On Gaussian Approximation for Density Evolution of Low-Density Parity-Check Codes

Minyue Fu
School of EE&CS, University of Newcastle, Australia
Email: minyue.fu@newcastle.edu.au

Abstract— **This paper is concerned with density evolution for iterative decoding of low-density parity-check (LDPC) codes. We first study the problem of density evolution computation for regular LDPC codes. For this, we propose a simple computational algorithm based on the ergodicity theory. This method is shown to match very well with explicit calculations of density functions. The second problem we study is about the approach of Gaussian approximation to density evolution. We point out that it is inappropriate to use the mean of the density only to model the iterative decoding process. Instead, both the mean and variance are needed for Gaussian approximation. Finally, we consider the problem of density evolution for irregular LDPC codes. For this, we extend the density evolution algorithm for regular LDPC codes to irregular LDPC codes. We then illustrate that Gaussian approximation is also valid provided that the degree distributions are not wide. A dynamic model is also presented based on Gaussian approximation.**

Keywords: Low-density parity-check codes, irregular LDPC codes, iterative decoding, MAP decoding.

I. Introduction

This paper is inspired by a series of work by Richardson, Urbanke and their colleagues [1], [2], [3], [4] where they have studied the so-called *density evolution* problem for the decoding process of LDPC codes. Density evolution is about keeping track of the density functions for the messages in the decoding process. This is known to be a very useful tool for both understanding the decoding process and code design.

In this paper, we also study the problem of density evolution for LDPC codes. Firstly, we consider regular LDPC codes and present an alternative method for density evolution. This method is based on the ergodicity theory and simple to apply. We also show how this method compares with the Fourier transform based method in [1]. Secondly, we study the Gaussian approximation method as discussed in [2]. We give some justification about the validity of Gaussian approximation. We then show that the single-parameter model as developed in [2] is inadequate in general. Consequently, we develop a two-input, two-output model which can adequately describe the dynamic behavior of the LDPC decoding process. This model shows that the dynamic behavior is far more nonlinear and complex than a single-parameter model can reveal. Finally, we extend the work on regular LDPC codes to irregular LDPC codes. More specifically, the aforementioned density evolution algorithm is extended to irregular LDPC codes. We also point out that Gaussian approximation is still valid provided that the degree distributions of the variable and check nodes are reasonably narrow. Based on this observation, a similar two-input, two-output model is presented.

II. Regular LDPC codes

Under the assumption that the bipartite graph of a given LDPC code has no cycles of length 2ℓ or less, which is referred to as the *cycle-free* case, the density functions in an iterative decoding process up to ℓ-th iterations can be computed by treating all variables in each iteration independent. In this paper, we only consider the sum-product rule [5] for decoding. Denote by u_0 the log-likelihood ratio (LLR) of the received signal at variable nodes, by v the message from a variable node to a check node and by u the message from a check node to a variable node. Let d_v and d_c denote the numbers of edges for each variable node and check node, respectively. Then, at each variable node, update rule is given by

$$v = u_0 + \sum_{i=1}^{d_v-1} u_i \tag{1}$$

where $u_i, i = 1, 2, \cdots, d_v - 1$, are the incoming messages from the neighbors of the variable node except the check node that will get the message v. The update rule for each check node is given by

$$u = 2\tanh^{-1} \prod_{i=1}^{d_c-1} \tanh\frac{v_i}{2} \tag{2}$$

where $v_i, i = 1, 2, \cdots, d_c - 1$, are the incoming messages from the neighbors of the check node except the variable node that will get the message u. The decoding process starts by using (1) with $u_i, i = 1, 2, \cdots, d_v-1$, equal to zero initially. It is then followed by (2) to complete an iteration. This process is then repeated ℓ times.

Under the cycle-free assumption and the assumption that the samples of u_0 at different variable nodes are independent with identical distributions (i.i.d.), it is straightforward to see that $u_i, i = 1, 2, \cdots, d_v - 1$, are i.i.d. and so are $v_i, i = 1, 2, \cdots, d_c - 1$. This fact allows us to compute the probability density functions of u_i and v_i easily. In the sequel, we denote by $P^{(k)}$ and $Q^{(k)}$, $k = 1, 2, \cdots$, the densities of u_i and v_i in iteration k. We will also denote by P_0 the density of u_0. Also denote the mapping from the density of v to $\tanh v$ by Γ.

Based on the i.i.d. property above, it is proposed in [1] to use Fourier transform to update $P^{(k)}$ and $Q^{(k)}$. It follows

from (1) that, for $k \geq 1$,

$$Q^{(k+1)} = P_0 \otimes (\otimes_{i=1}^{d_v-1} P^{(k)}) \tag{3}$$

where $\otimes$ denotes convolution. Similarly,(2) leads to

$$P^{(k+1)} = \Gamma^{-1}(\otimes_{i=1}^{d_v-1} \Gamma(Q^{(k+1)})) \tag{4}$$

Taking the Fourier transform, the (3) becomes

$$\mathcal{F}(Q^{(k+1)}) = \mathcal{F}(P_0)(\mathcal{F}(P^{(k-1)}))^{d_v-1} \tag{5}$$

The use of Fourier transform for computing $P^{(k+1)}$ is somewhat more complicated because it requires some transformation of $P^{(k+1)}$ and $Q^{(k)}$; see [1] for details.

The density evolution method we propose is based on the ergodicity theory. To start with, we point out a *hidden* assumption in the update rule (1)-(2). As mentioned above, the update rule requires the cycle-free assumption. The required code length grows exponentially fast as ℓ increases. For a realistic iteration number (e.g., 20), the code length can be regarded as of infinite size. That is, the hidden assumption is that the code is of infinite length practically.

We now assume that the code length is infinite. Since the update rule (1)-(2) is stationary (i.e., invariant with respect to the iteration number), by the ergodicity theory (see, e.g., [11]), we know that the update rule preserves ergodicity, if the input signal is ergodic. Since the initial samples of u_0 are i.i.d., they form an ergodic random process. Hence, the messages in every iteration are all ergodic. It is a well-known property of ergodicity that any statistical parameter of the random process can be arbitrarily closely approximated by averaging over a sufficient number of samples. Examples of statistical parameters include the bit error probability and the density function itself if it is quantized into a finite support.

Using the ergodic properties as stated above, our proposed algorithm is simply stated as follows:

- Step 0: Choose a large number N and generate N samples of u_0 according to the given density P_0.
- Step 1 (for variable nodes): For iteration 0, copy the samples of u_0 to v as in (1). For other iterations, take the N samples of u from the previous iteration. For each $i = 1, 2, \cdots, d_v - 1$, randomly interleave the samples of u to generate N samples of u_i. Then compute the N samples of v using (1).
- Step 2 (for check nodes): For each iteration, take the N samples of v as calculated above. For each $i = 1, 2, \cdots, d_c - 1$, randomly interleave the samples of v to generate N samples of v_i. Then compute the N samples of u using (2).

Note that the interleavers are used to guarantee the independence of samples with the same density function.

We claim that the complexity of this algorithm is $O(\ell N(d_v + d_c))$ for ℓ iterations. This simply follows from the fact that the complexity for random interleaving of N samples is $O(N)$. To see the latter, we can use a random generator which generates a random real number from 0 to 1. We start with a vector V of $1, 2, \cdots, N$ and take the size of the vector as $S = N$. In each step, we generate a random number x and multiply it by S and round it up to generate an index i between 1 to S. We then concatenate $V(i)$ to a new vector, copy the last element of V to $V(i)$ and reduce S by 1. Repeat the above step until V is empty. It is clear that only N such steps are required. Therefore, our claim holds.

Now we compare the complexity of our algorithm with the Fourier transform based algorithm in [1]. Clearly, this implementation is done using fast Fourier transform (FFT) and requires $\tilde{N}$ uniform sampling points. Since it is known [6] that their distributions are approximately Gaussian and their mean and variance typically diverges to large values after several iterations, $\tilde{N}$ should be sufficiently large to cover the dynamic range of distributions and their accuracies. For a given $\tilde{N}$, the computational complexity of each Fourier transform is $\tilde{N} \log_2 \tilde{N}$. Therefore, the computational complexity for implementing (5) is $O(\ell \tilde{N} \log_2 \tilde{N})$. Similarly, it can be shown that the complexity for implementing (2) is $O(\ell \tilde{N} \log_2 \tilde{N})$. Hence, the complexity of the Fourier transform based algorithm is $O(\ell \tilde{N} \log_2 \tilde{N})$.

Clearly, if $\tilde{N}$ is compatible to N and both are not very large, the Fourier algorithm is more efficient because it does not depend on d_v and d_c. But if both $\tilde{N}$ and N are very large and they are compatible, the proposed algorithm becomes competitive. The proposed algorithm also has two additional advantages: 1) It uses the update rule directly; 2) It does not require determining where to place the sampling points for evaluating the error probability.

To test how the proposed algorithm performs, Table I lists the noise thresholds for different regular LDPC codes computed by this algorithm and the FFT-based algorithm. The code variables are assigned to ± 1 and additive white Gaussian noise (AWGN) with zero mean and variance σ_0^2 is used. The values $\sigma_0^{\rm FFT}, \sigma_0^{\rm prop}$ and $\sigma_0^{\rm bound}$ denote the thresholds of σ_0 computed using the FFT-based algorithm, our algorithm and Shannon bound, respectively. The threshold is computed by a simple bisection method which tests if a given σ_0 leads to a finite signal-to-noise ratio (S/N) for the messages or if the S/N grows indefinitely as the iteration number increases. It is known [1] that such a threshold is guaranteed to exist. The data for $\sigma_0^{\rm FFT}$ are cited from [1]. It is clear from this table that the proposed algorithm approximates very well for $N = 10^5$ or above. For $N = 10^5$, it takes only a few seconds using Matlab on a Pentium 4 PC to test if a given σ_0 value exceeds the threshold. For $N = 10^6$, it takes a couple of minutes. When implemented in C language, the test takes only a few seconds for $N = 10^6$.

III. Gaussian Approximation

It is well known that the densities of the messages are approximately Gaussian when the channel noise is AWGN. This fact was probably first observed by Wiberg [6] using simulations. This observation has been used in numerous

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

(d_v, d_c)	rate	σ_0^{FFT}	σ_0^{prop} $N = 10^5$	σ_0^{prop} $N = 10^6$	σ_0^{bound}
(3,6)	0.5	0.8809	0.8795	0.8801	1.0000
(4,8)	0.5	0.8376	0.834	0.8371	1.0000
(5,10)	0.5	0.7936	0.791	0.7924	1.0000
(3,5)	0.4	1.0093	1.00	1.005	1.1616
(4,6)	1/3	1.0109	1.009	1.01	1.3048
(3,4)	0.25	1.2667	1.256	1.2605	1.5538
(4,10)	0.6	0.7481	0.745	.7462	0.87794
(3,9)	2/3	0.7082	0.705	0.707	0.81115
(3,12)	0.75	0.6320	0.628	0.6315	0.73954

TABLE I
COMPARISON OF DENSITY EVOLUTION METHODS

papers to approximate the density evolutions of both LDPC codes [3] and turbo codes [7], [8], [9], [10]. However, most density evolution methods based on Gaussian approximations use a single parameter model. This parameter can be the S/N of the density [8], [9], the so-called mutual information of the density [7], [8], [10], or simply the mean of the density [3]. For LDPC codes, [3] justifies the use of a single parameter model by using the so-called *symmetry condition* which requires a density function $f(x)$ to satisfy $f(x) = f(-x)e^x$. More specifically, it is argued in [3] that if a density obeys both a Gaussian distribution and the symmetry condition, then the mean μ and variance σ^2 has the relationship $\sigma^2 = 2\mu$. It follows that a single parameter (e.g., the mean) is sufficient to characterize the density.

However, it has been pointed out in [12] that the use of a single parameter to model the decoding process is inadequate for turbo codes. The main reason is that the true density is not exactly Gaussian. Although the true density may obey the symmetry condition, if the symmetry condition is applied to the approximated density, it may grossly distort the density. Indeed, it is shown in [12] that the mean-variance curve of the true density deviates significantly from $\sigma^2 = 2\mu$, especially when the number of iterations is large.

In this section, we intend to do the following. Firstly, we explain why the message densities in LDPC decoding are approximately Gaussian. This analysis is very similar to [12] for decoding of turbo codes. Secondly, we show using the density evolution method given in the previous section why a single parameter model is inappropriate to model the message densities. Finally, we illustrate how a two-parameter model looks like for a typically LDPC decoding process.

A. Justification of Gaussian Approximation

To explain why the message densities are approximately Gaussian, we revisit the update rule (1)-(2). It is clear from (1) that if $u_i, i = 0, 1, \cdots, d_v - 1$, are all approximately Gaussian and i.i.d., then v is approximately Gaussian. To see why (2) preserves the Gaussian property approximately, we denote the mapping (2) by $f_{d_c-1}(v_1, \cdots, v_{d_c-1})$. It is easy to check that

$$f_{m+1}(v_1, \cdots, v_{m+1}) = f_2(f_m(v_1, \cdots, v_m), v_{m+1}) \quad (6)$$

for any $m > 2$. Thus, it is sufficient to show that

$$u = f_2(v_1, v_2)$$

is approximately Gaussian if v_1 and v_2 are approximately Gaussian and i.i.d. It is simple to check that

$$f_2(v_1, v_2) = \ln(\exp(v_1) + \exp(v_2)) - \ln(\exp(v_1 + v_2) + 1)$$

Since v_1 and v_2 are approximately Gaussian, $v_1 + v_2$ is approximately Gaussian and $\exp(v_1), \exp(v_2)$ and $\exp(v_1 + v_2)$ are approximately lognormal (i.e., they become approximately Gaussian by taking log). Moreover, 1 is a special lognormal distribution because $\ln 1 = 0$ is a special Gaussian distribution (with zero mean and zero variance). It is well known (see, e.g., [13]) that the sum of lognormal distributions is approximately lognormal. By continuity, the sum of approximately lognormal distributions is still approximately lognormal. It follows that $\exp(v_1) + \exp(v_2)$ and $\exp(v_1 + v_2) + 1$ are approximately lognormal, and hence $\ln(\exp(v_1) + \exp(v_2))$ and $\ln(\exp(v_1 + v_2) + 1)$ are approximately Gaussian and their sum is approximately Gaussian as well.

Now let us comment on the accuracy of the Gaussian approximation. It is known [13] that when two lognormal distributions are i.i.d., the approximation that their sum is still lognormal tends to be most accurate. The approximation tends to become poor when the two distributions are very different. Using this, we see that $\ln(\exp(v_1)+\exp(v_2))$ is well approximately by a Gaussian distribution because v_1 and v_2 are i.i.d.. On the other hand, $\ln(\exp(v_1 + v_2) + 1)$ tends to give a poorer approximation because $\exp(v_1 + v_2)$ and 1 have very different distributions. It is clear that if $v_1 + v_2$ is large, we have $\ln(\exp(v_1 + v_2) + 1) \approx v_1 + v_2$, which is approximately Gaussian. When $v_1 + v_2$ is small, we have $\ln(\exp(v_1 + v_2) + 1) \approx \exp(v_1 + v_2)$, which is indeed lognormal.

Going back to (1), one useful observation is that even if the inputs u_i are not very Gaussian-like, the output v tends to be more Gaussian-like. This is especially true when d_v is large, following from the law of large numbers. Another useful observation is that when $u_i, i = 1, 2, \cdots, d_v - 1$, are relatively small compared with u_0, the output v tends to be well approximately by a Gaussian distribution because u_0 is Gaussian.

Therefore, our conclusion can be summarized as follows:

- The message densities given update rule (1)-(2) are approximately Gaussian.
- The accuracy of Gaussian approximation tends to be much more accurate at the variable nodes than the check nodes.
- At the variable nodes, the accuracy of Gaussian approximation tends to be more accurate when the messages from the check nodes are very small (in terms of S/N) or large.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

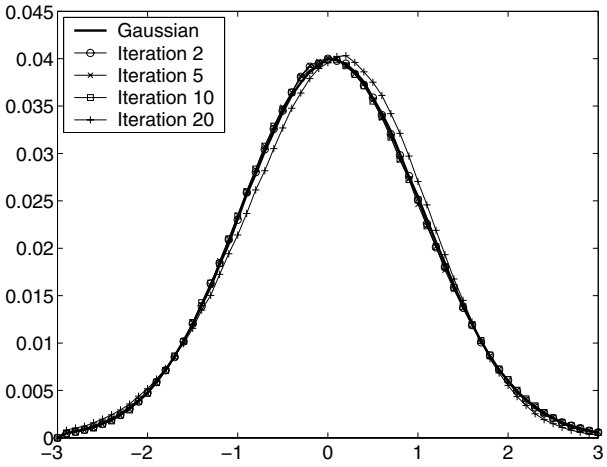

Fig. 1. Density functions at Variable Nodes

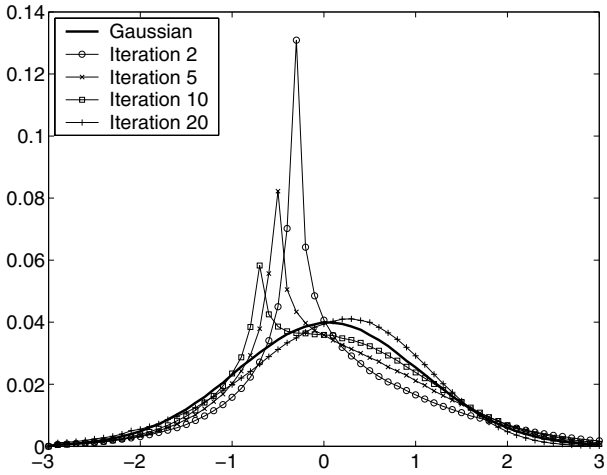

Fig. 2. Density functions at Check Nodes

- At the check nodes, the accuracy of Gaussian approximation tends to be more accurate when the message from the variable nodes are not very small.

Note that these properties are all known [6], [2], but we have now given a justification. To demonstrate these properties, we show in Figures 1-2 the densities of the variable nodes and check nodes, respectively, for a (3,6) LDPC code with $\sigma_0 = 0.85$. We see that the densities are very close to Gaussian at variable nodes, but not so close at check nodes.

B. Mean vs. Variance

Given that the message densities at the variable nodes are well approximated by Gaussian distributions, we can obviously model the density evolution by tracking the mean and variance only. Our purpose here is to examine the relationship between the mean and variance of the message densities. Since we know that the densities are more Gaussian-like for variable nodes, we only study them here.

The key observation we want to show here is that it is grossly inaccurate to assume the relationship of $\sigma^2 = 2\mu$ for the message densities. This is illustrated in Figure 3 which plots the mean-variance curve for the variable nodes of a (3,6) LDPC code with $\sigma_0 = 0.86$.

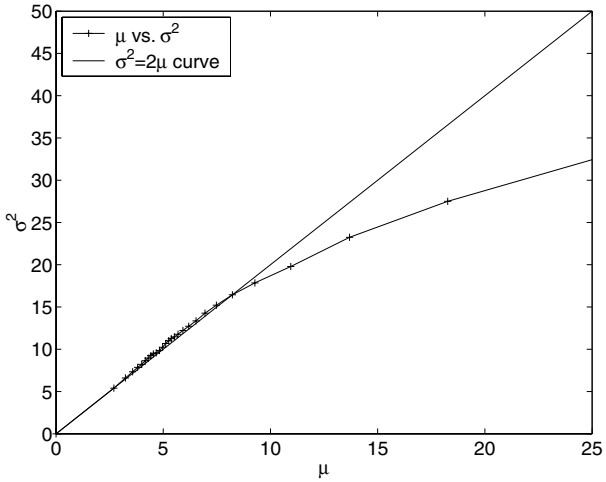

Fig. 3. Mean vs. Variance Curve

It is clear from the figure that the mean-variance curve deviates substantially from the $\sigma^2 = 2\mu$ curve. Indeed, the approximation $\sigma^2 = 2\mu$ is very good initially. [1] The curve then deviates slightly towards $\sigma^2 > 2\mu$. At some point, this trend reverses and the curve crosses $\sigma^2 = 2\mu$. Afterwards, the curve deviates grossly towards $\sigma^2 < 2\mu$. We note that this is a typical behavior for regular LDPC codes. The same behavior applies to turbo decoding [12].

We note that Gaussian approximation does violate the symmetry condition [3]. If the symmetry condition is forced onto the approximated density, it may grossly distort the density. In reality, relaxing the symmetry condition hardly matters because the density function is typically near zero in the negative region (assuming all 1's are transmitted).

It is reported in [2] that the single-parameter model based on the mean value only gives a good approximation for the threshold of σ_0. This is because as σ_0 approaches the threshold, the mean value of the message is typically small and $\sigma^2 \approx 2\mu$ is reasonably valid.

C. Dynamic Model of LDPC Decoding

Using the Gaussian approximation, we understand that the density update rule can be modeled approximately as a two-input, two-output (TITO) mapping. More specifically, denoting the mean and variance of the density at the variable node in iteration k by μ_k and σ_k^2, the TITO mapping can be expressed as

$$(\mu_{k+1}, \sigma_{k+1}) = \mathcal{M}(\mu_k, \sigma_k) \tag{7}$$

which is a dynamic model of the decoding process. The state of the process is (μ_k, σ_k).

For a given regular LDPC code with a large block size, the mapping can be calculated numerically using the algorithm described in the previous section.

[1] This is not surprising because u_0 satisfies $\sigma^2 = 2\mu$; see, e.g., [12].

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

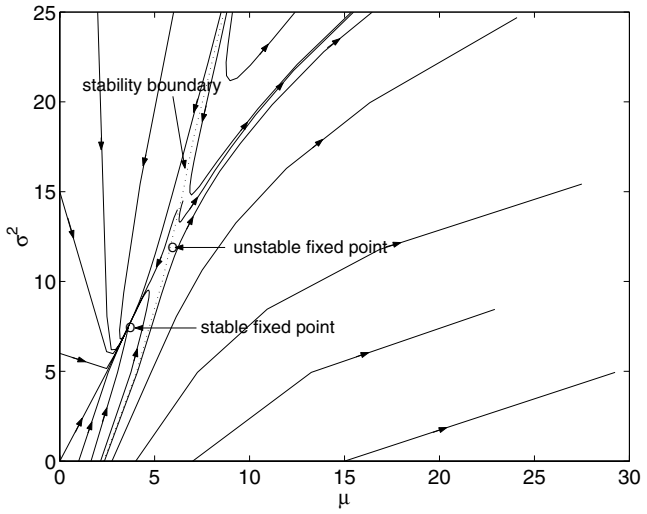

Fig. 4. Dynamic Model of Regular LDPC Decoding

To illustrate this dynamic model, we apply it to a $(3, 6)$ LDPC code with $\sigma_0 = 0.95$. Note that this value is above the threshold in Table I and is chosen deliberately to demonstrate the complex dynamic behavior of the decoding process. Given any (μ_k, σ_k), we take u_i, $i = 1, 2, \cdots, d_v - 1$, to be a set of i.i.d. Gaussian distributions with mean and variance equal to μ_k and σ_k^2, respectively, and carry out density evolution as described in the previous section to compute the resultant μ_{k+1} and σ_{k+1}^2. The result is shown in Figure 4.

As we see in Figure 4, the dynamic behavior is very nonlinear. The state space of (μ_k, σ_k) is divided into 2 regions by a *stability boundary*. There are also two fixed points, a stable one to the left of the stability boundary and an unstable one right on the stability boundary. If the initial state is to the left side of this boundary, the trajectory of (μ_k, σ_k) always converges to the stable fixed point. This corresponding to the "failed" decoding case because the final S/N is finite (or decoding error probability is nonzero). However, if the initial state is to the right of this boundary, it always diverges with S/N approaching infinity (or decoding error probability to zero). If the initial state starts on the stability boundary, it always converges towards the unstable fixed point and then either converges to the stable fixed point or diverges to a high S/N as described above.

The step size for the state varies depending on its location. In general, the closer (further away) it is to either of the fixed points, the slower (or faster) the state moves.

As the channel noise density σ_0 decreases, the two fixed points will move towards together. At the σ_0 reaches the threshold, the two points merge. After that, they both disappear along with the stability boundary, i.e., the whole state space becomes one region only; and S/N always converges to infinity no matter where the decoding starts.

We point out that the dynamic behavior of the LDPC decoding process as described above holds for all regular LDPC codes. In fact, the decoding dynamics for turbo codes behave in the same way, as reported in [12]. This is not surprising because turbo decoding also uses the sum-product rule. Moreover, this dynamic behavior is also valid for other update rules (e.g., max-Log-MAP rule, soft Viterbi algorithm).

Apart from the use in understanding the behavior of the decoding process, the dynamic model as described here is also useful for several other purposes. Firstly, it helps understand how sensitive the threshold of σ_0 by examining the movement of the two fixed points when σ_0 is moving away from the threshold. Secondly, it helps design a faster decoder. Indeed, since the step size for decoding depends on the state, a sub-optimal but faster decoder can be used where the step size is expected to be large and an optimal decoder (i.e. the sum-product rule) is used when the step size may become small. Using a sub-optimal decoder will change the decoding trajectory. But because the final decoding outcome is independent of the initial state when the channel noise is below the threshold, switching between decoders as described above does not affect the final decoding results. Provided the switching is optimized, such a decoder can potentially be faster than the optimal decoder without sacrificing any decoding errors. See [12] for an example of a fast decoder for turbo codes. Thirdly, the model is useful for partial re-transmission design. More specifically, in the event the decoding of a particular block fails, one may estimate where the state is (which can be approximated without knowing the correct codeword) and use this information to estimate how much additional S/N is needed for successful decoding.

IV. Irregular LDPC Codes

Irregular LDPC codes employ different degrees at either variable nodes or check nodes or both. Following the notation in [3], we use two polynomials to describe the degree distributions at variable nodes and check nodes,

$$\lambda(x) = \sum_{i=2}^{d_v} \lambda_i x^{i-1}; \quad \rho(x) = \sum_{i=2}^{d_c} \rho_i x^{i-1} \tag{8}$$

respectively. In the above, d_v (resp. d_c) is the maximum degree of the variable (resp. check) node and λ_i (resp. ρ_i) is the probability for an edge to connect to a degree i variable (resp. check) node.

Recall the notation $P_0, P^{(k)}, Q^{(k)}$ and Γ. It is shown in [2] that the update rule for $P^{(k)}$ and $Q^{(k)}$ is given by

$$Q^{(k+1)} = P_0 \otimes \lambda(P^{(k)}) \tag{9}$$

$$P^{(k+1)} = \Gamma^{-1}(\rho(\Gamma(Q^{(k+1)}))) \tag{10}$$

where

$$\lambda(P) = \sum_i \lambda_i P^{\otimes(i-1)}; \quad \rho(Q) = \sum_i \rho_i Q^{\otimes(i-1)}$$

That is, only the *averaged* density needs to be updated.

We now modify the density evolution algorithm in Section II to suit irregular LDPC codes. Note that the

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

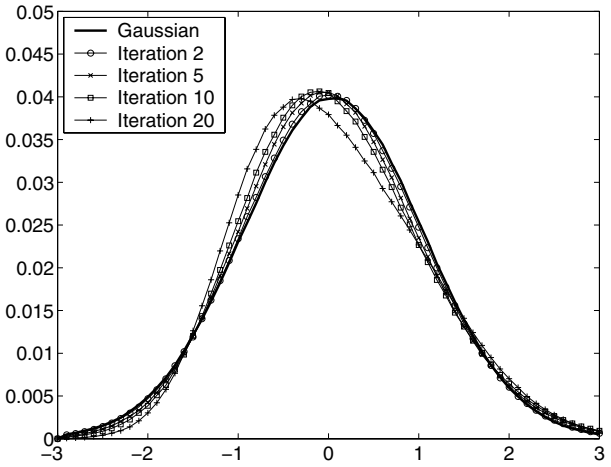

Fig. 5. Density functions of Irregular LDPC Codes at Variable nodes

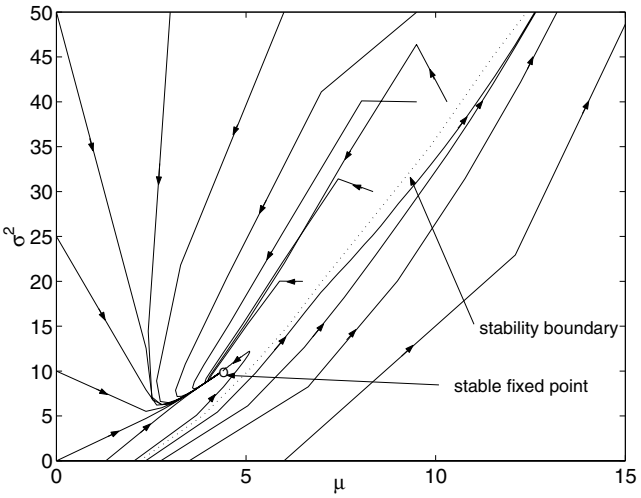

Fig. 6. Dynamic Model of Irregular LDPC Decoding

samples of $P^{\otimes i}$ can be generated by adding the samples of $P^{\otimes(i-1)}$ and the samples P. Similarly, the samples of $Q^{\otimes i}$ can be generated by using the samples of $Q^{\otimes(i-1)}$ and the samples of Q through the operation $2\tanh^{-1}\{\tanh(x/2)\tanh(y/2)\}$. Also note that for N_i samples of the averaged density, only $N_i = \lambda_i N$ (or $\rho_i N$) samples of degree i samples are needed because the samples of the averaged density can be formed by concatenating all the N_i samples of the degree i densities.

Next we consider Gaussian approximation for irregular LDPC codes. Since only the averaged density needs to be considered in each iteration, the key question here is whether the averaged density can be approximated by a Gaussian distribution if the individual densities are Gaussian or approximately Gaussian. The answer to this question is obvious no in general because the sum of two Gaussian densities (not the sum of two Gaussian variables) tends to have two peaks. However, when the individual densities are such that their means are close and their variances are not too small, the sum tends to be approximately Gaussian.

Following the discussion above, we see that the averaged density can be approximated by Gaussian distributions if the degree distributions are not very wide. This is supported by simulations as shown in Figure 5, where a rate 1/2 irregular LDPC code with $\lambda(x) = 0.38354x + 0.04237x^2 + 0.57409x^3$ and $\rho(x) = 0.24123x^4 + 0.75877x^5$ is used. The channel noisy density $\sigma_0 = 0.88$ is used. We see from the Gaussian approximation is reasonably good, although not as good as in Figure 1.

Based on Gaussian approximation, a two-input, two-output dynamic model is also naturally developed for irregular LDPC codes. Figure 6 shows the dynamic model for the same rate 1/2 irregular code but with $\sigma_0 = 0.92$. We see that the state space also exhibits two regions and a stable fixed point, but it does not appear to have an unstable fixed point. The main difference, which is rather interesting, is that the σ^2 vs. μ curves tend to grow superlinearly when μ is large, whereas in the regular LDPC case the growth is sublinear.

REFERENCES

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

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

[3] S-Y. Chung, T. J. Richardson and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 657-670, Feb. 2001.

[4] S-Y. Chung, G. D. Forney, Jr., T. J. Richardson and R. L. Urbanke, "On the design of low-density parity-check codes within 0.0045dB of the Shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58-60, Feb. 2001.

[5] D. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399-431, March 1999.

[6] N. Wiberg, "Codes and decoding on general graphs," Ph.D. Dissertation, Department of Electrical Engineering, Linköping University, 1996.

[7] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 17271737, Feb. 2001.

[8] D. Divsalar, S. Dolinar, and F. Pollara, "Iterative turbo decoder analysis based on density evolution," *IEEE J. Selected Areas in Commun.*, vol. 19, no. 5, pp. 891-907, 2001.

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

[10] J. W. Lee and R. E. Blahut, "Generalized EXIT chart and BER analysis of finite-length turbo codes," *Proc. GlobeCom 2003*, San Fransisco, Dec. 2003.

[11] R. M. Gary, *Probability, Random Processes, and Ergodic Properties*. Springer-Verlag, New York, 1988.

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

[13] N. C. Baulieu, A. Abu-Dayya and P. J. McLane, "Estimating the distribution of independent lognormal random variables," *IEEE Trans. Commun.*, vol. 43, no. 12, pp. 2869-2873, 1995.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.