THE TURBO DECODER AS A LEAST SQUARES COST GRADIENT DESCENT

John M. Walsh, C. Richard Johnson, Jr.[*]

Cornell University
School of Elec. and Comp. Eng.
Ithaca, NY 14853

Phillip A. Regalia[†]

The Catholic University of America
Dept. of Elec. Eng. and Comp. Sci.
Washington DC, 20064

ABSTRACT

In this paper we show that the iterative decoding algorithm can be viewed as descending a nonlinear least squares cost function. When at least one component code has a likelihood function that can be written as the product of its bitwise marginals, the iterative decoding algorithm is exactly a steepest descent on this cost function. Furthermore, when the iterative decoder converges to infinite log likelihood ratios, we show that its trajectories must locally descend the cost function. Conditions are then given under which the iterative decoder may be thought of as globally descending this cost function. This suggests, together with its positive definiteness, that the proposed cost function makes a suitable Lyapunov function.

1. INTRODUCTION

The introduction of turbo codes in [1] brought the performance of error control codes closer than ever to theoretically attainable limits. A key element of the success of turbo codes is the iterative decoding algorithm, which is suboptimal, yet performs remarkably well. A coherent explanation of the favorable convergence properties of the algorithm, however, remains elusive, and deducing regions of convergence or even fixed points have stumped even the best efforts. A wealth of literature has sprouted with clever explanations, although these explanations are often based on asymptotic approximations. EXIT charts [2] and density evolution [3], for example, have been generally lauded, but these methods of analysis appeal to approximations which are only valid for very large block lengths. Connections with belief propagation [4], and the sum product algorithm [5], while providing intuitive underpinnings for the algorithm, fall short of explaining convergence due to the existence of loops in the turbo decoder graph. Information geometry [6, 7, 8, 9, 10] provides for an elegant description of the decoding algorithm, but here too arguments for convergence have been inhibited by the inability to describe intrinsic information extraction as an information projection on an invariant set. Due to this time variant nature of the intrinsic information extraction, attempts at connections with projection algorithms in information spaces [11], such as the EM algorithm, have also encountered shortcomings.

In this paper, we argue for a simplistic interpretation of the iterative decoding algorithm as descent on a particular cost function. In the case where the iterative decoder has already been proven to converge [8], the iterative decoder is exactly a steepest descent on this cost function. Furthermore, when the turbo decoder converges to infinite log likelihood ratios and thus bitwise probabilities of zero or one, it must locally descend this cost function. This motivates the use of this cost function as a Lyapunov function for the turbo decoder. Finally, we show that the iterative decoder does not always descend this cost function, and thus we give conditions under which it does.

We begin the paper by introducing our notation as we provide a brief review of the operation of the turbo decoder. Next, we introduce the cost function and calculate its gradient, and note the connection between a gradient descent algorithm on this cost and the iterative decoder. A sufficient condition is given for the turbo decoder to descend this cost function, and an appeal to a simulation shows that this condition is generally not globally satisfied. However, it is shown next that whenever the turbo decoder converges bitwise probabilities of zero or one, it must at least locally be a gradient descent on this cost function. Given this connection between convergence and descent, we state conditions for global descent on this cost. We conclude the paper by mentioning possible extensions and reinterpretations of the results.

[*]John Walsh and C. Richard Johnson, Jr. were supported in part by Applied Signal Technology, Texas Instruments, and NSF Grants CCF-0310023 and INT-0233127. Contact jmw56@cornell.edu for any inquiries regarding this paper.

[†]P. A. Regalia was supported in part by the Network of Excellence in Wireless Communications (NEWCOM), E. C. Contract no. 507325 while at the Groupe des Ecoles des Télécommunications, INT, 91011 Evry France.

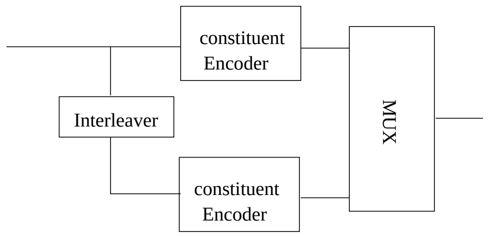

Fig. 1. A parallel concatenated turbo code. The MUX selects both the systematic and parity check bits from one of the component codes and just the parity check bits of the other. If puncturing is used, some of the parity check bits are never transmitted.

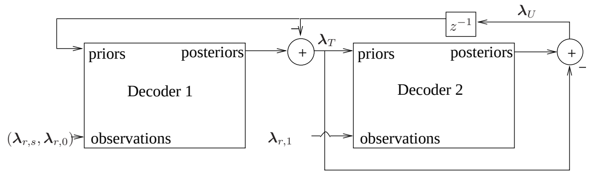

Fig. 2. The turbo decoder.

2. THE TURBO DECODER

In this section, we introduce our notation, which is heavily influenced by the information geometric analysis of turbo decoding [7, 8, 9, 10]. We also give a brief review of a description of the operation of the turbo decoder. For simplicity, we will consider the (parallel concatenated) turbo code structure as shown in Fig. 1 [1] and the iterative decoder structure as shown in Fig. 2. Let N be the number of systematic bits in a block, and let $\mathbf{B}$ be the $2^N \times N$ matrix whose ith row is binary representation of the integer i.

Let $\mathbf{B}_i$ denote the i-th row of $\mathbf{B}$. If $p(\mathbf{B}_i)$ is a probability mass function over the outcomes $\{\mathbf{B}_i\}_{i=0}^{2^N-1}$, we will find it convenient to work with its logarithmic coordinates [8, 7, 10]

$$\theta(\mathbf{B}_i) = \log p(\mathbf{B}_i) - \log p(\mathbf{B}_0)$$

which, when listed in a vector $\boldsymbol{\theta}$ for i from 0 to $2^N - 1$, are called the θ coordinates. This amounts to expressing $p(\mathbf{B}_i)$ as

$$p(\mathbf{B}_i) = \exp(\theta(\mathbf{B}_i) - \psi)$$

in which $\psi = \log\left(\sum_i \exp[\theta(\mathbf{B}_i)]\right)$ is a normalization constant to ensure that outcomes $p(\mathbf{B}_i)$ sum to one. One may show [8], [7] that a probability mass function $p(\mathbf{B}_i)$ is a product density (i.e., coincides with the product of its bitwise marginal functions) if and only if its θ coordinates become

$$\boldsymbol{\theta} = \begin{bmatrix} \theta(\mathbf{B}_0) \\ \theta(\mathbf{B}_1) \\ \vdots \\ \theta(\mathbf{B}_{2^N-1}) \end{bmatrix} = \mathbf{B}\boldsymbol{\lambda}$$

for some vector $\boldsymbol{\lambda}$, which may be identified as the log marginal ratios of the distribution:

$$\boldsymbol{\lambda}_i = \log[\mathsf{Pr}(\xi_i = 1)/\mathsf{Pr}(\xi_i = 0)]$$

Let N_1 be the number of parity check bits generated from the first component code, including any punctured bits, and let N_2 be the number of parity check bits generated by the second component code, also including any punctured bits. Suppose that the vector of bitwise log likelihood ratios (LLRs) at the output of the channel (including zeros at the locations of the punctured bits) are $\boldsymbol{\lambda}_r \in \mathbb{R}^{N+N_1+N_2}$. Reorder these into a new vector

$$\boldsymbol{\lambda}'_r = (\boldsymbol{\lambda}_{r,s}, \boldsymbol{\lambda}_{r,0}, \boldsymbol{\lambda}_{r,1})$$

so that all of the LLRs associated with the systematic bits are at the beginning of the vector $\boldsymbol{\lambda}_{r,s}$, followed by all of the LLRs associated with the parity check bits of the first code $\boldsymbol{\lambda}_{r,0}$, followed by all of the LLRs associated with the parity check bits of the second code $\boldsymbol{\lambda}_{r,1}$. Now, consider the codebook we are using. In particular, if we were to consider every possible value of the systematic bits, encoding each possibility and reordering it into the (systematic,parity check 1, parity check 2) order described above, and then stack these reordered codewords on top of each other, we would get a binary matrix of all the codewords, $\mathbf{C} \in \{0,1\}^{2^N \times (N+N_1+N_2)}$. This matrix would have the form

$$\mathbf{C} = [\mathbf{B}|\mathbf{C}_0|\mathbf{C}_1]$$

where now the i row is the (systematic, parity 1, parity 2)-reordered codeword if the systematic block that was encoded was the binary representation of the integer i. Each of the component decoders could use some of the observed channel LLRs and its own codebook to generate a wordwise likelihood function. A component decoder may then be regarded as bitwise marginalizing its likelihood function weighted with the pseudo prior wordwise pmf obtained by multiplying the marginal bitwise prior probabilities that it was given as an input. The likelihood function that the first decoder uses, for instance, has θ coordinates [7, 8, 10]

$$\boldsymbol{\theta}_{\mathbf{c}_0 \odot \mathbf{r}} = [\mathbf{B}|\mathbf{C}_0] \begin{bmatrix} \boldsymbol{\lambda}_{r,s} \\ \boldsymbol{\lambda}_{r,0} \end{bmatrix}$$

while, the likelihood function the second decoder uses has θ coordinates

$$\boldsymbol{\theta}_{\mathbf{c}_1 \odot \mathbf{r}} = \mathbf{C}_1 \boldsymbol{\lambda}_{r,1}$$

Let the extrinsic information from the first component decoder be denoted by $\boldsymbol{\lambda}_T$ and let the extrinsic information from the second decoder be denoted by $\boldsymbol{\lambda}_U$. During the turbo decoder iterations, the extrinsic information is obtained by marginalizing the word-wise densities whose θ coordinates are $\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0 \odot \mathbf{r}}$ and $\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1 \odot \mathbf{r}}$, respectively and then subtracting off intrinsic information. If we denote by π the map which takes a pmf's theta coordinates to its bitwise LLRs, then the turbo decoder may be described as iterating

$$\boldsymbol{\lambda}_T^{(k+1)} = \pi\left(\mathbf{B}\boldsymbol{\lambda}_U^{(k)} + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}\right) - \boldsymbol{\lambda}_U^{(k)} \quad (1)$$

$$\boldsymbol{\lambda}_U^{(k+1)} = \pi\left(\mathbf{B}\boldsymbol{\lambda}_T^{(k+1)} + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}\right) - \boldsymbol{\lambda}_T^{(k+1)} \quad (2)$$

Here, the second decoder's extrinsic information values from a previous iteration are used as pseudo-priors for the first decoder. The pseudo prior LLRs are then subtracted from the pseudo posterior LLRs, and the result, called the extrinsic information, is fed as pseudo priors into the second decoder, etc. The turbo decoder is said to have "converged" when the pseudo posteriors from the two decoders agree.

3. A COST FUNCTION

Consider the cost function

$$\mathsf{J}_{ls}(\boldsymbol{\lambda}_U, \boldsymbol{\lambda}_T) = \|\pi\left(\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}\right) - \pi\left(\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}\right)\|_2^2$$

In particular, since $\boldsymbol{\lambda}_U$ and $\boldsymbol{\lambda}_T$ are inherently linked in the decoding algorithm, we are going to add the constraint that $\boldsymbol{\lambda}_U$ be the extrinsic information produced by the second decoder when $\boldsymbol{\lambda}_T$ is used as a pseudo prior. Using (1) this then gives the form of the cost

$$\begin{aligned}\mathsf{J}_{ls} &= \|\pi\left(\mathbf{B}(\pi(\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}) - \boldsymbol{\lambda}_T) + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}\right) \\ &\quad - \pi\left(\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}\right)\|_2^2\end{aligned}$$

To take the gradient of the cost, it will be useful to use the following form of π [8]

$$\pi(\boldsymbol{\theta}) = \log\left(\mathbf{B}^T\exp(\boldsymbol{\theta})\right) - \log\left((\mathbf{1}-\mathbf{B})^T\exp(\boldsymbol{\theta})\right)$$

This then gives

$$\begin{aligned}&\nabla_{\boldsymbol{\lambda}_U}\pi(\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}) = \\ &\mathsf{Diag}[\mathbf{B}^T\exp(\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}})]^{-1} \\ &\mathbf{B}^T\mathsf{Diag}[\exp(\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}})]\mathbf{B} \\ &-\mathsf{Diag}[(\mathbf{1}-\mathbf{B})^T\exp(\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}})]^{-1} \\ &(\mathbf{1}-\mathbf{B})^T\mathsf{Diag}[\exp(\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}})]\mathbf{B} \\ &= \mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}\end{aligned}$$

where we introduced the notation

$$\begin{aligned}[\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}]_{i,j} &= \mathsf{Pr}_{\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}[\xi_j = 1|\xi_i = 1] \\ &\quad - \mathsf{Pr}_{\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}[\xi_j = 1|\xi_i = 0]\end{aligned}$$

where here $\mathsf{Pr}_{\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}$ means the probability with respect to the pmf whose theta coordinates are $\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}$. Defining similarly

$$\begin{aligned}[\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}}]_{i,j} &= \mathsf{Pr}_{\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}}[\xi_j = 1|\xi_i = 1] \\ &\quad - \mathsf{Pr}_{\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}}[\xi_j = 1|\xi_i = 0]\end{aligned}$$

We can now infer that

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}_T}\mathsf{J}_{ls} &= 2\Big(\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}(\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}} - \mathbf{I}) \\ &\quad - \mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}}\Big) \\ &\quad (\pi(\mathbf{B}(\pi(\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}) - \boldsymbol{\lambda}_T) + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}) \\ &\quad - \pi(\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}))\end{aligned}$$

Defining the matrix

$$\boldsymbol{\Lambda} = \left(\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}(\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}} - \mathbf{I}) - \mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}}\right)$$

then a gradient descent on J_{ls} would take the form

$$\begin{aligned}\boldsymbol{\lambda}_T^{(k+1)} &= \boldsymbol{\lambda}_T^{(k)} - 2\mu\boldsymbol{\Lambda} \\ &\quad [\pi(\mathbf{B}(\pi(\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}) - \boldsymbol{\lambda}_T) + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}) \\ &\quad - \pi(\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}})] \quad (3)\end{aligned}$$

In the next section, we connect this form with the operation of the iterative decoder.

4. GRADIENT DESCENT AND ITERATIVE DECODING

To see the connection between the iterative decoder (1) and (2), and the gradient descent (3), begin by eliminating $\boldsymbol{\lambda}_U$ between (1) and (2) to get

$$\begin{aligned}\boldsymbol{\lambda}_T^{(k+1)} &= \pi\left(\mathbf{B}\left(\pi(\mathbf{B}\boldsymbol{\lambda}_T^{(k)} + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}) - \boldsymbol{\lambda}_T^{(k)}\right) + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}\right) \\ &\quad - \pi\left(\mathbf{B}\boldsymbol{\lambda}_T^{(k)} + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}\right) + \boldsymbol{\lambda}_T^{(k)} \quad (4)\end{aligned}$$

So we see if we have

$$-2\mu\boldsymbol{\Lambda} = \mathbf{I}$$

then (3) and (1)/(2) are the same. In other words, in this case the turbo decoder is exactly a steepest gradient descent on the least squares cost J_{ls} that we have proposed. In fact, this occurs when one of the two codes admits a wordwise likelihood function which can be written as a product of its marginals. In this case, either $\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_U + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}} = \mathbf{I}$ or $\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}} = \mathbf{I}$, and this, in turn, forces $\boldsymbol{\Lambda} = -\mathbf{I}$, which gives $\mu = \frac{1}{2}$. Thus, we have the following theorem.

Thm. 1: When one of the two component codes admits a likelihood function which can be written as the product of the bitwise marginals of its systematic bits, the turbo decoder is exactly a gradient descent on J_{ls} with step size $\mu = \frac{1}{2}$.

In fact, existing stability results [8], show that in this case the turbo decoder converges in one iteration, thus the minimum of the cost is found in one step.

Building on this result, we can see that J_{ls} is a natural Lyapunov function for the system, in that the system converges to bitwise probabilities of zero or one only if it at least locally descends J_{ls}. This specific case for the fixed point encompasses a majority of the cases of the actual use of turbo decoder, seeing as having all zero or one bitwise probabilities is a common occurrence when the decoder is operating properly.

Cor. 1: If the iterative decoder converges to bitwise marginal probabilities all in the set $\{0, 1\}$ then it must descend J_{ls} at least within the vicinity of the convergent point.

To see this, simply note that if all the bitwise marginal probabilities are one or zero, so that

$$\mathsf{Pr}_{\mathbf{B}\boldsymbol{\lambda}_U+\boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}[\xi_i = 1] \in \{0,1\} \forall i \in \{0,\ldots,N-1\}$$

then only one binary word, call it $\mathbf{y}$, has any probability, and thus the wordwise pmf is a Kronecker delta function.

$$\mathsf{Pr}_{\mathbf{B}\boldsymbol{\lambda}_U+\boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}[\boldsymbol{\xi}] = \begin{cases} 1 & \boldsymbol{\xi} = \mathbf{y} \\ 0 & \text{o.w.} \end{cases}$$

This, in turn, can be written as the product of indicator functions for the proper values for each of the bits.

$$\mathsf{Pr}_{\mathbf{B}\boldsymbol{\lambda}_U+\boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}[\boldsymbol{\xi}] = \prod_{i=0}^{N-1} \delta[\boldsymbol{\xi}_i - \mathbf{y}_i]$$

where δ is the Kronecker delta function. This then gives $\boldsymbol{\Lambda} = -\mathbf{I}$, since the second order marginals are just the product of the first order marginals, and the turbo decoder is thus, at least within the vicinity of this fixed point, a steepest gradient descent on this cost function. $\square$

More generally, it is possible to sacrifice steepest gradient descent for descent. If

$$\mathbf{H} \quad = \quad \boldsymbol{\Lambda} + \boldsymbol{\Lambda}^T \tag{5}$$

can be shown to be negative definite, then the gradient will still point downhill, and (1) and (2) combined will move in a direction that is downhill with respect to the cost J_{ls}. To see this, let $\mathbf{x}$ denote a gradient vector, pointing in the steepest ascent direction, so that

$$\begin{aligned} \mathbf{x} \quad = \quad & \boldsymbol{\Lambda}\left(\pi(\mathbf{B}(\pi(\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}) - \boldsymbol{\lambda}_T) + \boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}\right) \\ & -\pi\left(\mathbf{B}\boldsymbol{\lambda}_T + \boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}\right)) \end{aligned}$$

Then $-\mathbf{x}$ points in the steepest descent direction. Observe now that

$$\mathbf{x}^T\mathbf{H}^{-1}\mathbf{x} < 0$$

if (5) holds and $\boldsymbol{\Lambda}$ is negative definite, so that the subspace angle between $-\mathbf{x}$ and $\mathbf{H}\mathbf{x}$ is less than 90 degrees. Hence

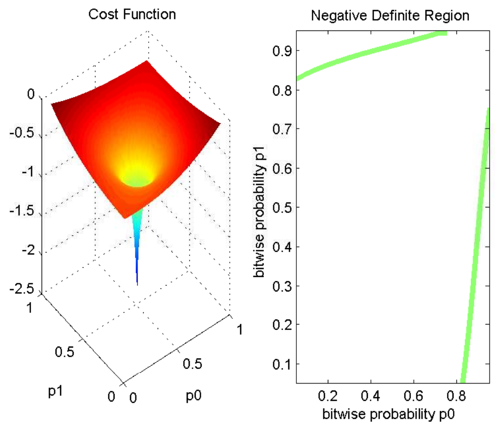

Fig. 3. J_{ls} alongside a plot showing which regions in which $\mathbf{H}$ is negative definite for a block size of 2 bits, a simple parity check code, and a particular noise realization. The x and y axes in this plot are the bitwise marginal probabilities that the corresponding bit is equal to one. Here, the large interior is where $\mathbf{H}$ is negative definite.

$\boldsymbol{\Lambda}\mathbf{x}$ still points downhill. Here we used the fact that if $\mathbf{H}^{-1}$ is negative definite, then $\mathbf{H}$ is negative definite.

A natural question to ask now is whether or not we can always expect $\mathbf{H}$ to be negative definite everywhere. The unfortunate answer is no. Fig. 3 shows the proposed cost function along side a plot dividing the probabilities associated with two bits under $\boldsymbol{\lambda}_T$ into the regions in which $\mathbf{H}$ is negative definite and in which $\mathbf{H}$ is not negative definite for a particular noise realization. Similarly, the plot in Fig. 4 shows the three bit case for a particular noise realization. Here, it is ”under” the drawn surface that $\mathbf{H}$ is negative definite. One observation is that the set in which $\mathbf{H}$ is negative definite does not appear to be convex.

Still, the arguments above show that if the noise realization and code structure allow for $\mathbf{H}$ to be everywhere negative definite, the iterative decoder globally descends J_{ls}. We summarize this conclusion in the final theorem of the paper.

Thm. 2: If

$$\begin{aligned} \mathbf{H} = \mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_U+\boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}(\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_T+\boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}} - \mathbf{I}) - \mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_T+\boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}} \\ + \left(\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_U+\boldsymbol{\theta}_{\mathbf{c}_0\odot\mathbf{r}}}(\mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_T+\boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}} - \mathbf{I}) - \mathbf{Q}_{\mathbf{B}\boldsymbol{\lambda}_T+\boldsymbol{\theta}_{\mathbf{c}_1\odot\mathbf{r}}}\right)^T \\ = \boldsymbol{\Lambda} + \boldsymbol{\Lambda}^T \end{aligned}$$

is negative definite $\forall \boldsymbol{\lambda}_T \in \mathbb{R}^N$, then the turbo decoder descends J_{ls}.

In the case that $\mathbf{H}$ were negative definite for all $\boldsymbol{\lambda}_T$, then the iterative decoder would move in a direction of decreasing J_{ls} at every step. This does not imply, however, that the turbo decoder converges, for cyclic behavior can also be explained in this framework, corresponding to the step size $\mu = \frac{1}{2}$ being too large. This suggests that the connection to J_{ls} under these conditions could analyze both limiting fixed point stable and cyclic behavior. At the very least, under the conditions in Thm. 2, J_{ls} provides an intuitive ”property restoral” framework to the behavior of the iterative decoder.

5. CONCLUSIONS

In this paper, we introduced a least squares cost function that the turbo decoder was then shown to descend. In particular, when one of the two codes used a likelihood function which was capable of being written as the product of its bitwise marginals the turbo decoder admits an exact interpretation as a gradient descent on J_{ls}. More generally, if the turbo decoder converges to a vector of bitwise marginal probabilities in $\{0, 1\}^N$, then it must at least locally descend J_{ls}. Finally, we gave conditions under which the turbo decoder may be regarded as descending J_{ls}. A salient, but also difficult, possible extension of the work would involve connecting the negative definiteness of $\mathbf{H}$ to properties of the code likelihood functions.

6. REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near shannon limit error-correcting coding and decoding: Turbo-codes.,” in *ICC 93*, Geneva, May 1993, vol. 2, pp. 1064–1070.

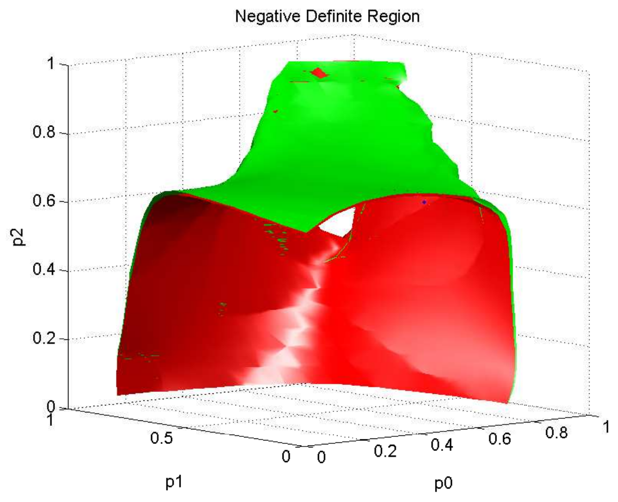

Fig. 4. Here we plot the edge of the region where $\mathbf{H}$ is negative definite for a particular noise realization and a 3 bit block with a single parity check code. Here, it is ”under” the surface that $\mathbf{H}$ is negative definite.

[2] S. ten Brink, “Convergence behavior of iteratively decoded parallel concatenated codes.,” *IEEE Trans. Commun.*, vol. 49, pp. 1727–1737, Oct. 2001.

[3] H. El Gamal and A. R. Hammons, Jr., “Analyzing the turbo decoder using the gaussian approximation,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 671–686, Feb. 2001.

[4] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, “Turbo decoding as an instance of pearls belief propagation algorithm.,” *IEEE J. Select. Areas Commun.*, vol. 16, pp. 140–152, Feb. 1998.

[5] F. R. Kshischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[6] M. Moher and T. A. Gulliver, “Cross-entropy and iterative decoding.,” *IEEE Trans. Inform. Theory*, vol. 44, pp. 3097–3104, Nov. 1998.

[7] S. Ikeda, T. Tanaka, and S. Amari, “Information geometry of turbo and low-density parity-check codes,” *IEEE Trans. Inform. Theory*, vol. 50, pp. 1097 – 1114, June 2004.

[8] T. Richardson, “The geometry of turbo-decoding dynamics.,” *IEEE Trans. Inform. Theory*, vol. 46, pp. 9–23, Jan. 2000.

[9] P. Duhamel B. Muquest and M. de Courville, “Geometrical interpretations of iterative ’turbo’ decoding,” in *Proceedings ISIT*, June 2002.

[10] J. Walsh, P. Regalia, and C. R. Johnson, Jr., “A refined information geometric interpretation of turbo decoding,” in *To appear at ICASSP ’05*, Mar. 2005.

[11] I. Csiszár and F. Matúš, “Information projections revisited,” *IEEE Trans. Inform. Theory*, vol. 49, pp. 1474–1490, June 2003.