure 6. Uncoded block transmission model. Neither encoding at the transmitter nor decoding at the receiver. The detection reduces to a vector equalization EQ.

not the case in practice. Therefore, suboptimum equalization schemes are applied, which trade-off performance against complexity.

4. Continuous-time single-layer recurrent neural networks for vector equalization

The dynamical behavior of continuous-time single-layer recurrent neural networks of dimension N', abbreviated in the following by RNN[4], is given by the state-space equations [22]:

$$\begin{aligned}\Upsilon_e \cdot \frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} &= -\boldsymbol{u}(t) + \boldsymbol{W}\cdot\boldsymbol{v}(t) + \boldsymbol{W}_0\cdot\boldsymbol{e},\\ \boldsymbol{v}(t) = \boldsymbol{\varphi}(\boldsymbol{u}(t)) &= [\varphi_1(u_1(t)),\ \varphi_2(u_2(t)),\ \cdots,\ \varphi_{N'}(u_{N'}(t))]^T.\end{aligned} \tag{5}$$

In Eq. (5), $\boldsymbol{\Upsilon}_e$ is a diagonal and positive definite matrix of size $N' \times N'$. $\boldsymbol{v}(t)$ is the output, $\boldsymbol{u}(t)$ is the inner state, $\boldsymbol{e}$ is the external input. $\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{e} \in \mathbb{C}^{N'}$.$\varphi_j(\cdot) : j \in \{1, 2, \ldots, N'\}$ is the j-th activation function. $\boldsymbol{W} = \{w_{jj'} : j, j' \in \{1, 2, \cdots, N'\}\} \in \mathbb{C}^{N'\times N'}$, $\boldsymbol{W}_0 = \mathrm{diag}_v\{[w_{10}, w_{20}, \cdots, w_{N'0}]^T\} \in \mathbb{R}^{N'\times N'}$ are the weight matrices. The real-valued RNN (all variables and functions in Eq. (5) are real-valued) is shown in **Figure 7**, which is known as "additive model" or "resistance-capacitance model" [23]. In this case, $w_{jj'} = \frac{R_j}{R_{jj'}}$ is the weight coefficient between the output of the j'-th neuron and the input of the j-th neuron, $w_{j0} = \frac{R_j}{R_{j0}}$ is the weight coefficient of the j-th external input. We also notice that the feedback $\boldsymbol{W}\cdot\boldsymbol{v}$ in Eq. (5) and **Figure 7** is a linear function of the output $\boldsymbol{v}$. Moreover, $\boldsymbol{\Upsilon}_e$ can be given in this case as $\boldsymbol{\Upsilon}_e = \mathrm{diag}_v\{[R_1\cdot C_1, R_2\cdot C_2, \ldots, R_{N'}\cdot C_{N'}]^T\}$.

As a nonlinear dynamical system, the stability of the RNN is of primary interest [16]. This has been proven under specific conditions by Lyapunov's stability theory in [24] for real-valued RNN and in [22, 25] for complex-valued ones, among others. The RNN in Eq. (5) represents a general purpose structure. Based on N', φ, $\boldsymbol{W}$, $\boldsymbol{W}_0$ a wide range of optimization problems can be solved. First and most well-investigated applications of the RNN include the content addressable memory [24, 26], analog-to-digital converter (ADC) [27] and the traveling salesman problem [28]. In all these cases, no training is needed since the RNN is acting as a dynamical solver. This feature is desirable in many engineering fields like signal processing, communications, automatic control, etc., and has first been exploited by Hopfield in his

[4]The abbreviation RNN in this chapter inherently includes the continuous-time and the single-layer properties.

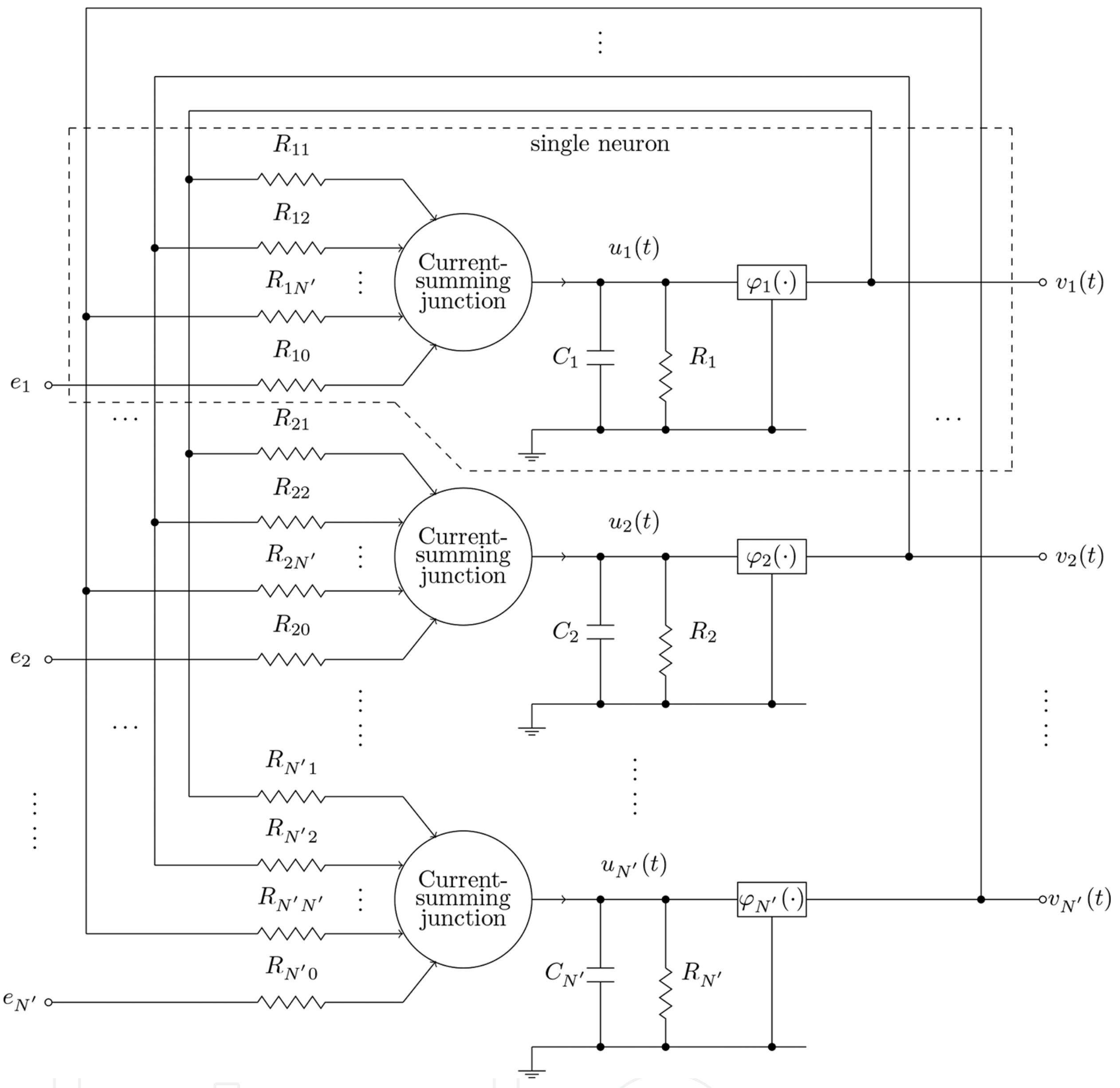


Figure 7. Continuous-time single-layer real-valued recurrent neural network. $\boldsymbol{v}(t)$ is the output, $\boldsymbol{u}(t)$ is the inner state, $\boldsymbol{e}$ is the external input and $\varphi(\cdot)$ is the activation function. This model is known as "additive model" or "resistance-capacitance model" [23].

pioneering work [24, 29], where information has been stored in a dynamically stable RNN. We focus in the following on the vector equalization.

Remark 2. The dimension of a real-valued RNN is the same as the number of neurons.

Remark 3. Two real-valued RNNs each of N' neurons are required to represent one complex-valued RNN (with dimension N'). This is possible by separating Eq. (5) into real and imaginary parts. However, this doubles in general the number of connections per neuron (and hence the number of multiplications) because of the required connections (represented by $\boldsymbol{W}_i$) between the two real-valued RNNs as it can be seen from the following equation:

$$\boldsymbol{\Upsilon}_e \cdot \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \boldsymbol{u}_r(t) \\ \boldsymbol{u}_i(t) \end{bmatrix} = -\begin{bmatrix} \boldsymbol{u}_r(t) \\ \boldsymbol{u}_i(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{W}_r & -\boldsymbol{W}_i \\ \boldsymbol{W}_i & \boldsymbol{W}_r \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{v}_r(t) \\ \boldsymbol{v}_i(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{W}_0 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_0 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{e}_r \\ \boldsymbol{e}_i \end{bmatrix}. \tag{6}$$

$\boldsymbol{\Upsilon}_e$ in this case is a diagonal positive definite matrix of size $2 \cdot N' \times 2 \cdot N'$ and

$$\begin{aligned} \boldsymbol{u}(t) &= \boldsymbol{u}_r(t) + \iota \boldsymbol{u}_i(t) \quad , \quad & \boldsymbol{e} &= \boldsymbol{e}_r + \iota \boldsymbol{e}_i \\ \boldsymbol{v}(t) &= \boldsymbol{v}_r(t) + \iota \boldsymbol{v}_i(t) \quad , \quad & \boldsymbol{W} &= \boldsymbol{W}_r + \iota \boldsymbol{W}_i. \end{aligned}$$

A. Vector equalization based on RNN

The usage of the RNN for vector equalization became known for multiuser interference cancellation in CDMA environments [30, 31]. However, this was limited to the binary phase-shift keying (BPSK) symbol alphabet $\boldsymbol{\psi} = \{-1, +1\}$. This has been generalized to complex-valued symbol alphabets in [21] by combining the results of references [20, 22, 32][5]. Based thereon, it has been proven that the RNN ends in a *local* minimum of Eq. (4) if the following relations are fulfilled [21], cf. Eqs. (1), (2), (5) and **Figures 6** and **7**.

$$\begin{array}{ccc} \boldsymbol{e} = \tilde{\boldsymbol{x}} & \boldsymbol{v} = \breve{\boldsymbol{x}} & N' = N \\ \boldsymbol{W}_0 = \boldsymbol{R}_d^{-1} & \boldsymbol{W} = \boldsymbol{I} - \boldsymbol{R}_d^{-1} \cdot \boldsymbol{R} & \boldsymbol{\varphi}(\cdot) = \boldsymbol{\theta}^{(opt)}(\cdot) \end{array} \tag{7}$$

and therefore $\hat{\boldsymbol{x}} = \mathrm{DECI}(\boldsymbol{v})$. **Figure 8** shows an example of an eight quadrature amplitude modulation (8 QAM) symbol alphabet and its corresponding DECI function. The relations in Eq. (7) are obtained by the comparison between the maximum likelihood function of the vector equalization and the Lyapunov function of the RNN.

The dynamical behavior of the vector equalization based on RNN can be given as, cf. Eqs. (1), (5), (7)

$$\begin{aligned} \boldsymbol{\Upsilon}_e \cdot \frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} &= -\boldsymbol{u}(t) + \breve{\boldsymbol{x}}(t) + \boldsymbol{R}_d^{-1} \cdot \boldsymbol{R} \cdot [\boldsymbol{x} - \breve{\boldsymbol{x}}(t)] + \boldsymbol{R}_d^{-1} \cdot \tilde{\boldsymbol{n}}, \\ \breve{\boldsymbol{x}}(t) = \boldsymbol{\theta}^{(opt)}\big(\boldsymbol{u}(t)\big) &= [\theta_1^{(opt)}\big(u_1(t)\big), \theta_2^{(opt)}\big(u_2(t)\big), \cdots, \theta_N^{(opt)}\big(u_N(t)\big)]^T. \end{aligned} \tag{8}$$

The locally asymptotical stability of Eq. (8) based on Lyapunov functions has been proved in [21] (based on [22]) for separable symbol alphabets $\boldsymbol{\psi}^{(s)}$. When Eq. (8) reaches an equilibrium point $\boldsymbol{u}_{ep}$, i.e., $\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = \boldsymbol{0}_N \Rightarrow \boldsymbol{u} = \boldsymbol{u}_{ep}$, Eq. (8) can be rewritten as

$$\boldsymbol{u}_{ep} = \breve{\boldsymbol{x}}_{ep} + \boldsymbol{R}_d^{-1} \cdot \boldsymbol{R} \cdot [\boldsymbol{x} - \breve{\boldsymbol{x}}_{ep}] + \boldsymbol{R}_d^{-1} \cdot \tilde{\boldsymbol{n}}. \tag{9}$$

If additionally, a correct equalization is achieved, i.e., $\breve{\boldsymbol{x}}_{ep} = \boldsymbol{x}$, the inner state is

[5]For *discrete-time* single-layer recurrent neural networks for vector equalization, we refer to references [19, 33].

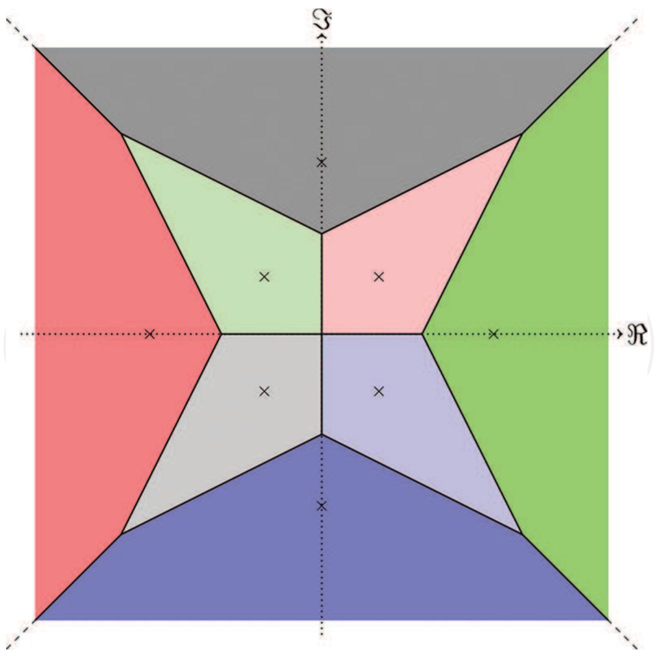


Figure 8. An example of an 8 QAM symbol alphabet and its corresponding DECI function. Each element of the symbol alphabet (marked with ×) has its own "decisions region" visualized by different colors. The function DECI delivers that element of the symbol alphabet, where the input argument lies in its corresponding decision region.

$$\boldsymbol{u}_{ep} = \boldsymbol{x} + \underbrace{\boldsymbol{R}_d^{-1} \cdot \tilde{\boldsymbol{n}}}_{\boldsymbol{n}_e} . \tag{10}$$

Thus, the RNN as vector equalizer, Eq. (8) acts as "analog dynamical solver" and there is no need for a training. The covariance matrix of $\boldsymbol{n}_e$ is $\boldsymbol{\Phi}_{\boldsymbol{n}_e\boldsymbol{n}_e} = \frac{N_0}{2} \cdot \boldsymbol{R}_d^{-1} \cdot \boldsymbol{R} \cdot \boldsymbol{R}_d^{-1}$. We define

$$\boldsymbol{\Sigma}_{\boldsymbol{n}_e} = \text{diag}_v\{[\sigma_1^2, \sigma_2^2, \cdots, \sigma_N^2]^T\} = \text{diag}_m\{\boldsymbol{\Phi}_{\boldsymbol{n}_e\boldsymbol{n}_e}\} = \frac{N_0}{2} \cdot \boldsymbol{R}_d^{-1}. \tag{11}$$

In Eq. (7), $\theta^{(opt)}(\cdot)$ is the optimum activation function and depends on the symbol alphabet $\boldsymbol{\psi}$. For BPSK (a real-valued case)

$$\theta^{(opt)}(u) = \tanh\left(\frac{u}{\sigma^2}\right). \tag{12}$$

where σ^2 is given in Eq. (11).

Remark 4. For separable symbol alphabets, $\boldsymbol{\psi} = \boldsymbol{\psi}^{(s)} \Rightarrow \theta^{(opt)}(u = u_r + \imath u_i) = \theta_r^{(opt)}(u_r) + \imath\theta_i^{(opt)}(u_i)$ [19].

B. Analog hardware implementation aspects: equalization

The analog signal processing as a matter of topical importance for modern receiver architectures was recognized in [34], where an analog vector equalizer—designed in BiCMOS technology—was considered as a promising application for the analog processing of baseband signals. The equalizer accepts sampled vector symbols in analog form with an advantage that the equalizer does not require an ADC at the input interface. At very high data rates, the exclusion of an ADC softens the trade-off between chip area requirement and overall power consumption. We discuss in the following section the main features/challenges of the analog implementation of the vector equalizer based on RNN.

Structure: An RNN of dimension N' (in general $2 \cdot N'$ neurons) is capable to act as a vector equalizer as long as the block size at the transmitter side N (over all possible symbol alphabets, coding schemes and block sizes) is as maximum as N', i.e., $N \leq N'$.

Activation function: The definition of the optimum activation function $\theta^{(opt)}(\cdot)$ is not general, but depends on the symbol alphabet under consideration. Different symbol alphabets need different activation functions. However, we have proven in [20] that for square QAM symbol alphabets—the most relevant ones in practice—$\theta^{(opt)}(\cdot)$ can be approximated as a sum of a limited number of shifted and weighted hyperbolic tangent functions. Square QAM symbol alphabets are separable ones, cf. Remark 4. The analog implementation of the hyperbolic tangent well befits the large-signal transfer function of transconductance stages based on bipolar differential amplifiers [3, 34].

Adaptivity: A vector equalizer must be capable to adapt to different and time-variant interference levels. The adaptivity is regulated by the measurement of the block transmit matrix $\boldsymbol{R}$, a task performed by a "channel estimation unit" (CEU). The weight matrices $\boldsymbol{W}$ and $\boldsymbol{W}_0$ are then computed as in Eq. (7) and forwarded to the RNN (**Figure 9**). Thus, the weight matrices $\boldsymbol{W}$ and $\boldsymbol{W}_0$ are *not* the outcome of any training algorithm but related directly to $\boldsymbol{R}$, cf. Eq. (7). This represents a typical example for the mixed-signal integrated circuit, where the weight coefficients are (obtained and) stored digitally, converted into analog values, later used as weight coefficients for the analog RNN [8].

For the j-th neuron in the additive model **Figure 7**, the ratio between two resistors R_j and $R_{jj'}$ (R_j and R_{j0}) is used to configure each weight coefficient $w_{jj'}$ (w_{j0}). According to the additive

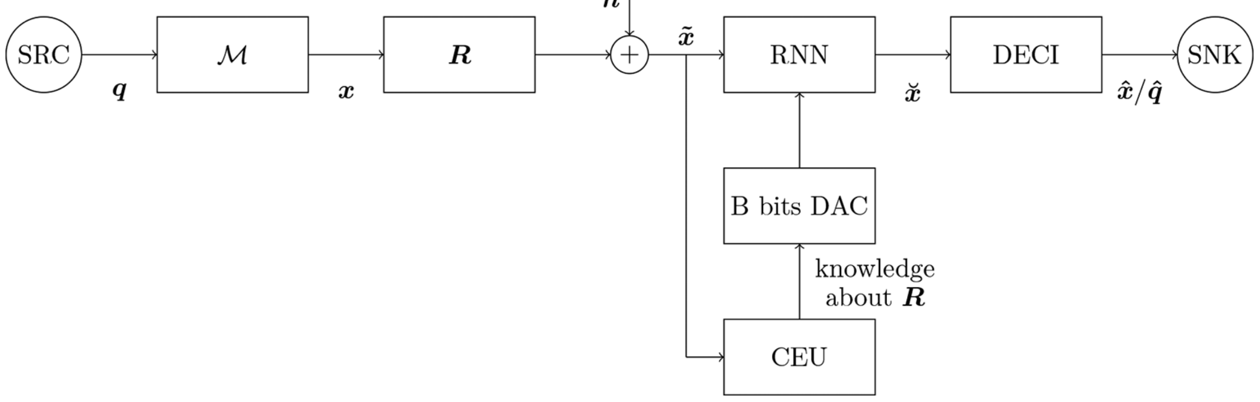


Figure 9. Uncoded block transmission model. The detection reduces to a vector equalization EQ. The channel estimation unit (CEU) estimates the block transmit matrix $\boldsymbol{R}$.

model, $R_{jj'}$ and R_{j0} can assume both positive and negative values, and the absolute value theoretically extends from R_j to infinite (for $w_{jj'} \in [-1, +1]$). This puts serious limitations to the direct implementation of the model. In [3], we showed how this difficulty can be overcome by using a Gilbert cell as a four-quadrant analog multiplier. A Gilbert cell [35] is composed of two pairs of differential amplifiers with cross-coupled collectors, and is controlled by a differential voltage input G_{ji} applied at the base gate of the transistors. When biased with a differential tail current $I_{ji} = I_{ji}^{+} - I_{ji}^{-}$, the differential output current $I_{ji,w} = I_{ji,w}^{+} - I_{ji,w}^{-}$ is a fraction w of the tail current I_{ji}, as a function of the input voltage G_{ji}:

$$I_{ji,w} = I_{ji,w}^{+} - I_{ji,w}^{-} = f_{Gc}(I_{ji} = I_{ji}^{+} - I_{ji}^{-}, G_{ji}) = w \cdot I_{ji} \in [-I_{ji}, +I_{ji}]. \tag{13}$$

Accuracy: Locally asymptotical Lyapunov stability can be guaranteed for the RNN in Eqs. (5), (8) if, among others, the hermitian property is verified for the weight matrix $\boldsymbol{W}$ (the symmetric property in the real-valued case). Inaccuracies in the weights' representation may jeopardize the Lyapunov stability and impact the performance of the vector equalizer. The first cause of weights' inaccuracy may arise from the limited accuracy of the analog design in terms of components' parasitics, devices' mismatch, process variation, just to name a few. Those inaccuracies (if modest) are expected to slightly degrade the performance without causing a catastrophic failure, thanks to the high nonlinearity of the equalization algorithm. Moreover, it has been shown in [8, 36] that in some cases, they produce beneficial effects: These imperfections incorporate some kind of simulated annealing which enables escaping local minima by allowing occasionally "uphill steps" since the Lyapunov stable RNN is a gradient-like system. This feature is emulated in discrete-time by *stochastic* Hopfield networks [23]. Non-precision of the weights may also arise from an insufficient resolution of the digital-to-analog converter (DAC) (**Figure 9**). On the other hand, an overzealous DAC design increases the chip area, the power consumption and adds complexity to the interface between the analog vector equalizer and the digital CEU. In this case, a conservative approach suggests to use a DAC with enough resolution to match the precision used by the CEU.

Interneuron connectivity and reconfigurability: Scaling the architecture of an analog VLSI design is not straightforward. A vector equalizer based on recurrent neural networks is composed by the repetition of equal sub-systems, i.e., the neurons. Using a bottom-up approach, the first step to scale the system involves the redesign of the single neuron in order to handle more feedback inputs. In a successive step, the neurons are connected together and a system-level simulation is performed to check the functionality of the system. However, several design choices must be made during the process and it is not guaranteed that the optimum architecture for a certain number of neurons is still the best choice when the number of neurons changes. For large N, the block transmit matrix $\boldsymbol{R}$, defining the weight matrix $\boldsymbol{W}$, is usually sparse. If a maximum number of nonzero elements over the rows of $\boldsymbol{R}$ is assumed, the requirement for a full connectivity between the neurons in **Figure 7** can be relaxed, and only a maximum number of connections per neuron will be necessary. In this case, however, in addition to the "adaptivity", the RNN must be reconfigured according to the position of the nonzero elements in $\boldsymbol{R}$. The hardware simplification given by the partial connectivity may be counterbalanced by the necessity of a further routing (e.g., multiplexing/demultiplexing) of the feedback. For special cases, where the block transmit matrix can be reordered around the diagonal, more independent RNNs can be simply used in parallel. In **Figures 3(b)** and **3(c)**, four independent RNNs, each of dimension four, can be used in parallel. Additionally, for specific transmission schemes such as MIMO-OFDM in **Figure 4**, the connectivity can be assumed limited (number of transmit antennas minus one) and fixed (crosstalk only between *same* subcarriers, when used simultaneously on different transmit antennas).

Example 1. In **Figure 4**, eight RNNs (number of subcarriers) each of dimension of three (number of transmit antennas) can be used in parallel. Each neuron has two feedback inputs.

C. Circuit design

We review here the main features of the analog circuit design of an RNN as vector equalizer working with the BPSK symbol alphabet and composed of four neurons. Detailed explanation can be found in reference [3]. The RNN is realized in IHP 0.25 μm SiGe BiCMOS technology (SG25H3). A simplified schematic of a neuron is shown in **Figure 10**. Schematics of gray boxes are presented in **Figure 11**.

The dynamical behavior of the circuit in **Figures 10** and **11** is described as [3]

$$\begin{aligned} \boldsymbol{\Upsilon} \cdot \frac{\mathrm{d}\boldsymbol{u}'(t)}{\mathrm{d}t} &= -\boldsymbol{u}'(t) + \boldsymbol{W} \cdot \boldsymbol{v}'(t) + \boldsymbol{W}_0 \cdot \boldsymbol{e}', \\ \frac{R \cdot I_t}{N-1} \cdot \tanh\left(\frac{u'(t)}{2 \cdot V_t}\right) &= v'(t), \\ \tau \cdot \boldsymbol{I} &= \boldsymbol{\Upsilon}. \end{aligned} \tag{14}$$

which is equivalent to Eq. (5). $\tau = R \cdot C$ is the time constant of the circuit. R is shown in **Figure 10** and C is a fictitious capacitance between the nodes and u_j^+ and $u_j^-.V_t$ is the thermal voltage and I_t is the tail current in **Figure 11**. The circuit is fully differential and the differential currents and voltages are denoted as, cf. **Figures 10** and **11**:

$$\begin{array}{ccc} I_{ji} = I_{ji}^+ - I_{ji}^- & I_j = I_j^+ - I_j^- & I_o = I_o^+ - I_o^-, \\ I_{ji,w} = I_{ji,w}^+ - I_{ji,w}^- & u'_j = u_j^+ - u_j^- & e'_j = e_j^+ - e_j^- \end{array} \tag{15}$$

(1) Performance: Simulation results based on the above described analog RNN are shown in **Figure 12**. The interference is described by the channel matrix $\boldsymbol{R}_{test}$.

$$\boldsymbol{R}_{test} = \begin{bmatrix} 1 & 0.24 & -0.34 & -0.57 \\ 0.24 & 1 & 0.32 & 0.29 \\ -0.34 & 0.32 & 1 & 0.25 \\ -0.57 & 0.29 & 0.25 & 1 \end{bmatrix}$$

The black dashed line shows the bit error rate (BER) for a BPSK symbol alphabet in an AWGN channel (an interference-free channel). Performance achieved by the maximum likelihood

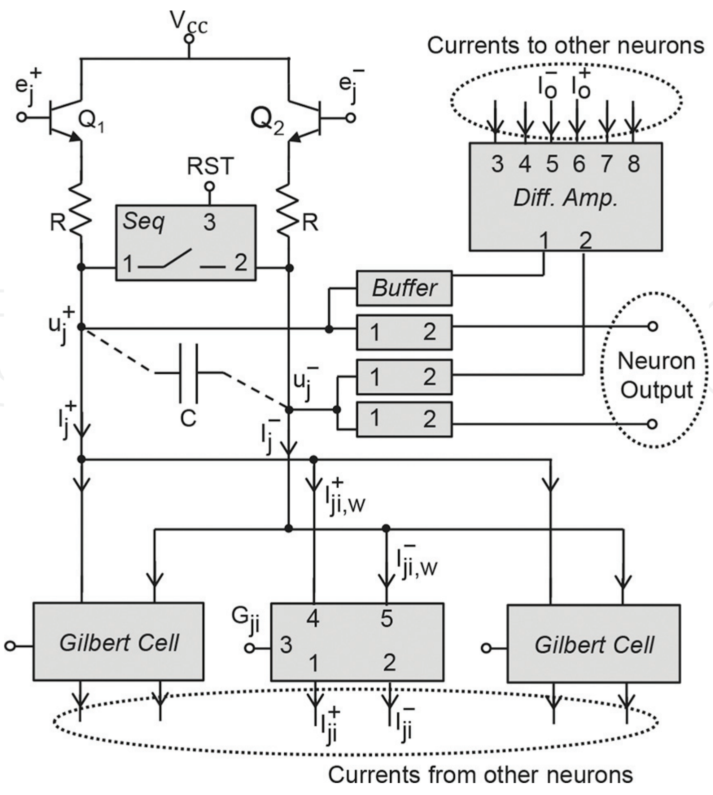


Figure 10. A simplified schematic of a single neuron as a part of a (four neurons) RNN analog vector equalizer. u'_j is the inner state, e'_j is the external input and G_{ji} is used for adapting the weight coefficient w_{ji} from the output of the i-th neuron to the input of the j-th neuron. The circuit is fully differential [3].

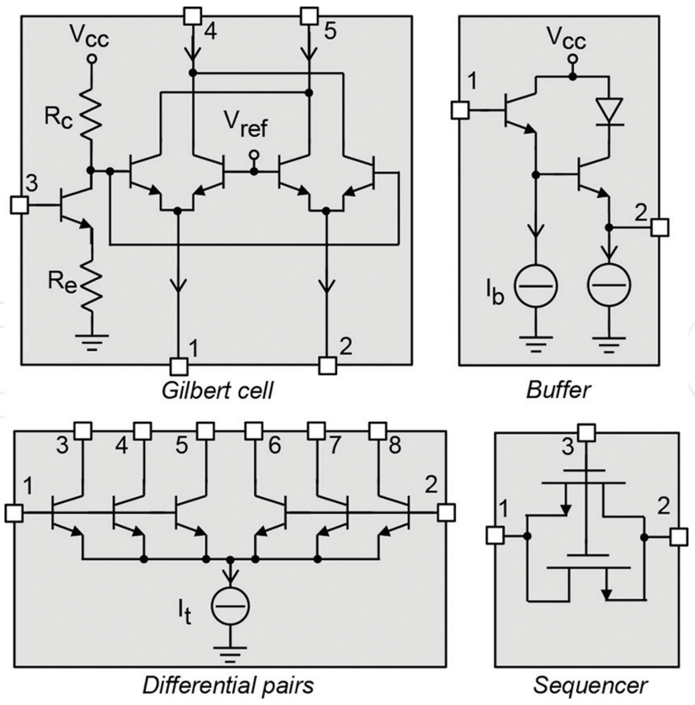


Figure 11. Details of the circuit building blocks. Gilbert cell used as a four-quadrant analog multiplier, buffer stages, BJT differential pairs for the generation of the hyperbolic tangent function and a metal-oxide-semiconductor field-effect transistor (MOSFET) switch used as a sequencer [3].

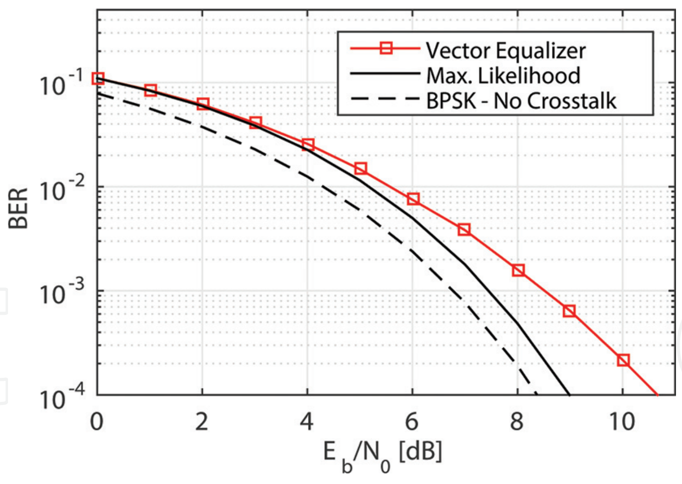


Figure 12. BER vs. E_b/N_0 for the analog RNN vector equalizer. Evolution time equals $10 \cdot \tau$. BPSK symbol alphabet and channel matrix $\boldsymbol{R}_{test}$.

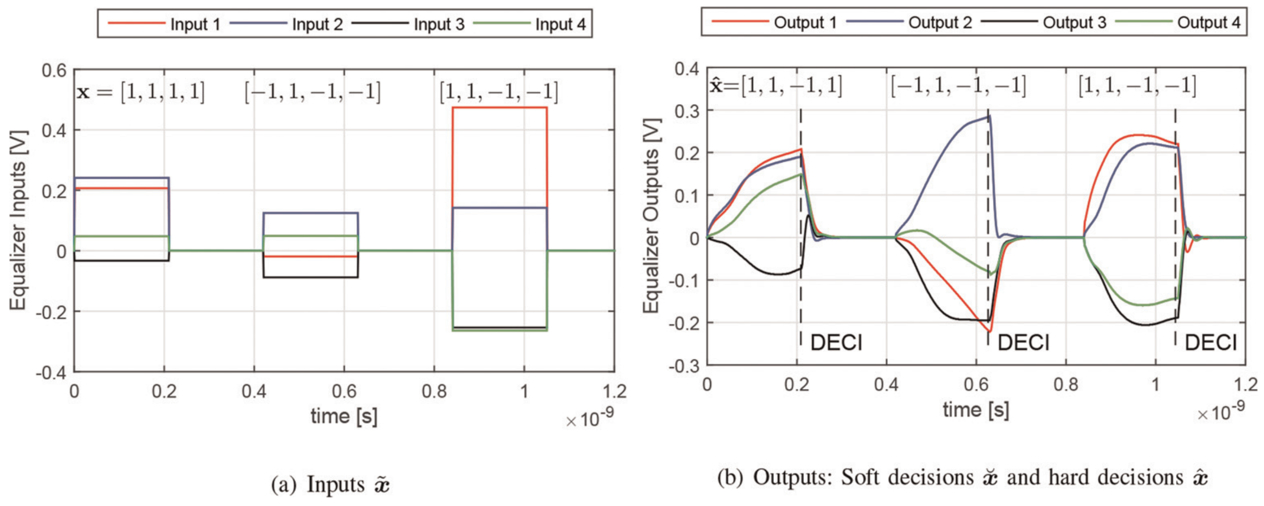


Figure 13. An example of a transient simulation for the analog RNN vector equalizer. (a) Inputs $\tilde{x}$ (b) Outputs: soft decisions $\breve{x}$ and hard decisions $\breve{x}$.

algorithm in Eq. (4) is included as a solid black line. The performance of the analog RNN vector equalizer[6] is presented in a solid red line with square markers. Compared to the optimum algorithm, the signal-to-noise ratio (SNR) loss for the analog RNN vector equalizer can be quantified in approximately 1.7 dB at a BER of 10^{-4}. This loss in SNR emphasizes the suboptimality of the RNN as vector equalizer and depends on the channel matrix. **Figure 13** shows an example of a transient simulation for the analog RNN vector equalizer. The time constant is approximately τ = 40 ps. The SNR ratio is set to 2 dB and a series of three receive vectors are equalized in sequence. Because of the channel matrix and noise, the sampled vectors at the input of the equalizer $\tilde{x}$ present different signs and values, compared to the sent

[6]Analog RNN vector equalizer refers to the described analog hardware-implemented RNN for vector equalization.

vectors $\boldsymbol{x}$ (shown in square brackets). The equalization of each receive vector lasts $10 \cdot \tau$. First half of this interval (evolution time) is used to reach a stable state, while the second half of the interval (reset time) is used to return to a predefined inner state (all-zero state) before the equalization of a new vector starts. At the end of the evolution time, a decision is made based on the sign of the output vector (the decision function DECI for BPSK is a sign function). In our example, a comparison between the sent and the recovered bits shows an error of one bit out of twelve, equivalent to a BER$\approx \frac{1}{12}$, a result in line with the BER shown in **Figure 12**.

Remark 5. The evolution and reset times are the two limiting factors for the maximum throughput of the analog RNN vector equalizer. However, they cannot be unlimitedly minimized since the RNN needs a minimum evolution time to reach an equilibrium point representing a local minimum of the Lyapunov function, i.e., a local minimum of Eq. (4).

(2) Energy efficiency: The energy efficiency of a hardware "architecture" is the ratio between the power requirement (Watt) of the architecture and its achievement in a given time period. In our case, the throughput of the equalizer represents the achievement. Combining the value of τ and the power consumption, the abovementioned analog vector equalizer is expected to win the competition versus common digital signal processing, thanks to three to four orders of magnitude better energy efficiency [3].

5. Channel coding

Channel coding (including encoding at the transmitter side COD and decoding at the receiver side DEC) aims to enable an error-free transmission over noisy channels with maximum possible transmit rate. This is done by adding redundancy (extra bits) at the transmitter side, i.e., the bijective map from $\boldsymbol{q}$ to $\boldsymbol{q}_c$ (**Figure 14**,) such that the codewords $\boldsymbol{q}_c$ are sufficiently distinguishable at the receiver side even if the noisy channel corrupts some bits during the transmission. **Figure 14** shows a coded transmission over an AWGN channel.

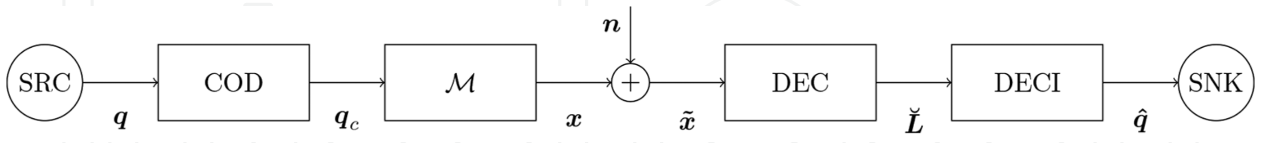


Figure 14. Coded transmission over an BER channel.

For every received codeword, the optimum decoding (the maximum likelihood one) needs to calculate the distance between the received codeword and all possible codewords $\mathcal{C}$, which makes it infeasible for realistic cases (except for convolutional codes which are not considered here). We focus on binary LDPC codes and their corresponding suboptimum decoding algorithm: the *belief propagation* with BPSK symbol alphabet. LDPC codes [37] belong to the class of binary linear block codes and have been shown to achieve an error rate very close to the Shannon limit (a performance lower bound) for the AWGN channel and have been implemented in many practical systems such as the satellite digital video broadcast (DVB-S2)

[38]. A binary linear block code is characterized by a binary parity check matrix $\boldsymbol{H}$ of size $(n-k) \times n$ for $n > k$.

6. Continuous-time single-layer high-order recurrent neural networks for belief propagation

One of the largest drawbacks of RNNs is their quadratic Lyapunov function [39]. Optimization problems associated with cost functions of higher degree cannot be solved "satisfactorily" by RNNs. Increasing the order of the Lyapunov function leads to a nonlinear feedback in the network. In doing so, we obtain the single-layer high-order recurrent neural network, named differently in literature, depending on the nonlinear feedback [39–42].

Remark 6. High-order recurrent neural networks are in the literature exclusively real-valued.

Figure 15 shows the continuous-time single-layer high-order recurrent neural network, abbreviated in the following by HORNN[7].

The dynamical behavior is given by

$$\begin{aligned}
\boldsymbol{\Upsilon}_d \cdot \frac{\mathrm{d}\check{\boldsymbol{u}}(t)}{\mathrm{d}t} &= -\check{\boldsymbol{u}}(t) + \check{\boldsymbol{W}} \cdot \check{\boldsymbol{f}}\left(\check{\boldsymbol{v}}(t)\right) + \check{\boldsymbol{W}}_0 \cdot \check{\boldsymbol{e}}, \\
\check{\boldsymbol{v}}(t) &= \check{\boldsymbol{\varphi}}\left(\check{u}(t)\right) = \left[\check{\varphi}_1\left(\check{u}_1(t)\right), \check{\varphi}_2\left(\check{u}_2(t)\right), \cdots, \check{\varphi}_{\check{n}}\left(\check{u}_{\check{n}}(t)\right)\right]^T, \\
\boldsymbol{\Upsilon}_d &= \mathrm{diag}_v\left\{\left[\check{R}_1 \cdot \check{C}_1, \check{R}_2 \cdot \check{C}_2, \ldots, \check{R}_{\check{n}} \cdot \check{C}_{\check{n}}\right]^T\right\}.
\end{aligned} \tag{16}$$

The parameters in Eq. (16) can be linked to **Figure 15** in the same way as Eq. (5) linked to **Figure 7**. $\check{\boldsymbol{f}}(\check{\boldsymbol{v}})$ is a real-valued continuously differentiable vector function. In addition, $\check{\boldsymbol{f}}(\boldsymbol{0}_{\check{n}}) = \boldsymbol{0}_{\check{n}}$. It is worth mentioning that the term "high-order" in this case refers to the interconnections between the neurons rather than the degree of the differential equation describing the dynamics. As for RNNs, this is still of first order, cf. Eq. (16).

Remark 7. In the special case $\check{\boldsymbol{f}}(\check{\boldsymbol{v}}) = \check{\boldsymbol{v}}$, the HORNN reduces to the (real-valued) RNN.

In order to apply HORNNs to solve optimization tasks, their stability has to be investigated. A property without which the behavior of dynamical systems is often suspected [39]. This was the topic of many publications [39–42]. A common denominator of the locally asymptotical stability proof of the HORNN based on Lyapunov functions is

- $\check{\varphi}(\cdot)$ is continuously differentiable and a strictly increasing function.
- The right side of the first line of Eq. (16) can be rewritten as a gradient of a scalar function.

[7]The abbreviation HORNN in this chapter inherently includes the continuous-time, single-layer and real-valued properties.

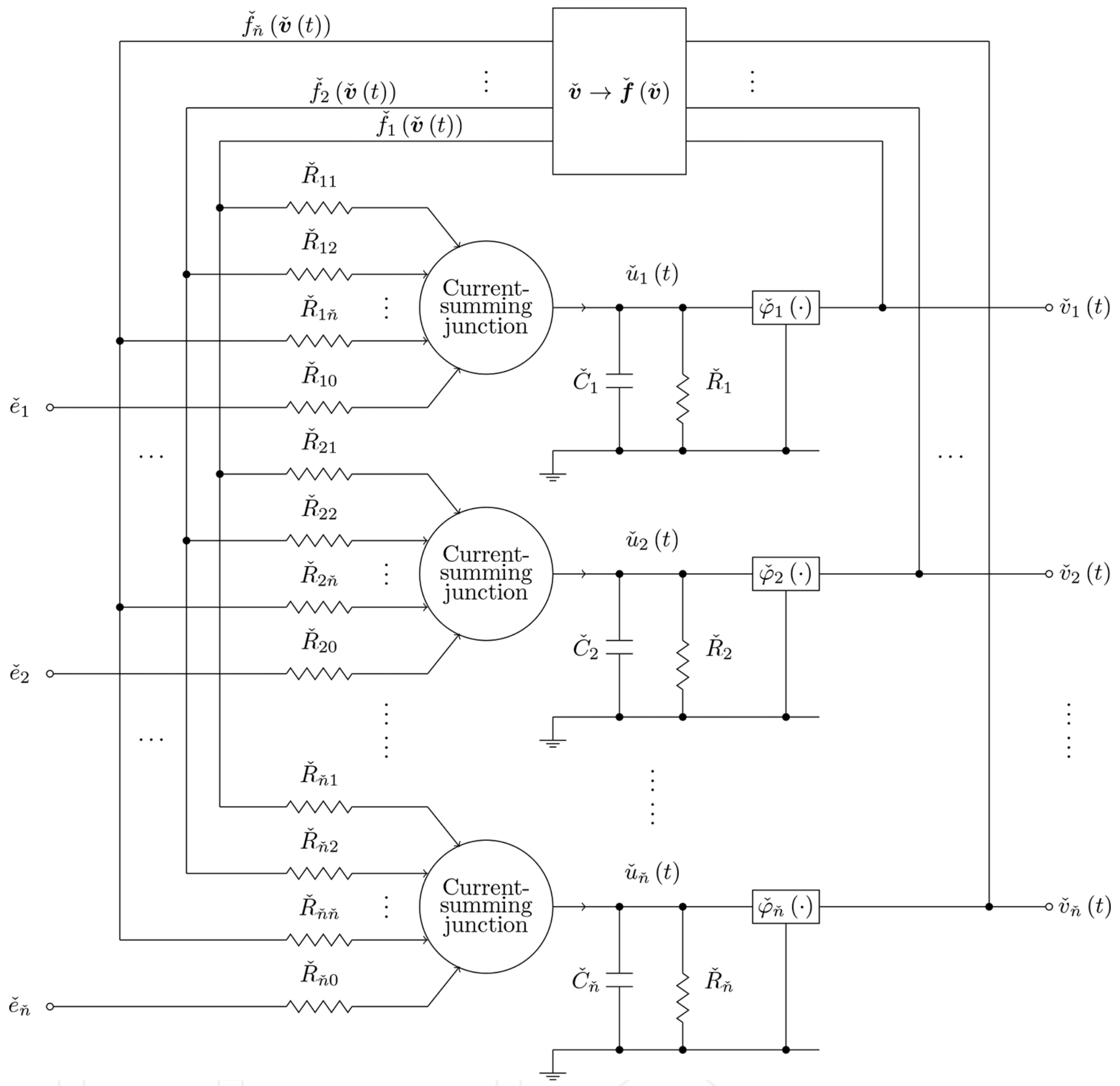


Figure 15. Continuous-time single-layer real-valued high-order recurrent neural network. $\check{v}(t)$ is the output, $\check{u}(t)$ is the inner state, $\check{e}$ is the external input and $\check{\varphi}(\cdot)$ is the activation function. $\check{\boldsymbol{f}}(\check{\boldsymbol{v}})$ is a real-valued continuously differentiable vector function with $\check{\boldsymbol{f}}(\boldsymbol{0}_{\check{n}}) = \boldsymbol{0}_{\check{n}}$ [21].

A. Belief propagation based on HORNN

Originally proposed by Gallager [37], belief propagation is a suboptimum graph-based decoding algorithm for LDPC codes. The corresponding graph is bipartite (n parity nodes and $n - k$ check nodes) and known as Tanner graph [43]. This is shown in **Figure 16** for the Hamming code with the parity check matrix $\boldsymbol{H}_{\text{Hamming}}$ Eq. (17) where $n = 7$, $k = 4$. The belief propagation algorithm iteratively exchanges "messages" between parity and check nodes.

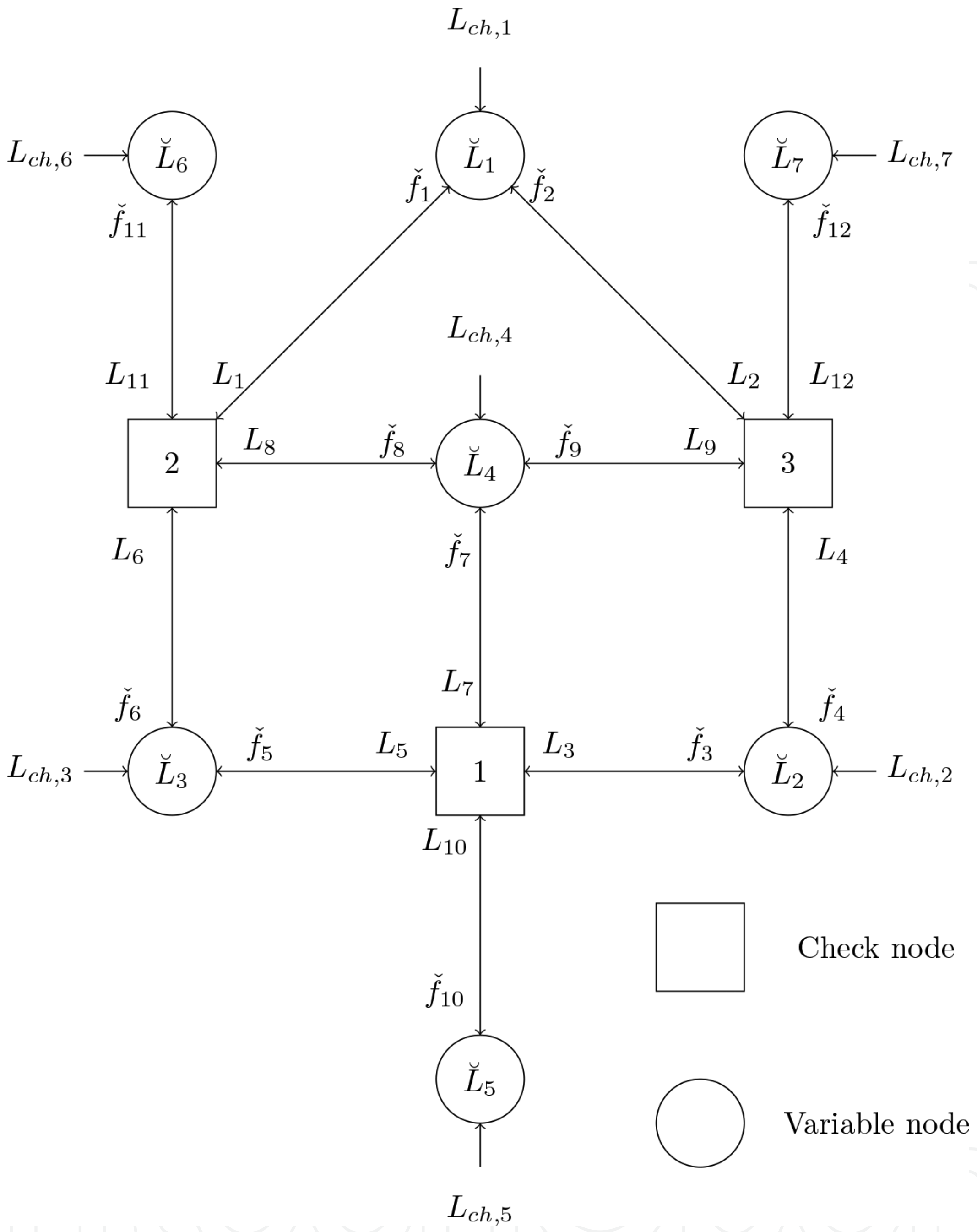


Figure 16. Tanner graph of the systematic Hamming code $n = 7$ and $k = 4$.

$$\boldsymbol{H}_{\text{Hamming}} = \left[\begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right] \tag{17}$$

For every binary linear block code characterized by the binary parity check matrix $\boldsymbol{H}$ of size $(n - k) \times n$ for $n > k$, three *binary* matrices $\boldsymbol{P}_{n_h \times n_h}$, $\boldsymbol{S}_{n_h \times n_h}$ and $\boldsymbol{B}_{n_h \times n}$ can be uniquely defined [44, 45] such that Eq. (16) and **Figure 15** perform continuous-time belief propagation if the following relations are fulfilled:

$$\check{u} = \boldsymbol{L}, \tag{18a}$$

$$\check{e} = \boldsymbol{B} \cdot \boldsymbol{L}_{ch}, \tag{18b}$$

$$\check{\varphi}(\cdot) = \tanh\left(\frac{\cdot}{2}\right), \tag{18c}$$

$$\check{v} = \check{\varphi}(\boldsymbol{L}), \tag{18d}$$

$$\check{W} = \boldsymbol{P}, \tag{18e}$$

$$\check{W}_0 = \boldsymbol{I}_{n_h \times n_h}, \tag{18f}$$

$$\check{f}_j = 2 \cdot \text{atanh}\left\{ \prod_{j' \in \text{pos}[\boldsymbol{S}(j,:)=1]} \check{v}_{j'} \right\} \quad \text{for } j, j' \in \{1,\ 2,\ \ldots, n_h\}, \tag{18g}$$

$$\check{n} = n_h. \tag{18h}$$

In Eq. (18)[8],

- k is the length of the information word ($\boldsymbol{q}$ in **Figures 1** and **14**).
- n is the length of the codeword ($\boldsymbol{q}_c$ in **Figures 1** and **14**).
- $n_h = \mathbf{1}^T_{\left(1 \times (n-k)\right)} \cdot \boldsymbol{H} \cdot \mathbf{1}_{(n \times 1)} \in [k+1,\ n \cdot (n-k)]$ is the number of nonzero elements in $\boldsymbol{H}$.
- $\boldsymbol{L}_{ch,(n \times 1)}$ is the vector of intrinsic log-likelihood ratio (LLR), which depends on the transition probability of the channel. For $\boldsymbol{q}_{c,j}$ (the j-th element of $\boldsymbol{q}_c$ for $j \in \{1, 2, \cdots, n\}$) it is given as

$$\boldsymbol{L}_{ch,j} = \ln \frac{p(\dot{x}_j = \tilde{x}_j | \boldsymbol{q}_{c,j} = 0)}{p(\dot{x}_j = \tilde{x}_j | \boldsymbol{q}_{c,j} = 1).} \tag{19}$$

  In the last relation, $\dot{x}_j$ is the variable of the conditioned probability density function $p(\dot{x}_j | \boldsymbol{q}_{c,j})$. $\ln(\cdot)$ is the natural logarithm. For an AWGN channel, $\mathcal{N}(0, \sigma_n^2) : \boldsymbol{L}_{ch,j} = \frac{\tilde{x}_j}{2 \cdot \sigma_n^2}$.

- $\boldsymbol{L}_{(n_h \times 1)}$ is the "message" sent from the variable nodes to the check nodes.
- $\check{\boldsymbol{f}}_{(n_h \times 1)}$ is the "message" sent from check nodes to variable nodes.
- $\boldsymbol{I}_{(n_h \times n_h)}$ is an identity matrix of size $n_h \times n_h$.
- $\text{pos}[\boldsymbol{S}(j,\ :) = 1]$ delivers the positions of the nonzero elements in the j-th row of the matrix $\boldsymbol{S}$.

[8] $\boldsymbol{L}$, $\check{\boldsymbol{L}}$ and $\boldsymbol{L}_{ch}$ are vectors.

The dynamical behavior of belief propagation can be described based on Eqs. (16), (18) and **Figures 14**, **15**, and **16** [45]

$$\begin{aligned} \boldsymbol{\Upsilon}_d \cdot \frac{\mathrm{d}\boldsymbol{L}(t)}{\mathrm{d}t} &= -\boldsymbol{L}(t) + \boldsymbol{P} \cdot \check{\boldsymbol{f}}\left(\check{\boldsymbol{v}}(t)\right) + \boldsymbol{B} \cdot \boldsymbol{L}_{ch}, \\ \check{\boldsymbol{v}}(t) &= \mathbf{tanh}\left(\frac{\boldsymbol{L}(t)}{2}\right), \\ \check{\boldsymbol{L}}(t) &= \boldsymbol{B}^T \cdot \check{\boldsymbol{f}}\left(\check{\boldsymbol{v}}(t)\right) + \boldsymbol{L}_{ch}. \end{aligned} \tag{20}$$

$\check{\boldsymbol{L}}(t)$ is the soft-output of the decoding algorithm, cf. **Figures 5** and **14**. The discrete-time description is given as [44]

$$\begin{aligned} \boldsymbol{L}[l+1] &= \boldsymbol{P} \cdot \check{\boldsymbol{f}}(\check{\boldsymbol{v}}[l]) + \boldsymbol{B} \cdot \boldsymbol{L}_{ch}, \\ \check{\boldsymbol{v}}[l] &= \mathbf{tanh}\left(\frac{\boldsymbol{L}[l]}{2}\right), \\ \check{\boldsymbol{L}}[l] &= \boldsymbol{B}^T \cdot \check{\boldsymbol{f}}(\check{\boldsymbol{v}}[l]) + \boldsymbol{L}_{ch}. \end{aligned} \tag{21}$$

B. Dynamical behavior of belief propagation

In a series of papers, Hemati *et. al.* [12, 17, 46–49] also modeled the dynamics of analog belief propagation as a set of first-order nonlinear differential equations Eq. (20). This was motivated from a circuit design aspect, where $\boldsymbol{\Upsilon}_d$ (the same is valid for $\boldsymbol{\Upsilon}_e$) can be seen as a bandwidth limitation of the analog circuit, realized taking advantage of the low-pass filter behavior of transmission lines **Figure 17**. We have shown in [45] that the model in **Figure 17** also has important dynamical properties when compared with the discrete-time belief propagation Eq. (21) [44]. Particularly, the equilibrium points of the continuous-time belief propagation of Eq. (20) coincide with the fixed points of the discrete-time belief propagation of Eq. (21). This has been proved in [45]. In both cases

$$\boldsymbol{L}_{ep} = \boldsymbol{P} \cdot \check{\boldsymbol{f}}\left(\mathbf{tanh}\left(\frac{\boldsymbol{L}_{ep}}{2}\right)\right) + \boldsymbol{B} \cdot \boldsymbol{L}_{ch}. \tag{22}$$

The absolute stability of belief propagation Eqs. (20), (21) was proven for repetition codes (one of the simplest binary linear block codes) in [44, 45]. In this case

$$\check{\boldsymbol{L}}_{ep} = \mathbf{1}_{(n \times n)} \cdot \boldsymbol{L}_{ch}. \tag{23}$$

Far away from repetition codes, it has been noticed that iterative decoding algorithms (belief propagation is one of them) exhibit depending on the SNR a wide range of phenomena associated with nonlinear dynamical systems such as existence of multiple fixed points, oscillatory behavior, bifurcation, chaos and transit chaos [50]. Equilibrium points are reached at "relatively" high SNR. The analysis in reference [50] is limited to the discrete-time case.

Remark 8. The HORNN in **Figure 15**, Eqs. (18), (20) for belief propagation acts as a computational model.

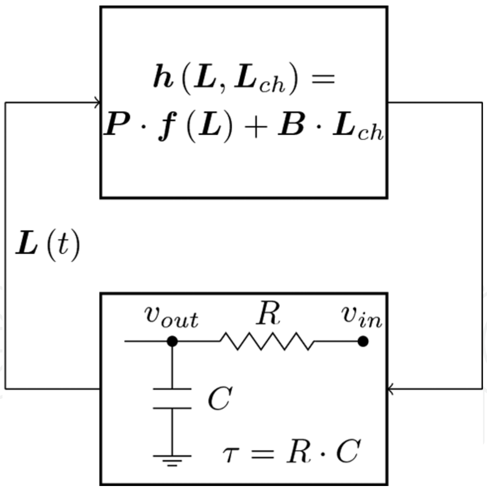

$\boldsymbol{h}(\boldsymbol{L}, \boldsymbol{L}_{ch}) =$
$\boldsymbol{P} \cdot \boldsymbol{f}(\boldsymbol{L}) + \boldsymbol{B} \cdot \boldsymbol{L}_{ch}$
$\boldsymbol{L}(t)$
v_{out}
R
v_{in}
C
$\tau = R \cdot C$

Figure 17. A simple model for analog decoding as presented in [46].

C. Analog hardware implementation aspects: decoding

Many analog hardware implementation aspects have been already mentioned in Section 4-B. We mention here only additional aspects exclusively related to the analog belief propagation based on HORNN.

Structure: In practice, different coding schemes (different parity check matrices $\boldsymbol{H}$) with various (k, n) constellations are applied to modify the code rate $r_c = k/n$ depending on the channel state. The HORNN in **Figure 15** is capable to act as a continuous-time belief propagation (decoder) as long as the number of neurons $\check{n}$ in **Figure 15** equals (or is larger than) the maximum number of nonzero elements over all parity check matrices and all (k,n) constellations, i.e., $\check{n} \geq \max_{\boldsymbol{H}} n_h$.

Adaptivity: No training is needed. $\check{\boldsymbol{W}}_0$ and $\check{\boldsymbol{W}}$ are directly related to the parity check matrix $\boldsymbol{H}$. In contrast to the analog RNN vector equalizer, the weight coefficients are binary, i.e., the weight matrices $\check{\boldsymbol{W}}_0$ and $\check{\boldsymbol{W}}$ define a feedback to be either existent or not. In such a case for **Figure 15**, $\check{R}_{jj'}$, $\check{R}_{j0} \in \{\check{R}_j, \infty\}$. Moreover, there is no need for high-resolution DAC for the weight coefficients.

Interneuron connectivity: No full connection is needed since the matrix $\boldsymbol{P}$ for LDPC codes is sparse. The number of connections per neuron must equal the maximum number of nonzero elements in $\boldsymbol{P}$ row-wise over all considered coding schemes and equals $\max_{\boldsymbol{H}} \boldsymbol{P} \cdot \mathbf{1}_{(n_h \times 1)}$. If this is fulfilled and if interneuron connectivity control is available, the structure in **Figure 15** becomes valid for all considered coding scheme.

Vector function connectivity: For different coding schemes, the number of the arguments $\check{v}'_j$ to evaluate the function $\check{f}_j$ changes, cf. Eq. (18g). The maximum number of the arguments depends on the number of the nonzero elements in $\boldsymbol{S}$ row-wise and equals $\max_{\boldsymbol{H}} \boldsymbol{S} \cdot \mathbf{1}_{(n_h \times 1)}$. Thus, implementing the function $\check{f}_j$ according to this maximum number enables evaluating the function $\check{f}_j$ for all considered coding schemes.

Remark 9. For a specific coding scheme, the interneuron connectivity can be made fixed. The resulted HORNN structure in this case is valid also for all codeword lengths resulted after performing a puncturing of the original code.

Remark 10. Both, the interneuron connectivity and the weight adaptation play a significant role, in the equalization as well as in the decoding. It can safely be said that they represent the major challenge of the circuit, since the analog circuit must be capable to perform equalization and decoding for a given number of possible combinations of block size, symbol alphabet, coding scheme, etc. Particularly for the decoding, the advantage of having a non-full connectivity is counterbalanced by a double (and very complex) (de)multiplexing of the signals (once for the vector function $\check{\boldsymbol{f}}$ and once for the interneuron connectivity).

7. Joint equalization and decoding

Turbo equalization is a joint iterative equalization and decoding scheme. In this case, a symbol-by-symbol maximum aposteriori probability (s/s MAP) equalizer exchanges in an iterative way reliability values $\boldsymbol{L}$ with a (s/s MAP) decoder [51, 52]. This concept is inspired from the decoding concept of turbo codes, where two (s/s MAP) decoders exchange iteratively reliability values [53]. Despite its good performance, the main drawback of the turbo equalizer is the very high complexity of the s/s MAP-equalizer for multipath channels with long impulse response (compared with symbol duration) and/or symbol alphabets with large cardinality. Therefore, a suboptimum equalization (and a suboptimum decoding) usually replace the s/s-MAP ones (**Figure 18**).

One discrete-time joint equalization and decoding approach has been introduced in [52] and is shown in **Figure 19**. $\tilde{x}$, $\boldsymbol{R}_d$ and $\boldsymbol{R}$ are as in Eq. (2) and z^{-1} is a delay unit. We notice that there are two different (iteration) loops in **Figure 19**: the equalization loop (the blue one) on symbol basis (in the sense of $\boldsymbol{\psi}$) and the decoding loop (the dashed one) on bit basis. $a = \{1, 2, 3, \cdots\}$, $\rho \in \mathbb{N}$, i.e., after each ρ equalization loops, one decoding loop is performed. The conversion between symbol basis and bit basis ($\boldsymbol{u}$ to $\boldsymbol{L}_{ch}$) is performed by $\boldsymbol{\theta}_{S/L}(\cdot)$, the way around ($\check{\boldsymbol{L}}$ to $\check{\boldsymbol{x}}$) by $\boldsymbol{\theta}_{L/S}(\cdot)$. The expressions for $\boldsymbol{\theta}_{L/S}(\cdot)$ and $\boldsymbol{\theta}_{L/S}(\cdot)$ can be found in [52]. However, for BPSK, they are given as

$$\begin{aligned} \theta_{S/L}(u) &= \frac{2}{\sigma^2} \cdot u, \\ \theta_{L/S}(\check{L}) &= \tanh\left(\frac{\check{L}}{2}\right). \end{aligned} \tag{24}$$

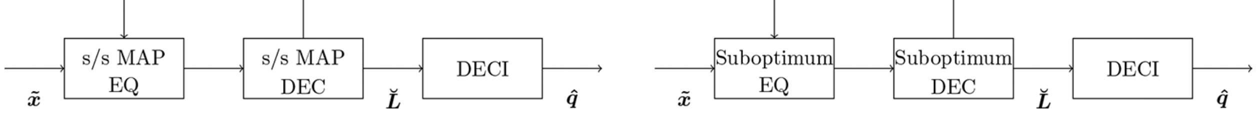


Figure 18. Two examples for joint equalization and decoding. Notice the feedback from the decoder to the equalizer, i.e., turbo principle.

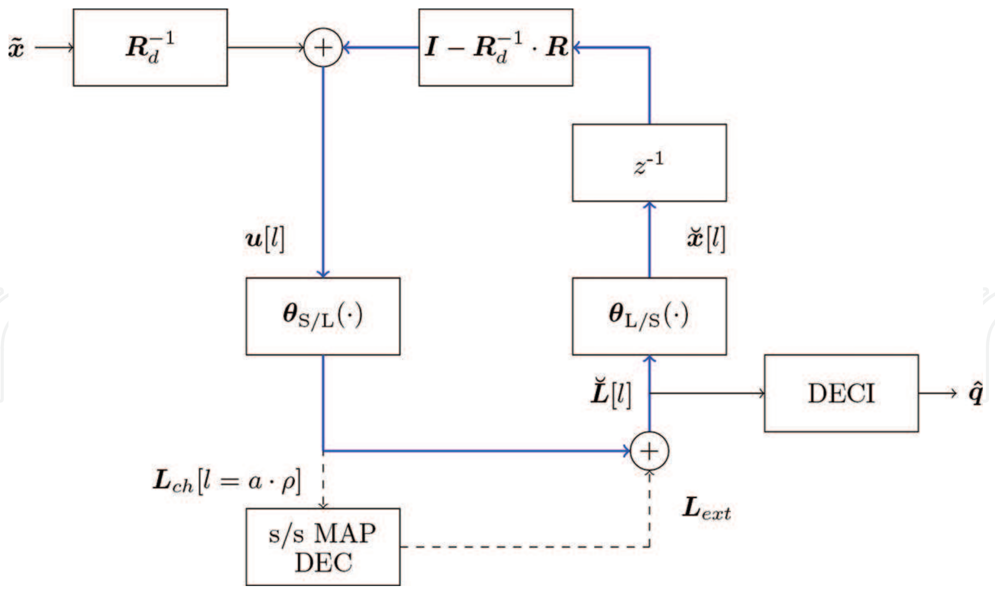


Figure 19. Joint equalization and decoding as described in [52]. L_{ext} represents the "knowledge" obtained by exploiting the redundancy of the code.

σ^2 is given in Eq. (11). If we consider *only* the equalization loop in **Figure 19**, we notice that it describes exactly the dynamical behavior of discrete-time recurrent neural networks [19, 25, 33, 54–56]

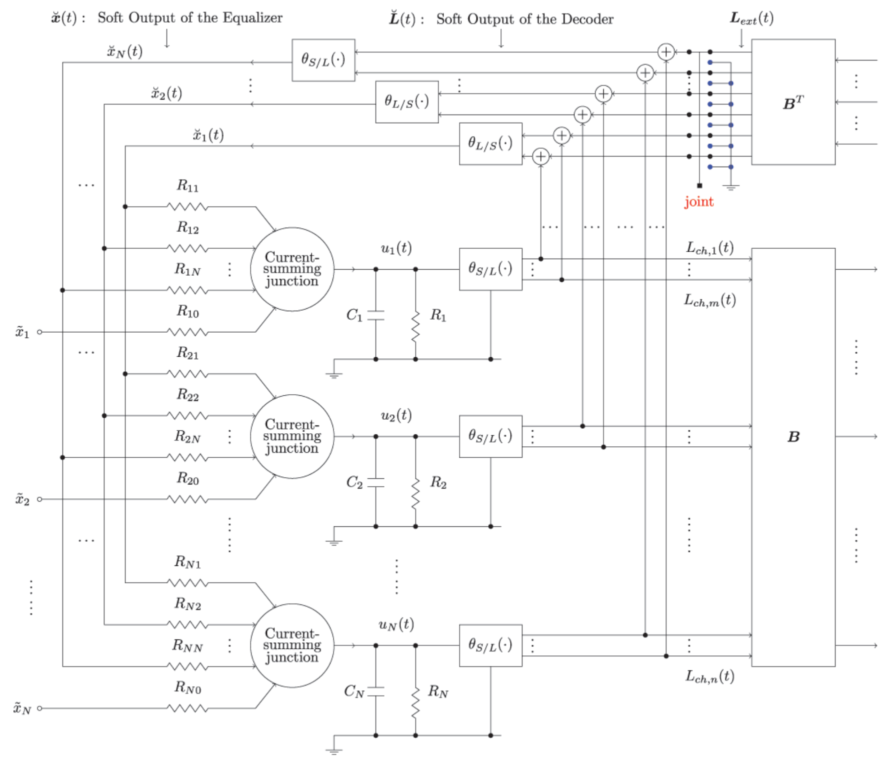
$$\begin{aligned} u[l+1] &= [I - R_d^{-1} \cdot R] \cdot \breve{x}[l] + R_d^{-1} \cdot \tilde{x}, \\ \breve{x}[l] &= \theta_{L/S}(\theta_{S/L}(u[l])). \end{aligned} \tag{25}$$

Remark 11. If $\theta_{L/S}\left(\theta_{S/L}(u)\right) = \theta^{(opt)}(u)$, Eqs. (8), (25) share the same equilibrium/fixed points. For BPSK, it can be easily shown based on Eqs. (12), (24) that this is fulfilled.

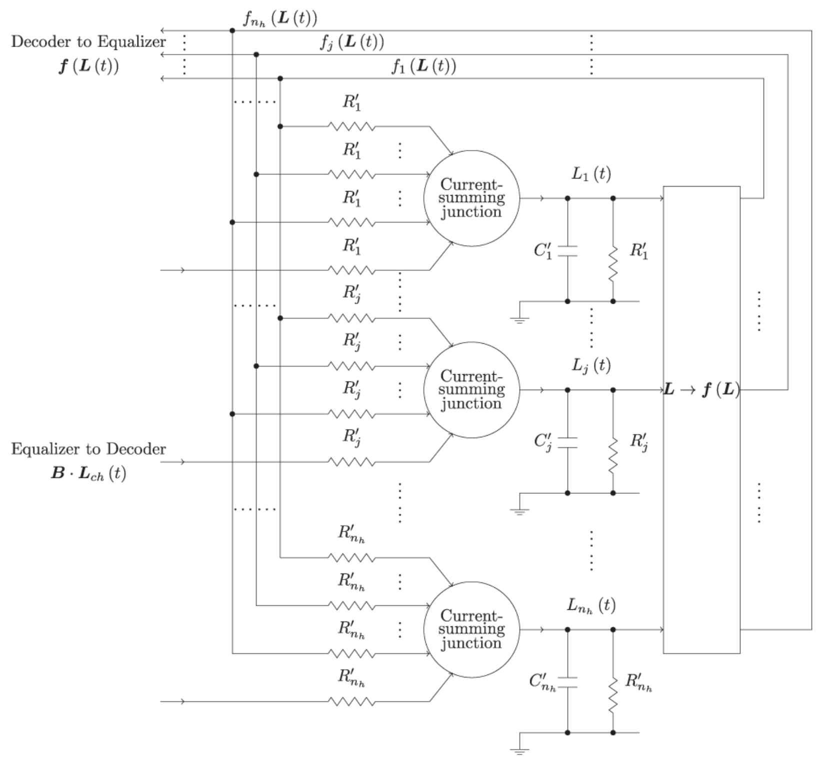
8. Continuous-time joint equalization and decoding

Motivated by the expected improvement of the energy efficiency by analog implementation compared with the conventional digital one, we map in this section the joint equalization and decoding structure given in **Figure 19** to a continuous-time framework. s/s MAP DEC in **Figure 19** is replaced by a suboptimum decoding algorithm: the *belief propagation*. Moreover, equalization and decoding loops in **Figure 19** are replaced by RNN and HORNN as discussed previously in Sections 4-A and 6-A, respectively. The introduced structure serves as a computational model for an analog hardware implementation and does not need any training.

Figure 20 shows a novel continuous-time joint equalization and decoding based on recurrent neural network structures. The dynamical behavior of the whole system is described by the following differential equations:



(a) The equalization part.



(b) The decoding part.

Figure 20. Continuous-time *joint* equalization and decoding based on a recurrent neural network structure for real-valued symbol alphabet (for complex-valued ones, cf. Remark 3). $\check{L}(t)$, $\check{x}(t)$ are the soft output of the decoder and the equalizer, respectively. (a) The equalization part. (b) The decoding part.

$$\boldsymbol{\Upsilon}_e \cdot \frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = -\boldsymbol{u}(t) + \boldsymbol{W} \cdot \check{\boldsymbol{x}}(t) + \boldsymbol{W}_0 \cdot \tilde{\boldsymbol{x}}, \tag{26a}$$

$$\boldsymbol{L}_{ch}(t) = \boldsymbol{\theta}_{\mathrm{S/L}}\Big(\boldsymbol{u}(t)\Big), \tag{26b}$$

$$\boldsymbol{\Upsilon}_d \cdot \frac{\mathrm{d}\boldsymbol{L}(t)}{\mathrm{d}t} = -\boldsymbol{L}(t) + \boldsymbol{P} \cdot \boldsymbol{f}\Big(\boldsymbol{L}(t)\Big) + \boldsymbol{B} \cdot \boldsymbol{L}_{ch}(t), \tag{26c}$$

$$\check{\boldsymbol{L}}(t) = \underbrace{\boldsymbol{B}^T \cdot \boldsymbol{f}\Big(\boldsymbol{L}(t)\Big)}_{\boldsymbol{L}_{ext}(t)} + \boldsymbol{L}_{ch}(t), \tag{26d}$$

$$\check{\boldsymbol{x}}(t) = \boldsymbol{\theta}_{\mathrm{L/S}}\Big(\check{\boldsymbol{L}}(t)\Big), \tag{26e}$$

$$f_j = 2 \cdot \mathrm{atanh}\left\{ \prod_{j' \in \mathrm{pos}[\boldsymbol{S}(j,:)=1]} \tanh\left(\frac{L_{j'}}{2}\right) \right\}. \tag{26f}$$

- Eq. (26a) and **Figure 20(a)** describe the continuous-time vector equalization, cf. Eqs. (1), (5), (7), (8).
- Eq. (26c) and **Figure 20(b)** describe the continuous-time belief propagation, cf. Eqs. (16), (18), (20).

Comparing **Figure 20** with **Figures 7** and **15**, we notice that

- The function $\boldsymbol{\varphi}(\cdot)$ in **Figure 7** (the optimum activation function $\boldsymbol{\theta}^{(opt)}(\cdot)$) has been split into two functions $\boldsymbol{\theta}_{\mathrm{S/L}}(\cdot)$ and $\boldsymbol{\theta}_{\mathrm{L/S}}(\cdot)$ in **Figure 20(a)**. For BPSK symbol alphabet and based on Eqs. (12), (24), it can be easily shown that $\boldsymbol{\theta}^{(opt)}(\boldsymbol{u}) = \boldsymbol{\theta}_{\mathrm{L/S}}\Big(\boldsymbol{\theta}_{\mathrm{S/L}}(\boldsymbol{u})\Big)$, cf. Remark 11.
- The functions $\check{\boldsymbol{\varphi}}(\cdot)$ and $\check{\boldsymbol{f}}(\cdot)$ in **Figure 15**, Eq. (18) have been merged to one function $\boldsymbol{f}(\cdot)$ in **Figure 20(b)**, Eq. (26f) $\boldsymbol{f}(\boldsymbol{L}) = \check{\boldsymbol{f}}\Big(\check{\boldsymbol{\varphi}}(\boldsymbol{L})\Big)$. $\check{\boldsymbol{L}}(t)$ and $\check{\boldsymbol{x}}(t)$ are the soft output of the decoder and the equalizer, respectively.

A. Special cases

The novel structure in **Figure 20** is general and stays valid for the following cases:

- **Separate equalization and decoding**: In this case, **Figure 20(a)** is modified such that no feedback from decoder to equalizer is applied. This is shown in **Figure 21(a)**. *Only* at the end of the separate equalization and decoding process, the output is given as $\check{\boldsymbol{L}} = \boldsymbol{L}_{ext} + \boldsymbol{L}_{ch}$. We distinguish between two cases
  1. Equalization and decoding take place separately *at the same time*.
  2. Successive equalization and decoding: *only* after the end of the equalization process, $\boldsymbol{L}_{ch}$ are forwarded to the decoder and the decoder starts the evolution. We focus on this case.

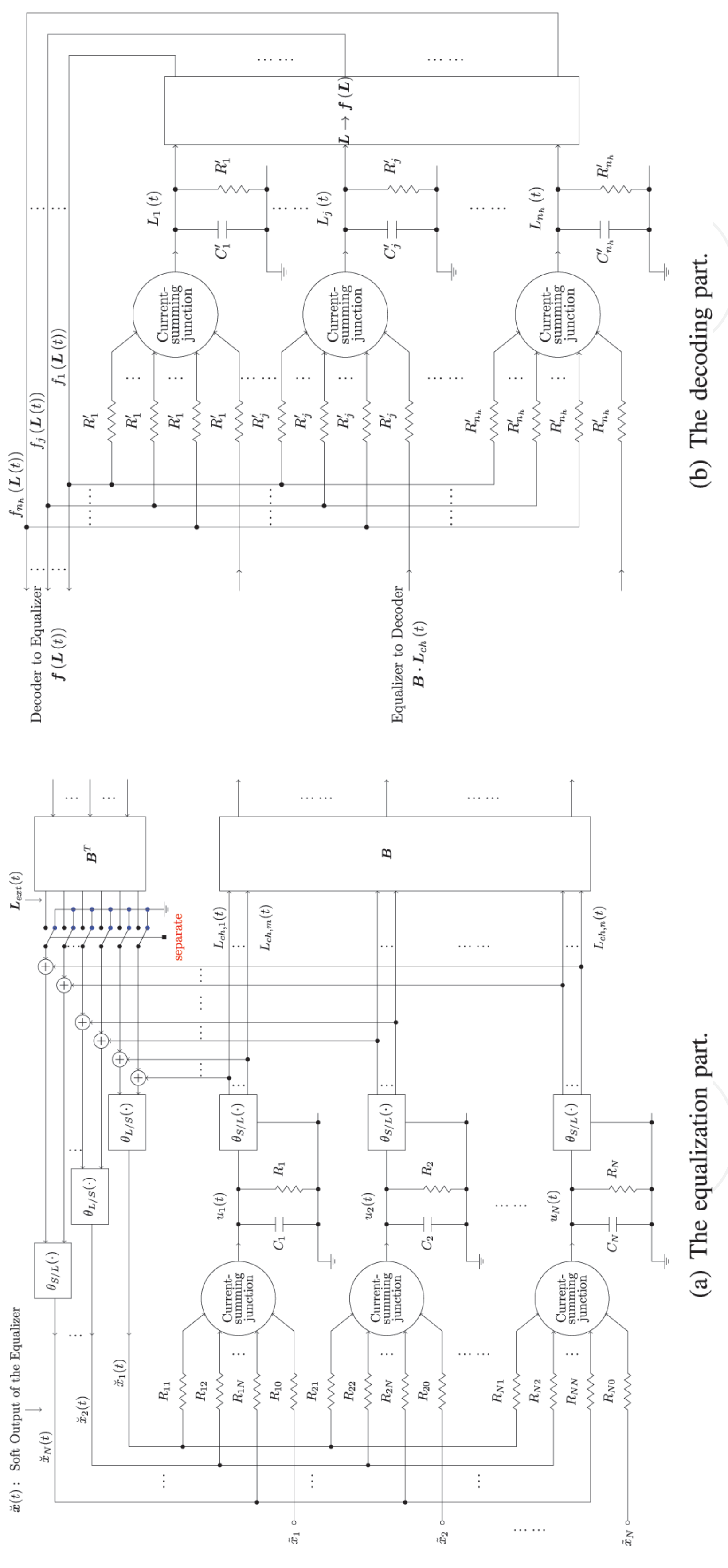

$\check{\boldsymbol{x}}(t)$: Soft Output of the Equalizer
$\check{x}_N(t)$
$\check{x}_2(t)$
$\check{x}_1(t)$
$\theta_{S/L}(\cdot)$
$\theta_{L/S}(\cdot)$
$\boldsymbol{L}_{ext}(t)$
$\boldsymbol{B}^T$
separate
$\boldsymbol{B}$
$L_{ch,1}(t)$
$L_{ch,m}(t)$
$L_{ch,n}(t)$
Current-summing junction
R_{11} R_{12} R_{1N} R_{10}
R_{21} R_{22} R_{2N} R_{20}
R_{N1} R_{N2} R_{NN} R_{N0}
$u_1(t)$ $u_2(t)$ $u_N(t)$
R_1 R_2 R_N
C_1 C_2 C_N
$\tilde{x}_1$ $\tilde{x}_2$ $\tilde{x}_N$
(a) The equalization part.
Decoder to Equalizer
$\boldsymbol{f}(\boldsymbol{L}(t))$
$f_{n_h}(\boldsymbol{L}(t))$
$f_j(\boldsymbol{L}(t))$
$f_1(\boldsymbol{L}(t))$
Equalizer to Decoder
$\boldsymbol{B}\cdot\boldsymbol{L}_{ch}(t)$
R'_1 R'_j R'_{n_h}
C'_1 C'_j C'_{n_h}
$L_1(t)$ $L_j(t)$ $L_{n_h}(t)$
$\boldsymbol{L}\to\boldsymbol{f}(\boldsymbol{L})$
(b) The decoding part.

Figure 21. Continuous-time *separate* equalization and decoding based on a recurrent neural network structure for real-valued symbol alphabet (for complex-valued ones, cf. Remark 3). $\check{\boldsymbol{x}}(t)$ is the soft output of the equalizer.

- **Coded transmission over an AWGN channel**: In this case, $\boldsymbol{R} = \boldsymbol{I}$ and hence based on Eq. (7) $\boldsymbol{W} = \boldsymbol{0}$, $\boldsymbol{W}_0 = \boldsymbol{I}$. Under these conditions, Eq. (26a) becomes linear and can easily be solved $\boldsymbol{u}(t) \rightarrow \tilde{x}$ and $\boldsymbol{L}_{ch}(t)$ in Eq. (26b) becomes time independent $\boldsymbol{L}_{ch}$. In this case, Eq. (26c) reduces to Eqs. (16), (18).
- **Uncoded transmission over an "interference-causing" channel**: In this case, $\boldsymbol{P} = \boldsymbol{0}$, $\boldsymbol{B} = \boldsymbol{0}$ and Eq. (26c) becomes $\boldsymbol{L} = \boldsymbol{0}_{n_h}$. Under these conditions, Eq. (26a) reduces to Eqs. (5), (7), (8) (notice, however, Remark 11).

B. Throughput, asynchronicity and scheduling

The diagonal elements in $\boldsymbol{\Upsilon}_d$ define the duration of the transient response the HORNN needs in order to converge eventually (in case of convergence). The larger they are, the longer is the transient response and consequently the less is the decoding throughput. The same is valid for $\boldsymbol{\Upsilon}_e$. The diagonal elements of $\boldsymbol{\Upsilon}_e$ based on our analog RNN vector equalizer are in the range of a few tens of picoseconds.

Unequal diagonal elements in $\boldsymbol{\Upsilon}_e$ (and $\boldsymbol{\Upsilon}_d$) represent some kind of continuous-time asynchronicity [46]. Asynchronicity in discrete-time RNNs is desirable since it provides the ability to avoid limit cycles, which can probably occur in synchronous discrete-time RNNs [54, 57].

Assuming $\boldsymbol{\Upsilon}_d = \tau_d \cdot \boldsymbol{I}$ and $\boldsymbol{\Upsilon}_e = \tau_e \cdot \boldsymbol{I}$, we notice that the ratio τ_e/τ_d is comparable to the scheduling problem in the discrete-time joint equalization and decoding case. More precisely, how many iterations ρ within the equalizer should be performed before a decoding process takes place, cf. **Figure 19**. This is optimized usually by simulations and is case dependent.

From a dynamical point of view, the case $\tau_e/\tau_d \ll 1$ (or $\tau_d/\tau_e \ll 1$) could be seen as a singular perturbation (in time). In this case, one part of **Figure 20** can be seen as "frozen" compared with the other part.

Remark 12. We notice that the parameters of the transmission model (block transmit matrix, symbol alphabet, block size, channel coding scheme) are utilized to define the parameters of the continuous-time recurrent neural network structure in **Figure 20** such that no training is needed. This represents in practice a big advantage especially for analog hardware. However, to enable different coding schemes and symbol alphabets, either a full connectivity or a vector and interneuron connectivity controls are needed. Both structures are challenging from a hardware implementation point of view.

Remark 13. For the ease of depiction, **Figures 20** and **21** assume that one transmitted block contains exactly one codeword. This is not necessarily the case in practice. As an example, if one transmitted block contains two codewords, one RNN and two parallel HORNNs will be needed. On the other hand, if one codeword lasts over two transmitted blocks, two parallel RNNs and one HORNN is needed.

9. Simulation results

We simulate the dynamical system as given in Eq. (26) and **Figure 20** based on the first Euler method [58]. We assume:

- BPSK modulation scheme (symbol alphabet $\boldsymbol{\psi} = \{-1, +1\}$).
- Each transmitted block contains one codeword, cf. Remark 13.
- $\boldsymbol{\Upsilon}_d = \boldsymbol{\Upsilon}_e = \tau \cdot \boldsymbol{I}$.
- Channel coding scheme: An LDPC code with $k = 204$, code rate 0.5 ($n = 408$) and column weight 3 taken from [59].
- Multipath channels [60]:

  –Proakis-a abbreviated in the following by its channel impulse response h_a leading to a small interference.

  –Proakis-b abbreviated in the following by its channel impulse response h_b leading to a moderate interference.

The impulse response of h_a and h_b are

$$\begin{aligned} h_a &= [\,0.04 \quad -0.05 \quad 0.07 \quad -0.21 \quad -0.5 \quad 0.72 \quad 0.36 \quad 0 \quad 0.21 \quad 0.03 \quad 0.07\,], \\ h_b &= [\quad 0.407 \quad 0.815 \quad 0.407 \quad]. \end{aligned}$$

The block transmission matrix $\boldsymbol{R}$ is a banded Toeplitz matrix of the autocorrelation function of the channel impulse response [61]. The following cases are considered:

- Uncoded transmission over AWGN channel. The bit error rate can be obtained analytically and is given as $\frac{1}{2} \cdot \operatorname{erfc}\left\{\frac{E_b}{N_0}\right\}$ [62]. $\operatorname{erfc}(\cdot)$ is the complementary error function and E_b is the energy per bit.
- Coded transmission over AWGN channel and continuous-time decoding at the receiver (HORNN-belief propagation).
- Uncoded transmission over (the abovementioned) multipath channels and continuous-time equalization at the receiver (RNN-equalization).
- Coded transmission over (the abovementioned) multipath channels. We distinguish between joint equalization and decoding (**Figure 20**) and separate equalization and decoding (**Figure 21**). In the latter case, equalization is performed firstly, and consequently the decoding.

The evolution time for the whole system in all cases is $20 \cdot \tau$, i.e., all simulated scenarios deliver the same throughput. For separate equalization and decoding, the evolution time of the equalization equals the evolution time of the decoding and equals 10 τ. The simulation results are shown in **Figure 22**. We notice the following:

- Joint equalization and decoding outperforms the separate one, which is a fact we know from the discrete-time case. Our proposed model in **Figure 20** is capable of "transforming" this advantage to the continuous-time case.
- For the channel h_a, the BER (for continuous-time joint equalization and decoding) is close to the coded BER curve.

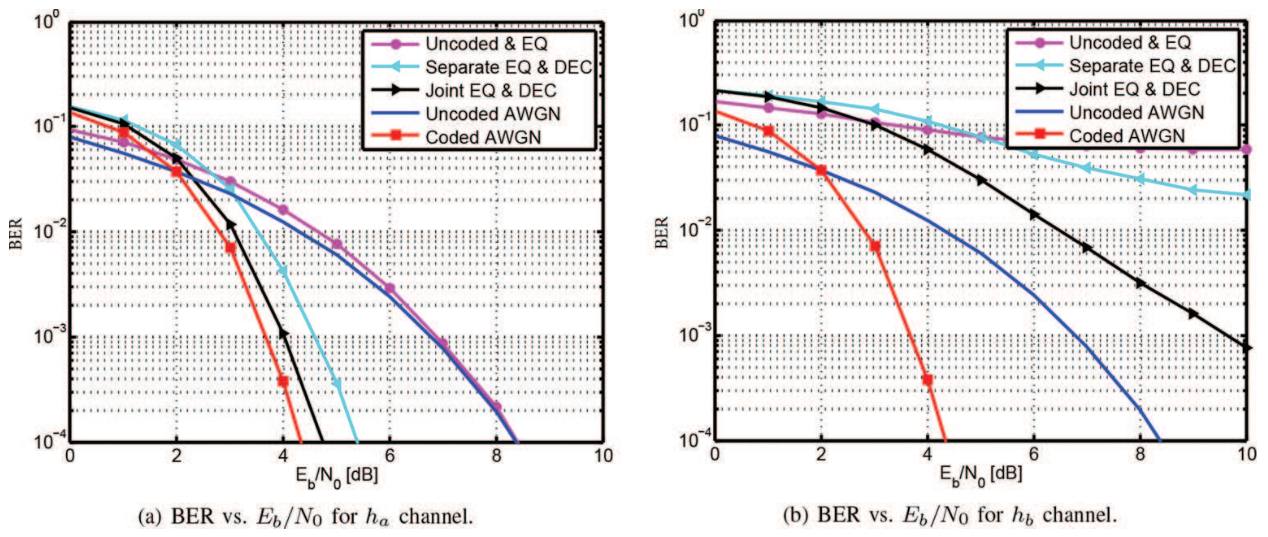


Figure 22. BER vs. E_b/N_0 for evolution time equals $20 \cdot \tau$. Continuous-time (joint) equalization and decoding. BPSK symbol alphabet.

- For the channel h_b, there exists a gap between the obtained results and the coded AWGN curve. This was expected, since h_b represents a more severe multipath channel compared with h_a.
- If only equalization performance is considered, we compare between "Uncoded & EQ" curves and "Uncoded BER" curves. In **Figure 22(a)**, the vector equalizer based on continuous-time recurrent neural networks is capable to remove already all interferences caused by the multipath channel h_a, whereas in **Figure 22(b)**, the "Uncoded & EQ" curve approaches an error floor.

Remark 14. Interleaving and antigray mapping often encountered in the context of iterative equalization and decoding can be easily integrated in the proposed model in **Figure 20**. Antigray mapping will influence the functions $\boldsymbol{\theta}_{\mathrm{S/L}}(\cdot)$ and $\boldsymbol{\theta}_{\mathrm{L/S}}(\cdot)$, whereas interleaving affects the matrix $\boldsymbol{B}$.

10. Conclusion

Joint equalization and decoding is a detection technique which possesses the potential for improving the bit error rates of the transmission at the cost of additional computational complexity at the receiver. Joint equalization and decoding is being considered only for the discrete-time case. However, for high data rates, the energy consumption of a digital implementation becomes a limiting factor and shortens the lifetime of the battery. Improving the energy efficiency revives the analog implementation option for joint equalization and decoding algorithms, particularly taking advantage of the nonlinearity of the corresponding algorithms.

Continuous-time recurrent neural networks serve as promising computational models for analog hardware implementation and stand out due to their Lyapunov stability (the proved existence of attracting equilibrium points under specific conditions) and special suitability for analog VLSI. They have often been applied for solving optimization problems even without the need for a training. The drop of the training is particularly favorable for analog hardware implementation.

In this chapter, we introduced a novel continuous-time recurrent neural network structure, which is capable to perform continuous-time joint equalization and decoding. This structure is based on continuous-time recurrent neural networks for equalization and continuous-time high-order recurrent neural networks for belief propagation, a well-known decoding algorithm for low-density parity-check codes. In both cases, the behavior of the underlying dynamical system has been addressed, Lyapunov stability and simulated annealing are a few examples. The parameters of the transmission system (channel matrix, symbol alphabet, block size, channel coding scheme) are used to define the parameters of the proposed recurrent neural network such that *no training* is needed.

Simulation results showed that the superiority of joint equalization and decoding preserves, if this is done in analog according to our proposed model. Compared with the digital implementation, the analog one is expected to improve the energy efficiency and consume less chip area. We confirmed this for the analog hardware implementation of the equalization part. In this case, the analog vector equalization achieves an energy efficiency of a few picojoule per equalized bit, which is three to four orders of magnitude better than the digital counterparts. Additionally, analog hardware implementation aspects have been discussed. We showed as an example the importance of the interneuron connectivity, especially pointing out the challenges represented either by the hardware implementation of a massively distributed network, or by the routing of the signals using (de)multiplexers.

Author details

Mohamad Mostafa[1]*, Giuseppe Oliveri[2], Werner G. Teich[3] and Jürgen Lindner[3]

*Address all correspondence to: Mohamad.Mostafa@DLR.de

1 German Aerospace Center (DLR), Institute of Communications and Navigation, Wessling, Germany

2 Ulm University, Institute of Electron Devices and Circuits, Ulm, Germany

3 Ulm University, Institute of Communications Engineering, Ulm, Germany

References

[1] G. P. Fettweis, K.-C. Chen, and R. Tafazoli, “Green radio: Energy efficiency in wireless networks,” *Journal of Communications and Networks*, vol. 12, no. 2, pp. 99–102, April 2010.

[2] C. A. Chan, A. F. Gygax, E. Wong, C. A. Leckie, A. Nirmalathas, and D. C. Kilper, "Methodologies for assessing the use-phase power consumption and greenhouse gas emissions of telecommunications network services," *Environmental Science & Technology*, vol. 47, no. 1, pp. 485–492, January 2013.

[3] G. Oliveri, M. Mostafa, W. G. Teich, J. Lindner, and H. Schumacher, "Advanced low power, high speed nonlinear analog signal processing: An analog VLSI example," in *Wireless Innovation Forum, European Conference on Communications Technologies and Software Defined Radio*, Erlangen, Germany, 5–9 October 2015.

[4] H.-A. Löliger, "Analog decoding and beyond," in *IEEE Information Theory Workshop*, 2–7 September 2001, pp. 126–127.

[5] H.-A. Löliger, F. Tarköy, F. Lustenberger, and M. Helfenstein, "Decoding in analog VLSI," *IEEE Communications Magazine*, vol. 37, no. 4, pp. 99–101, April 1999.

[6] C. Mead, *Analog VLSI and neural systems*, ser. Addison-Wesley VLSI system series. Addison-Wesley, 1989.

[7] E. A. Vittoz, "Analog VLSI signal processing: Why, where and how?" *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 8, no. 1, pp. 27–44, February 1994.

[8] S. Draghici, "Neural networks in analog hardware - Design and implementation issues," *International Journal of Neural Systems*, vol. 10, no. 1, pp. 19–42, February 2000.

[9] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 1-3, pp. 239–255, December 2010.

[10] S. L. Goh and D. P. Mandic, "A complex-valued RTRL algorithm for recurrent neural networks," *Neural Computation*, vol. 16, no. 12, pp. 2699–2713, December 2004.

[11] J. Lindner, "MC-CDMA in the context of general multiuser/multisubchannel transmission methods," *European Transactions on Telecommunications*, vol. 10, no. 4, pp. 351–367, July-August 1999.

[12] S. Hemati and A. H. Banihashemi, "Dynamics and performance analysis of analog iterative decoding for low-density parity-check LDPC codes," *IEEE Transactions on Communications*, vol. 54, no. 1, pp. 61–70, January 2006.

[13] J. Hagenauer, E. Offer, C. Méasson, and M. Mörz, "Decoding and equalization with analog non-linear networks," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 659–680, November-December 1999.

[14] M. Dangl, *Iterative estimation and detection for single carrier block transmission*. Der Andere Verlag, Dissertation, Ulm University, Institute of Information Technology, 2007.

[15] A. Cichocki and R. Unbehauen, *Neural networks for optimization and signal processing*. John Wiley & Sons Ltd. & B. G. Teubner, Stuttgart, 1993, ch. 2.

[16] H. Tang, K. C. Tan, and Z. Yi, "Various computational models and applications," in *Neural networks: Computational models and applications*, ser. Studies in computational intelligence, J. Kacprzyk, Ed. Springer-Verlag Berlin Heidelberg, 2007, vol. 53, ch. 5, pp. 57–79.

[17] S. Hemati and A. Yongacoglu, "Dynamics of analog decoders for different message representation domains," *IEEE Transactions on Communications*, vol. 58, no. 3, pp. 721–723, May 2010.

[18] W. G. Teich, M. A. Ibrahim, F. Waeckerle, and R. F. H. Fischer, "Equalization for fiber-optic transmission systems: Low-complexity iterative implementations," in *17th ITG-Fachtagung Photonische Netze*, Leipzig, Germany, May 2016.

[19] A. Engelhart, *Vector detection techniques with moderate complexity.* VDI Verlag GmbH, Dissertation, Ulm University, Institute of Information Technology, 2003.

[20] M. Mostafa, W. G. Teich, and J. Lindner, "Approximation of activation functions for vector equalization based on recurrent neural networks," in *6th International Symposium on Turbo Codes and Iterative Information Processing*, Bremen, Germany, 18-22 August 2014, pp. 52–56.

[21] M. Mostafa, *Equalization and decoding: A continuous-time dynamical approach*. Der Andre-Verlag, Dissertation, Ulm University, Institute of Communications Engineering, 2014, ch. 1 & 3.

[22] K. Kuroe, N. Hashimoto, and T. Mori, "On energy function for complex-valued neural networks and its applications," in *9th International Conference on Neural Information Processing*, vol. 3, 18-22 November 2002, pp. 1079–1083.

[23] S. Haykin, *Neural networks: A comprehensive foundation*. USA: Macmillan college publishing company, Inc., 1994.

[24] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," in *Proceedings of the National Academy of Sciences of the USA*, vol. 81, no. 10, pp. 3088–3092, May 1984.

[25] M. Yoshida and T. Mori, "Global stability analysis for complex-valued recurrent neural networks and its application to convex optimization problems," in *Complex-valued neural networks: Utilizing high-dimensional parameters*, T. Nitta, Ed. Information science reference, 2009, ch. 5, pp. 104–114.

[26] J. M. Zurada, I. Cloete, and W. van der Poel, "Generalized Hopfield networks for associative memories with multi-valued stable states," *Neurocomputing*, vol. 13, pp. 135–149, 1996.

[27] D. Tank and J. J. Hopfield, "Simple 'neural' optimization network: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Transactions on Circuits and Systems*, vol. 33, no. 5, pp. 553–541, May 1986.

[28] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," *Biological Cybernatics*, vol. 52, no. 3, pp. 141–152, 1985.

[29] J. J. Hopfield, "Neurons networks and physical systems with emergent collective computational abilities," in *Proceedings of the National Academy of Sciences of the USA*, vol. 79, no. 8, pp. 2554–2558, April 1982.

[30] G. I. Kechriotis and E. S. Manolakos, "Hopfield neural network implementation for optimum CDMA multiuser detector," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 131–141, January 1996.

[31] T. Miyajima, T. Hasegawa, and M. Haneishi, "On the multiuser detection using a neural network in code-division multiple-access communications," *IEICE Transactions on Communications*, vol. E76-B, pp. 961–968, 1993.

[32] M. Mostafa, W. G. Teich, and J. Lindner, "Vector equalization based on continuous-time recurrent neural networks," in *6th IEEE International Conference on Signal Processing and Communication Systems*, Gold Coast, Australia, 12-14 December 2012, pp. 1–7.

[33] W. G. Teich and M. Seidl, "Code division multiple access communications: Multiuser detection based on a recurrent neural network structure," in *4th IEEE International Symposium on Spread Spectrum Techniques and Applications*, vol. 3, Mainz, Germany, 22-25 September 1996, pp. 979–984.

[34] H. Schumacher, A. C. Ulusoy, G. Liu, and G. Oliveri, "Si/SiGe ICs for ultra-broadband communications: The analog signal processing perspective," in *IEEE MTT-S International Microwave Symposium Digest (IMS)*, Seattle, WA, June 2013, pp. 1–4.

[35] B. Gilbert, "A precise four-quadrant multiplier with subnanosecond response," *IEEE Journal of Solid-State Circuits*, vol. 3, no. 4, pp. 365–373, December 1968.

[36] A. F. Murray and A. V. W. Smith, "Asynchronous arithmetic for VLSI neural systems," *Electronic Letters*, vol. 23, no. 12, pp. 642–643, June 1987.

[37] R. G. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, January 1962.

[38] "Digital video broadcasting (DVB), Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications; Part 1: DVB-S2," ETSI EN 302 307-1 V1.4.1, Tech. Rep., 11 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_en/302300_302399/30230701/01.04.01_60/en_30230701v010401p.pdf

[39] A. Dembo, O. Farotimi, and T. Kailath, "High-order absolutely stable neural networks," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 1, pp. 57–65, January 1991.

[40] M. Vidyasagar, "Minimum-seeking properties of analog neural networks with multilinear objective functions," *IEEE Transactions on Automatic Control*, vol. 40, no. 8, pp. 1359–1375, August 1995.

[41] E. B. Kosmatopoulos and M. A. Christodoulou, "Structural properties of gradient recurrent high-order neural networks," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 42, no. 9, pp. 592–603, September 1995.

[42] X. Xu, N. K. Huang, and W. T. Tsai, "A Generalized Neural Network Model," *Technical report*, Computer Science Department, Inst. of Technology, Univ. vol. 1, supplement 1, pp. 150, 1988.

[43] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. IT-27, no. 5, pp. 533–547, September 1981.

[44] B. S. Rüffer, C.-M. Kellett, P.-M. Dower, and S. R. Weller, "Belief propagation as a dynamical system: The linear case and open problems," *Control Theory & Applications, IET*, vol. 4, no. 7, pp. 1188–1200, July 2010.

[45] M. Mostafa, W. G. Teich, and J. Lindner, "Comparison of belief propagation and iterative threshold decoding based on dynamical systems," in *IEEE International Symposium on Information Theory*, Istanbul, Turkey, 7-12 July 2013, pp. 2995–2999.

[46] S. Hemati and A. H. Banihashemi, "Comparison between continuous-time asynchronous and discrete-time synchronous iterative decoding," in *IEEE Global Telecommunications Conference*, vol. 1, 29 November-3 December 2004, pp. 356–360.

[47] S. Hemati and A. H. Banihashemi, "Convergence speed and throughput of analog decoders," *IEEE Transactions on Communications*, vol. 55, no. 5, pp. 833–836, May 2007.

[48] S. Hemati and A. Yongacoglu, "On the dynamics of analog min-sum iterative decoders: An analytical approach," *IEEE Transactions on Communications*, vol. 58, no. 5, pp. 2225–2231, August 2010.

[49] S. Hemati, A. H. Banihashemi, and C. Plett, "A 0.18 μm CMOS analog min-sum iterative decoder for a (32,8) low-density paroty-check LDPC code," *IEEE Journal on Solid-State Circuits*, vol. 41, no. 11, pp. 2531–2540, November 2006.

[50] L. Kocarev, F. Lehman, G. M. Maggio, B. Scanavino, Z. Tasev, and A. Vardi, "Nonlinear dynamics of iterative decoding systems: Analysis and applications," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1366–1384, April 2006.

[51] C. Douillard, M. Jézéquel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *European Transactions on Telecommunications*, vol. 6, no. 5, pp. 507–511, 1995.

[52] C. Sgraja, A. Engelhart, W. G. Teich, and J. Lindner, "Combined equalization and decoding for BFDM packet transmission schemes," in *1st International OFDM-Workshop*, vol. 3, Hamburg, Germany, 21-22 September 1999, pp. 1–5.

[53] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo codes(1)," in *IEEE International Conference on Communications ICC*, vol. 2, Geneva, Switzerland, 1993, pp. 1064–1070.

[54] M. Mostafa, W. G. Teich, and J. Lindner, "Local stability analysis of discrete-time, continuous-state, complex-valued recurrent neural networks with inner state feedback," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 830–836, April 2014.

[55] A. Engelhart, W. G. Teich, J. Lindner, G. Jeney, S. Imre, and L. Pap, "A survey of multiuser/multisubchannel detection schemes based on recurrent neural networks," *Wireless Communications and Mobile Computing, Special Issue on Advances in 3G Wireless Networks*, vol. 2, no. 3, pp. 269–284, May 2002.

[56] E. Goles, E. Chacc, F. Fogelman-Soulié, and D. Pellegrin, "Decreasing energy functions as a tool for studying threshold functions," *Discrete Applied Mathematics*, vol. 12, no. 3, pp. 261–277, 1985.

[57] E. Goles, and S. Martinez, "A short proof on the cyclic behaviour of multithreshold symmetric automata," *Information and Control*, vol. 51, no. 2, pp. 95–97, November 1981.

[58] D. G. Zill, *A first course in differential equations with modeling applications*. Cengage Learning, Inc., 2009, ch. 9, pp. 340–345.

[59] T. J. C. MacKay. Encyclopedia of sparse graph codes. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[60] J. G. Proakis, *Digital Communications*. McGraw-Hill, Inc, 1995.

[61] R. M. Gray, *Toeplitz and Circulant Matrices: A Review*, ser. Foundations and Trends in Technology. Now Publishers, 2006.

[62] J. Lindner, *Informations übertragung*. Springer-Verlag Berlin Heidelberg, 2005.