Nonlinear Dynamics of Iterative Decoding Systems: Analysis and Applications

Ljupco Kocarev, *Fellow, IEEE*, Frederic Lehmann, Gian Mario Maggio, *Senior Member, IEEE*, Bartolo Scanavino, *Member, IEEE*, Zarko Tasev, and Alexander Vardy, *Fellow, IEEE*

***Abstract*—Iterative decoding algorithms may be viewed as high-dimensional nonlinear dynamical systems, depending on a large number of parameters. In this work, we introduce a simplified description of several iterative decoding algorithms in terms of the *a posteriori* average entropy, and study them as a function of a single parameter that closely approximates the signal-to-noise ratio (SNR). Using this approach, we show that virtually all the iterative decoding schemes in use today exhibit similar qualitative dynamics. In particular, a whole range of phenomena known to occur in nonlinear systems, such as existence of multiple fixed points, oscillatory behavior, bifurcations, chaos, and transient chaos are found in iterative decoding algorithms. As an application, we develop an adaptive technique to control transient chaos in the turbo-decoding algorithm, leading to a substantial improvement in performance. We also propose a new stopping criterion for turbo codes that achieves the same performance with considerably fewer iterations.**

***Index Terms*—Bifurcation theory, chaos, dynamical systems, iterative decoding, low-density parity-check codes, product codes, turbo codes.**

I. Introduction

DURING the past decade, it has been recognized that two classes of codes, namely, turbo codes [4] and low-density parity-check (LDPC) codes [13], [19], [23], perform at rates extremely close to the Shannon limit. Both classes of codes are based on a similar philosophy: constrained random code ensembles, decoded using iterative decoding algorithms. It is known [1], [15], [21], [27] that iterative decoding algorithms may be viewed as complex nonlinear dynamical systems. The goal of the present work is to contribute to the in-depth understanding of these powerful families of error-correcting codes from the point of view of the well-developed theory of nonlinear dynamical systems [26], [30].

Richardson [21] has developed a geometrical interpretation of the turbo-decoding algorithm, and formalized it as a discrete-time dynamical system defined on a continuous set. In the formalism of [21], the turbo-decoding algorithm appears as an iterative algorithm aimed at solving a system of $2k$ equations in $2k$ unknowns, where k is the number of information bits for the turbo code at hand. If the algorithm converges to a codeword, then this codeword constitutes a solution to this system of equations. Conversely, solutions to these equations provide fixed points of the turbo-decoding algorithm, when seen as a nonlinear mapping [9]. In a follow-up paper by Agrawal and Vardy [1], a bifurcation analysis of the fixed points in the turbo-decoding algorithm, as function of the signal-to-noise ratio (SNR), has been carried out. Further recent work has been concerned with the convergence behavior of iterative decoding systems, such as turbo codes [5] and LDPC codes [17], when the block length n tends to infinity. These papers open new promising directions for analyzing and designing coding schemes based on iterative decoding.

In this paper, following the approach of [21] and [1], we study iterative decoding algorithms as discrete-time nonlinear dynamical systems. The original contributions with respect to [1], [21] include a simplified measure of the performance of an iterative decoding algorithm in terms of *a posteriori* average entropy. In addition, we present a detailed characterization of the dynamical systems at hand and carry out an in-depth bifurcation analysis. For example, in [1], Lyapunov exponents were computed *only* at the fixed points (using the Jacobian matrix calculated at the fixed points). Herein, we compute the largest Lyapunov exponent of the chaotic attractors as well as the average chaotic transient lifetime. Furthermore, we show that, in general, iterative decoding algorithms exhibit a whole range of phenomena known to occur in nonlinear dynamical systems [20]. These phenomena include the existence of multiple fixed points, oscillatory behavior, chaos (for a finite block length), and transient chaos. Some of this was briefly discussed in the earlier work of Tasev, Popovski, Maggio, and Kocarev [27]. In this paper, we give a much more detailed and rigorous characterization of these phenomena. Moreover, we extend the results reported by Lehmann and Maggio in [17] for regular LDPC codes to the case of *irregular* LDPC codes, emphasizing the differences in dynamical behavior.

Subsequently, we show how the general principles distilled from our analysis may be applied to enhance the performance

Manuscript received April 25, 2003; revised January 13, 2006. This work was supported in part by STMicroelectronics, Inc., by the National Science Foundation, by the University of California through the DiMi Program, and by the CIFRE convention no. 98300.

L. Kocarev is with the Institute for Nonlinear Science, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: lkocarev@ucsd.edu).

F. Lehmann is with GET/INT, Department CITI, F-91011 Evry, France (e-mail: frederic.lehmann@int-evry.fr).

G. M. Maggio is with ST Microelectronics, Inc., Advanced System Technology, CH-1228, Geneva, Switzerland (e-mail: maggio@ieee.org).

B. Scanavino is with CERCOM, Politecnico di Torino, 10120 Torino, Italy (e-mail: bartolo.scanavino@polito.it).

Z. Tasev is with Kyocera Wireless Corp., San Diego, CA 92121 USA (e-mail: ztasev@kyocera-wireless.com).

A. Vardy is with the Department of Electrical and Computer Engineering, the Department of Computer Science and Engineering, and the Department of Mathematics, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: vardy@kilimanjaro.ucsd.edu).

Communicated by R. Urbanke, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2006.871054

of existing iterative decoding schemes. In particular, preliminary results on control of transient chaos, reported in [15], are further developed in this paper, and supported by the frame statistics with and without control. Another original contribution of this paper are new stopping criteria for turbo codes, which are derived starting from the definition of *a posteriori* average entropy herein (see Section IV-C).

In summary, this paper presents a self-contained comprehensive analysis of the dynamical behavior of most iterative decoding systems currently known, with a view toward improving their performance. For the interested reader, we should also mention the related recent work of Fu [12]. That work, which is completely independent[1] from ours, confirms some of our results concerning the dynamical behavior of turbo codes and suggests several interesting applications.

The rest of this paper is organized as follows. The next section is a primer on the theory of nonlinear dynamical systems. In Section III, we briefly review several well-known coding schemes based on iterative decoding, namely, parallel concatenated codes (PCC), serially concatenated codes (SCC), product codes (PC), and LDPC codes. Section IV is devoted to the investigation of nonlinear dynamics of parallel-concatenated turbo codes, using the classical turbo code of Berrou, Glavieux, and Thitimajshima [4] as a case study. For this code, the turbo decoding algorithm is first described as a high-dimensional discrete-time dynamical system, depending on a large number of parameters. Subsequently, we propose a simplified description of the dynamics of this algorithm, as a function of a single parameter that is closely related to the channel SNR. We then carry out a detailed bifurcation analysis. Similar considerations apply to other classes of codes. In particular, in Section V, we describe the nonlinear dynamics of SCCs, PCs, and LDPC codes (both regular and irregular). Finally, in Section VI, we discuss some applications based on the theory of nonlinear dynamic systems. First, we develop a simple technique for controlling the transient chaos in turbo decoding, thereby reducing the number of iterations required to reach an unequivocal fixed point. We then propose a new stopping criterion for iterative decoding, based on the average entropy of an information block. We conclude the paper with a brief discussion in Section VII.

II. Nonlinear Dynamical Systems

We define a differentiable discrete-time dynamical system by an evolution equation of the form

$$\boldsymbol{x}_{n+1} = f(\boldsymbol{x}_n)$$

where f is a differentiable function and the variables $\boldsymbol{x}$ vary over a state-space M, which can be $\mathbb{R}^m$ or a compact manifold. Computer experiments with iterative decoding algorithms usually exhibit transient behavior followed by what appears to be an asymptotic regime. Thus, iterative decoding algorithms are dissipative systems. In general, for dissipative systems, there exists a set $U \subseteq M$, which is asymptotically contracted by the time evolution to a compact set—that is, the set $\bigcap_{n\geq 0} f^n(U)$ is compact. The long-term behavior of a general dissipative dynamical system can be always categorized into one of the following two attractor classes:

- *Regular attractors:* fixed points, periodic orbits, and limit cycles (certain dynamical systems can also have an attractor that is a product of limit cycles, called a torus);
- *Irregular attractors:* chaotic and strange attractors.

Precise definitions of the terms above, along with several illustrative examples, are given later in this section.

A. One-Dimensional Chaotic Maps

We now give precise definitions of chaos for the one-dimensional case. Namely, we consider systems of the form

$$x_{n+1} = f(x_n), \qquad \text{where } x \in I \subset \mathbb{R}. \tag{1}$$

The fundamental property of chaotic systems is their sensitivity to initial conditions. From a practical point of view, this property implies that the trajectories (time evolution) of two nearby initial conditions diverge from each other quickly and their separation increases exponentially on the average. In the following two definitions, we assume that I is a subset of $\mathbb{R}$.

Definition 1 (topological transitivity): A function $f : I \to I$ is said to be topologically transitive if for every pair of open sets $U, V \subset I$, there is an integer $k > 0$ such that $f^k(U) \cap V \neq \emptyset$.

A topologically transitive map has points which, under the iteration of f, eventually move from any (arbitrarily small) neighborhood into any other neighborhood. We note that if a map has a dense orbit, then it is topologically transitive. For compact subsets of $\mathbb{R}$, the converse is also true.

Definition 2 (sensitive dependence): A function $f : I \to I$ has sensitive dependence of initial conditions if there is a $\delta > 0$ such that for every $x \in I$ and every open interval J containing x, there is a $y \in J$ with $|f^n(x) - f^n(y)| > \delta$ for some $n \geq 0$.

A map has sensitive dependence of initial conditions if, for all $x \in I$, there exist points arbitrary close to x which, under the iteration of f, eventually separate from x by at least δ. We stress that not all points near x need to be separated from x under the iteration of f, but there must be at least one such point in every neighborhood of x.

Definition 3 (chaotic set): Let V be an arbitrarry set. A function $f : V \to V$ is said to be chaotic on V if

1) f has sensitive dependence of initial conditions;
2) f is topologically transitive; and
3) periodic points of f are dense in V.

Example 1 (Logistic Map): Let a be a real number with $a > 4$, and consider the map $f : \mathbb{R} \to \mathbb{R}$ given by

$$f(x) = ax(1-x). \tag{2}$$

Clearly, the maximum value of f, achieved at $x = 1/2$, is $a/4$, which is strictly greater than 1 by the choice of a. Hence, there exists an open interval A_0 centered at $1/2$ with the following property: if $x \in A_0$, then $f(x) > 1$. But $f(x) > 1$ implies that $f^n(x) < 0$ for all $n \geq 2$. It follows that A_0 is a set of the points which immediately escape from the unit interval $I = [0, 1]$. In

[1]The manuscript by Fu [12] was submitted seven months *after* the submission date of the present paper.

general, let us define the set of points which escape from the interval I at the $(n+1)$th iteration as follows:

$$A_n \overset{\text{def}}{=} \{x \in I : f^i(x) \in I \text{ for } i \leq n \text{ and } f^{n+1}(x) \notin I\}.$$

Since A_0 is an open interval centered at $1/2$, the set $I - A_0$ consists of two closed intervals: B_0 and B_1. This, in turn, implies that A_1 consists of two disjoint open intervals. Continuing this argument inductively, we find that A_n consists of 2^n disjoint open intervals and $I - (A_0 \cup A_1 \cdots \cup A_n)$ consists of 2^{n+1} closed intervals B_i. Moreover, it is easy to see that f^{n+1} maps each B_i monotonically onto I. This implies that f^n has at least 2^n fixed points for all $n \geq 1$. Now, let us define

$$\Gamma \overset{\text{def}}{=} I - \bigcup_{n=0}^{\infty} A_n.$$

Then it can be shown that Γ is a Cantor set and the quadratic map $f(x) = ax(1-x)$ in (2) is chaotic on Γ. □

A set A is said to be *invariant* if $f(A) = A$. We say that a compact invariant set A is *topologically transitive* if there exists an $x \in A$ such that $\omega(x) = A$, where $\omega(x)$ is the set of limit points of the orbit $f^n(x)_{n\geq 0}$. Equivalently, a compact invariant set A is topologically transitive if the map $f : A \to A$ is topologically transitive (see Definition 1).

Definition 4 (attracting set): A closed invariant set $\mathcal{A}$ is called an attracting set if there is a neighborhood U of $\mathcal{A}$ such that $\forall x \in U, \forall n \geq 0, f^n(x) \in U$ and $f^n(x) \to \mathcal{A}$ as $n \to \infty$.

Definition 5 (basin of attraction): The domain or basin of attraction of an attracting set $\mathcal{A}$ is given by $\cup_{n\leq 0} f^n(\mathcal{A})$.

Note that even if f is noninvertible, f^{-1} still makes sense for sets: $f^{-1}(A)$ is the set of points in $\mathbb{R}$ that maps into A under f. Extending this idea, we can define $f^{-n}(A)$ inductively for all $n > 1$. Thus, $\cup_{n\leq 0} f^n(\mathcal{A})$ in Definition 5 is well-defined.

Definition 6 (attractor, chaotic attractor): An attractor is a topologically transitive attracting set. A chaotic attractor is a topologically transitive attracting set which is also chaotic.

Example 2: Consider again the map $f(x) = ax(1-x)$ in (2), but now choose $a \in (3,4)$ and think of f as a map from $[0,1]$ into $[0,1]$. Depending on the value of $a \in (3,4)$, three cases are possible. This logistic map either

- has a periodic attractor, which is unique and attracts almost every point; or
- has a nonchaotic attractor: almost every point is attracted to a Cantor set on which the map is not sensitive to initial conditions; or
- has a chaotic attractor: almost every point is attracted to a finite union of intervals, which is a chaotic set. □

Definition 7 (repelling hyperbolic set and chaotic saddle): A set Γ is a repelling hyperbolic set for f if Γ is closed, bounded, and invariant under f and there exists an integer n_0 such that $|(f^n)'(x)| > 1$ for all $n \geq n_0$ and for all $x \in \Gamma$. A repelling hyperbolic set which is also chaotic is called a chaotic saddle.

For example, the set Γ in Example 1 is a chaotic saddle. Here is another explicit example of a chaotic saddle.

Example 3 (piecewise linear map): Consider a piecewise linear function $f : \mathbb{R} \to \mathbb{R}$ defined by

$$f(x) = \begin{cases} a_1 x + c_1 & -\infty < x \leq b_1 \\ -a_2 x + c_2 & b_1 < x \leq b_2 \\ a_3 x + c_3 & b_2 < x \leq b_3 \\ \cdots & \cdots \\ -a_{2d} x + c_{2d} & b_{2d-1} < x < +\infty \end{cases} \tag{3}$$

where the parameters $a_i > 1, b_i \in (0,1)$, and c_i are chosen so that $f(0) = 0, f(1) = 1$, and the map $f(\cdot)$ is continuous. If $a_i b_i < 1$ for all i, the map has two attractors: a chaotic one, which is a subset of $(0,1)$, and a fixed point located at $-\infty$. The basin of attraction of the chaotic attractor is the interval $(0,1)$. On the other hand, if $a_i b_i > 1$ for all i, the map has only one attractor: the fixed point at $-\infty$. However, the map admits a chaotic saddle: this is the set of initial points which stay in the unit interval when time n goes to infinity. □

B. Lyapunov Exponents

Lyapunov exponents are useful quantitative indicators of asymptotic expansion and contraction rates in a dynamical system. They, therefore, have fundamental bearing upon stability and bifurcations. In particular, the local stability of fixed points (or periodic orbits) depends on whether the corresponding Lyapunov exponents are negative or not. For invariant sets with more complex dynamics, such as chaotic attractors and chaotic saddles, the situation is much more subtle. Invariant measures in this case are often not unique—for instance, associated to each (unstable) periodic orbit contained in an attractor is a Dirac ergodic measure whose support is the orbit. Ergodic measures are thus not unique if there is more than one periodic orbit (as is the case with most chaotic attractors), and each ergodic measure carries its own Lyapunov exponents.

Theorem 1 (multiplicative ergodic theorem): Let ρ be a probability measure on the space M. Let $f : M \to M$ be a measure-preserving map such that ρ is ergodic. Let $D_{\boldsymbol{x}} f = (\partial f_i / \partial x_j)$ denote the matrix of partial derivatives of the components f_i at $\boldsymbol{x}$. Define the matrix

$$T_{\boldsymbol{x}}^n \overset{\text{def}}{=} \left(D_{f^{n-1}(\boldsymbol{x})} f\right) \left(D_{f^{n-2}(\boldsymbol{x})} f\right) \cdots \left(D_{f(\boldsymbol{x})} f\right) D_{\boldsymbol{x}} f = D_{\boldsymbol{x}} f^n$$

and let $T_{\boldsymbol{x}}^{n*}$ be the adjoint of $T_{\boldsymbol{x}}^n$. Then for ρ-almost all $\boldsymbol{x}$, the following limit exists:

$$\Lambda_{\boldsymbol{x}} \overset{\text{def}}{=} \lim_{n\to\infty} \left(T_{\boldsymbol{x}}^{n*} T_{\boldsymbol{x}}^n\right)^{\frac{1}{2n}}. \tag{4}$$

Definition 8 (Lyapunov exponents): Logarithms of the eigenvalues of $\Lambda_{\boldsymbol{x}}$ in (4) are called Lyapunov exponents.

We denote the Lyapunov exponents by $\lambda_1 \geq \lambda_2 \geq \cdots$ or by $\lambda^{(1)} > \lambda^{(2)} > \cdots$ when they are no longer repeated by their multiplicity $m^{(i)}$. It is well known [20], [26] that if ρ is ergodic, then Lyapunov exponents are almost everywhere constant.

The finite set $\{\lambda_i\}$ captures the asymptotic behavior of the derivative along almost every orbit, in such a way that a positive λ_i indicates eventual expansion (and, hence, sensitive dependence on initial conditions) while negative λ_i indicate contraction along certain directions.

The exponential rate of growth of distances is given, in general, by λ_1—if one picks a vector at random, then its growth rate is λ_1. The growth rate of a k-dimensional Euclidean volume element is given by $\lambda_1 + \lambda_2 + \cdots + \lambda_k$.

Remark: There are well-developed techniques for computing *all* the Lyapunov exponents of a dynamical system (cf. [18]). Here, we describe a simple method for numerically computing the *largest* Lyapunov exponent, which will be used throughout this paper. Consider a trajectory $\boldsymbol{x}_{n+1} = f(\boldsymbol{x}_n)$ of an m-dimensional dynamical system $f : \mathbb{R}^m \to \mathbb{R}^m$, initialized at $\boldsymbol{x}_0$, as well as a nearby trajectory with initial condition $\boldsymbol{x}_0 + \Delta\boldsymbol{x}_0$. Let us write for convenience $w_n = \Delta\boldsymbol{x}_n$. Then the time evolution of w_n is given by $w_{n+1} = M(\boldsymbol{x}_n)w_n$, where $M = \partial f/\partial\boldsymbol{x}$ is the Jacobian matrix of f. Let $\|\cdot\|$ denote the Euclidean norm and define $w_i = w_{i-1}/d_i$, where $d_i = \|w_{i-1}\|$, for $i = 1, 2, \ldots$. Then the quantity $\frac{1}{n}\sum_{i=1}^{n} \ln d_i$ approaches the largest Lyapunov exponent as $n \to \infty$. For more details on this, we refer the reader to [18].

C. Largest Lyapunov Exponent as a Classification Tool

In experiments with iterative decoding we have encountered a number of qualitatively different attractors, each associated with a different type of time evolution. We found that the largest Lyapunov exponent is a good indicator for deducing what kind of state the ergodic measure ρ is describing.

Attracting Fixed Points and Periodic Points: A point $Q \in \mathbb{R}^m$ is a *fixed point* of $f: \mathbb{R}^m \to \mathbb{R}^m$ if $f(Q) = Q$. It is an *attracting fixed point* if there exists a neighborhood U of Q such that $\lim_{n\to\infty} f^n(x) = Q$ for all $x \in U$. The asymptotic measure of an attracting fixed point is the measure $\rho = \delta_Q$, where δ_Q is the Dirac delta function at Q. This measure is invariant and ergodic. A point Q is a *periodic point* of period k if $f^k(Q) = Q$. The least positive k for which $f^k(Q) = Q$ is called the prime period of Q; then Q is a fixed point of f^k. The set of all iterates of a periodic point forms a periodic trajectory (orbit). The Lyapunov exponents in both cases are simply related to the eigenvalues of the Jacobian matrix evaluated at Q (cf.[1]) and, therefore, are all negative. Thus, the largest Lyapunov exponent satisfies $\lambda_1 < 0$ in this case.

Attracting Limit Cycle: Let Γ be a closed curve in $\mathbb{R}^m$, homeomorphic to a circle. If Γ is an attractor, a volume element is not contracted along the direction of Γ. Therefore, its asymptotic measure has one Lyapunov exponent equal to zero while all the others are negative. Thus, $\lambda_1 = 0$.

Chaotic Attractor: An attractor A is *chaotic* if its asymptotic measure (natural measure) has a positive Lyapunov exponent. If any one of the Lyapunov exponents is positive, a volume element is expanded in some direction at an exponential rate and neighboring trajectories are diverging. This is precisely the sensitive dependence on initial conditions (cf. Definition 2). It follows that for chaotic attractors, we have $\lambda_1 > 0$ while $\sum_{i=1}^{m} \lambda_i < 0$ (since the invariant set A is an attractor).

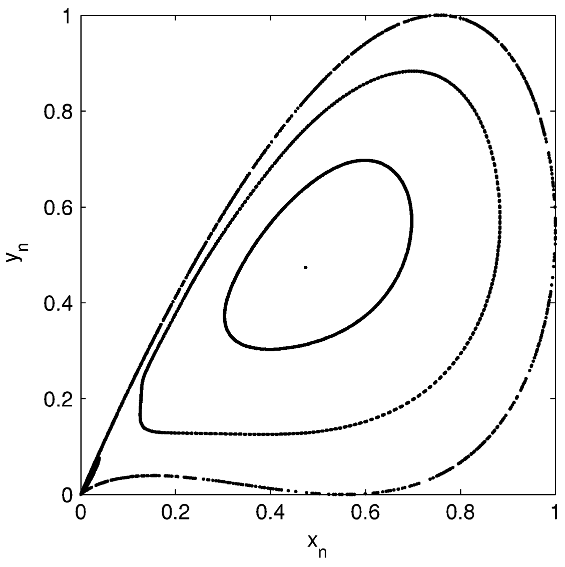


Fig. 1. Route to chaos for the simple two-dimensional map given by (5). The values of the parameter a corresponding to the invariant sets, starting from the fixed point toward chaos, are $1.9, 2.1, 2.16$, and 2.27.

Example 4: Let $a \in \mathbb{R}$ be a parameter, and consider the two-dimensional system defined on the plane $\mathbb{R}^2$ as follows:

$$x_{n+1} = y_n, \quad y_{n+1} = ay_n(1 - x_n). \tag{5}$$

The system has a fixed point at $x = y = \frac{a-1}{a}$, which is stable for $1 < a \leq 2$. As a passes through the value 2, this fixed point looses stability and spawns an attracting invariant circle. This circle grows as the parameter a increases, becoming noticeably warped. When $a = 2.27$, the circle completely breaks down, forming a chaotic attractor. Fig. 1 shows a typical route to chaos for this system, as well as the different attractors: fixed point, limit-cycle, and chaotic attractor. $\square$

Example 5: Consider a three-dimensional dynamical system parameterized by a constant $\alpha \in \mathbb{R}$ along with three constant vectors $\boldsymbol{a} = (a_1, a_2, a_3), \boldsymbol{b} = (b_1, b_2, b_3)$, and $\boldsymbol{c} = (c_1, c_2, c_3)$ in $\mathbb{R}^3$, and defined by the following recursions:

$$\begin{aligned}
X_{11}(n+1) &= F_{11}(\boldsymbol{X}_2(n); \boldsymbol{a}, \boldsymbol{b})\\
X_{12}(n+1) &= F_{12}(\boldsymbol{X}_2(n); \boldsymbol{a}, \boldsymbol{b})\\
X_{13}(n+1) &= F_{13}(\boldsymbol{X}_2(n); \boldsymbol{a}, \boldsymbol{b})\\
X_{21}(n+1) &= F_{21}(\boldsymbol{X}_1(n+1); \boldsymbol{a}, \boldsymbol{c})\\
X_{22}(n+1) &= F_{22}(\boldsymbol{X}_1(n+1); \boldsymbol{a}, \boldsymbol{c})\\
X_{22}(n+1) &= F_{23}(\boldsymbol{X}_1(n+1); \boldsymbol{a}, \boldsymbol{c})
\end{aligned} \tag{6}$$

where the pair of system variables $\boldsymbol{X}_1 = (X_{11}, X_{12}, X_{13})$ and $\boldsymbol{X}_2 = (X_{21}, X_{22}, X_{23})$ vary over $\mathbb{R}^3$ and the six iteration maps $F_{11}(\cdot; \boldsymbol{a}, \boldsymbol{b})$, $F_{12}(\cdot; \boldsymbol{a}, \boldsymbol{b})$, $F_{13}(\cdot; \boldsymbol{a}, \boldsymbol{b})$, $F_{21}(\cdot; \boldsymbol{a}, \boldsymbol{c})$, $F_{22}(\cdot; \boldsymbol{a}, \boldsymbol{c})$, and $F_{23}(\cdot; \boldsymbol{a}, \boldsymbol{c})$ are functions from $\mathbb{R}^3$ to $\mathbb{R}$ parameterized by the constants $\alpha, a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3$. These functions are given explicitly in (8) on the bottom of the following page.

In order to obtain a one-dimensional representation of this dynamical system, we define the quantity

$$E(n) \overset{\text{def}}{=} \frac{1}{3}\sum_{j=1}^{3} H_2(p_j(n)) \tag{7}$$

where $H_2(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is the binary entropy function and

$$p_j(n) \overset{\text{def}}{=} \frac{1}{1+\exp(X_{1j}(n)+X_{2j}(n)+\alpha a_j)}, \quad \text{for } j = 1,2,3.$$

The significance of the quantity $E(n)$ in (7) will be discussed later (see Section IV-C). It is clear, however, that we could try to study the time evolution of $E(n)$ in order to obtain some insight into the dynamical behavior of the system in (6) and (8), at the bottom of the page. For example, Figs. 2 and 3 show, in terms of $E(n)$, limit-cycle attractors for this system, under different settings of parameter values $\boldsymbol{a}, \boldsymbol{b}$, and $\boldsymbol{c}$. In both figures, we have assumed that the system is initialized with $\boldsymbol{X}_2(0) = (0,0,0)$.

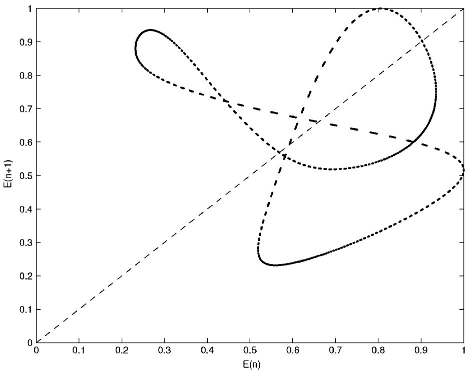


Fig. 2. Limit-cycle attractor, in terms of the evolution of $E(n)$, for the system of Example 5 parameterized by $\alpha = 3.125$, $\boldsymbol{a} = (-0.4486, 2.1564, 0.3722)$, $\boldsymbol{b} = (1.8710, 1.9027, 0.9954)$, $\boldsymbol{c} = (-0.4469, 1.0577, 1.4175)$.

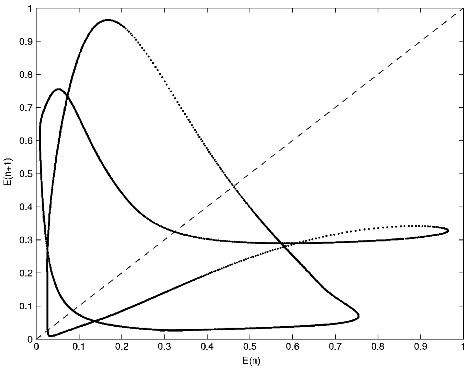


Fig. 3. Limit-cycle attractor, in terms of the evolution of $E(n)$, for the system of Example 5 parameterized by $\alpha = 3.125$, $\boldsymbol{a} = (0.6746, -0.2706, 1.2499)$, $\boldsymbol{b} = (0.4669, 2.7908, 1.6528)$, and $\boldsymbol{c} = (0.1764, 2.6040, 1.0167)$.

D. Transient Chaos

Chaotic saddles are nonattracting closed chaotic invariant sets having a dense orbit. A trajectory starting from a random initial condition in a state-space region that contains a chaotic saddle typically stays near the saddle, exhibiting chaotic-like dynamics, for a finite amount of time before eventually exiting the region and asymptotically approaching a final state, that is usually nonchaotic. Thus, in this case, chaos is only transient.

The natural measure for a chaotic saddle can be defined as follows. Let U be the region that encloses a chaotic saddle. If we iterate N_0 initial conditions, chosen uniformly in U, then the orbits which leave U never return to U. Let N_n be the number of orbits that have not left U after n iterates. For large n, this number will decay exponentially with time

$$\frac{N_n}{N_0} \sim \exp\left\{-\frac{n}{\tau}\right\}.$$

We say that τ is the *lifetime* of the chaotic transient. Let W be a subset of U. Then the natural measure of W is

$$\mu(W) \overset{\text{def}}{=} \lim_{n\to\infty}\lim_{N_0\to\infty}\frac{N_n(W)}{N_n}$$

where $N_n(W)$ is the number of orbit points which fall in W at time n. The last two equations imply that if the initial conditions are distributed according to the natural measure and evolved in time, then the distribution will decay exponentially at the rate $\alpha = 1/\tau$. Points which leave U after a long time do so by being attracted along the stable manifold of the saddle, bouncing around on the saddle in a (perhaps) chaotic way, and then exiting along the unstable manifold. For the natural measure μ of a chaotic saddle, one has

$$\alpha = \sum_{\lambda_i>0}\lambda_i - h(\mu)$$

$$\begin{aligned}
F_{11}(X_{22},X_{23};\boldsymbol{a},\boldsymbol{b}) &= \ln\frac{e^{\alpha(b_1+b_2)}+e^{\alpha(a_3+b_1+b_2+b_3)+X_{23}}+e^{\alpha(a_2+b_1+b_3)+X_{22}}+e^{\alpha(a_2+a_3+b_1)+X_{22}+X_{23}}}{1+e^{\alpha(a_3+b_3)+X_{23}}+e^{\alpha(a_2+b_2+b_3)+X_{22}}+e^{\alpha(a_2+a_3+b_2)+X_{22}+X_{23}}}\\
F_{12}(X_{21},X_{23};\boldsymbol{a},\boldsymbol{b}) &= \ln\frac{e^{\alpha(b_2+b_3)}+e^{\alpha(a_3+b_2)+X_{23}}+e^{\alpha(a_1+b_1+b_3)+X_{21}}+e^{\alpha(a_1+a_3+b_1)+X_{21}+X_{23}}}{1+e^{\alpha(a_3+b_3)+X_{23}}+e^{\alpha(a_1+b_1+b_2)+X_{21}}+e^{\alpha(a_1+a_3+b_1+b_2+b_3)+X_{21}+X_{23}}}\\
F_{13}(X_{21},X_{22};\boldsymbol{a},\boldsymbol{b}) &= \ln\frac{e^{\alpha b_3}+e^{\alpha(a_2+b_2)+X_{22}}+e^{\alpha(a_1+b_1+b_2+b_3)+X_{21}}+e^{\alpha(a_1+a_2+b_1)+X_{21}+X_{22}}}{1+e^{\alpha(a_2+b_2+b_3)+X_{22}}+e^{\alpha(a_1+b_1+b_2)+X_{21}}+e^{\alpha(a_1+a_2+b_1+b_3)+X_{21}+X_{22}}}\\
F_{21}(X_{12},X_{13};\boldsymbol{a},\boldsymbol{c}) &= \ln\frac{e^{\alpha(c_2+c_3)}+e^{\alpha(a_3+c_1+c_3)+X_{13}}+e^{\alpha(a_2+c_2)+X_{12}}+e^{\alpha(a_2+a_3+c_1)+X_{12}+X_{13}}}{1+e^{\alpha(a_3+c_1+c_2)+X_{13}}+e^{\alpha(a_2+c_3)+X_{12}}+e^{\alpha(a_2+a_3+c_1+c_2+c_3)+X_{12}+X_{13}}}\\
F_{22}(X_{11},X_{13};\boldsymbol{a},\boldsymbol{c}) &= \ln\frac{e^{\alpha c_3}+e^{\alpha(a_3+c_1+c_2+c_3)+X_{13}}+e^{\alpha(a_1+c_2)+X_{11}}+e^{\alpha(a_1+a_3+c_1)+X_{11}+X_{13}}}{1+e^{\alpha(a_3+c_1+c_2)+X_{13}}+e^{\alpha(a_1+c_2+c_3)+X_{11}}+e^{\alpha(a_1+a_3+c_1+c_3)+X_{11}+X_{13}}}\\
F_{23}(X_{11},X_{12};\boldsymbol{a},\boldsymbol{c}) &= \ln\frac{e^{\alpha(c_1+c_2)}+e^{\alpha(a_2+c_1+c_2+c_3)+X_{12}}+e^{\alpha(a_1+c_1+c_3)+X_{11}}+e^{\alpha(a_1+a_2+c_1)+X_{11}+X_{12}}}{1+e^{\alpha(a_2+c_3)+X_{12}}+e^{\alpha(a_1+c_2+c_3)+X_{11}}+e^{\alpha(a_1+a_2+c_2)+X_{11}+X_{12}}}.
\end{aligned} \tag{8}$$

where λ_i are the Lyapunov exponents and $h(\cdot)$ is the measure-theoretic (or the Kolmogorov–Sinai) entropy [20, p. 235].

We next consider a simple example where the characteristics of a chaotic saddle, namely, the chaotic transient lifetime and the Lyapunov exponents, can be computed in a closed form.

Example 6 (piecewise linear map): Consider the piecewise linear map (3) with $a_i b_i > 1$ for all i. In other words, every linear piece maps a part of the unit interval onto an interval containing the entire unit interval. This requires $|f'(x)| > 1$ everywhere, and thus all periodic orbits are unstable and the chaotic saddle is the closure of the set of all finite periodic orbits. Note that the chaotic saddle is a subset of the unit interval. For the natural measure μ of this chaotic saddle, one finds

$$\alpha = -\log \sum_{i=1}^{2d} a_i^{-1}$$

$$\lambda = \frac{\sum_{k=1}^{2d} a_k^{-1} \log a_k}{\sum_{i=1}^{2d} a_i^{-1}}$$

$$h(\mu) = \frac{\sum_{k=1}^{2d} a_k^{-1} \log\left(a_k \sum_{i=1}^{2d} a_i^{-1} \log a_i\right)}{\sum_{i=1}^{2d} a_i^{-1}}$$

where $\lambda = \lambda_1$ is the largest Lyapunov exponent. Note that this system satisfies the relation $h(\mu) = \lambda - \alpha$, which is characteristic of a chaotic saddle. □

III. Iterative Decoding Algorithms

We now briefly describe the coding schemes, and the associated iterative decoding algorithms, considered in this paper.

A. Turbo Concatenated Codes

For simplicity, we restrict our attention to concatenated codes (CC) involving two systematic constituent block codes separated by an interleaver. If convolutional codes are used, we assume that the trellis has been terminated. We let k and R denote the number of information bits and the coding rate of the CC, respectively. Throughout the paper, we assume that the codes are binary and the channel is a memoryless Gaussian channel with binary phase-shift keying (BPSK) modulation ($0 \rightarrow +1$ and $1 \rightarrow -1$). We let σ denote the standard deviation of the channel noise.

Parallel concatenated codes (PCC): PCCs, or turbo codes [3], [4], are illustrated in Fig. 4(a). The k information bits are permuted by a random interleaver, then both the information bits and the permuted information bits are fed to the two constituent encoders. The output codeword is formed by multiplexing the information bits with the redundancy produced by the two encoders.

Serially Concatenated Codes (SCC): SCCs [2] are illustrated in Fig. 4(b). An outer code adds redundancy to the k information bits, and the resulting outer codeword is permuted by a random interleaver. The inner encoder treats the output of the interleaver as information bits and adds more redundancy. The output of the inner encoder is the transmitted codeword.

Product codes (PC): PCs [11] are illustrated in Fig. 4(c). The k information bits are placed in an array of k_{R} rows and k_{C} columns. Then the k_{R} rows are encoded by the $(n_{\mathrm{C}}, k_{\mathrm{C}})$ *row code* and the resulting n_{C} columns are encoded by the $(n_{\mathrm{R}}, k_{\mathrm{R}})$ *column code*. Note that the only difference between PC and SCC is that the random interleaver is replaced by a block interleaver.

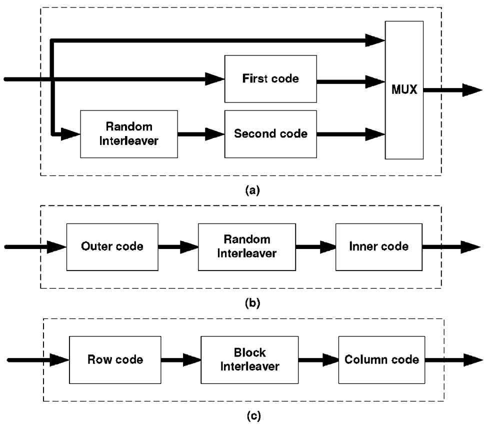


Fig. 4. Three types of turbo concatenated codes: (a) PCC, (b) SCC, (c) PC.

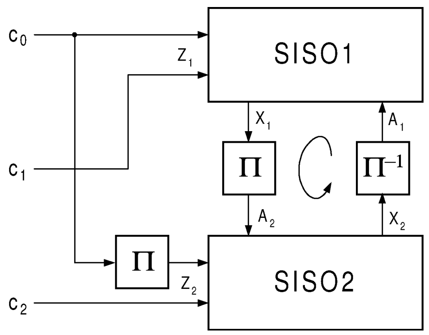


Fig. 5. Block diagram of an iterative decoder for a turbo concatenated code.

B. Iterative Decoding of Turbo Concatenated Codes

The block diagram of an iterative decoder for a generic CC is illustrated in Fig. 5. The decoding process iterates between two decoders denoted SISO1 and SISO2. We assume that each of the two decoders is a maximum *a posteriori* probability (MAP) decoder (although, in practice, approximations to MAP decoding are often employed). SISO1 uses channel observations $\boldsymbol{Z}_1$ and *a priori* information $\boldsymbol{A}_1$, in the form of log-likelihood ratios (LLRs), to generate *a posteriori* bit-by-bit LLRs $\boldsymbol{L}_1$. The *extrinsic* information is then given by $\boldsymbol{X}_1 = \boldsymbol{L}_1 - \boldsymbol{Z}_1 - \boldsymbol{A}_1$. After interleaving, $\boldsymbol{X}_1$ is used as the *a priori* information $\boldsymbol{A}_2$, in conjunction with $\boldsymbol{Z}_2$, by SISO2 to generate *a posteriori* bit-by-bit LLRs $\boldsymbol{L}_2$. Then, the extrinsic information $\boldsymbol{X}_2 = \boldsymbol{L}_2 - \boldsymbol{Z}_2 - \boldsymbol{A}_2$ is used as the *a priori* information for SISO1, after deinterleaving. And so on, for a specified number of iterations.

SISO1 corresponds to the decoding of the first code, the outer code, and the row code for PCC, SCC, and PC, respectively. Similarly, SISO2 corresponds to the decoding of the second code, the inner code, and the column code for PCC, SCC, and PC, respectively. Throughout the paper, we assume that all the quantities exchanged by the decoders are in the form of LLRs and can be modeled [23] as independent and identically distributed (i.i.d.) random variables having a symmetric Gaussian distribution.

This iterative decoding system may be viewed as a closed-loop dynamical system, with SISO1 and SISO2 acting as the constituent decoders in the corresponding open-loop system.

C. LDPC Codes and Their Decoding

An LDPC code is defined by a bipartite graph consisting of variable and check nodes appropriately related by edges. Let d_v and d_c be the maximum variable and check node degrees, respectively. We denote the variable (resp., check) degree-distribution polynomials [22] by

$$\lambda(x) = \sum_{i=2}^{d_v} \lambda_i x^{i-1} \quad \text{and} \quad \rho(x) = \sum_{i=2}^{d_c} \rho_i x^{i-1}$$

where λ_i (resp., ρ_i) is the fraction of variable (resp., check) nodes of degree i. If $\lambda(x) = x^{d_v-1}$ and $\rho(x) = x^{d_c-1}$, the bipartite graph is regular and the corresponding LDPC code is said to be regular; otherwise, the code is irregular.

The message-passing (or sum–product) algorithm [16], [23] used to decode LDPC codes proceeds as follows. The message v sent by a variable node to a check node on an edge e is the log-likelihood ratio of that variable node, given the LLRs of the check nodes u_i, received on all incoming edges except e, and given the channel log-likelihood ratio u_0 of the variable node itself. Thus,

$$v = u_0 + \sum_i u_i. \tag{9}$$

The message u sent by a check node to a variable node on an edge e is the log-likelihood ratio of that check node, given the LLRs of the variable nodes v_i received on all incoming edges except e. Thus,

$$\tanh\left(\frac{u}{2}\right) = \prod_i \tanh\left(\frac{v_i}{2}\right). \tag{10}$$

Equations (9) and (10) constitute one decoding iteration. Initially, each variable node (message) is initialized with the channel log-likelihood ratio u_0 of the corresponding bit.

IV. Dynamics of Parallel-Concatenated Turbo Codes

In this section, we study the nonlinear dynamics of parallel-concatenated turbo codes. We then derive a simplified representation qualitatively describing the dynamics of the iterative decoding algorithm for this class of codes. Similar considerations apply to other types of codes as well (our bifurcation analysis for these codes is presented in Section V).

As a case study, we shall consider (in Sections IV-D and IV-E) the classical rate-$1/3$ turbo code of Berrou, Glavieux, and Thitimajshima [4], for which the two constituent codes are identical memory-4 recursive convolutional codes, with the feedforward polynomial $1 + D^4$ and the feedback polynomial $1 + D + D^2 + D^3 + D^4$. The interleaver length is $k = 1024$ bits throughout (except in Sections IV-A and IV-E).

Fig. 6 shows the performance of this code, as a function of the number of iterations. Referring to Fig. 6 (see also [1] and other papers), the performance of the turbo decoding algorithm can be classified into three distinct regions.

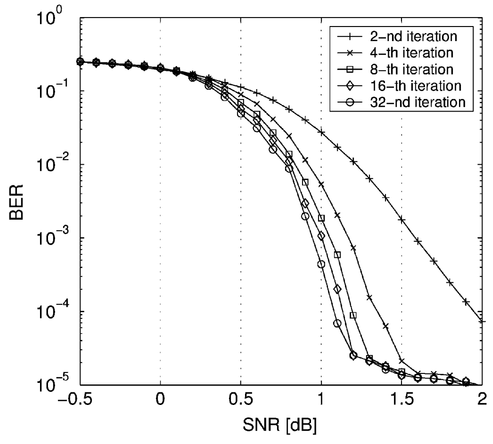


Fig. 6. Performance of the classical rate-$1/3$ parallel-concatenated turbo code with interleaver length 1024, for increasing number of iterations. The waterfall region corresponds to SNRs between about 0.25 and 1.25 dB.

Low SNR Region: For very low values of SNR, the extrinsic LLRs often converge to values leading to a large number of incorrect decisions. The corresponding performance curves decrease slowly with SNR.

High SNR Region: For very high SNRs, the extrinsic LLRs often converge to values leading to mostly correct decisions. However, the corresponding performance curves reach an "error floor" and decrease slowly with SNR.

Waterfall Region: The transition between the aforementioned SNR regions is called the "waterfall region," as the performance curves decrease sharply with SNR in this region.

In this section, we study the dynamics of iterative turbo decoding, qualitatively and quantitatively, in each of these regions.

A. A Simple Example

Consider a simple parallel-concatenated turbo code of [3], for which the constituent codes are identical memory-3 recursive systematic convolutional codes, with the feedforward polynomial $1 + D^2 + D^3$ and feedback polynomial $1 + D + D^3$. Assume that both encoders are initialized to the all-zero state.

For the sake of this example, let us first assume that $k = 3$. Thus, there are only three information bits $\boldsymbol{i} = (i_1, i_2, i_3)$. We choose the cyclic interleaver $\pi(\boldsymbol{i}) = (i_3, i_1, i_2)$. Let us denote the parity bits generated by constituent encoders, upon input $\boldsymbol{i}$ and $\pi(\boldsymbol{i})$ by $\boldsymbol{p}_1(\boldsymbol{i}) = (p_1^1, p_2^1, p_3^1)$ and $\boldsymbol{p}_2(\boldsymbol{i}) = (p_1^2, p_2^2, p_3^2)$, respectively. Then, we have

$$\begin{aligned} \boldsymbol{p}_1(\boldsymbol{i}) &= (i_1, i_1 + i_2, i_2 + i_3) \\ \boldsymbol{p}_2(\boldsymbol{i}) &= (i_3, i_1 + i_3, i_1 + i_2) \end{aligned} \tag{11}$$

where all the summations are modulo 2. Let $\boldsymbol{c}_0 = (c_1^0, c_2^0, c_3^0)$, $\boldsymbol{c}_1 = (c_1^1, c_2^1, c_3^1)$, and $\boldsymbol{c}_2 = (c_1^2, c_2^2, c_3^2)$ denote the channel output sequences corresponding to the input sequences $\boldsymbol{i}, \boldsymbol{p}_1(\boldsymbol{i})$, and $\boldsymbol{p}_2(\boldsymbol{i})$, respectively. Further, let $\boldsymbol{X}_1 = (X_{11}, X_{12}, X_{13})$ and $\boldsymbol{X}_2 = (X_{21}, X_{22}, X_{23})$ denote the extrinsic information at the output of two decoders, SISO1 and SISO2, respectively. We order the vectors $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ so that they correspond to the information bits in the natural, noninterleaved, order (thus, both

X_{11} and X_{21} reflect the extrinsic information for the bit i_1). Referring to Fig. 5, while noting that $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are just permuted versions of $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, respectively, we can write

$$\begin{aligned} X_{11}(n+1) &= F_{11}(\boldsymbol{X}_2(n); \boldsymbol{c}_0, \boldsymbol{c}_1) \\ X_{12}(n+1) &= F_{12}(\boldsymbol{X}_2(n); \boldsymbol{c}_0, \boldsymbol{c}_1) \\ X_{13}(n+1) &= F_{13}(\boldsymbol{X}_2(n); \boldsymbol{c}_0, \boldsymbol{c}_1) \\ X_{21}(n+1) &= F_{21}(\boldsymbol{X}_1(n+1); \boldsymbol{c}_0, \boldsymbol{c}_2) \\ X_{22}(n+1) &= F_{22}(\boldsymbol{X}_1(n+1); \boldsymbol{c}_0, \boldsymbol{c}_2) \\ X_{22}(n+1) &= F_{23}(\boldsymbol{X}_1(n+1); \boldsymbol{c}_0, \boldsymbol{c}_2) \end{aligned} \tag{12}$$

where n denotes the iteration number. We now wish to give explicit closed-form expressions for the functions $F_{11}(\cdot; \boldsymbol{c}_0, \boldsymbol{c}_1)$, $F_{12}(\cdot; \boldsymbol{c}_0, \boldsymbol{c}_1)$, $F_{13}(\cdot; \boldsymbol{c}_0, \boldsymbol{c}_1)$, $F_{21}(\cdot; \boldsymbol{c}_0, \boldsymbol{c}_2)$, $F_{22}(\cdot; \boldsymbol{c}_0, \boldsymbol{c}_2)$, and $F_{23}(\cdot; \boldsymbol{c}_0, \boldsymbol{c}_2)$. As shown, for example, in [1], the extrinsic log-likelihood ratio for the jth information bit at the output of the SISO1 decoder is given by

$$X_{1j} = \ln \frac{\sum_{\boldsymbol{i}: i_j=1} p(\boldsymbol{c}_0|\boldsymbol{i}) p(\boldsymbol{c}_1|\boldsymbol{i}) p(\boldsymbol{i})}{\sum_{\boldsymbol{i}: i_j=0} p(\boldsymbol{c}_0|\boldsymbol{i}) p(\boldsymbol{c}_1|\boldsymbol{i}) p(\boldsymbol{i})} - \frac{2}{\sigma^2} c_j^0 - X_{2j} \tag{13}$$

for all $j = 1, 2, 3$, where $p(\boldsymbol{i})$ is the "*a priori*" probability of the information bits given by

$$p(\boldsymbol{i}) \sim \exp(i_1 X_{21} + i_2 X_{22} + i_3 X_{23}) \tag{14}$$

while

$$p(\boldsymbol{c}_0|\boldsymbol{i}) \sim \exp\left(\frac{2}{\sigma^2}\left(i_1 c_1^0 + i_2 c_2^0 + i_3 c_3^0\right)\right) \tag{15}$$

$$p(\boldsymbol{c}_1|\boldsymbol{i}) \sim \exp\left(\frac{2}{\sigma^2}\left(p_1^1 c_1^1 + p_2^1 c_2^1 + p_3^1 c_3^1\right)\right). \tag{16}$$

Similarly, the extrinsic log-likelihood ratio for the jth information bit at the output of the SISO2 decoder is given by

$$X_{2j} = \ln \frac{\sum_{\boldsymbol{i}: i_j=1} p(\boldsymbol{c}_0|\boldsymbol{i}) p(\boldsymbol{c}_2|\boldsymbol{i}) p(\boldsymbol{i})}{\sum_{\boldsymbol{i}: i_j=0} p(\boldsymbol{c}_0|\boldsymbol{i}) p(\boldsymbol{c}_2|\boldsymbol{i}) p(\boldsymbol{i})} - \frac{2}{\sigma^2} c_j^0 - X_{1j} \tag{17}$$

for all $j = 1, 2, 3$, where $p(\boldsymbol{c}_0|\boldsymbol{i})$ is still given by (15), while $p(\boldsymbol{c}_2|\boldsymbol{i})$ and $p(\boldsymbol{i})$ are now given by

$$p(\boldsymbol{i}) \sim \exp(i_1 X_{11} + i_2 X_{12} + i_3 X_{13}) \tag{18}$$

$$p(\boldsymbol{c}_2|\boldsymbol{i}) \sim \exp\left(\frac{2}{\sigma^2}\left(p_1^2 c_1^2 + p_2^2 c_2^2 + p_3^2 c_3^2\right)\right). \tag{19}$$

Using (11), we can write (16) and (19) explicitly in terms of the information bits i_1, i_2, i_3. Now, substituting (14)–(16) and (18), (19) into the summations of (13) and (17), we finally obtain the explicit closed-form expressions for the iteration maps $F_{11}(\cdot)$, $F_{12}(\cdot)$, $F_{13}(\cdot)$, $F_{21}(\cdot)$, $F_{22}(\cdot)$, and $F_{23}(\cdot)$ in (12). We find that these maps are given by (8), with $\boldsymbol{a} = \boldsymbol{c}_0, \boldsymbol{b} = \boldsymbol{c}_1$, $\boldsymbol{c} = \boldsymbol{c}_2$, and $\alpha = 2/\sigma^2$. Thus, what we have here is precisely the three-dimensional dynamical system of Example 5. It follows that Figs. 2 and 3 depict limit-cycle attractors for this iterative decoder, under a certain choice of parameter values (in particular, for $\sigma = 0.8$ which corresponds to $\alpha = 3.125$).

B. Turbo Decoding as a Nonlinear Mapping

We now show how the derivations in Section IV-A extend to the case of general parallel concatenated turbo codes. As before, let $\boldsymbol{i}$ denote the sequence (of length k) of information bits at the input to the turbo encoder. Let $\boldsymbol{p}_1(\boldsymbol{i})$ and $\boldsymbol{p}_2(\boldsymbol{i})$ be the parity bits produced by the first and second encoder, respectively. Let $\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2$ denote the channel outputs corresponding to the input sequences $\boldsymbol{i}, \boldsymbol{p}_1(\boldsymbol{i})$, and $\boldsymbol{p}_2(\boldsymbol{i})$, respectively. Then, as in (12), the turbo decoding algorithm can be described by a discrete-time dynamical system of the form

$$\begin{aligned} \boldsymbol{X}_1(n+1) &= \boldsymbol{F}_1(\boldsymbol{X}_2(n); \boldsymbol{c}_0, \boldsymbol{c}_1) \\ \boldsymbol{X}_2(n+1) &= \boldsymbol{F}_2(\boldsymbol{X}_1(n+1); \boldsymbol{c}_0, \boldsymbol{c}_2) \end{aligned} \tag{20}$$

where $\boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathbb{R}^k$ is the extrinsic information exchanged by the two SISO decoders, whereas $\boldsymbol{F}_1 = (F_{11}, F_{12}, \ldots, F_{1k})$ and $\boldsymbol{F}_2 = (F_{21}, F_{21}, \ldots, F_{2k})$ are nonlinear functions, parameterized by the channel output $\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2$, which depend on the constituent codes. Specifically, as in (13), $\boldsymbol{F}_1$ is given by

$$F_{1j}(\boldsymbol{X}_2; \boldsymbol{c}_0, \boldsymbol{c}_1) = \ln \frac{\sum_{i=1}^{2^k} f_{ij}}{\sum_{i=1}^{2^k} g_{ij}} - \frac{2}{\sigma^2} c_j^0 - X_{2j} \tag{21}$$

for $j = 1, 2, \ldots, k$, where f_{ij} and g_{ij} are exponential functions of $\boldsymbol{X}_2, \boldsymbol{c}_0, \boldsymbol{c}_1$. A similar expression holds for the function $\boldsymbol{F}_2$. The exact form of f_{ij} and g_{ij} depends on the specific constituent codes and on the specific SISO decoding algorithm. As shown in [21], the system (20) always depends smoothly on its $2k$ variables $\boldsymbol{X}_1, \boldsymbol{X}_2$, and $3k$ parameters $\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2$.

Assuming *a priori* a uniform probability distribution of the information bits, the initial conditions in (20) should be set[2] at the origin: $\boldsymbol{X}_1 = \boldsymbol{X}_2 = \boldsymbol{0}$. At each iteration n, the decoder computes the $2k$ values of $\boldsymbol{X}_1, \boldsymbol{X}_2$. A decision on the jth bit can be made according to the sign of the log-likelihood ratio

$$L_j(n) \overset{\text{def}}{=} \ln \frac{p_j^1(n)}{p_j^0(n)} = X_{1j}(n) + X_{2j}(n) + \frac{4}{\sigma^2} c_j^0 \tag{22}$$

where $p_j^0(n)$ and $p_j^1(n)$ are the probabilities that the jth information bit is 0 or 1, respectively. Note that, if c_j^0 is known, $p_j^0(n)$ and $p_j^1(n)$ can be computed from $X_{1j}(n)$ and $X_{2j}(n)$, and *vice versa*, using (22). Let us write $\boldsymbol{p}_0 = (p_1^0, p_2^0, \ldots, p_k^0)$. Then the system (20) can be rewritten in equivalent form as

$$\boldsymbol{p}_0(n+1) = \boldsymbol{G}(\boldsymbol{p}_0(n); \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2) \tag{23}$$

where $\boldsymbol{G}$ is a nonlinear function. The dynamical system (23) is again high dimensional with a large number of parameters. However, it is advantageous in that $\boldsymbol{p}_0 \in [0,1]^k$. A typical trajectory of the turbo decoding algorithm in the form of (20) starts at the origin and converges to an attractor (chaotic or nonchaotic), usually located in the region of large (positive or negative) values of $\boldsymbol{X}_1, \boldsymbol{X}_2$. In contrast, a typical trajectory in the form of (23) starts at the point $(1/2, 1/2, \ldots, 1/2)$ and converges to an attractor that is always in $[0,1]^k$.

C. A Simplified Model of Iterative Decoding

The dynamical systems (20) and (23) are much too complex for detailed analysis. In order to study the dynamics of these systems, we suggest the following simplified representation of turbo decoding trajectories. Define

$$\begin{aligned} E(n) &\overset{\text{def}}{=} \frac{1}{k} \sum_{j=1}^{k} H_2\left(p_j^0(n)\right) \\ &= -\frac{1}{k} \sum_{j=1}^{k} \left(p_j^0(n) \log_2 p_j^0(n) + p_j^1(n) \log_2 p_j^1(n)\right) \end{aligned} \tag{24}$$

[2]However, one may use other initial conditions; for instance, to compute the Lyapunov exponents or the average chaotic transient lifetime.

The quantity E represents the *a posteriori average entropy*, which gives a measure of the reliability of the decisions for a given length-k information sequence. Note that if all bits are detected correctly, p_j is either 0 or 1 for all j, and $E=0$. On the other hand, if all bits are equally probable (that is, ambiguous), we have $E=1$. Note that $E\to 0$ does not automatically mean that the decisions are correct, but rather that the turbo decoding algorithm is very confident about its decisions. However, the excellent performance of turbo decoding seems to indicate that, in most cases, the decisions are, in fact, correct when $E(n)\to 0$.

In what follows, we consider three types of plots: $E(n)$ versus n, $E(n{+}1)$ versus $E(n)$, and E versus SNR for $n\to\infty$. By plotting the iterates of $E(n)$, we obtain a simple representation of the trajectories of the turbo-decoding algorithm in the interval $[0,1]$. On the other hand, recursive plots of $E(n{+}1)$ versus $E(n)$ are useful to visualize the invariant sets of the decoding algorithm. Finally, a plot of $\lim_{n\to\infty} E(n)$ versus SNR represents a bifurcation diagram.

Although both (20) and (23) depend on $3k$ parameters, we will study these systems as a function of a single parameter (which closely approximates the channel SNR), following the approach developed by Agrawal and Vardy [1]. As in [1], we assume that the noise samples corresponding to the channel observations $\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2$, represented in vector form as

$$\boldsymbol{\nu} = (\nu_1, \nu_2, \ldots, \nu_{3k}), \tag{25}$$

are such that the noise ratios $\nu_1/\nu_2, \nu_2/\nu_3, \ldots, \nu_{3k-1}/\nu_{3k}$ are fixed. Then the noise sequence $\boldsymbol{\nu}$ in (25) is completely determined by the sample variance

$$\hat{\sigma}^2 = \frac{1}{3k}\sum_{i=1}^{3k} \nu_i^2. \tag{26}$$

It follows that one can parameterize the system by the single parameter $\hat{\sigma}$. Note that $\hat{\sigma}^2$ is a good approximation of the channel noise variance σ^2, since k is typically a large integer. Thus, the SNR is well approximated by $E_b/N_0 \simeq 1/(2R\hat{\sigma}^2)$, where E_b is the energy per information bit, N_0 is the noise power spectral density, and R is the code rate.

D. Bifurcation and Chaos in Turbo Decoding

We now give a qualitative description of the behavior of the turbo decoding algorithm, for small interleaver lengths (e.g., $k=1024$). Our conclusions are based on comprehensive simulations of the algorithm for SNRs ranging from $-\infty$ to $+\infty$ with different realizations of the noise (different noise ratios $\nu_1/\nu_2, \nu_2/\nu_3, \ldots, \nu_{3k-1}/\nu_{3k}$). Fig. 7 schematically summarizes the results of our bifurcation analysis: The turbo decoding algorithm exhibits all three types of attractors previously discussed: fixed points (or periodic-orbit attractors), limit-cycle attractors, and chaotic attractors, which correspond to negative, zero, and positive largest Lyapunov exponents, respectively.

1) Fixed Points and Bifurcations: As shown in [1], the turbo decoding algorithm admits two types of fixed points—indecisive and unequivocal.

Unequivocal fixed point: In this case, p_j^0 is close to 0 or 1 for all j, so the log-likelihoods L_j assume large values, and consequently $E\simeq 0$. Decisions corresponding to an unequivocal

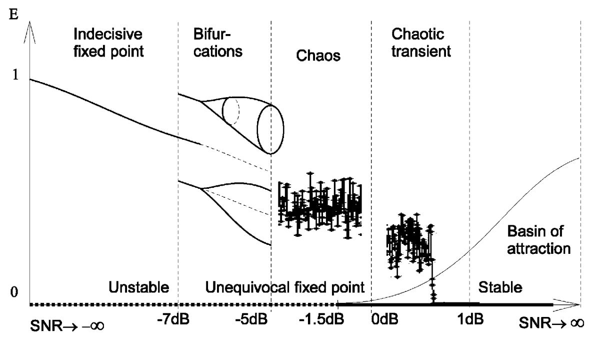


Fig. 7. Schematic bifurcation diagram of the turbo decoding algorithm in (20), for small to moderate interleaver lengths. Unequivocal fixed point corresponds to $E=0$; it is shown using a bold dotted line when is unstable and a bold solid line when is stable. The basin of attraction for the unequivocal fixed point is also shown schematically as a function of SNR.

fixed point will typically form a valid codeword, which usually coincides with the transmitted codeword.

Indecisive fixed point: At this fixed point, the turbo decoding algorithm is ambiguous regarding the values of the information bits, with p_j^0 being close to $1/2$ for all j. Thus, $L_j \simeq 0$ and $E\simeq 1$. Decisions corresponding to this fixed point typically will *not* form a valid codeword.

It is proved in [1] that for asymptotically low SNRs, the turbo decoding algorithm has a unique indecisive fixed point. Experiments show that not only is this true, but the SNR required for the existence and stability of this fixed point is not extremely low: we found that the indecisive fixed point is stable for all SNRs less than -7 dB.

When the SNR approaches $-\infty$, the average entropy for the indecisive fixed point approaches the limit $E=1$. We found that the indecisive fixed point moves toward smaller values of E with increasing SNR. It eventually looses its stability (or disappears), typically in the SNR range of -7 to -5 dB.

As is well known [20], [30], there are three ways in which a fixed point of a discrete-time dynamical system may loose its stability: when the Jacobian matrix evaluated at the fixed point admits complex conjugate eigenvalues on the unit circle (Neimark–Sacker bifurcation), or an eigenvalue at $+1$ (tangent bifurcation), or an eigenvalue at -1 (flip bifurcation). In our experiments, we have observed all three types of bifurcations in the turbo-decoding algorithm.

For high SNRs, the turbo-decoding algorithm typically converges to an unequivocal fixed point. In each instance of the algorithm that we have analyzed, an unequivocal fixed point existed for all values of SNR from $-\infty$ to $+\infty$. This point is always represented by the average entropy value of $E=0$, and becomes stable at about -1.5 dB. However, when the SNR is less than about 0 to 0.5 dB, the decoding algorithm "cannot see" this fixed point, since the initial conditions are not within its basin of attraction. The basin of attraction of the unequivocal fixed point grows with SNR.

2) Chaotic Attractors and Transient Chaos: The turbo-decoding algorithm also has limit cycle and chaotic attractors. Given a general dynamical system with only nonchaotic, asymptotically stable behavior, how do chaotic attractors arise as a parameter of the system varies?

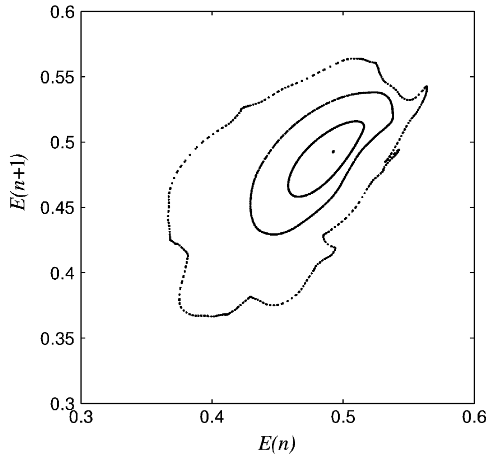


Fig. 8. Torus-breakdown route to chaos for the classical rate-$1/3$ turbo code. The values of SNR corresponding to the invariant sets, starting from the fixed point toward chaos, are: -6.7 dB, -6.6 dB, -6.5 dB, -6.2 dB.

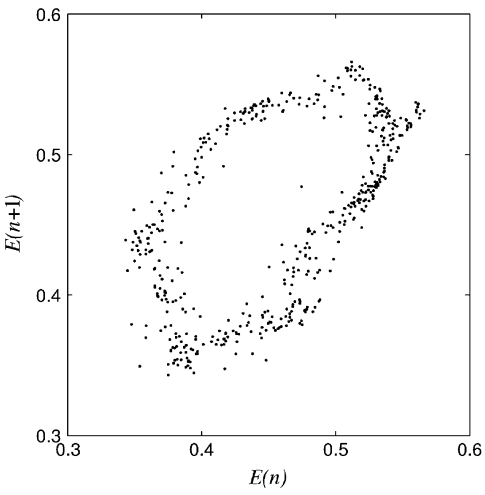


Fig. 9. Chaotic attractor for the classical rate-$1/3$ turbo code, with interleaver of length $k = 1024$. This attractor is observed at the SNR of -6.1 dB.

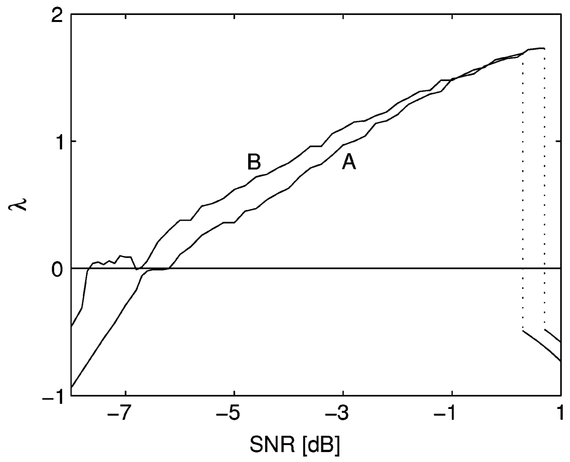


Fig. 10. Largest Lyapunov exponent λ_1 versus SNR for two different noise realizations (curves A and B). Negative value of λ_1 reflects the existence of an (attracting) fixed point, the case $\lambda_1 = 0$ corresponds to (attracting) limit cycle, whereas positive values of λ_1 indicate chaos. Note the abrupt transition of the value (and sign) of λ_1 for high SNR, caused by the unequivocal fixed point.

Several ways (routes) by which this can occur are well documented [20]. These include infinite period-doubling cascades, intermittency, crises, and torus breakdown.

Our analysis indicates that torus breakdown route to chaos is dominant in the turbo-decoding algorithm. This route is evident, for example, in Fig. 8, where a fixed point undergoes a Neimark–Sacker bifurcation giving rise to a periodic orbit (limit-cycle attractor), which then further bifurcates leading to a chaotic attractor. Comparing Fig. 8 with Fig. 1, we see that turbo-decoding algorithm exhibits the same qualitative dynamical behavior (and the same route to chaos) as the two-dimensional map of Example 4, given by (5).

Remark: We found that such quasi-periodic route to chaos is generic for turbo concatenated codes for moderate values of the interleaver length k (on the order of 1000). This is reasonable, since the eigenvalues of a random high-dimensional system are spread throughout the complex plane, with relatively few of them on the real axis. For other values of the interleaver length (or other high-dimensional systems), the route to chaos may be much more complicated, involving multiple Neimark–Sacker, inverse Neimark–Sacker, and other bifurcations, with regions of chaos interspersed with region of quasi-periodicity.

From the schematic bifurcation diagram in Fig. 7, we see that turbo-decoding algorithm exhibits chaotic behavior for a relatively large range of SNRs. An example of a typical chaotic attractor is shown in Fig. 9. The largest Lyapunov exponent of this attractor was computed to be $\lambda_1 = 0.051$. The attractor persists for all values of SNR in the interval (-6.1 dB, 0.5 dB).

In Fig. 10, we have plotted the largest Lyapunov exponent for the natural measure of the turbo-decoding algorithm (starting with initial conditions at the origin), for two different noise realizations. Curve A corresponds to the same noise realization that was used to produce Figs. 8 and 9. Notice that in the SNR region of -6.6 dB to -6.2 dB, the largest Lyapunov exponent is equal to zero, which indicates a limit-cycle attractor.

In the waterfall region (cf. Fig. 6), the turbo-decoding algorithm converges either to the chaotic invariant set or to the unequivocal fixed point, but only after a long transient behavior, which indicates the existence of a chaotic nonattracting invariant set in the vicinity of the fixed point.

In some cases, the algorithm spends several thousand iterations before reaching the fixed-point solution. For example, for the noise realization represented by Curve A in Fig. 10, at the SNR of 0.8 dB, the average chaotic transient lifetime is 378 iter-ations. As the SNR increases, the average chaotic transient lifetime decreases. In Table I, we report the results of the following experiment. For each SNR, we generate 1000 different noise realizations (1000 different parameter frames) and compute the number of decoding trajectories that approach the fixed point in less than a given number of iterations. For example, at the SNR of 0.6 dB, there are 492 frames that converge to the unequivocal fixed point in five or less iterations, another 226 frames converge in 10 or less iterations, and so on, while 58 frames remain chaotic after 2000 iterations (which means that their trajectory either approaches a chaotic attractor or that the transient chaos lifetime is very large).

TABLE I
DISTRIBUTION OF FRAMES THAT CONVERGE TO THE UNEQUIVOCAL FIXED POINT AS A FUNCTION OF THE NUMBER OF ITERATIONS

Number of Iterations	SNR [dB]				
	0.2	0.4	0.6	0.8	1.0
$1 < n \leqslant 5$	60	193	492	772	927
$5 < n \leqslant 10$	101	226	243	152	56
$10 < n \leqslant 20$	49	93	84	35	8
$20 < n \leqslant 50$	51	75	45	12	5
$50 < n \leqslant 100$	49	47	21	5	1
$100 < n \leqslant 200$	30	40	19	4	1
$200 < n \leqslant 500$	40	43	18	5	1
$500 < n \leqslant 1000$	29	29	10	3	0
$1000 < n \leqslant 2000$	39	25	10	2	0
$2000 < n$	552	229	58	10	1

We note that this transient chaos behavior is *characteristic* of the waterfall region of turbo decoding for small to moderate interleaver lengths. In Section VI-A, we show how this fact can be exploited to enhance the performance of turbo codes.

E. Qualitative Description for Small and Large Code Lengths

We found that there are several stark differences between the dynamics of the turbo-decoding algorithm for small interleaver lengths and large interleaver lengths (e.g., $k = 2^{18}$). We now give a qualitative description of these differences.

Although the turbo-decoding algorithm is a high-dimensional dynamical system, it apparently has only a few active variables. That is, out of the total of $2k$ variables, most are "slave" to just a few. In other words, the dynamics of the algorithm can be effectively described by the mapping

$$\boldsymbol{Y}(n+1) = \boldsymbol{F}(\boldsymbol{Y}(n)) \tag{27}$$

where $\boldsymbol{Y}$ is an m-dimensional vector, with $m \ll k$, representing an appropriate combination of the variables $\boldsymbol{X}_1, \boldsymbol{X}_2$ in (20). We believe that $m = 1$ for large k. That is, for $k \to \infty$ turbo decoding can be always described as a one-dimensional map!

In some cases, even for $k = 1024$, we have found that the turbo-decoding algorithm behaves as a one-dimensional system. However, for small k, the number of frames for which the decoding algorithm can be described as one-dimensional map is also small. In contrast, for large k, we found that for *all* frames, the turbo-decoding algorithm is actually a one-dimensional map.

Unfortunately, we do not have a closed analytical form for the function $\boldsymbol{F}$ and/or the variable $\boldsymbol{Y}$ in (27). Nevertheless, we now present a qualitative analysis of the dynamics of the turbo decoder in the case where it can be approximated by a one-dimensional map. To do so, we will use an "equivalent" map, with the average entropy E playing the role of the variable $\boldsymbol{Y}$ in (27). A schematic representation of this map is given in Fig. 11.

We first discuss the case of small k, following the outline of Section IV-D. The decoding algorithm admits two kinds of fixed points: stable indecisive and unstable unequivocal (Fig. 11a). As the SNR grows, indecisive fixed point bifurcates—for example, via the tangent bifurcation. Fig. 11b shows the map immediately after the bifurcation. Observe that the trajectory in Fig. 11b spends a long time in the vicinity of the vanishing fixed point,

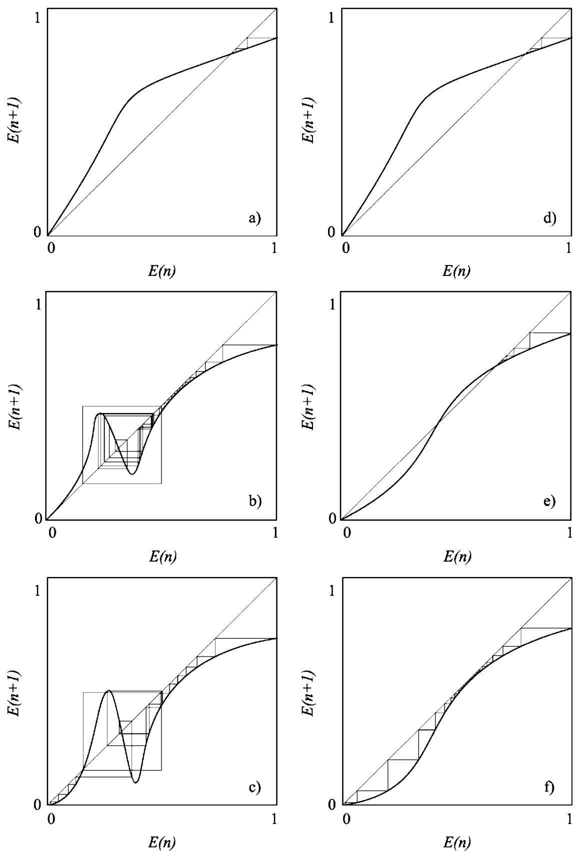


Fig. 11. Schematic one-dimensional map describing the qualitative dynamics of the turbo-decoding algorithm. Graphs a), b), and c) are for small k (e.g., $k = 1024$), while graphs d), e), and f) are for large k (e.g., $k = 2^{18}$). Note that for large k the algorithm does not exhibit chaotic behavior.

before it approaches a chaotic attractor. If we further increase the SNR, the map would look as shown in Fig. 11c. Namely, a typical trajectory spends a long time in the vicinity of the chaotic attractor, eventually escaping this region and converging to a stable unequivocal fixed point.

We next consider the case $k \to \infty$. The corresponding map is depicted in Fig. 11d, e, and f. For low SNRs, this map has two fixed points: the stable indecisive and the unstable unequivocal (see Fig. 11d). The map may have three fixed points at some SNRs, as in Fig. 11e. However, it does *not* exhibit chaotic behavior for *any* SNR. When the SNR exceeds a certain threshold (after the tangent bifurcation), the trajectory of the decodingalgorithm approaches the stable unequivocal fixed point (Fig. 11e). Comparing Fig. 11a, b, c with Fig. 11d, e, f, one can see how the nonlinear map $\boldsymbol{F}$ in (27) transforms as k increases.

In order to further support our argument, we show in Fig. 12 several trajectories of the turbo-decoding algorithm plotted as $E(n+1)$ versus $E(n)$. The trajectory in Fig. 12a approaches the stable indecisive fixed point. A different trajectory is plotted in Fig. 12b and c (with Fig. 12b being a zoom of the upper-right corner of Fig. 12c). This trajectory spends some time in the vicinity of the vanishing fixed point (Fig. 12b), and then approaches a chaotic attractor (Fig. 12 c). The cloud of points in Fig. 12c indicates that either the one-dimensional map has many minima and maxima in this region or that the dynamics of the

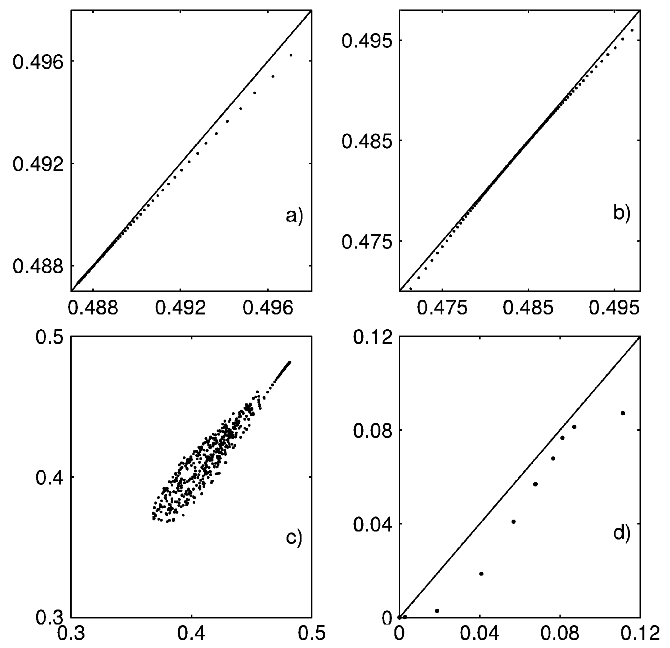


Fig. 12. One-dimensional trajectories of the turbo decoding algorithm: a) The trajectory approaches stable indecisive fixed point at -7.65 dB (for $k = 1024$); b), c) the trajectory in the vicinity of a tangent bifurcation at -7.64 dB (for $k = 1024$); d) the trajectory for $k = 2^{18}$ at 0.5 dB.

algorithm are effectively high dimensional. While $k = 1024$ in Fig. 12a, b, and c, Fig. 12d shows a trajectory of the turbo-decoding algorithm for $k = 2^{18}$. Note the absense of chaotic behavior in this figure.

Remark: The work of ten Brink [5] further supports the conclusion that for $k \to \infty$ the turbo-decoding algorithm can be described as a one-dimensional map. In [5], mutual information transfer characteristics of decoders are proposed as a tool for understanding the convergence behavior of iterative decoding systems. The exchange of extrinsic information is then visualized as a decoding trajectory in the so-called EXIT chart. The analysis of [5] clearly shows that the turbo-decoding algorithm can be approximated as a one-dimensional map, at least for $k \to \infty$. The results of this section can be viewed as an extension of [5] to small k (on the order of 1000), for which the dynamical system at hand is rich in nonlinear phenomena, including chaos.

However, an important difference between the present work and [5] lies in the approach used to arrive at the (same) conclusions. The EXIT chart method of [5] iterates the average mutual information (computed numerically, using Monte Carlo techniques) between the constituent decoders. Herein, we consider the turbo decoder as a dynamical system parameterized by a single parameter (which closely approximates the channel SNR), following the approach developed in [1]. We then use methods rooted in the theory of nonlinear dynamical systems.

Remark: For LDPC codes, an approximate one-dimensional representation of the iterative decoding algorithm for $k \to \infty$ was derived in closed form in [17]. The thresholds obtained with this approximate model are in close agreement with the values computed through the density evolution method of [23] or through Monte Carlo simulations [10].

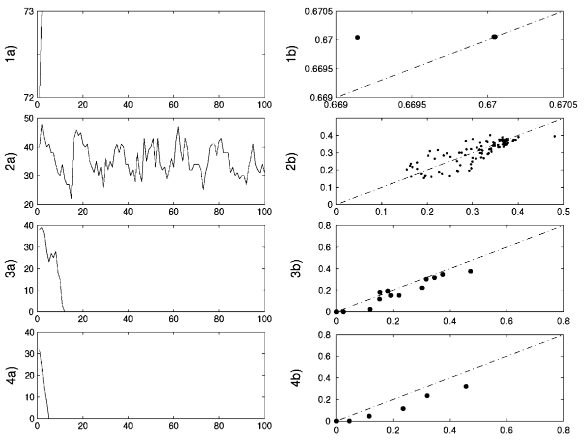


Fig. 13. Iterative decoding trajectories for a serially concatenated turbo code. a) Number of decision errors versus the iteration number. b) $E(n+1)$ versus $E(n)$. Values of SNR are: 1) -10 dB, 2) 0.40 dB, (3) 0.60 dB, and (4) 0.80 dB.

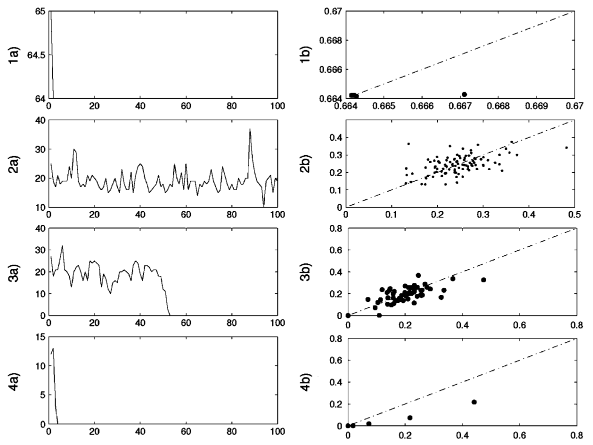


Fig. 14. Iterative decoding trajectories for a serially concatenated turbo code. a) Number of decision errors versus the iteration number. b) $E(n+1)$ versus $E(n)$. Values of SNR are: 1) -10 dB, 2) 0.00 dB, 3) 0.20 dB, and 4) 0.40 dB.

V. Dynamics of Other Iterative Coding Schemes

We now briefly discuss the nonlinear dynamics exhibited by serially concatenated turbo codes, product codes, and LDPC codes (both regular and irregular). We find that these dynamics are, in principle, similar to those of parallel concatenated turbo codes, although there are several important differences.

A. Serially Concatenated Turbo Codes

In serial concatenation [2], a rate-k/N code is obtained by concatenating a rate-k/N' outer code with a rate-N'/N inner code through an interleaver. As a case study, we consider a rate-$1/4$ serially concatenated turbo code made up of two 4-state constituent convolutional codes. Both codes are obtained from the same recursive systematic code (mother code) of rate $1/2$, with parity-check polynomials $\{1+D+D^2, 1+D^2\}$. In our simulations, the frame length was $k = 200$ information bits, and an s-random interleaver of [8] was used.

In Figs. 13 and 14, we have plottted the number of errors in the information bits, as well as $E(n+1)$ as a function of $E(n)$. We show decoder trajectories for two different noise realizations (one in Fig. 13 and the other in Fig. 14) across a range of

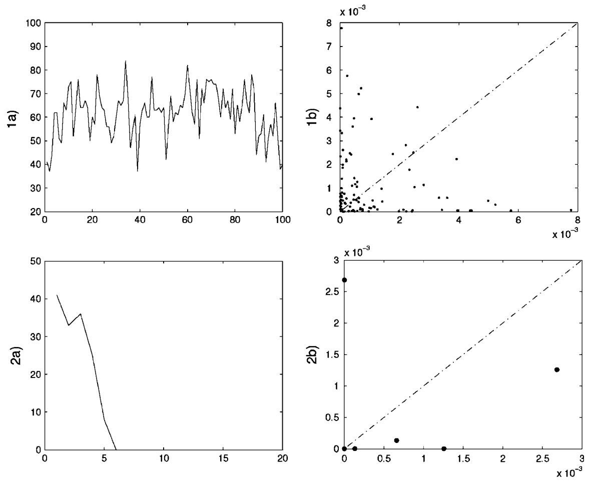


Fig. 15. Iterative decoding trajectories for the $[\mathrm{BCH}\,(32,26)]^2$ product code. a) Number of decision errors versus the iteration number. b) $E(n{+}1)$ versus $E(n)$. Values of SNR are: 1) 1.60 dB and 2) 1.61 dB.

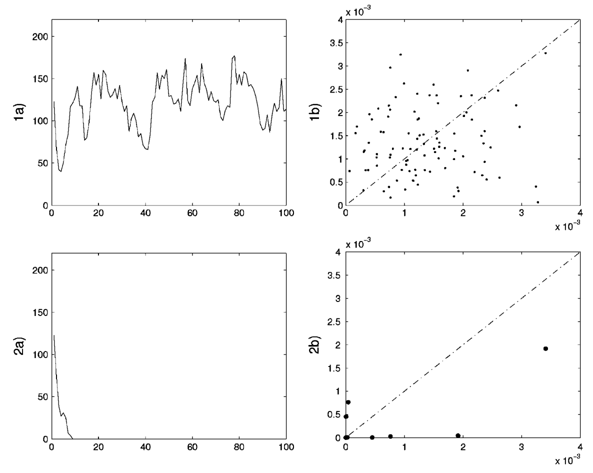


Fig. 16. Iterative decoding trajectories for the $[\mathrm{BCH}\,(64,51)]^2$ product code. a) Number of decision errors versus the iteration number. b) $E(n{+}1)$ versus $E(n)$. Values of SNR are: 1) 2.770 dB and 2) 2.771 dB.

SNRs from -10.0 to 1.0 dB. When the SNR is low, both trajectories converge to an indecisive fixed point: the decoder output is not a valid codeword and the average entropy remains stuck at $E \simeq 0.66$ (Figs. 13.1 and 14.1). When the SNR is sufficiently high, both trajectories converge to an unequivocal fixed point with $E = 0$ (Figs. 13.4 and 14.4). In the intermediate cases, we see that the decoder trajectories may become chaotic or exhibit a chaotic transient.

B. Product Codes

We next consider iterative decoding of the $[\mathrm{BCH}\,(32,26)]^2$ and the $[\mathrm{BCH}\,(64,51)]^2$ product codes. That is, in both cases, the row code and the column code are taken as the same Bose–Chaudhuri–Hocquenghem (BCH) code. As component soft-input soft-output (SISO) decoders, we employ suboptimum soft decoders based on the Chase algorithm [6]. Using the techniques developed in Section IV, we study the trajectories of the resulting iterative decoding algorithm.

As in Section V-A, we plot the number of errors in the information bits as a function of the iteration number n, as well as $E(n{+}1)$ versus $E(n)$. Fig. 15 shows a typical decoding trajectory for the $[\mathrm{BCH}\,(32,26)]^2$ product code. At low SNRs, no indecisive fixed point exists, but the trajectory is already chaotic. At an SNR of 1.60 dB, just before the chaotic attractor looses its stability, $E(n)$ repeatedly reaches a value close to 0 before going back to chaotic behavior. Finally, at an SNR of 1.61 dB, the trajectory converges, after a very short transient, to an unequivocal fixed point. Similar behavior is observed in Fig. 16 for the $[\mathrm{BCH}\,(64,51)]^2$ product code.

C. Regular LDPC Codes

Consider the ensemble of regular $(216,3,6)$ LDPC codes (the SNR threshold [23] for *infinite-length* $(3,6)$ LDPC codes is 1.1 dB). We pick a code at random from this ensemble and study typical trajectories of the iterative decoding algorithm described in Section III-C. We find that, as in the case of parallel concatenated turbo codes, the trajectories of this decoding algorithm exhibit indecisive and unequivocal fixed points, periodic orbits, chaotic invariant sets, and chaotic transients.

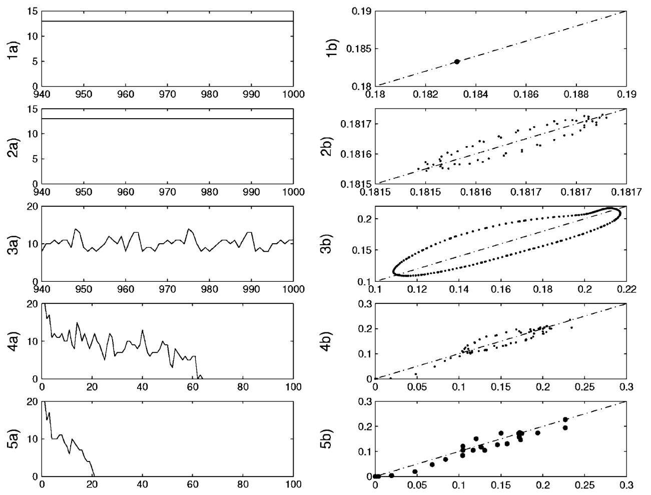


Fig. 17. Iterative decoding trajectories for a $(216,3,6)$ LDPC code illustrating the occurrence of a Neimark–Sacker bifurcation. a) Number of decision errors versus the iteration number. b) $E(n{+}1)$ versus $E(n)$. Values of SNR are: 1) 1.19 dB, 2) 1.20 dB, 3) 1.44 dB, 4) 1.45 dB, and 5) 1.52 dB.

The first example of a typical trajectory is shown in Fig. 17. Fig. 17.1 reveals the existence of a stable indecisive fixed point at low SNRs (the iterates of the number of errors and average entropy are plotted for iterations $n \geq 940$ to skip the initial transient behavior). Fig. 17.2 is characteristic of the Neimark–Sacker bifurcation. The indecisive fixed point becomes unstable at the SNR of 1.20 dB and is surrounded by a small periodic closed orbit. Observe that this does not affect any of the bit decisions, since the variations in the LLRs are not large enough to induce a sign change. Fig. 17.3 shows that if the SNR is further increased to 1.44 dB, the periodic closed orbit becomes larger. Note that, in this case, the number of errors also exhibits a periodic behavior because of the periodic sign changes in the LLRs. When the SNR is increased to 1.45 dB in Fig. 17.4, the trajectory eventually converges to an unequivocal fixed point, with $E \simeq 0$ and correct decisions throughout. Note the presence of a chaotic transient during the first 63 iterations, which indicates the existence of a nonattracting chaotic in-

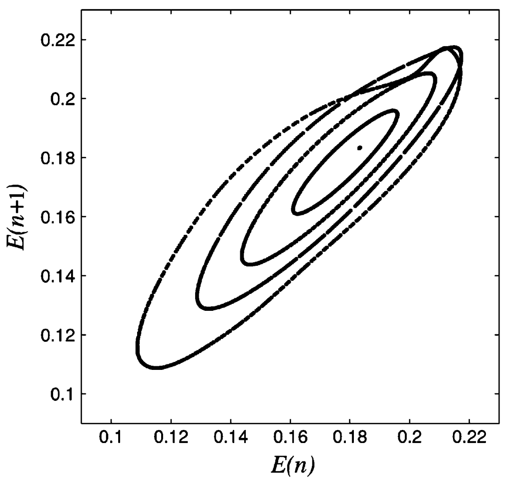


Fig. 18. Torus-breakdown route to chaos for a $(216, 3, 6)$ LDPC code. The values of SNR corresponding to the invariant sets, starting from the fixed point toward chaos, are: 1.19 dB, 1.23 dB, 1.27 dB, 1.33 dB, and 1.44 dB.

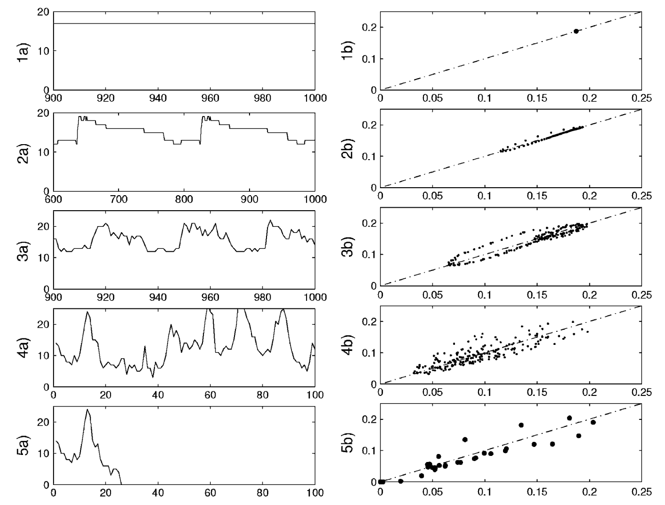


Fig. 19. Iterative decoding trajectories for a $(216,3,6)$ LDPC code illustrating the occurrence of a tangent bifurcation. a) Number of decision errors versus the iteration number. b) $E(n+1)$ versus $E(n)$. Values of SNR are as follows: 1) 0.59 dB, 2) 0.60 dB, 3) 0.76 dB, 4) 1.64 dB, and 5) 1.76 dB.

variant set near the fixed point. A similar behavior is observed at the SNR of 1.52 dB in Fig. 17.5, but the duration of the chaotic transient is reduced to only 21 iterations. Further increasing the SNR reduces the duration of the transient chaos even more until it disappears completely: the average entropy decreases monotonically with the number of iterations. This indicates that the size of the basin of attraction of the unequivocal fixed point grows with SNR.

We find that regular LDPC codes exhibit, qualitatively, the same torus-breakdown route to chaos that was already observed for turbo codes in Section IV-D. This is illustrated in Fig. 18 (which corresponds to the same noise realizations that were used to produce Fig. 17). Comparing Fig. 18 with Fig. 8 and 1 clearly shows similar dynamical behavior.

A second typical trajectory, which is depicted in Fig. 19, is characteristic of a tangent bifurcation. Fig. 19.1 shows a stable indecisive fixed point at an SNR of 0.59 dB. Fig. 19.2 shows the beginning of a bifurcation: the indecisive fixed point disappears at 0.60 dB, and the trajectory converges to a periodic closed orbit. The fact that the closed orbit is tangent to the bisectrix line $E(n+1) = E(n)$ is a remnant of the tangent bifurcation. Observe that the corresponding decision errors are, again, periodic. Figs. 19.3 and 19.4 show a typical torus-breakdown route to chaos, where a closed orbit is gradually transformed into a chaotic attractor. Finally, Fig. 19.5 shows that the chaotic attractor eventually looses its stability: the trajectory then converges to an unequivocal fixed point after a chaotic transient.

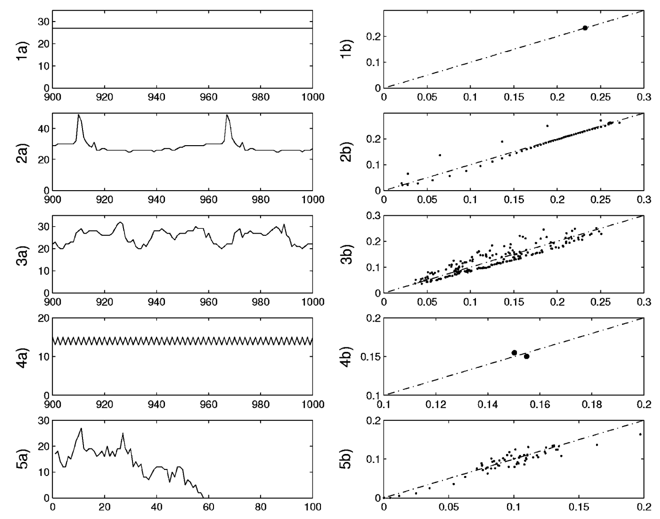


Fig. 20. Iterative decoding trajectories for a $(216, 3, 6)$ LDPC code illustrating the occurrence of a flip bifurcation. a) Number of decision errors versus the iteration number. b) $E(n+1)$ versus $E(n)$. Values of SNR are as follows: 1) -1.1 dB, 2) -1.0 dB, 3) -0.03 dB, 4) 0.79 dB, and 5) 0.87 dB.

The trajectory shown in Fig. 20 is similar to that of Fig. 19. However, the dynamics in Fig. 20 are more complex because of the occurrence of a flip bifurcation. At SNR slightly less than 0.79 dB, the trajectory converges to a stable chaotic attractor. At 0.79 dB, the system exhibits a periodic window, and the trajectory converges to a stable period-2 periodic point. This is manifested by the two points in Fig. 20.4b, alternatively visited by the iterates of the average entropy $E(n)$. The periodic point eventually bifurcates at 0.87 dB and, once again, the trajectory converges to a unequivocal fixed point after a chaotic transient.

D. Irregular LDPC Codes

Now consider the ensemble of irregular $(2000, \lambda(x), \rho(x))$ LDPC codes with degree distributions

$$\begin{aligned}\rho(x) &= 0.24123x^4 + 0.75877x^5\\ \lambda(x) &= 0.38354x + 0.04237x^2 + 0.57409x^3.\end{aligned} \tag{28}$$

These polynomials were optimized using the density-evolution techniques of [22]. The infinite-length SNR threshold for the resulting ensemble is 0.8085 dB.

One of the primary differences between regular and irregular LDPC codes is that the irregular codes exhibit multiple fixed-points at low SNRs, in contrast to the unique indecisive fixed point observed in all prior cases (cf. [1]). This is not surprising,

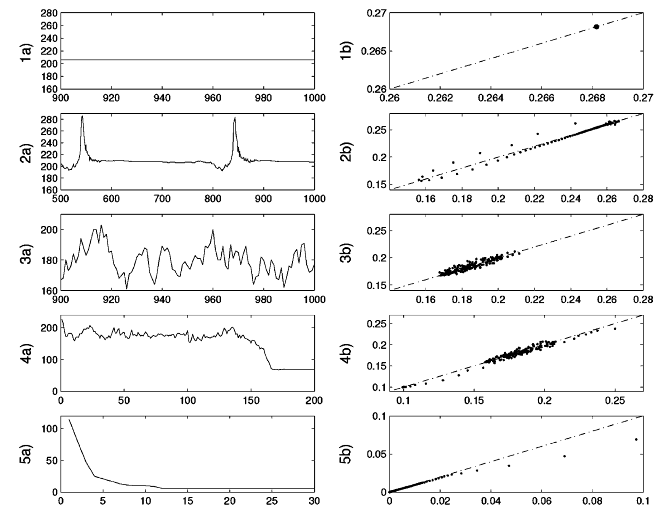


Fig. 21. Iterative decoding trajectories for an irregular LDPC code of (28), undergoing a tangent bifurcation. a) Number of decision errors versus the iteration number. b) $E(n+1)$ versus $E(n)$. Values of SNR are: 1) 0.32 dB, 2) 0.34 dB, 3) 0.70 dB, 4) 0.72 dB, and 5) 2.48 dB.

since the density evolution approach of [22] exhibits a similar behavior for infinite-length irregular LDPC codes.

Fig. 21 illustrates this phenomenon. Specifically, Fig. 21.1 and 21.2 shows the disappearance of an indecisive fixed point via a tangent bifurcation and the formation of a closed periodic orbit. As the SNR increases, a torus-breakdown route to chaos takes place until the chaotic attractor looses its stability at about 0.72 dB (see Fig. 21.3 and 21.4). The trajectory then converges to another indecisive fixed point with $E \approx 0.1$. If the SNR is further increased, the new fixed point gradually moves down and the corresponding trajectory becomes tangent to the bisectrix line $E(n+1) = E(n)$, as shown in Fig. 21.5. Eventually, an unequivocal fixed point ($E \simeq 0$) is reached at a sufficiently high SNR. Nevertheless, the number of decision errors remains at an "error floor" of six residual errors.

VI. Applications

This section is devoted to applications of the findings from the nonlinear dynamical analyses in Sections IV and V. Once again, we use parallel concatenated turbo codes as a case study.

A. Ultra-Fast Convergence

We consider an application of nonlinear control theory [28], [7] in order to speed up the convergence of the turbo decoding algorithm. We have developed a simple adaptive control mechanism to reduce the long transient behavior in the algorithm. A block diagram of the turbo decoder *with adaptive control* is depicted in Fig. 22. Our control function $g(\,\cdot\,)$ is given by

$$g(\boldsymbol{X}_i) = \alpha \boldsymbol{X}_i e^{-\beta|\boldsymbol{X}_i|} \tag{29}$$

where $\boldsymbol{X}_1, \boldsymbol{X}_2$ are the extrinsic information variables in (20), while α and β are parameters. In simulations, we have used $\alpha = 0.9$ and $\beta = 0.01$, although similar results were obtained with other values of $\alpha \in [0.8, 1]$ and $\beta \in [0.001, 0.01]$.

We point that the adaptive control algorithm of Fig. 22 and (29) is very simple, and can be readily implemented (either in software and/or in hardware) without significantly increasing the complexity of the decoding algorithm.

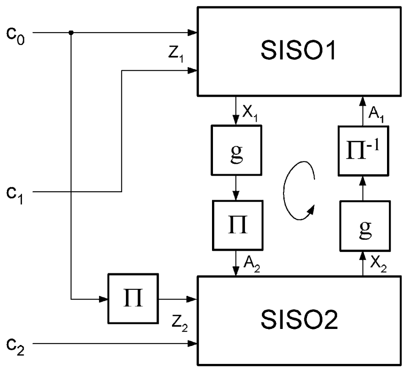


Fig. 22. Block diagram of the turbo decoder with control of the transient chaos. The control function is given by $g(\boldsymbol{X}_i) = \alpha \boldsymbol{X}_i e^{-\beta|\boldsymbol{X}_i|}$, where α and β are appropriately chosen real constants. Conventional turbo codes correspond to the identity control function $g(\boldsymbol{X}_i) = \boldsymbol{X}_i$ (see Fig. 5).

The intuition behind our control strategy can be explained as follows. Let us write $g(\boldsymbol{X}_i)$ as

$$g(\boldsymbol{X}_i) = (g_{i1}X_{i1}, g_{i2}X_{i1}, \ldots, g_{ik}X_{ik}) \tag{30}$$

where g_{ij} are the corresponding attenuation/gain factors. The probability that the jth information bit is zero is can be written as $p_j^0 = 1/(1+e^{L_j})$, where $L_j = X_{1j} + X_{2j} + \frac{4}{\sigma^2}c_j^0$. If L_j is small, then the attenuation factor g_{ij} in (30) is close to 1 (since α is close to 1 and β is small). In other words, the control algorithm does nothing. If, however, L_j is large, then the control algorithm reduces the value of g_{ij}, thereby attenuating the effect of X_{ij} on the decoder. If the jth bit is a part of a valid codeword, the control algorithm does not affect the decoding: the turbo decoding algorithm makes a decision for the jth bit with probability close to 1 with or without control. However, if the jth bit is not a part of a valid codeword and the turbo decoding algorithm "struggles" to find the valid codeword, the attenuation effect of g_{ij} helps a great deal in reducing the long transient behavior. We found that the average chaotic transient lifetime with control is only nine iterations, as compared to about 350 iterations without control.

The performance of our control strategy in (29) is reported in Fig. 23. On average, turbo decoding with control exhibits a gain of 0.25 to 0.3 dB over the conventional turbo-decoding al-gorithm. Note that the turbo-decoding algorithm with control, stopped after eight iterations, shows better performance than the conventional turbo-decoding algorithm stopped after 32 iterations. Thus, adaptive control produces an algorithm that is four times faster, while providing about 0.2-dB gain over the conventional turbo-decoding algorithm. On the other hand, we can see from Fig. 23 that control is not very effective in the error-floor region. This is to be expected since the iterative decoding process does not exhibit transient chaos in this region.

The error frame statistics with and without control, as a function of SNR, are reported in Fig. 24. The simulation results corresponding to the application of the adaptive control method, for the case of the Max-Log-MAP algorithm, are shown in Fig. 25 as a function of the number of iterations. From this figure, one can see the coding gain due to adaptive control, as the number

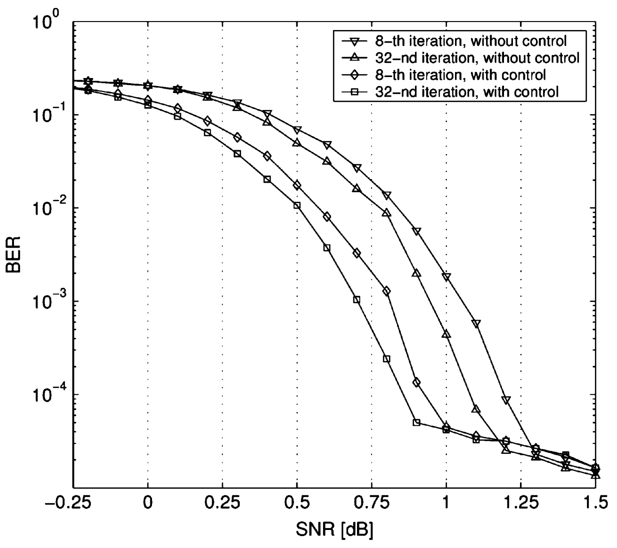


Fig. 23. Performance of the classical rate-$1/3$ parallel concatenated turbo code for interleaver length 1024, with/without control of the transient chaos.

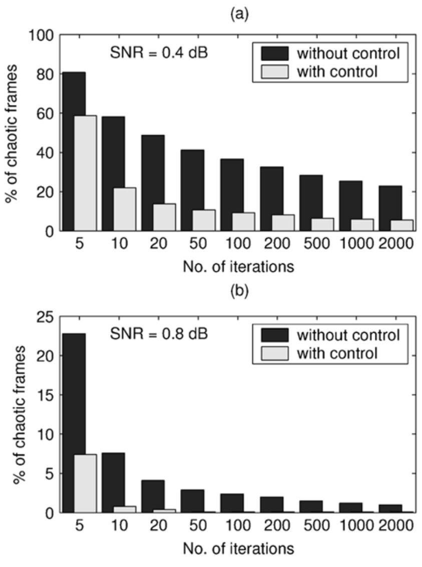


Fig. 24. Histogram showing the number of frames which remain chaotic after a certain number of iterations, with and without control of the transient chaos.

of iterations progresses. After $n = 8$ iterations, the turbo-decoding algorithm with control exhibits an average gain of about 0.2 dB versus the case without control.

B. New Stopping Criteria

In this subsection, we propose novel stopping rules for parallel concatenated turbo codes and compare these rules with the current state of the art. It is well known [1], [24], [25], [31], [32] that rather than performing a fixed number of iterations for each transmitted frame, one can stop the iterative decoding algorithm on a frame-by-frame basis using a properly defined stopping rule. In principle, this allows to reduce the average number of iterations and, consequently, save computation.

In what follows, we develop two stopping rules based upon the *a posteriori* average entropy E, as defined in (24) of Section IV-C. These rules complement each other in trying to identify as early as possible situations where the decoding algorithm

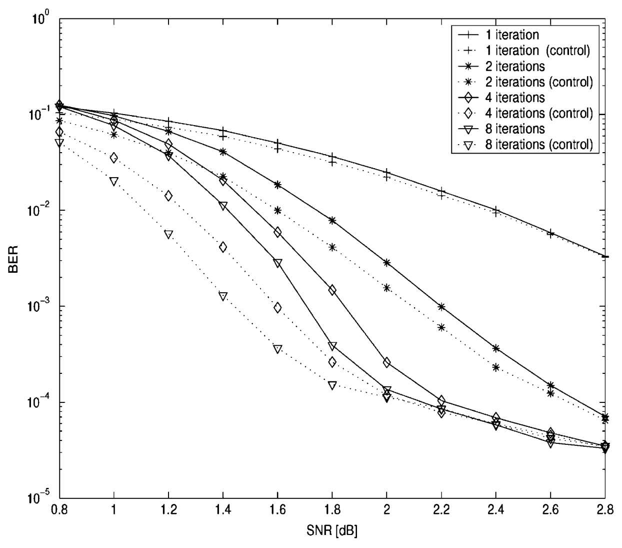


Fig. 25. Performance of the classical turbo code for interleaver length 1024, using Max-Log-MAP algorithm, with/without control of the transient chaos.

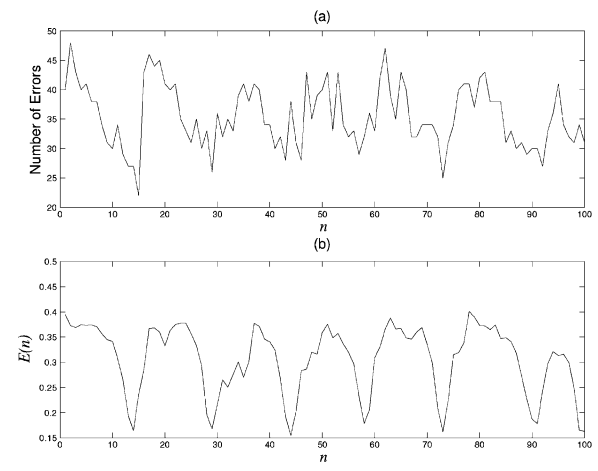


Fig. 26. Typical chaotic trajectories for the classical turbo decoder of [4] at an SNR of 0.0 dB. (a) Number of decision errors versus n. (b) $E(n)$ versus n.

either has reached a fixed point or, most likely, it will never reach one in a reasonable number of iterations.

1) Zero-Entropy and Sign-Entropy-Derivative Detection: Our first stopping rule is based simply on monitoring the iterates of $E(n)$. When the decoding algorithm converges to an unequivocal fixed point, the *a posteriori* entropy of the system tends to zero. Therefore, we fix a small threshold E_{th} and stop the algorithm whenever $E(n) \leq E_{\text{th}}$. We call the corresponding stopping rule ZED (zero-entropy detection).

In order to avoid unnecessary iterations when the algorithm exhibits chaotic behavior, we can stop the iterations as soon as we recognize this behavior. Fig. 26 shows typical chaotic trajectories of a (classical) turbo decoder. From these trajectories, it can be evinced that an appropriate stopping criterion may be based upon detecting abrupt changes in the slope of entropy evolution. Thus, we also propose a stopping rule called SEDD (sign-entropy-derivative detection), which stops the iterations whenever the derivative of the entropy changes its sign more than N_s times, where N_s is a small constant.

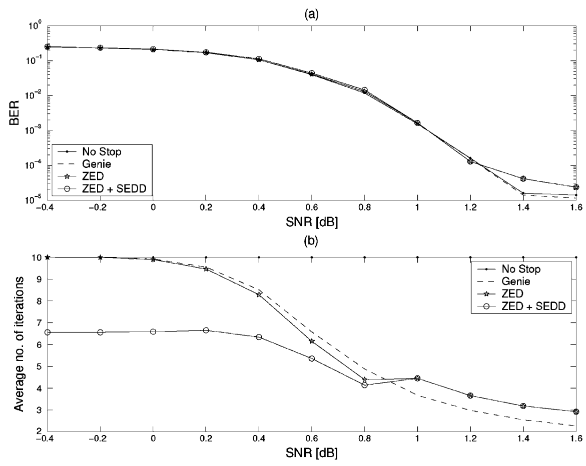


Fig. 27. Performance of the classical turbo code of [4] with the proposed ZED and SEDD stopping criteria, but without control of the transient chaos. (a) BER versus SNR. (b) Average number of iterations versus SNR.

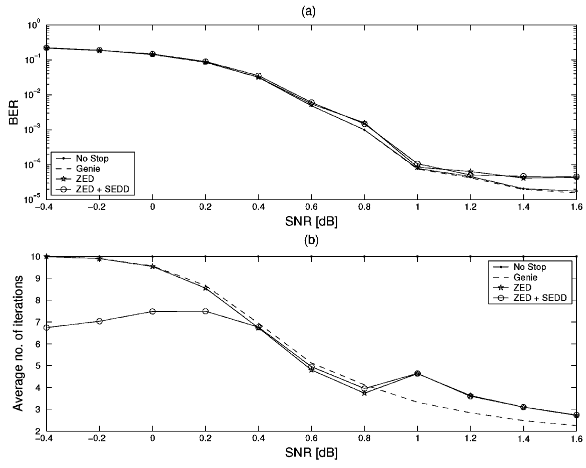


Fig. 28. Performance of the classical turbo code of [4] with the proposed ZED and SEDD stopping criteria and control of the transient chaos. (a) BER versus SNR. (b) Average number of iterations versus SNR.

As a comparison benchmark, we will use the so-called *genie stopping rule*. This rule is based on the fact that the best way to determine whether the decoder has reached a fixed point consists of evaluating the number of residual decision errors after each iteration. If this number is zero, we stop the process, otherwise we let the decoder reach the maximum allowed number of iterations. With this method, one obtains the performance of the decoder that always reaches the maximum number of iterations, while effectively lowering the average number of iterations. Of course, in order to implement this stopping rule, one needs to know the information bits or, indeed, be a genie.

In Figs. 27 and 28, we see the results of implementing both ZED and SEDD for the classical rate-$1/3$ parallel concatenated turbo code of [4], with and without control of the transient chaos. To optimize the performance, we devised an adaptive stopping strategy as a function of SNR. For the ZED stopping rule, we set the threshold at $E_{\rm th} = 0.06$ for SNRs less than 0.8 dB. In this range of SNRs, we essentially match the genie performance, in terms of both bit-error rate (BER) and the average number of iterations. For SNRs higher than 0.8 dB, we reduce the threshold to $E_{\rm th} = 0.04$, because we are entering the error-floor region, where most of the erroneous frames contain only a few decision errors (after, say, 10 iterations). Thus, we need to lower the threshold in order to distinguish this type of frames from the error-free frames, for which $E = 0$.

Figs. 27 and 28 also show the results of combining the two stopping criteria: ZED + SEDD. The threshold value for SEDD was $N_s = 2$ in all cases. As expected, we found that SEDD works especially well for low SNRs, where a lot of frames exhibit chaotic behavior. In contrast, ZED is more effective in the waterfall and the error-floor regions. Comparing the performance of SEDD in the two figures, we see that the average number of iterations with control is higher than the one without control, especially in the neighborhood of the waterfall region. Indeed, control of the transient chaos helps some frames with long transient behavior converge to the unequivocal fixed point Notably, our stopping criterion does not stop this decoding process, thus maintaining the gains in terms of BER.

2) Comparison With Current State of the Art: We now compare the performance of the proposed stopping criteria with the current state of the art. First, let us recall the some of the main methods in the literature [1], [14], [24], [25], [31], [32] for stopping the iterations of an iterative decoder.

Mean Reliability (**Mean**). This stopping criterion, described in [32], is based upon computing, after each iteration n, the average of the absolute values of the LLRs, namely, the quantity

$$A(n) \stackrel{\rm def}{=} \frac{1}{k}\sum_{j=1}^{k} |L_j(n)|.$$

The iterative decoding process is stopped after iteration n, if $A(n) \geq \theta_A$, where θ_A is a fixed threshold.

Minimum LLR (**MIN**). This stopping criterion [29] is based upon evaluating the minimum LLR, namely

$$B(n) \stackrel{\rm def}{=} \min_{1\leq j\leq k} |L_j(n)|.$$

The iterative decoding process is stopped after iteration n, if $B(n) \geq \theta_B$, where θ_B is a fixed threshold.

Sum-Reliability (**Sum**). The sum criterion [14] is based upon evaluating the sum of the absolute values of the LLRs

$$C(n) \stackrel{\rm def}{=} \sum_{j=1}^{k} |L_j(n)|.$$

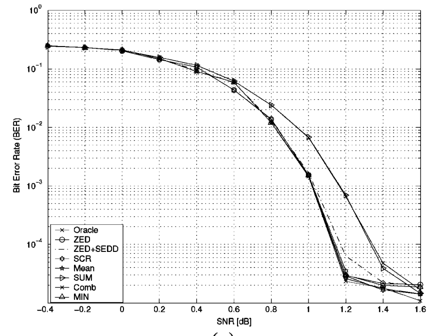
The iterative decoding process is stopped after iteration n, if $C(n) - C(n{-}1) \leq 0$.

Combined Minimum LLR and Sum-Reliability (**Comb**). This stopping criterion [14] is a straightforward combination of the two stopping criteria above. The iterative decoding process is stopped after iteration n, if

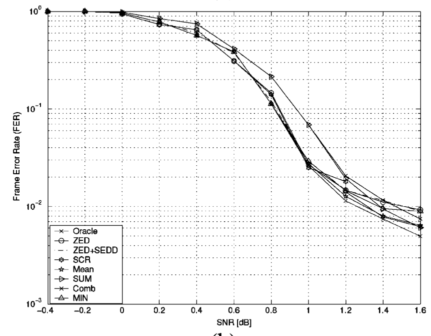
$$C(n) - C(n{-}1) \leq 0 \text{ or } B(n) \geq \theta_B.$$

Sign-Change Ratio (**SCR**). The SCR criterion of [24] evaluates, after each iteration n, the sign-change decision function $f_n(j)$ defined as follows:

$$f_n(j) \stackrel{\rm def}{=} \begin{cases} 1, & \text{if } \operatorname{sgn}[X_{2,j}(n)] \neq \operatorname{sgn}[X_{2,j}(n{-}1)] \\ 0, & \text{if } \operatorname{sgn}[X_{2,j}(n)] = \operatorname{sgn}[X_{2,j}(n{-}1)] \end{cases}$$



(a)



(b)

Fig. 29. Performance comparison of the different stopping criteria for turbo codes, in terms of (a) BER and (b) FER versus SNR.

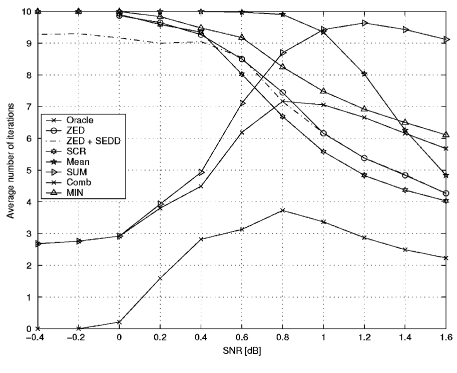


Fig. 30. Performance comparison of the different stopping criteria for turbo codes, in terms of the average number of iterations as function of SNR.

for all $j \in \{1, 2, \ldots, k\}$. The decoder then computes the average sign-change ratio, given by

$$D(n) \overset{\text{def}}{=} \frac{1}{k} \sum_{j=1}^{k} f_n(j).$$

The iterative decoding process is stopped after iteration n, if $D(n) \leq \theta_D$, where θ_D is a fixed threshold.

In Fig. 29, we compare the performance of the proposed stopping rules ZED and ZED + SEDD, in terms of both BER and FER (frame-error rate), with the performance of the five stopping criteria above. For the Mean, SCR, MIN, and Comb stopping rules, the value of the threshold θ has been optimized for best performance. Fig. 30 reports our simulation results for all the stopping criteria in terms of the average number of iterations. We have also included in Figs. 29 and 30 the so-called "oracle" stopping rule. This hypothetical rule consists of the following: if a frame cannot be correctly decoded using the maximum number of iterations (10 in our case), then no iterations at all are counted in Fig. 30; otherwise, we count the *minimum* number of iterations required to decode this frame.

It can be observed from Fig. 29 that the performance of the ZED and ZED $+$ SEDD stopping rules is comparable to that of the best known stopping criteria. In fact, all the stopping rules we have considered, except for Sum and Comb, result in about the same BER over a wide range of SNRs. The performance of the Sum and Comb stopping rules is noticeably (by about 0.2 dB) inferior. Discounting the Sum and Comb stopping rules, it can be seen from Fig. 30 that the ZED $+$ SEDD stopping criterion results in the lowest number of iterations, especially for low values of SNR. For high values of SNR, we are doing just slightly worse than the best (SCR) of the five stopping rules we have chosen as a comparison benchmark.

VII. Conclusion

Iterative decoding algorithms can be viewed as high-dimensional dynamical systems parameterized by a large number of parameters. We have introduced a simplified description of iterative decoding for several turbo-concatenated codes and for LDPC codes. Using this model, we have shown that a whole range of phenomena known to occur in nonlinear systems, including the existence of multiple fixed points, oscillatory behavior, bifurcations, chaos, and transient chaos, are found in iterative decoding algorithms. The observed behavior depends (apart from SNR) on the particular noise realization.

As an application of the theory developed here, we have devised a simple adaptive technique to control transient chaos (characteristic of the waterfall region) in the turbo-decoding algorithm. This results in an ultra-fast convergence and significant performance gains. Finally, we have proposed a novel stopping criterion for turbo codes, based on the average entropy of an information block. This stopping criterion is shown to reduce the average number of iterations and to benefit from the use of our adaptive control technique.

Acknowledgment

The authors are grateful to the anonymous referees, whose comments have improved the presentation of this work.

References

[1] D. Agrawal and A. Vardy, "The turbo decoding algorithm and its phase trajectories," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 699–722, Feb. 2001.

[2] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 909–926, May 1998.

[3] C. Berrou, "The ten-year-old turbo codes are entering into service," *IEEE Commun. Mag.*, vol. 41, pp. 110–116, Aug. 2003.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE Int. Conf. Communications*, Geneva, Switzerland, 1993, pp. 1064–70.

[5] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[6] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 1, pp. 170–182, Jan. 1972.

[7] M. Dhamala and Y.-C. Lai, "Controlling transient chaos in deterministic flows with applications to electrical power systems and ecology," *Phys. Rev. E*, vol. 59, pp. 1646–1655, Feb. 1999.

[8] S. Dolinar and D. Divsalar, "Weight distribution for turbo codes using random and nonrandom permutations," *JPL Progr. Rep.*, pp. 42–122, Aug. 1995.

[9] L. Duan and B. Rimoldi, "The iterative turbo decoding algorithm has fixed points," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2993–2995, Nov. 2001.

[10] H. El Gamal and A. R. Hammons Jr., "Analyzing the turbo decoder using the Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 671–686, Feb. 2001.

[11] P. Elias, "Error-free coding," *IRE Trans. Inf. Theory*, vol. IT-4, no. 4, pp. 29–37, Sep. 1954.

[12] M. Fu, "Stochastic analysis of turbo decoding," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 81–100, Jan. 2005.

[13] R. G. Gallager, "Low-density parity check codes," *IEEE Trans. Inf. Theory*, vol. IT-8, no. 1, pp. 21–28, Jan. 1962.

[14] F. Gilbert, F. Kienle, and N. Wehn, "Low complexity stopping criteria for UMTS turbo-decoders," in *Proc. 57th IEEE Vehicular Technology Conf.*, Cheju, Korea, Apr. 2003, pp. 2376–2380.

[15] L. Kocarev, Z. Tasev, and A. Vardy, "Improving turbo codes by control of transient chaos in turbo-decoding algorithms," *Electron. Lett.*, vol. 38, pp. 1184–1186, Sep. 2002.

[16] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[17] F. Lehmann and G. M. Maggio, "An approximate analytical model of the message passing decoder of LDPC codes," in *Proc. IEEE Int. Symp. Information Theory*, Lausanne, Switzerland, Jun./Jul. 2002, p. 31.

[18] A. J. Lichtenberg and M. A. Lieberman, *Regular and Chaotic Dynamics*. New York: Springer-Verlag, 1992.

[19] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[20] E. Ott, *Chaos in Dynamical Systems*. New York: Cambridge Univ. Press, 1993.

[21] T. J. Richardson, "The geometry of turbo decoding dynamics," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 9–23, Jan. 2000.

[22] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[23] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[24] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117–1120, Aug. 1999.

[25] A. Shibutani, H. Suda, and F. Adachi, "Reducing average number of turbo decoding iterations," *Electron. Lett.*, vol. 35, pp. 701–702, 1999.

[26] S. H. Strogatz, *Nonlinear Dynamics and Chaos*. Cambridge, MA: Perseus, 1994.

[27] Z. Tasev, P. Popovski, G. M. Maggio, and L. Kocarev, "Bifurcations and chaos in turbo decoding algorithms," in *Chaos and Bifurcation Control: Theory and Applications (Lecture Notes in Control Information Sciences)*. New York: Springer-Verlag, 2003.

[28] T. Tel, "Controlling transient chaos," *J. Phys., Ser A: Mathematical and General*, vol. 24, pp. L1359–L1368, Dec. 1991.

[29] Z. Wang and K. K. Parhi, "Decoding metrics and their applications in VLSI turbo decoders," in *Proc. IEEE Conf. Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, Jun. 2000, pp. 3370–3373.

[30] S. Wiggins and F. John, *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. New York: Springer-Verlag, 1990.

[31] Y. Wu, B. D. Woerner, and W. J. Ebel, "A simple stopping criterion for turbo decoding," *IEEE Commun. Lett.*, vol. 4, no. 8, pp. 258–260, Aug. 2000.

[32] F. Zhai and I. J. Fair, "Techniques for early stopping and error detection in turbo decoding," *IEEE Trans. Commun.*, vol. 51, no. 10, pp. 1617–1623, Oct. 2003.