

Published in IET Control Theory and Applications  
 Received on 12th March 2008  
 Revised on 26th August 2008  
 doi: 10.1049/iet-cta:20080090



ISSN 1751-8644

# Bifurcations in iterative decoding and root locus plots

*C.M. Kellett S.R. Weller*

*School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan NSW 2308, Australia  
 E-mail: chris.kellett@newcastle.edu.au*

**Abstract:** A class of error correction codes called ‘low-density parity-check (LDPC)’ codes have been the subject of a great deal of recent study in the coding community as a result of their ability to approach Shannon’s fundamental capacity limit. Crucial to the performance of these codes is the use of an ‘iterative’ decoder. These iterative decoders are effectively high-dimensional, nonlinear dynamical systems and, consequently, control-theoretic tools are useful in analysing such decoders. The authors describe LDPC codes and the decoding algorithm and make a connection between the fixed points of the decoding algorithm and the well-known root locus plot. Through two examples of LDPC codes, the authors show that the root locus plot visually captures the bifurcation behaviour of iterative decoding that occurs at the signal-to-noise ratio predicted by Shannon’s noisy channel coding theorem.

## 1 Introduction

Error correction codes introduce redundancy into a digital sequence to allow communication receivers to correct for errors introduced by transmitting information over a noisy channel. Since Shannon first characterised the capacity of a communications channel in his seminal 1948 paper [1], until the early 1990s, decoding of error correction codes consisted almost entirely of algebraic methods. In particular, codewords are formed from strings of symbols chosen from a finite field and decoding consists of finding the codeword closest to the received word in some appropriate metric (see [2] and the references therein).

The efficacy of these methods is demonstrated by their widespread adoption throughout digital communications, from compact discs to mobile phones to deep space communications. Nonetheless, despite over 40 years worth of research, as of the early 1990s, the best error correction codes still remained over 3 dB away from the fundamental capacity limit described by Shannon in 1948. That changed in 1993 with the description of the turbo decoding algorithm by Berrou, Glavieux and Thitimajshima [3], which combined what would have been considered two relatively weak codes via a large interleaver and, most importantly, an ‘iterative’ decoding algorithm of manageable

complexity. Along with the rediscovery of Gallager’s low-density parity-check (LDPC) codes [4] by MacKay and Neal [5], iterative decoding has pushed digital communications performance to within a fraction of a decibel of Shannon’s fundamental limit [6]. However, basic questions about convergence and robustness properties of these algorithms remain unanswered.

A small handful of authors have studied the turbo decoding algorithm as a nonlinear dynamical system [7–9]. A common difficulty in these approaches is necessarily high order of the dynamical systems. However, by considering LDPC codes (under assumptions described in Section 3), the action of the decoding algorithm can be described by a one-dimensional dynamical system with a specific structure. This structure allows us to use the popular root locus plot to examine how fixed points vary as a function of the channel error parameter. In the following section, we describe a class of systems for which the root locus plot provides a useful graphical description of the fixed points. The purpose of this paper is to briefly describe a novel application area for control theoretical tools. Towards the end, in Section 3, we describe how a one-dimensional discrete-time system arises from iterative decoding. We present two specific codes and discuss what information can be gleaned from the root locus plot.

## 2 Root locus plots and fixed points

The root locus plot, originally introduced by Evans [10, 11], has long been a popular control design tool for single-input single-output linear time-invariant systems. This is due to the fact that it is possible to derive simple rules for quickly graphing how the location of closed-loop system poles varies as a function of controller gain. In particular, if we let  $s \in \mathbb{C}^2$  denote the Laplace variable, for an open-loop transfer function  $F(s)$ , the (negative feedback) closed-loop transfer function of Fig. 1 is given by

$$\frac{KF(s)}{1 + KF(s)} \quad (1)$$

where  $K \in \mathbb{R} > 0$ . Bounded-input bounded-output stability is therefore dependent on the location of the roots of  $1 + KF(s) = 0$ . Rewriting the transfer function  $F(s)$  in terms of its numerator and denominator polynomials, here denoted by  $n(s)$  and  $d(s)$ , respectively, the roots of  $1 + KF(s) = 0$  are equivalent to the roots of

$$d(s) + Kn(s) = 0 \quad (2)$$

Now consider the one-dimensional discrete-time dynamical system described by

$$x_{k+1} = \epsilon p_1(x_k) + p_2(x_k) \quad (3)$$

where  $p_1, p_2: \mathbb{R} \rightarrow \mathbb{R}$  are polynomials and  $\epsilon \in \mathbb{R}_{\geq 0}$ . It is possible to consider  $\epsilon < 0$ , but, for the sake of simplicity, we restrict our attention to  $\epsilon \geq 0$ .

Our interest is in how the equilibria of (3) vary as a function of the parameter  $\epsilon$ . We observe that the condition for an equilibrium of (3) is given by

$$\epsilon p_1(x) + p_2(x) = x \quad (4)$$

which we may rewrite as

$$-p_1(x) + \frac{1}{\epsilon}(p_2(x) - x) = 0 \quad (5)$$

which we recognise as (2). Therefore under the assumption that the degree of  $p_1(x)$  is greater than (or equal to) the degree of  $p_2(x) - x$ , we may use the root locus plot to visualise how fixed points vary as a function of  $\epsilon$ . In the context of stability for linear time-invariant (LTI) systems, an obvious (minimum) requirement is that all poles lie in the left-half

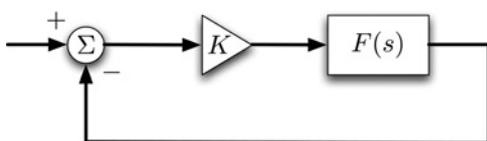


Figure 1 Negative feedback loop

plane. In the context of studying fixed points, however, the interest is in when (5) will have real roots, that is, we are interested in the portions of the locus which lie on (or near, as we shall see in the sequel) the real axis.

*Example 1:* One well-studied system of the form (3) is the so-called 'logistic map', given by

$$x_{k+1} = \epsilon x_k(1 - x_k) \quad (6)$$

The fixed point condition is then simply

$$(x^2 - x) + \frac{1}{\epsilon}x = 0 \quad (7)$$

The root locus plot for this system (Fig. 2) is quite simple as it consists of a pole/zero pair at the origin and a single other root at  $x = 1 - (1/\epsilon)$ .

This example points up two shortcomings of the root locus when considering fixed points for one-dimensional discrete-time dynamical systems. The first is that the root locus tells us nothing about the stability properties of the fixed points. The second shortcoming is that the root locus plot does not predict bifurcations or periodic orbits.

For example, the fixed point at zero is locally asymptotically stable for  $\epsilon \in [0, 1)$ , but unstable for  $\epsilon > 1$ . Similarly, the fixed point at  $1 - (1/\epsilon)$  is locally asymptotically stable for  $\epsilon \in (1, 3)$ . However, there is nothing remarkable about the fixed point when viewed on the root locus plot. Furthermore, as shown in the bifurcation diagram of Fig. 3, as  $\epsilon$  continues to increase, the fixed point at  $1 - (1/\epsilon)$  undergoes a bifurcation, becoming unstable and creating a period-two orbit. Through period-doubling, this period-two orbit eventually gives rise to chaotic behaviour. Again, this is not captured by the root locus plot. Nonetheless, as we will see in the sequel, the root locus plot provides insight into the behaviour of iterative decoding of error-correction codes. □

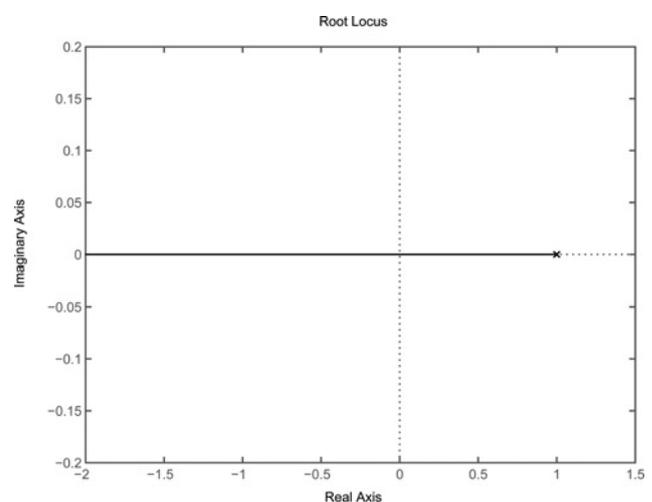


Figure 2 Root locus for fixed points of the logistic map

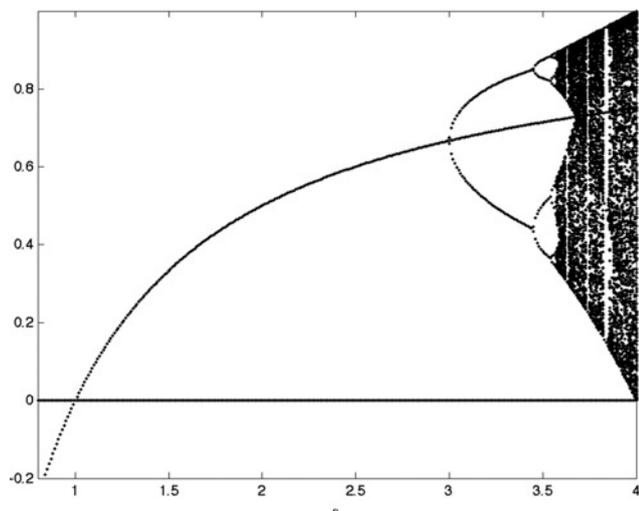


Figure 3 Bifurcation diagram for the logistic map

### 3 Iterative decoding

The type of error-correction codes we consider in this paper are so-called 'low-density parity-check (LDPC)' codes. LDPC codes were first proposed by Gallager in the early 1960s [4], but available computing power at the time was insufficient to demonstrate the effectiveness of such codes which rely on an iterative decoder to achieve their impressive performance characteristics. However, following the publication of the turbo decoding algorithm [3], such codes were rediscovered in the late 1990s by MacKay and Neal [5] and have subsequently been the subject of a great deal of research. Excellent sources for the material in this section include [12–16].

Parity-check codes rely on parity-check equations to verify correctness of a received codeword. For instance, suppose we wish to transmit two information bits {10}. Using an 'even parity' constraint, we require that the sum of the digits in a codeword, modulo 2, be zero. In other words, for the simple case above, we would append the parity bit 1, yielding the codeword {101}. Consequently, if we transmit {101} and at most one bit is changed because of the effects of noise in the channel, this error can be detected. Note that, in this case, the error cannot be corrected and, furthermore, if two bits are changed, we incorrectly decode the transmitted symbol. Error correction is possible, however, by appending extra parity bits.

Parity-check codes have a convenient graphical representation, called the Tanner graph, which is particularly useful for understanding iterative decoding algorithms. The Tanner graph is a bipartite graph with the vertex sets corresponding to bits (denoted by circles in Fig. 4) and parity-check equations (denoted by squares in Fig. 4), respectively. Edges connect bit vertices with the parity-check vertices corresponding to the parity-check equations in which the bits are involved.

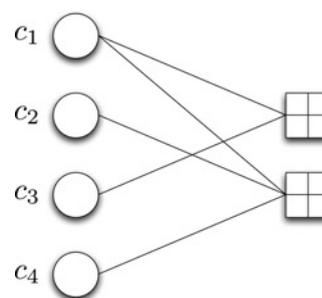


Figure 4 Tanner graph example

Consider again our desire to transmit two information bits  $c_1$  and  $c_2$ . We can define two parity bits,  $c_3$  and  $c_4$  via the parity-check equations

$$\begin{aligned} c_1 \oplus c_3 &= 0 & \text{and} \\ c_1 \oplus c_2 \oplus c_4 &= 0 \end{aligned} \quad (8)$$

where  $\oplus$  denotes modulo 2 addition. The Tanner graph corresponding to this code is shown in Fig. 4. Note that the vertex corresponding to  $c_1$  is connected to both check equations, whereas all other bit vertices are only connected to a single parity-check vertex. Note that the valid codewords  $\{c_1 c_2 c_3 c_4\}$  for (8) are {0000}, {0101}, {1011} and {1110}.

In general,  $k$  information bits are augmented with  $n - k$  parity-check bits, resulting in codewords of length  $n$ . The fraction  $R := k/n$  is known as the code 'rate', and captures the redundancy implicit in a code. High-rate codes ( $R < 1$ ) have low redundancy and correspondingly weak error correction performance.

The 'low-density' in LDPC refers to the edges in the Tanner graph being 'sparse'. In applications, Tanner graphs having as few as 0.1% of all possible edges are common. The desirability of a sparse graph will become evident when we describe the decoding algorithm below.

In what follows, we make use of the degree distribution polynomials

$$\lambda(x) = \sum_i \lambda_i x^{i-1}, \quad \rho(x) = \sum_i \rho_i x^{i-1} \quad (9)$$

as defined in [17]. The polynomial  $\lambda(x)$  describes the distribution of degrees for the bit vertices, whereas  $\rho(x)$  describes the distribution of degrees for the parity-check vertices. In particular,  $\lambda_i$  is the fraction of bit vertices of degree  $i$ . Similarly,  $\rho_i$  is the fraction of parity-check vertices of degree  $i$ . For example, suppose we desire a code where half the bit vertices participate in two parity-check equations, whereas the other half participate in three parity-check equations. Then the bit node degree distribution

polynomial is given by

$$\lambda(x) = \frac{1}{2}x + \frac{1}{2}x^2$$

A ‘regular’ code is one in which all bit vertices have the same degree and all parity-check vertices have the same degree. In other words,  $\lambda_j = 1$  for exactly one  $j$  and  $\lambda_i = 0$  for  $i \neq j$ . Similarly,  $\rho_\ell = 1$  for exactly one  $\ell$  and  $\rho_i = 0$  for all  $i \neq \ell$ . This is then referred to as a  $(\lambda, \rho)$ -regular LDPC code, where  $\lambda$  and  $\rho$  are the respective degrees of the bit vertices and parity-check vertices. For example, a (3,6)-regular LDPC code would have a Tanner graph in which all bit vertices are of degree 3 and all parity-check vertices are of degree 6. If a code is not regular, it is called ‘irregular’.

*Remark 1:* We have presented the Tanner graph representation of an LDPC code because it simplifies understanding the iterative decoding algorithm. A completely equivalent formulation, more commonly seen in the literature, involves describing the code via its ‘parity-check matrix’, usually denoted by  $H$ . The matrix  $H$  is simply the adjacency matrix for the Tanner graph and a codeword  $c$  is a binary vector satisfying

$$Hc = 0$$

when using modulo-2 arithmetic. In other words, all codewords lie in the nullspace of the parity-check matrix  $H$  over the binary field. In the above example, the parity-check matrix corresponding to (8) is given by

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

### 3.1 Binary erasure channel (BEC)

BEC is among the simplest channel models available. With the BEC, one can transmit either a one or a zero and the transmitted symbol is either received correctly or is ‘erased’. As shown in Fig. 5, a symbol is erased with probability  $\epsilon > 0$  and is received correctly with probability  $1 - \epsilon$ .

The simplicity of the BEC accounts for it being a popular subject of study. Fortunately, many of the results derived for the BEC hold for more general classes of channels (although the technical details in extending results are often significant). Furthermore, erasure channels are, in fact, a reasonable

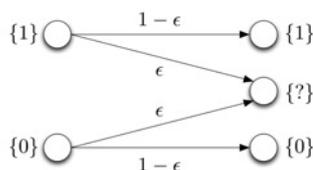


Figure 5 Binary erasure channel

abstraction of certain communications processes, for example, packetised data networks where packets may be dropped.

### 3.2 Iterative decoding

Iteratively decoding LDPC codes transmitted over the BEC is a straightforward procedure and is referred to as ‘message-passing’. Consider the Tanner graph in Fig. 4 and suppose the codeword {1011} is transmitted but {?01?} is received, that is, suppose the first and last bits are erased. We initialise the bit vertices with the received symbols and then pass them to the parity-check vertices, which then attempt to verify that the received bits indeed comprise a codeword. In this case, we see that the check equations correspond to

$$\begin{aligned} ? \oplus 1 &= 0 \quad \text{and} \\ ? \oplus 0 \oplus ? &= 0 \end{aligned}$$

Clearly, the second equation is indeterminate, but the first equation is only satisfied if the erased bit is, in fact, a 1. This information is then passed back to the bit vertices, and the bit  $c_1$  is set to 1. The message {101?} is then passed back to the check vertices, so that

$$\begin{aligned} 1 \oplus 1 &= 0 \quad \text{and} \\ 1 \oplus 0 \oplus ? &= 0 \end{aligned}$$

implying that  $c_4$  is 1. Consequently, we have correctly recovered the transmitted codeword {1011} via the steps {?01?}  $\rightarrow$  {101?}  $\rightarrow$  {1011}.

The above example illustrates the following general procedure: (i) initialise the bit vertices with the received symbols; (ii) pass these values to the parity-check equations and correct erasures for parity-check equations with only one erasure; (iii) replace the corrected erasures with their true value. Repeat steps (ii) and (iii) until all erasures have been corrected or until the number of erasures stops decreasing, at which point the decoding has failed.

Note that by insisting on sparsity of the Tanner graph, we ensure that most check equations correspond to only a few bit vertices, increasing the likelihood that at any step of the message-passing algorithm we will be able to replace an erasure with the actual transmitted value.

*Remark 2:* Iterative decoding has been deployed in numerous applications (including deep-space communications and the 3G mobile phone standard) and is being considered for every significant communications standard under discussion that involves error-correction coding. However, the above discussion describes how to correct erasures and does not obviously suggest how to deal with more complicated channels, such as the additive white Gaussian noise (AWGN) channel. In these more complicated situations, the message-passing structure described above remains the same, but the information passed along the edges of the graph becomes probabilistic information on each received bit. An

excellent introduction to probabilistic iterative decoding can be found in [12].

### 3.3 Model derivation

Given the simplicity of the binary erasure channel, it is possible to characterise the iterative decoding process in terms of a single parameter, namely, the channel erasure probability (see [15] or [16]). In what follows, we assume an infinite block length codeword with no cycles in the Tanner graph. The purpose of the assumption on the lack of cycles is so that information from a particular bit does not loop back to itself. The purpose of the infinite block length is to allow the decoding process to continue indefinitely.

We first discuss a model of how erasures propagate for a simple (2, 4)-regular code and then give a description for general LDPC codes.

A (2, 4)-regular code is such that each bit participates in two parity-check equations and each parity-check equation contains four bits. The tree structure implied by our above assumption means that we can expand the Tanner graph (locally) around each bit or parity-check node as shown in Fig. 6.

We wish to examine how the probability of erasures evolves at each step. Having performed some number of correcting steps, denote the probability that a bit remains erased by  $x$ . In Fig. 6, the ability to correct a potential erasure at the lower bit node (denoted by a circle) will depend on whether or not 'any' of the upper bit nodes are still unknown. In other words, the probability that the top parity-check node (shown as a cross-hatched square) will pass down an erasure is given by the probability that none of the upper bit nodes are currently an erasure. This probability is  $1 - (1 - x)^3$ . Now, whether or not an erasure gets passed down to the lower check node is the probability

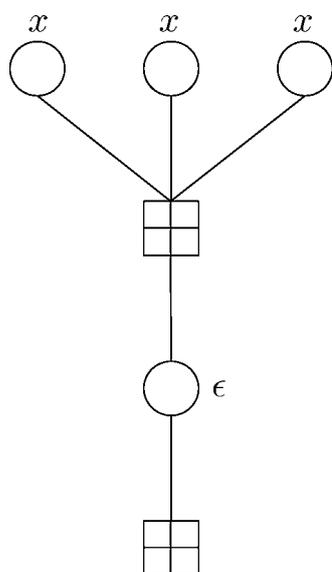


Figure 6 Local tree structure for a (2, 4)-regular code

that the lower bit node is uncorrected. The probability of this is given by the probability that the bit was initially erased ( $\epsilon$ ) and that it could not be corrected. As this procedure happens at each node at each time step, we see that the fraction of erasures evolves as

$$x_{k+1} = \epsilon(1 - (1 - x_k)^3) \quad (10)$$

More generally, consider a bit vertex of arbitrary degree  $d_b$  (i.e. the bit is connected to  $d_b$  parity-check vertices; see Fig. 7a). Again assume that the probability of an erasure on any incoming edge is denoted by  $x$ . Then the probability that an erasure will be passed down the tree will be the product of the initial erasure probability,  $\epsilon$ , and the probability that all of the parity-check vertices pass down an erasure. (This follows since the parity-check vertex below the bit vertex does not contribute to the calculation.) In other words, the probability of the bit vertex passing down an erasure is  $\epsilon x^{d_b-1}$ .

Consider now a parity-check vertex of arbitrary degree  $d_c$  (see Fig. 7b). The parity-check vertex will pass on an erasure if any of the incoming bit vertices is an erasure. (Again, this follows since the bit vertex below the parity-check vertex does not participate in the calculation.) With the probability of an erasure on an edge again denoted by  $x$ , we clearly see that the probability of the parity-check vertex passing down an erasure is given by  $1 - (1 - x)^{d_c-1}$ .

As before, the above probabilities will evolve at every node in the Tanner graph at each iteration. If we denote by  $x_k$  the expected fraction of bits which remain erased at the  $k$ th iteration, and make use of the degree distribution polynomials defined in (9), the expected fraction of erasures at the subsequent iteration is given by

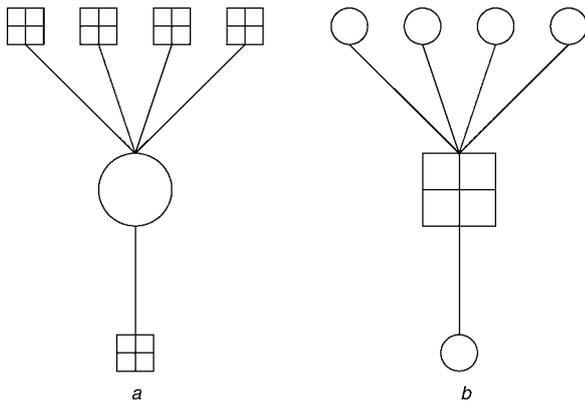
$$x_{k+1} = \epsilon\lambda(1 - \rho(1 - x_k)), \quad x_0 = \epsilon \quad (11)$$

*Remark 3:* Equation (11) is of fundamental importance in the analysis of iterative decoder performance for the BEC, allowing the expected fraction of erasures to be readily computed, subject only to the assumption of a cycle-free Tanner graph. The process of tracking the temporal evolution of the expected fractions of erasures is known as density evolution, and has been extended to channels other than the BEC, for example, the AWGN channel (see [18]).

The first question to pose in relation to (11) is whether or not the origin is locally asymptotically stable (LAS), as LAS implies that the fraction of erasures goes to zero, that is, we successfully recover the transmitted codeword. Examining the linearisation about the origin, we see that a condition for local asymptotic stability is given by

$$\epsilon\lambda'(0)\rho'(1) < 1 \quad (12)$$

This observation was first made in [18]. (We observe that choosing a code with all bit vertices having degree at least 3 will guarantee that the origin is always LAS.)



**Figure 7** Local tree structure for  
*a* check nodes and  
*b* variable nodes

Another item of crucial importance is the ‘threshold value’ of the erasure probability  $\epsilon$ . This is the value of  $\epsilon$  above which the ‘expected’ fraction of corrected erasures is bounded away from zero. In other words, if the channel noise is above the threshold value, recovery of the transmitted codeword is unlikely. Generally speaking, codes displaying higher-thresholds are preferred over those with lower thresholds, since more errors can be corrected. However, a high-threshold value is not the sole measure of a good code, for example, code rate and rate of convergence of the iterative algorithm are also important considerations.

### 3.4 Examples

We will consider two different regular LDPC codes as illustrative examples, that is, codes with single degrees each for the variable and check nodes. In particular, we will consider the systems corresponding to (2, 4)-regular and (3, 6)-regular LDPC codes. Regular codes have the advantage that they are significantly easier to analyse and implement. However, in order to generate codes that perform extremely close to capacity, it is necessary to move to irregular codes. For example, the code proposed in [6] is merely 0.0045 dB away from the Shannon limit, but requires variable nodes with degrees between 2 and 8000. We focus here on regular codes for simplicity.

*Example 2 [(2, 4)-regular LDPC code]:* We see that for a (2, 4)-regular LDPC code, the degree distribution polynomials are given by  $\lambda(x) = x$  and  $\rho(x) = x^3$ , and hence the expected fraction of errors at each decoding step is described by

$$x_{k+1} = \epsilon(1 - (1 - x_k)^3) = \epsilon(3x_k - 3x_k^2 + x_k^3)$$

The equation for a fixed point then yields

$$-3x + 3x^2 - x^3 + \frac{1}{\epsilon}x = 0 \tag{13}$$

Fig. 8 shows the root locus corresponding to (13). Of course, our interest is not in  $K \in (0, \infty)$ , as would normally be the case

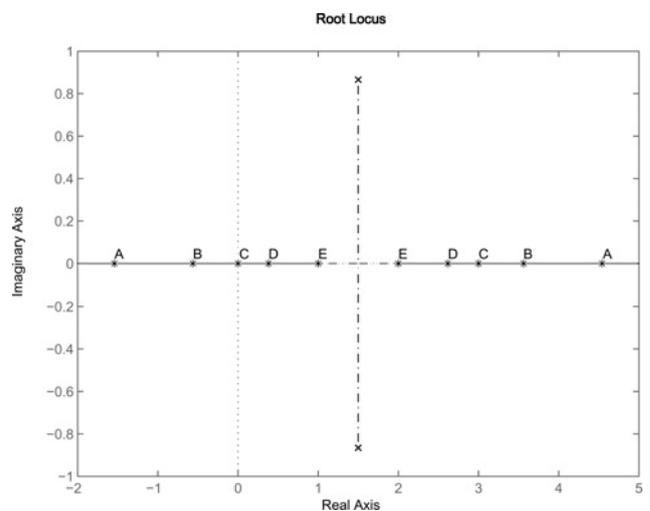
when considering feedback systems, but in  $\epsilon \in [0, 1]$ . This clearly corresponds to  $K \in [1, +\infty)$ . Furthermore, we are interested in varying  $\epsilon$  from zero to 1, that is, we wish to start with a zero probability of erasure and increase this probability, paying particular attention to the threshold value. At  $\epsilon = 0$ , the roots will be at the origin and  $\pm\infty$ . As  $\epsilon$  increases, the roots will move inwards until breaking off the axis at 1.5. Since (11) describes the ‘fraction’ of erasures at each iteration, our interest is in the situation when there are roots on the real axis between zero and 1.

We check the local stability condition (12) and see that it is satisfied for sufficiently small values of the erasure probability [i.e. for  $\epsilon < (1/3)$ ]. Observing the locus, as  $\epsilon$  increases, we see that a fixed point moves from the left-half plane, through the origin, into the right-half plane, and this occurs at the value of  $\epsilon = 1/3$ . Above this value, the fixed point at the origin is unstable, and the fixed point that moves to the right along the real axis is locally asymptotically stable. This is what we would expect as, above the threshold value, we cannot correct all the erasures introduced by the channel. Note that the third fixed point is outside the interval  $[0, 1]$  and is thus of no interest. □

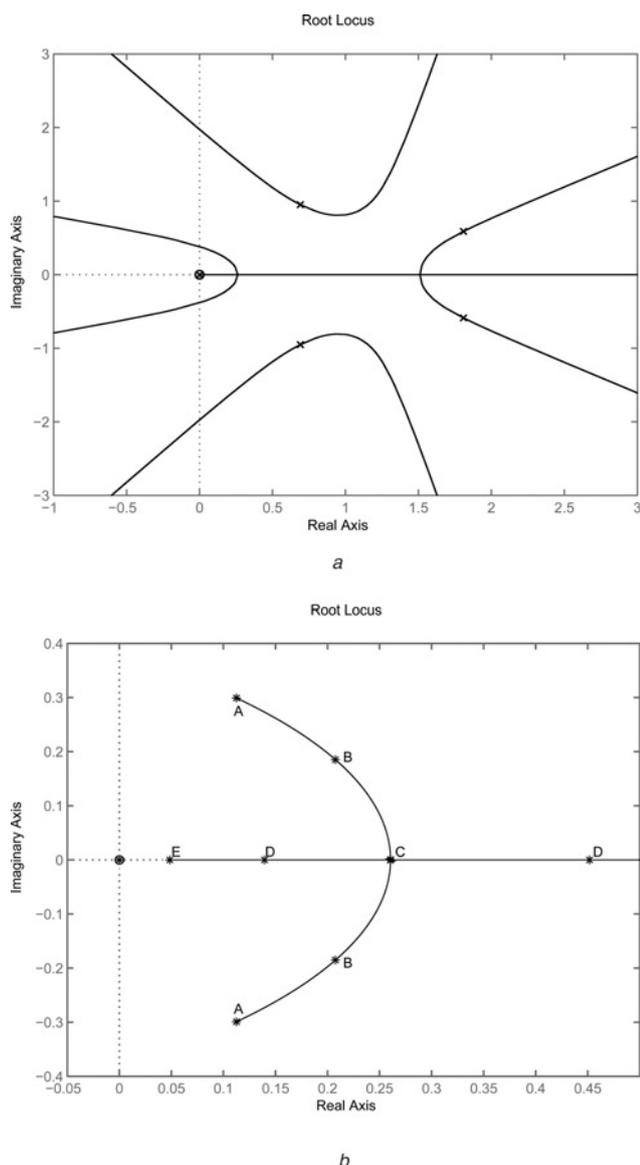
*Example 3 [(3, 6)-regular LDPC code]:* Next, consider the one-dimensional dynamical system corresponding to the (3, 6)-regular LDPC code. In this case,  $\lambda(x) = x^2$  and  $\rho(x) = x^5$ , and hence, the evolution of the fraction of erasures is given by

$$\begin{aligned} x_{k+1} &= \epsilon(1 - (1 - x_k)^5)^2 \\ &= \epsilon(25x_k^2 - 100x_k^3 + 200x_k^4 - 250x_k^5 \\ &\quad + 210x_k^6 - 120x_k^7 + 45x_k^8 - 10x_k^9 + x_k^{10}) \end{aligned} \tag{14}$$

Fig. 9a shows the root locus plot corresponding to the fixed point equation for (14), whereas Fig. 9b shows the relevant



**Figure 8** Root locus for the (2, 4)-regular LDPC code  
 Solid line corresponds to  $\epsilon \in [0, 1]$   
 (A)  $\epsilon = 0.1$ ; (B)  $\epsilon = 0.2$ ; (C)  $\epsilon = 1/3$  (threshold value);  
 (D)  $\epsilon = 0.5$ ; (E)  $\epsilon = 1$



**Figure 9** Root locus plot for (3, 6)-regular LDPC fixed points

a Root locus

b Locus for various values of  $\epsilon$ : (A)  $\epsilon = 1/5$ ; (B)  $\epsilon = 1/3$ ; (C)  $\epsilon = 0.4294$  (threshold value); (D)  $\epsilon = 1/2$ ; (E)  $\epsilon = 1$

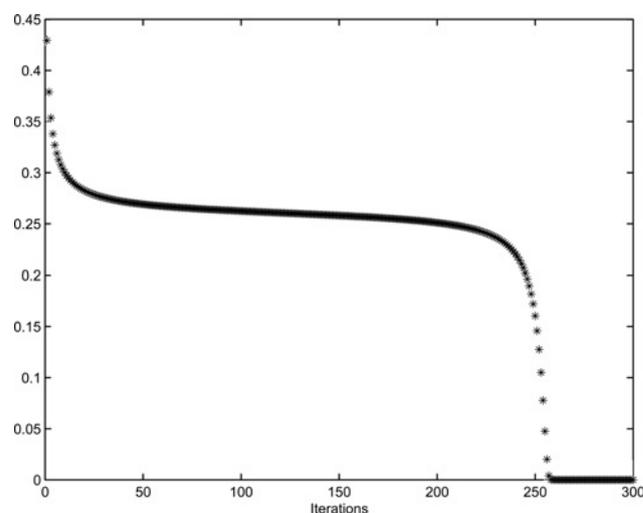
portion of the real axis and various values of  $\epsilon$ . As  $\epsilon$  increases from  $1/5$  to the threshold value of  $0.4294$  [16, §2.9.7], the roots move from points A to C. At point C, a new fixed point is created where the root locus breaks in to the real axis. As  $\epsilon$  continues to increase, this fixed point bifurcates, with one fixed point moving in towards the origin and another moving away from the origin. It is not difficult to show that the fixed point which moves in towards the origin is unstable and that which moves away from the origin is LAS.

Note that the origin is LAS for any value of  $\epsilon$  since  $\lambda'(0) = 0$ . This is consistent with the root locus plot in Fig. 9b as, for any finite value of  $\epsilon$ , the unstable fixed point will be bounded away from the origin. Furthermore, we can observe a sizable gap between the origin and the unstable fixed point at point (E) corresponding to  $\epsilon = 1$ .

Finally, we observe that the point where the root locus plot breaks in to the real axis between zero and 1, in Fig. 9b this is near 0.26, corresponds to the point where the convergence of the iterative decoding algorithm slows down when operating close to the threshold value. In particular, Fig. 10 shows the trajectory of (14) for a channel error probability of  $\epsilon = 0.4293$ , just below the threshold value of  $\epsilon = 0.4294$ . Note that convergence requires over 250 iterations. By contrast, taking  $x_0 = \epsilon = 0.2$  yields convergence in four iterations.  $\square$

The root locus plots for the (2, 4)-regular LDPC code (Fig. 8) and the (3, 6)-regular LDPC code (Fig. 9) provide important information about the behaviour of the decoding algorithm. For instance, when operating near threshold, the break-in point in Fig. 9 predicts where the decoding algorithm will be slow to improve the number of remaining erasures. Furthermore, there is an interest in how the decoding algorithm behaves for values 'above' threshold [19]. In the case of the (2, 4)-regular LDPC code, we observe that the iterative process will remove a larger fraction of errors than the (3, 6)-regular code since, for values slightly above threshold, the LAS fixed point will be near the origin. On the other hand, for the (3, 6)-regular LDPC code, using the iterative process for values above threshold will not be as beneficial. For example, consider  $\epsilon = 1/2$ , and note that, with this initial fraction of erasures, the algorithm will converge to the LAS fixed point at about 0.45 for the (3, 6)-regular code, whereas the (2, 4)-regular code converges to the fixed point at about 0.38. In other words, about 45% of the erasures will remain uncorrected for the (3, 6)-regular code whereas about 38% will remain uncorrected for the (2, 4)-regular code.

*Remark 4:* The bifurcation behaviour exhibited in the above examples is entirely consistent with Shannon's noisy coding theorem [1], which predicts for a code rate  $R$  a noise threshold  $\epsilon^*(R)$  above which the expected fraction of



**Figure 10** Trajectory for (11) with both initial condition and erasure probability given by  $x_0 = \epsilon = 0.4293$

erasures is strictly positive, whereas for noise values below the threshold error-free communication is possible (at least as long as the codeword length  $n$  tends to infinity).

*Remark 5:* We note that the message-passing decoder described in Section 3.2 is not the only decoder which gives rise to a one-dimensional dynamical system. For example, the so-called ‘Gallager A’ decoder for decoding LDPC codes transmitted over the binary symmetric channel gives rise to a one-dimensional difference equation (see [15]) that is amenable to the preceding root locus analysis. On the other hand, certain codes and decoders do not give rise to a root locus form description of fixed points. For example, the erasure fraction for a particular parallel concatenated turbo code evolves as (see [15])

$$x_{k+1} = \frac{x_k \epsilon^2 (2 - 2\epsilon + x_k \epsilon)}{(1 - \epsilon(1 - x_k))^2}$$

## 4 Conclusions

In this paper, we have described a family of codes and an iterative decoding algorithm whose behaviour can be characterised by a one-dimensional dynamical system. While these results are known in the information theory literature (see [15]), we have shown how the well-known root locus plot can be used to provide insight into the behaviour and performance of the decoding algorithm.

## 5 Acknowledgments

The authors are supported by the Australian Research Council under grant DP0771131. This work appeared in preliminary form at the 2006 IEEE Conference on Decision and Control.

## 6 References

- [1] SHANNON C.E.: ‘A mathematical theory of communication’, *Bell Syst. Tech. J.*, 1948, **27**, pp. 379–423, 623–656
- [2] VAN LINT J.H.: ‘Introduction to coding theory’ (Springer-Verlag, 1992, 2nd edn.)
- [3] BERROU C., GLAVIEUX A., THITIMAJSHIMA P.: ‘Near Shannon limit error-correcting coding and decoding: Turbo codes’. Proc. of IEEE Int. Conf. on Communication, 23–26 May 1993, vol. 2, pp. 1064–1070
- [4] GALLAGER R.G.: ‘Low-density parity-check codes’ (MIT Press, 1963)
- [5] MACKAY D.J.C., NEAL R.M.: ‘Near Shannon limit performance of low density parity check codes’, *Electron. Lett.*, 1997, **33**, (6), pp. 457–458
- [6] CHUNG S.-Y., FORNEY G.D., RICHARDSON T.J., URBANKE R.L.: ‘On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit’, *IEEE Commun. Lett.*, 2001, **5**, (2), pp. 58–60
- [7] AGRAWAL D., VARDY A.: ‘The turbo decoding algorithm and its phase trajectories’, *IEEE Trans. Inf. Theory*, 2001, **47**, (2), pp. 699–722
- [8] FU M.: ‘Stochastic analysis of turbo decoding’, *IEEE Trans. Inf. Theory*, 2005, **51**, (1), pp. 81–100
- [9] RICHARDSON T.: ‘The geometry of turbo-decoding dynamics’, *IEEE Trans. Inf. Theory*, 2000, **46**, (1), pp. 9–23
- [10] EVANS W.R.: ‘Graphical analysis of control systems’, *Trans. Am. Inst. Electr. Eng.*, 1948, **67**, pp. 547–551
- [11] EVANS W.R.: ‘Control system synthesis by root locus method’, *Trans. Am. Inst. Electr. Eng.*, **69**, pp. 66–69, 1950, Reprinted in *Control Theory: Twenty-Five Seminal Papers (1932–1981)*, Edited by Tamer Basar.
- [12] JOHNSON S.J.: ‘Introduction to low-density parity-check codes’, <http://www.sigpromu.org/sarah/publications.html>, 2006
- [13] JOHNSON S.J., WELLER S.R.: ‘Low-density parity-check codes: design and decoding’, ‘Wiley Encyclopedia of Telecommunications’ (John Wiley and Sons, 2003)
- [14] MACKAY D.J.C.: ‘Information theory, inference, and learning algorithms’ (Cambridge University Press, 2003)
- [15] RICHARDSON T., URBANKE R.L.: ‘Codes, Systems, and Graphical Models, volume 123 of The IMA Volumes in Mathematics and its Applications’, ‘An introduction to the analysis of iterative coding systems’. Springer, 2001, pp. 1–37
- [16] RICHARDSON T., URBANKE R.L.: ‘Modern coding theory’. Draft available <http://lthcwww.ep.ch/mct/index.php>, 2005
- [17] LUBY M.G., MITZENMACHER M., SHOKROLLAHI M.A., SPIELMAN D.A., STEMANN V.: ‘Practical loss-resilient codes’. Proc. of the 29th ACM Symp. on Theory of Computing, 1997, pp. 150–159
- [18] RICHARDSON T.J., URBANKE R.L.: ‘The capacity of low-density parity-check codes under message-passing decoding’, *IEEE Trans. Inf. Theory*, 2001, **47**, (2), pp. 599–618
- [19] MEASSON C., URBANKE R.L., MONTANARI A., RICHARDSON T.: ‘Life above threshold: from list decoding to area theorem and MSE’. Proc. of IEEE Information Theory Workshop, San Antonio, TX, 24–29 October 2004