Signal Space Characterization of Iterative Decoding

Brendan J. Frey , **Ralf Koetter** and **Alexander Vardy**

Abstract—**By tracing the flow of computations in the iterative decoders for low density parity check codes, we are able to formulate a signal-space view for a finite number of iterations in a finite-length code. On a Gaussian channel, maximum** *a posteriori* **codeword decoding (or "maximum likelihood decoding") decodes to the codeword signal that is closest to the channel output in Euclidean distance. In contrast, we show that iterative decoding decodes to the "pseudosignal" that has highest** *correlation* **with the channel output. The set of pseudosignals corresponds to "pseudocodewords", only a vanishingly small number of which correspond to codewords. We show that some pseudocodewords cause decoding errors, but that there are also pseudocodewords that frequently correct the deleterious effects of other pseudocodewords.**

I. Introduction

SIGNAL SPACE descriptions of coding and modulation form an insightful and utilitarian foundation of digital communication theory [1]. By viewing the operation of a communication system in signal space, we gain an intuitive understanding of how the system manipulates channel signals in an attempt to achieve good performance. This intuitive picture has been used to develop signal space structures such as signal constellations, set partitions and code lattices, which are integral parts of many high-performance communication systems [2–4].

It turns out that iterative decoders such as the turbodecoder and Gallager's decoder have a signal space description [5]. We show how a broad framework for describing iterative decoders leads to a signal space view of iterative decoding. In contrast to analysis of optimal decoding of ensembles of iteratively decodable codes [6, 7], we present a signal space analysis of the *iterative decoders*. In contrast to analysis of decoders that use an infinite number of iterations in infinite-length codes [8–10], we study finite numbers of iterations in finite-length codes. In contrast to research on iterative decoding in graphs defined on single cycles [11–13], we study graphs with multiple cycles. We begin to answer the question "What are the geometric properties in signal space of iterative decoding on the Gaussian channel?"

The "computation tree" was used in [14] to describe the flow of computations involved in decoding a particular codeword bit in an iterative decoder. To analyze code graphs with a single cycle, Aji *et al.* [11], Forney *et al.* [12] and Weiss [13] independently used the computation tree (actually, a chain in this case). Computation trees for code graphs with a large number of cycles have been studied by Wiberg [14] and Frey *et al.* [5].

As shown in [14], the computation tree defines a new code, called a "pseudocode'. Pseudocodewords play a role in the iterative decoder that is similar to the role of codewords in maximum likelihood decoding, but pseudocodewords are not generally codewords of the original code.

We show that each pseudocodeword corresponds to a "pseudosignal" in signal space and that the decision made for a particular codeword bit by iterative decoding is given by the *maximum-correlation* decision (Viterbi-type decoding) and the *correlation-weighted* decision (BCJR-type decoding) in signal space. In the latter case, the weights are given by the exponentials of the correlations between the pseudosignals and the channel output.

As an example, consider the (3,2,2) code whose optimal decoder uses 4 signal space points corresponding to the 4 codewords. These points are located on diametrically opposite faces of a cube, as shown in Fig. 1. For decoding a particular bit, these 4 points can be grouped into 2 points corresponding to a bit decision of 0 (shown as dark-shaded or red) and the other 2 points corresponding to a bit decision of 1 (shown as light-shaded or green).

We can describe the (3,2,2) code by including an extra *redundant* equation in the parity-check matrix, $H = [1, 1, 1; 1, 1, 1]$, and decode it by applying Gallager's iterative decoding algorithm [15]. Information is alternately passed from the 3 codeword bits to the 2 parity-check equations and vice versa. Although the (3,2,2) code is easily decoded optimally (in any sense of the word), it provides a simple example whose 3-dimensional signal space can be visualized. At the same time, the iterative decoder is complex enough to exhibit one of the fundamental properties of the signal space structure of iterative decoders: an explosion in the number of pseudocodewords relative to the number of actual codewords.

Fig. 2 shows the pseudosignals corresponding to a bit decision of 0 (dark-shaded or red) and a bit decision of 1 (light-shaded or green) for a particular bit after 3 iterations of iterative decoding. In this case, the bit decision obtained from Viterbi-type iterative decoding is given by the pseudosignal that has highest correlation with the channel output. The marginal *a posteriori* bit probability estimate obtained from BCJR-type iterative decoding is given by weighting the decision given by each pseudosignal by the exponential of the correlation with the channel output, scaled down by the estimated noise variance.

Comparing Fig. 2 with Fig. 1, note three effects: first, the signals corresponding to codewords are retained by the iterative decoder; second, these signals are skewed, in that in one dimension they have been shrunk away from the hypercube corners (see the codeword signals); third, decoding performance is determined by a mixture of the effects of the pseudosignals for the two possible bit decisions. This example shows that there are "good" pseudosignals, which are positively correlated with corners of the hypercube that give the correct decision and "bad" pseudosignals, which are positively correlated with corners of the hypercube that give the wrong decision.

By studying the pseudosignal structure of a $(3, 2, 2)$ code and a Hamming $(7, 4, 3)$ code, we gain insight into how decoding

Brendan Frey is on the faculty of Computer Science at the University of Waterloo and the faculty of Electrical and Computer Engineering at the University of Illinois at Urbana.

Ralf Koetter is on the faculty of Electrical and Computer Engineering at the University of Illinois at Urbana. R. Koetter's work partially supported by NSF grant CAREER-CCR-9984515

Alexander Vardy is on the faculty of Electrical and Computer Engineering at the University of California in San Diego.

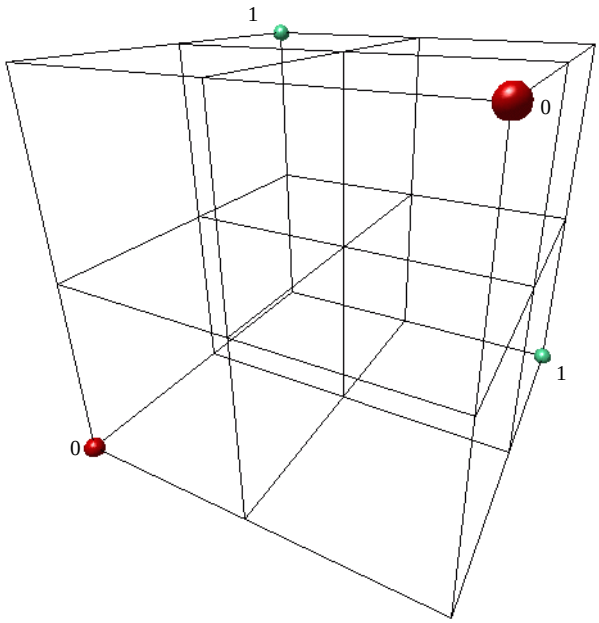

Fig. 1. Signal space positions of the codewords corresponding to a bit decision of 0 (dark-shaded or red) and a bit decision of 1 (light-shaded or green) for the (3,2,2) code.

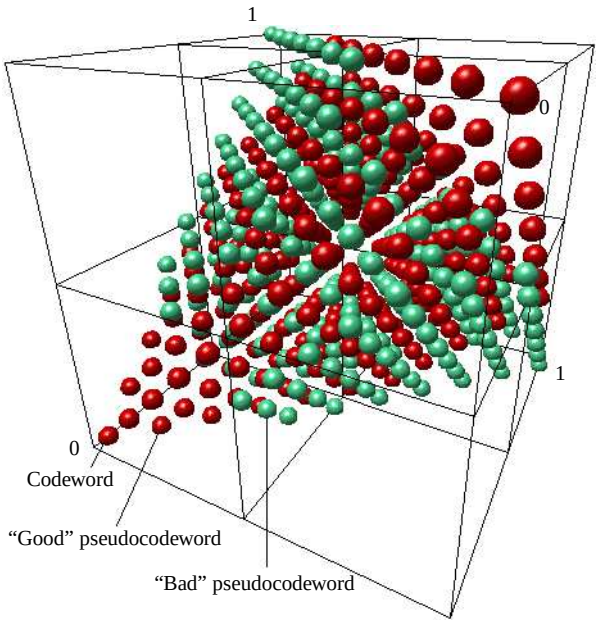

Fig. 2. Signal space positions of the pseudocodewords corresponding to a bit decision of 0 (dark-shaded or red) and a bit decision of 1 (light-shaded or green) after 3 iterations of iterative decoding for the (3,2,2) code described by the redundant parity-check matrix $H = [1, 1, 1; 1, 1, 1]$.

errors occur. For Viterbi-type iterative decoding, errors may arise when a bad pseudosignal is more highly correlated with the channel output than the codeword signal. This is the error mode used to develop a union bound on the error probability [14]. However, it turns out that in this situation, there are often good pseudosignals that are more highly correlated with the channel output than the bad pseudosignals, so that an error is not made. This result directly challenges the usefulness of the union bound.

The pseudosignal structure also lends insight into why BCJR-type iterative decoding improves on Viterbi-type iterative decoding. Even if the most highly correlated pseudosignal yields the wrong decision, there is often a cloud of highly correlated pseudosignals that yield the correct decision. In this case, the weighted average may give the correct decision.

We begin the main part of this paper with background material on a broad framework for describing codes using "factor graphs" and for describing iterative decoders as instances of the "sum-product" and "max-product" (a.k.a. "min-sum") algorithms (Sec. II). We suggest that readers familiar with the graphical model view of iterative decoding read only the definitions in this background section or see [14–16].

In Sec. IV, we use the computation tree to define a "pseudocode" that describes iterative decoding and we discuss the pseudocodes for an iteratively decoded 3-bit repetition code and a (3,2,2) code. We then show how iterative decoding emerges as correlation-based decoding in a signal space populated by "pseudosignals".

We then study two aspects of the signal space of iterative decoding: skewness and spurious pseudosignals, which do not correspond to codewords. In Sec. V, we show that one source of error in iterative decoding is caused by unequal scaling of each dimension of the signal space. We show how to compensate for "skewness" in a simple code.

Not surprisingly, the analysis of spurious pseudosignals is quite difficult. In Sec. VI, we discuss the role of bad pseudosignals, which cause decoding errors, and good pseudosignals, which can correct the errors introduced by bad pseudosignals. We show some examples of how spurious pseudosignals bear on the decoding performance in the (3,2,2) code and the Hamming (7,4,3) code. We also give a theorem on how the iterative decoder can be modified so that when it converges to a codeword, this codeword is the MAP codeword.

In Sec. VIII, we summarize our current view on the signal space structure of iterative decoding and outline some directions for further investigation.

II. Background

In this section, we review earlier work on factor graphs [17], computation trees [5, 11–14] and pseudocodes [14], and introduce our notation.

The trellis has proven to be a powerful graphical tool for understanding a variety of codes [18–20]. More recently, *graphical models* and their corresponding inference algorithms have proven to be very useful in describing the codes and iterative decoders for Gallager codes, turbocodes, serial turbocodes and product codes [16]. These graphical models include "Tanner graphs" [14, 21], Markov random fields [16, 22], Bayesian networks (a.k.a belief networks and causal diagrams) [16, 23] and "factor graphs" [17, 24].

Definition 1: *A* **factor graph** G *is a bipartite graph with one set of* N *vertices corresponding to variables* $\mathbf{x} = (x_1, \ldots, x_N)$ *and another set of* J *vertices corresponding to given local functions* $f_1, \ldots, f_J$*, such that the global function corresponding to the factor graph is given by*

$$f(\mathbf{x}) = \prod_{j=1}^{J} f_j(\mathbf{x}_j),$$

where $\mathbf{x}_j$ is the set of variables connected to f_j.

Sometimes, we associate a *singleton function* with each variable, in which case the global function is given by

$$f(\mathbf{x}) = \prod_{j=1}^{J} f_j(\mathbf{x}_j) \prod_{i=1}^{N} g_i(x_i), \tag{1}$$

where $g_1, \ldots, g_N$ are the singleton functions. In general, the functions may evaluate to any semi-ring [24, 25], but in this paper we will assume they evaluate to the real numbers.

A. Describing Codes with Factor Graphs

Iterative decoding can be viewed as the application of a simple message-passing algorithm in a factor graph that describes the constraints on the codeword symbols. In the simplest case, these constraints are just parity-check equations; in more complex cases, the constraints may contain state variables that are not part of the codeword. The most direct graphical description of a code is a factor graph for the codeword indicator function.

Definition 2: *A factor graph over codeword symbols $\mathbf{c}$ is said to "describe" a code C, if the global function f satisfies*

$$\mathbf{c} \in C \Leftrightarrow f(\mathbf{c}) = 1,$$
$$\mathbf{c} \notin C \Leftrightarrow f(\mathbf{c}) = 0.$$

The graphical structure of a code is often simplified by including some extra "state variables", which are not codeword symbols but help describe the codewords. In general, a state variable in a factor graph for a code C identifies subsets of C, but is not itself a codeword symbol.

For the sake of clarity, we will focus on graphical models without state (*e.g.*, Gallager codes), but the signal space concepts presented in this paper can also be applied to graphical models with state, (*e.g.*, turbocodes).

It is frequently useful to "fix" or "set" a variable in a factor graph to a particular value. To set x_i to the value a, let the singleton function for x_i be

$$g_i(x_i) = \begin{cases} 1 & \text{if } x_i = a \\ 0 & \text{if } x_i \neq a \end{cases} .$$

Example 1: *(7,4) Hamming code*. Fig. 3a shows the factor graph for a (7,4) Hamming code. The light discs represent codeword bits, whereas the dark discs with plus signs represent even parity indicator functions that evaluate to 1 if their variables have even parity (their sum *modulo* 2 equals 0) and evaluate to 0 otherwise. The global function is

$$f(c_1, \ldots, c_7) = f_1(c_1, c_2, c_3, c_4) f_2(c_2, c_4, c_5, c_6) \cdot f_3(c_3, c_4, c_6, c_7).$$

Given a vector of codeword symbols $\mathbf{c}$, $f(\mathbf{c}) = 1$ indicates that $\mathbf{c}$ is a codeword in the Hamming code and $f(\mathbf{c}) = 0$ indicates that $\mathbf{c}$ is not a codeword. ■

One simple approach to designing codes that turn out to give near-Shannon-limit performance is to simply connect up

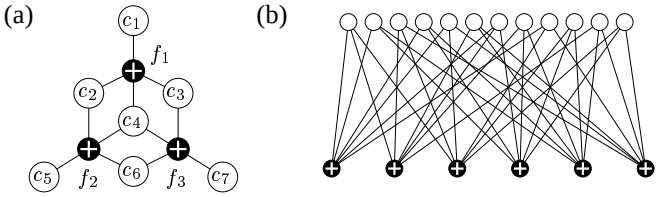

Fig. 3. Factor graphs for (a) a (7,4) Hamming code and (b) a short example of a Gallager code.

J parity-check functions to N codeword bits in a nearly random fashion. If the connectivity is made sparse (corresponding to a low-density parity-check matrix), we get a Gallager code [8, 15]. If the average degree of the parity-check nodes is d_{v} and the average degree of the codeword bit nodes is d_{f}, then the rate of the code is at least $1 - d_{\mathrm{f}}/d_{\mathrm{v}}$. Equality holds if all of the parity check equations are linearly independent. Fig. 3b shows a short (and thus not really sparsely-connected) example of a Gallager code. In this case, $d_{\mathrm{v}} = 3$, $d_{\mathrm{f}} = 6$, $R = 1/2$.

If the channel corrupts the symbols independently, the posterior distribution over codewords is

$$P(\mathbf{c}|\mathbf{y}) \propto f(\mathbf{c}) \prod_{k=1}^{N} p(y_k|c_k),$$

where $P(y_k|c_k)$, $k = 1, \ldots, N$ are the channel likelihoods. So, if we set the singleton function for each codeword symbol c_k equal to the channel likelihood $p(y_k|c_k)$, the resulting global function is proportional to the posterior distribution over the codeword symbols.

B. Computation trees for iterative decoders

The standard iterative decoders for Gallager codes [15, 26], turbocodes [27], repeat-accumulate codes [28], product codes [29] and serial turbocodes [30] can be viewed as instances of the sum-product algorithm in various factor graphs [16]. Also, Hagenauer *et al.*'s "log-likelihood algebra" [31] can be viewed as an instance of the sum-product update rule. See [14, 16] for further review.

The sum-product algorithm and its close relative the max-product algorithm specify how to compute messages representing evidence for values of variables in a graphical model and how to combine these messages locally to make inferences about the values of each variable.

By tracing the computations performed by the sum-product or max-product algorithm backward in time, we can construct a "computation tree" [5, 11–14].

Definition 3: *The* **computation tree** *$\bar{G}_k$ for an iterative decoder for codeword bit c_k in factor graph G is a factor graph constructed by creating a root node $\bar{c}_r$ corresponding to c_k in G and then recursively adding edges and leaf nodes to $\bar{G}_k$ that correspond to the messages passed in the iterative decoder. For each vertex that is created in $\bar{G}_k$, the corresponding local function or singleton function in G is copied.*

Each vertex in $\bar{G}_k$ corresponds to a unique vertex in G, but each vertex in G may have many corresponding vertices in $\bar{G}_k$. We

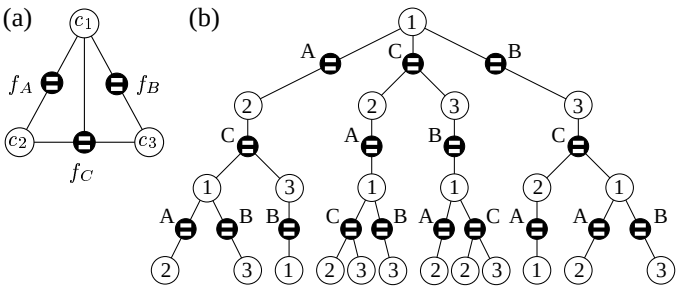

Fig. 4. (a) A factor graph for the 3 bit repetition code, where the local functions indicate equality. (b) The computation tree for 3 iterations of the alternating schedule terminating at c_1.

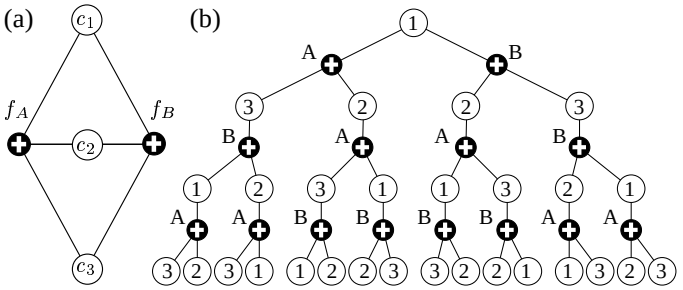

Fig. 5. (a) A factor graph for the (3,2,2) code, where the local functions indicate even parity. (b) The computation tree for 3 iterations of the alternating schedule terminating at c_1.

use an overbar, "¯", to denote variables and functions in the computation tree.

Definition 4: *For a factor graph G and a corresponding computation tree $\bar{G}_k$, the* **multiplicity** *m_j of vertex v_j in G is the number of times a copy of v_j appears in $\bar{G}_k$.*

It is particularly easy to construct the computation tree for a factor graph when the alternating message-passing schedule is used. To construct $\bar{G}_k$ for i iterations, pretend that c_k in G is the root of a tree and then copy G over to $\bar{G}_k$ using a breadth-first search – *ignoring the cycles* – until $\bar{G}_k$ has $2i$ layers of edges.

Example 2: *A computation tree for the 3-bit repetition code*. Fig. 4a shows a factor graph G for the 3-bit repetition code. We will use this example throughout this paper and we chose this highly asymmetric graph to emphasize properties of the decoder described later. There are 3 local functions that evaluate to 1 if their arguments are equal and evaluate to 0 otherwise. The computation tree $\bar{G}_1$ for 3 iterations of the alternating schedule terminating at c_1 is shown in Fig. 4b. To produce this graph using a breadth-first search, we begin by copying c_1 in G and making it the root in $\bar{G}_1$ (for visual clarity, only the subscript is shown in Fig. 4b). Next, copy f_A, f_B and f_C in G and make them children of the root in $\bar{G}_1$. For each of these "children" in G, copy their "children" over to $\bar{G}_1$. For "child" c_2 of f_A in G, copy it and make it the child of the copy of f_A just connected in $\bar{G}_1$. The final computation tree for 3 iterations has 6 layers of edges. In this computation tree, the multiplicities are $m_1 = 7$, $m_2 = 8$, $m_3 = 8$. ■

Example 3: *A computation tree for the (3,2,2) code*. Fig. 5a shows a factor graph G for the (3,2,2) code. There are 2 local functions that evaluate to 1 if their arguments have even parity and evaluate to 0 otherwise. The computation tree $\bar{G}_1$ for 3 iterations of the alternating schedule terminating at c_1 is shown in Fig. 5b. In this computation tree, the multiplicities are $m_1 = 9$, $m_2 = 10$, $m_3 = 10$. ■

C. Pseudocodes for iterative decoders

It turns out that iterative decoding is performing optimal decoding in the "pseudocode" defined by the computation tree.

Definition 5: *For an iterative decoder for codeword bit c_k in factor graph G for code C, the* **pseudocode** *$\bar{C}_k$ is the code described by the corresponding computation tree $\bar{G}_k$.*

For each codeword $\mathbf{c} \in C$, there is a corresponding pseudocodeword $\bar{\mathbf{c}} \in \bar{C}_k$ obtained by copying the value of each codeword bit c_m to the pseudocodeword bits that are copies of c_m. The pseudocodeword indicator function is a product of copies of local functions from the original factor graph. For configuration $\mathbf{c}$ the local functions in the original factor graph all evaluate to 1. So, the pseudocodeword indicator function will evaluate to 1 for the word $\bar{\mathbf{c}}$ constructed as described above.

We refer to pseudocodewords that do not have a corresponding codeword as *spurious pseudocodewords*.

Example 4: *A pseudocode for the 3-bit repetition code.* For the repetition code factor graph shown in Fig. 4a, the pseudocode described by the factor graph in Fig. 4b is also a repetition code, but has dimension $\bar{N} = 23$ instead of $N = 3$. Whereas the parameters of the original code are (3,1,3), the parameters of the new code are (23,1,23). This pseudocode does not have any spurious pseudocodewords. ■

Example 5: *A pseudocode for the (3,2,2) code.* Unlike the above example, the pseudocode described by the computation tree in Fig. 5b is very different from the original code. The pseudocode has dimension $\bar{N} = 29$ and there are 14 linearly independent parity checks. For a given pseudocodeword, we can obtain another pseudocodeword by flipping the values of any pair of leaf bits that are connected to the same check in the computation tree. So, the minimum distance is still 2. The resulting pseudocode has parameters (29,15,2), which are very different from the parameters of the original code. Since the original code has 4 codewords, this pseudocode has $2^{15} - 4 = 32764$ spurious pseudocodewords. ■

The computation tree is constructed by tracing the passage of messages in the original graph back in time. So, by construction, passing messages layer by layer from the leaves of the computation tree to the root gives evidence at the root that is equal to the evidence obtained by iterative propagation in the original graph. Also, since the root of the computation tree has been influenced by all other vertices in the computation tree, the evidence at the root is exact.

It follows that the evidence computed by the sum-product or max-product algorithm for variable c_k in factor graph G is equal

to the exact marginal or maximum for the root variable $\bar{c}_r$ in the computation tree $\bar{G}_k$.

This fact is used later to derive a signal space view of iterative decoding.

III. New Results on Computation Trees and Pseudocodes

For a given computation tree, the multiplicities can be computed using a recursion formula. Consider an augmented multiplicity vector $\mathbf{m}$ whose first N components are the multiplicities of the codeword bits and whose last set of components are the multiplicities of the local functions. Let $\mathbf{m}^{(l)}$ be the augmented multiplicity vector for a computation tree with l layers of edges. (For i iterations of decoding, we have $l = 2i$.) For a computation tree $\bar{G}_k$ corresponding to codeword bit c_k, let $\mathbf{m}^{(0)}$ be defined as a vector that contains a one in the kth position and zeros everywhere else. The vector $\mathbf{m}^{(1)}$ is defined as $\mathbf{m}^{(1)} = \mathbf{A}\mathbf{m}^{(0)} + \mathbf{m}^{(0)}$, where $\mathbf{A}$ is the adjacency matrix of the original graph.

Theorem 1: *For the alternating schedule, the vectors $\mathbf{m}^{(l)}$ satisfy the recurrence*

$$\mathbf{m}^{(l)} = \mathbf{A}\mathbf{m}^{(l-1)} - (\mathbf{D} - \mathbf{I})\mathbf{m}^{(l-2)}, \quad \textit{for } l \geq 2, \qquad (2)$$

where $\mathbf{A}$ is the adjacency matrix of the original graph, $\mathbf{D}$ is the diagonal degree matrix of the graph and $\mathbf{I}$ is the identity matrix.

Proof: For the 0th iteration this is obviously true, since the corresponding computation tree consists of a single vertex. A computation tree of depth 1 contains the neighbors of c_k and c_k itself, which yields the multiplicity vector $\mathbf{m}^{(1)}$. Let $\mathbf{n}^{(l)}$ be the multiplicity vector of the vertices that have distance exactly l from the root vertex. It is straightforward to verify that we have the equations $\mathbf{n}^{(0)} = \mathbf{m}^{(0)}$, $\mathbf{n}^{(1)} = \mathbf{A}\mathbf{n}^{(0)}$, and $\mathbf{n}^{(2)} = \mathbf{A}\mathbf{n}^{(1)} - \mathbf{D}\mathbf{n}^{(0)}$. For $l = 3, 4, \ldots$ the relationship

$$\mathbf{n}^{(l)} = \mathbf{A}\mathbf{n}^{(l-1)} - (\mathbf{D} - \mathbf{I})\mathbf{n}^{(l-2)} \qquad (3)$$

holds because the vertices at distance $l-1$ have exactly $\mathbf{A}\mathbf{n}^{(l-1)}$ neighbors. Among these neighbors the vertices at distance $l-2$ are counted $(\mathbf{D} - \mathbf{I})\mathbf{n}^{(l-2)}$ times. Simplifying the sum, $\mathbf{m}^{(l)} = \sum_{j=0}^{l} \mathbf{n}^{(j)}$, with $\mathbf{n}^{(-1)} = \mathbf{n}^{(-2)} = \mathbf{0}$, we obtain the claimed recurrence relation. ■

Example 6: *Multiplicities in the computation tree for the 3-bit repetition code.* Order the vertices in the factor graph in Fig. 4a (Example 2) $\{c_1, c_2, c_3, f_A, f_B, f_C\}$. The adjacency matrix is

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

TABLE I
Multiplicities for the 3-bit repetition code graph, for 0 through 20 iterations terminating at bit 1.

l	$i = l/2$	m_1	m_2	m_3
0	0	1	0	0
2	1	1	2	2
4	2	5	3	3
6	3	7	8	8
8	4	17	14	14
10	5	29	27	27
20	10	609	598	598
30	15	13,127	11,928	11,928
40	20	269,781	245,139	245,139

Using the recursion formula from Theorem 1, the multiplicities given in Table I are obtained for 0 through 20 iterations terminating at bit 1. Asymptotically, each of c_1 and c_2 appears in the tree 0.9156 times as often as c_0. ■

The recurrence in (2) can be solved for in closed form by introducing generating functions. To this end let $\mathbf{m}(z) = \sum_{l=0}^{\infty} \mathbf{m}^{(l)} z^l$. ¿From (2) we can derive the following equation

$$(\mathbf{I} - \mathbf{A}z + (\mathbf{D} - \mathbf{I})z^2)\mathbf{m}(z) = \mathbf{m}^{(0)} + z\mathbf{m}^{(0)}. \qquad (4)$$

The matrix $\mathbf{I} - \mathbf{A}z + (\mathbf{D} - \mathbf{I})z^2$ has nonzero determinant in the field of rational functions in z. Hence, equation (4) can be solved in the field of rational functions over the integers $\mathbf{Z}$ yielding $P_j(z)/Q_j(z)$ for the jth component of $\mathbf{m}(z)$, where $P_j(z)$ and $Q_j(z)$ are polynomials in $\mathbf{Z}[z]$. From the latter expression, we can derive a closed formula for the coefficients of vectors $\mathbf{m}$.

It is clear from Table I that even for small codes, the number of vertices in the computation tree grows very quickly with the number of layers.

Definition 6: *For each component j of the vector of vertex multiplicities, let γ_j be the maximum modulus solution to $Q_j(\gamma_j) = 0$. The* **growth rate** *of the computation tree is*

$$\lambda_{\max} = \max_j - \log|\gamma_j|.$$

$\lambda_{\max}$ bounds the multiplicities as follows.

Lemma 1: *The numbers $m_k^{(j)}$ and $n_k^{(j)}$ satisfy the inequality*

$$n_k^{(j)} \leq m_k^{(j)} < \exp(j(\lambda_{\max} + o(1))). \qquad (5)$$

Proof: The first inequality is trivial. By the properties of generating functions we know that the numbers $m_k^{(j)}$ are obtained by adding terms that grow exponentially in j. The fastest growing exponential term is given by $\exp(j\lambda_{\max})$, which is equivalent to the claimed statement. ■

Example 7: *Example 6 continued.*
We can solve for the multiplicities in closed form. The matrix

$\mathbf{B}(z) = \mathbf{I} - \mathbf{A}z + (\mathbf{D} - \mathbf{I})z^2$ is

$$\mathbf{B}(z) = \begin{pmatrix} 1+2\,z^2 & 0 & 0 & -z & -z & -z \\ 0 & 1+z^2 & 0 & -z & 0 & -z \\ 0 & 0 & 1+z^2 & 0 & -z & -z \\ -z & -z & 0 & 1+z^2 & 0 & \\ -z & 0 & -z & 0 & 1+z^2 & 0 \\ -z & -z & -z & 0 & 0 & 1+2\,z^2 \end{pmatrix}$$

$\mathbf{B}(z)^{-1}\mathbf{m}^{(0)}(1+z)$ gives the vector of generating functions for the multiplicities of each vertex in the the computation tree:

$$\mathbf{m}(z) = \begin{pmatrix} \frac{2z^6+z^4+z^2+1}{4z^7-4z^6+3z^5-3z^4-z^3+z^2-z+1} \\ \frac{\left(3z^2+2\right)z^2}{4z^7-4z^6+3z^5-3z^4-z^3+z^2-z+1} \\ \frac{\left(3z^2+2\right)z^2}{4z^7-4z^6+3z^5-3z^4-z^3+z^2-z+1} \\ \frac{\left(2z^4+2z^2+1\right)z}{4z^7-4z^6+3z^5-3z^4-z^3+z^2-z+1} \\ \frac{\left(2z^4+2z^2+1\right)z}{4z^7-4z^6+3z^5-3z^4-z^3+z^2-z+1} \\ \frac{\left(z^4+3z^2+1\right)z}{4z^7-4z^6+3z^5-3z^4-z^3+z^2-z+1} \end{pmatrix}.$$

For the computation tree in Example 6 we find that

$$\begin{aligned} \lambda_{\max} &= -\log\left|\frac{6\sqrt[3]{46+3\sqrt{249}}}{\left(46+3\sqrt{249}\right)^{2/3} - 5 + \sqrt[3]{46+3\sqrt{249}}}\right| \\ &\approx 0.3025 \end{aligned}$$

In comparison, from Table I we have $\log(m_1^{(40)}/m_1^{(30)})^{1/10} \approx 0.3023$. ■

These techniques for counting multiplicities are computationally feasible for medium code length. For example, evaluating (2) requires only multiplications of a vector of length $N+J$ with sparse $N+J \times N+J$ matrices which is easily accomplished for factor graphs with thousands of vertices.

IV. The Signal Space of Iterative Decoders

As described above, approximate iterative decoding in the factor graph for a code can be viewed as exact decoding in the computation tree derived from the factor graph. This computation tree is also a factor graph, and it describes a new code, called the "pseudocode". Since the sum-product and max-product algorithms are exact in trees, this view allows us to study iterative decoding and iteratively decoded codes by studying properties of the pseudocodes for the corresponding computation trees. We show that each pseudocodeword has a corresponding "pseudosignal" and that iterative decoders make decisions based on the *correlation* between pseudosignals and the channel output. As a result, iterative max-product (Viterbi-type) decoding is described by a solid angle tessellation of signal space, which is quite different from the Voronoi tessellation used by maximum-likelihood decoding.

Let $\psi : \{0,1\} \to \mathcal{R}$ be the mapping of a codeword bit to its channel signal. For binary antipodal signalling, we use

$$\psi(c_k) = \begin{cases} +1 & \text{if } c_k = 0, \\ -1 & \text{if } c_k = 1. \end{cases}$$

When the map is applied to a vector of N bits, it produces a point in $\mathcal{R}^N$. We indicate this by writing ψ in bold. Using this notation, the channel output vector $\mathbf{y}$ is a corrupted version of $\boldsymbol{\psi}(\mathbf{c})$.

For a given pseudocode, denote the vector of multiplicities for the N original codeword bits by $\mathbf{m} = (m_1, \ldots, m_N)^{\mathrm{T}}$, where "$^{\mathrm{T}}$" is vector transpose. Let $\ell_j(\bar{\mathbf{c}})$ be the number of times the bits in the pseudocodeword $\bar{\mathbf{c}}$ corresponding to bit c_j in the original code have the value 1. $\ell_j(\bar{\mathbf{c}})$ measures the weight of codeword bit c_j in the pseudocodeword. The vector of these weights is $\boldsymbol{\ell}(\bar{\mathbf{c}}) = (\ell_1(\bar{\mathbf{c}}), \ldots, \ell_N(\bar{\mathbf{c}}))^{\mathrm{T}}$. The scalar $m_j - 2\ell_j(\bar{\mathbf{c}})$ represents the weight in the jth dimension of signal space.

Definition 7: *The* **pseudosignal** *corresponding to pseudocodeword $\bar{\mathbf{c}}$ is the vector $\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}}) \in \mathcal{R}^N$.*

Since $\ell_j(\bar{\mathbf{c}}) \in \{0, \ldots, m_j\}$, we have $m_j - 2\ell_j(\bar{\mathbf{c}}) \in \{-m_j, \ldots, m_j\}$, *e.g.*, if all the bits in $\bar{\mathbf{c}}$ corresponding to c_j have the value 1, then $m_j - 2\ell_j(\bar{\mathbf{c}}) = -m_j$. In contrast to the channel signals, which can take on 2 values per dimension, $m_j - 2\ell_j(\bar{\mathbf{c}})$ can take on $2m_j + 1$ values per dimension. For the pseudosignals corresponding to codewords we have $\ell_j(\bar{\mathbf{c}}) \in \{0, m_j\}$ and, which sit on the corners of the skewed hypercube, $\bigotimes_{k=1}^{N}[-m_k, m_k]$. We note that in general different spurious pseudocodewords map to the same pseudosignal.

Theorem 2: *The bit decision made by iterative max-product decoding is given by the value of the root bit in the pseudocodeword $\bar{c}^*$ whose pseudosignal has maximum correlation with the channel output $\mathbf{y}$:*

$$\bar{c}^* = \underset{\bar{c} \in \bar{C}_k}{\operatorname{argmax}} \left(\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}})\right)^{\mathrm{T}} \mathbf{y}.$$

The probability ratio $\mathcal{E}_k(1)/\mathcal{E}_k(0)$ for bit c_k computed by iterative sum-product decoding is

$$\frac{\mathcal{E}_k(1)}{\mathcal{E}_k(0)} = \frac{\sum_{\bar{\mathbf{c}} \in \bar{C}_k : \bar{c}_r = 1} \exp[(\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}}))^{\mathrm{T}} \mathbf{y}/\sigma^2]}{\sum_{\bar{\mathbf{c}} \in \bar{C}_k : \bar{c}_r = 0} \exp[(\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}}))^{\mathrm{T}} \mathbf{y}/\sigma^2]},$$

Proof: Max-product decoding. Create a vector $\bar{\mathbf{y}} \in \mathcal{R}^{\bar{N}}$ of observations for the pseudocode as follows. Given a channel output vector $\mathbf{y} \in \mathcal{R}^N$, for $k = 1, \ldots, N$, set $\bar{y}_j = y_k$ for each pseudocodeword bit $\bar{c}_j$ that corresponds to codeword bit c_k. As described above, the bit decision made by iterative max-product decoding is given by the value of the root bit in

$$\bar{c}^* = \underset{\bar{c}}{\operatorname{argmax}}\, P(\bar{\mathbf{c}}|\bar{\mathbf{y}}). \tag{6}$$

From Sec. II we have

$$P(\bar{\mathbf{c}}|\bar{\mathbf{y}}) \propto \bar{f}(\bar{\mathbf{c}}) \prod_{j=1}^{\bar{N}} p(\bar{y}_j|\bar{c}_j), \tag{7}$$

where $\bar{f}(\bar{\mathbf{c}})$ is the pseudocodeword indicator function. Substituting (7) into (6), inserting the Gaussian channel likelihoods, $p(\bar{y}_j|\bar{c}_j) = \exp[-(\bar{y}_j - \psi(\bar{c}_j))^2/2\sigma^2]/\sqrt{2\pi\sigma^2}$, and noting that $\max_{\bar{\mathbf{c}}} \bar{f}(\bar{\mathbf{c}})(\cdot) = \max_{\bar{\mathbf{c}}\in\bar{C}_k}(\cdot)$, we obtain

$$\begin{aligned}\bar{c}^* &= \underset{\bar{c}\in\bar{C}_k}{\operatorname{argmax}} \prod_{j=1}^{\bar{N}} \exp[-(\bar{y}_j - \psi(\bar{c}_j))^2/2\sigma^2]/\sqrt{2\pi\sigma^2} \\ &= \underset{\bar{c}\in\bar{C}_k}{\operatorname{argmax}} -\sum_{j=1}^{\bar{N}} (\bar{y}_j - \psi(\bar{c}_j))^2. \end{aligned} \tag{8}$$

The sum over $\bar{N}$ terms is reduced to a sum over N terms using the multiplicities as follows:

$$\begin{aligned}&\sum_{j=1}^{\bar{N}} (\bar{y}_j - \psi(\bar{c}_j))^2 \\ &\quad= \sum_{k=1}^{N} \ell_k(\bar{\mathbf{c}})(y_k+1)^2 + (m_k - \ell_k(\bar{\mathbf{c}}))(y_k-1)^2 \\ &\quad= 2(2\boldsymbol{\ell}(\bar{\mathbf{c}}) - \mathbf{m})^{\mathrm{T}}\mathbf{y} + \sum_{j=1}^{N} m_j(y_j^2+1). \end{aligned} \tag{9}$$

Substituting this expression into (8) proves the theorem.

Sum-product decoding. As described above, the evidence obtained by iterative sum-product decoding is equal to the exact marginal for the root variable of the computation tree:

$$\frac{\mathcal{E}_k(1)}{\mathcal{E}_k(0)} = \frac{P(\bar{c}_r = 1|\bar{\mathbf{y}})}{P(\bar{c}_r = 0|\bar{\mathbf{y}})} = \frac{\sum_{\bar{\mathbf{c}}:\bar{c}_r=1} P(\bar{\mathbf{c}}|\bar{\mathbf{y}})}{\sum_{\bar{\mathbf{c}}:\bar{c}_r=0} P(\bar{\mathbf{c}}|\bar{\mathbf{y}})}$$

Substituting the Gaussian likelihoods and noting that $\sum_{\bar{\mathbf{c}}} \bar{f}(\bar{\mathbf{c}})(\cdot) = \sum_{\bar{\mathbf{c}}\in\bar{C}_k}(\cdot)$, we obtain

$$\frac{\mathcal{E}_k(1)}{\mathcal{E}_k(0)} = \frac{\sum_{\bar{\mathbf{c}}\in\bar{C}_k:\bar{c}_r=1} \exp[\sum_{j=1}^{\bar{N}} -(\bar{y}_j - \psi(\bar{c}_j))^2/2\sigma^2]}{\sum_{\bar{\mathbf{c}}\in\bar{C}_k:\bar{c}_r=0} \exp[\sum_{j=1}^{\bar{N}} -(\bar{y}_j - \psi(\bar{c}_j))^2/2\sigma^2]}$$

Substituting (9) into this expression proves the theorem. ■

In this view of iterative decoding, there is a pseudosignal $\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}})$ corresponding to each pseudocodeword $\bar{\mathbf{c}}$. When the correlation between $\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}})$ and $\mathbf{y}$ is high, the prediction made by $\bar{\mathbf{c}}$ is given high weight.

Example 8: *Pseudosignals for the 3-bit repetition code.* There are only two pseudocodewords for the computation tree shown in Fig. 4b and both correspond to codewords. It is evident from the computation tree that after 3 iterations $\mathbf{m} = (7,8,8)^{\mathrm{T}}$. For the all-zeros codeword $\bar{\mathbf{c}}_0 = (0,\ldots 0)^{\mathrm{T}}$, $\boldsymbol{\ell}(\bar{\mathbf{c}}_0) = (0,0,0)^{\mathrm{T}}$ and we have a pseudosignal $\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}}_0) = (7,8,8)^{\mathrm{T}}$. For the all-ones codeword $\bar{\mathbf{c}}_1 = (1,\ldots 1)^{\mathrm{T}}$, $\boldsymbol{\ell}(\bar{\mathbf{c}}_1) = (7,8,8)^{\mathrm{T}}$ and we have a pseudosignal $\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}}_1) = (-7,-8,-8)^{\mathrm{T}}$. ■

Example 9: *Pseudosignals for the (3,2,2) code.* Unlike the 3-bit repetition code, in this case there are many spurious pseudocodewords. ¿From Example 5, it is clear that after just 3 iterations in the factor graph of Fig. 5a there are 32,768 pseudosignals. It turns out that only 480 of the pseudosignals occupy unique positions in signal space and 4 of these unique positions correspond to codewords. Fig. 2 shows the positions of the unique pseudosignals. ■

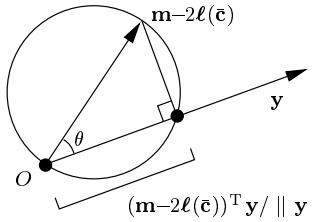

Fig. 6. A geometric expression for $(\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}}))^{\mathrm{T}}\mathbf{y}$ in signal space.

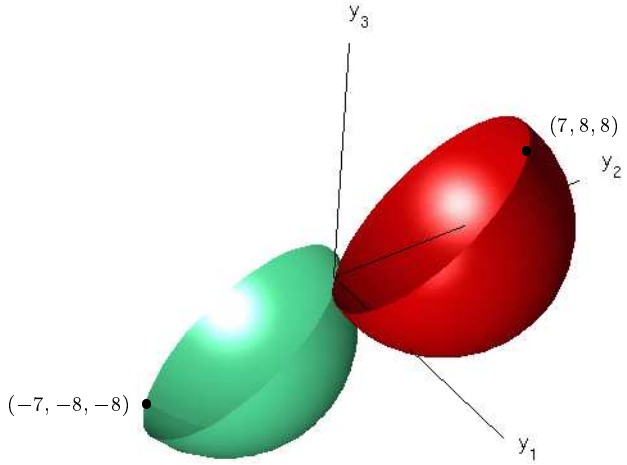

Fig. 7. A signal space view of 3 decoding iterations in the factor graph from Fig. 4 that describes the 3-bit repetition code. The dark-shaded (red) sphere corresponds to the codeword with $c_k = 0$, whereas the light-shaded (green) sphere corresponds to the codeword with $c_k = 1$.

Generally, the pseudosignals sit within a skewed hypercube with dimensions $\bigotimes_{k=1}^{N}[-m_k, m_k]$. The pseudosignals corresponding to codewords lie on the corners of this skewed hypercube.

Fig. 6 shows a simple way to express $(\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}}))^{\mathrm{T}}\mathbf{y}$ geometrically. $(\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}}))^{\mathrm{T}}\mathbf{y}$ can be viewed as the product of $\| \mathbf{y} \|$ and the distance from the origin O to the point where $\mathbf{y}$ intersects the sphere whose diameter is connected from O to $\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}})$. In other words, we project the pseudosignal onto the one-demension space that is spanned by the received vector. If the signal vector $\mathbf{y}$ does not intersect the sphere, we rewrite the correlation as $-(\mathbf{m} - 2\boldsymbol{\ell}(\bar{\mathbf{c}}))^{\mathrm{T}}(-\mathbf{y})$. Now, the above technique can be applied using the vector $-\mathbf{y}$, which obviously does intersect the sphere. Using this picture, we can visualize iterative decoding in signal space for low-dimensional codes.

For iterative max-product decoding, the bit decision is given by the bit prediction corresponding to the pseudocodeword sphere whose intersection with $\mathbf{y}$ is furthest from the origin.

Example 10: *Pseudosignal spheres for the iteratively decoded 3-bit repetition code.* The spheres used to geometrically determine signal correlations as described above are shown in Fig. 7. To give a better view, the pair of spheres have been cut

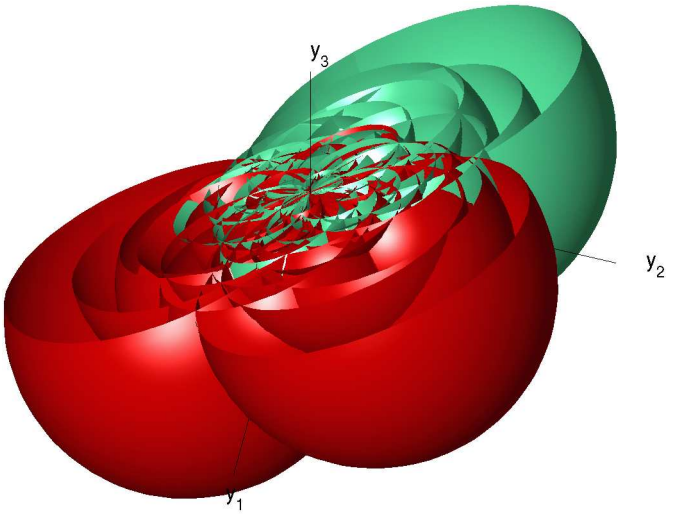

Fig. 8. A signal space view of 2 decoding iterations in the factor graph from Fig. 5a that describes the (3,2,2) code. The dark-shaded (red) spheres correspond to pseudocodewords that predict $c_k = 0$, whereas the light-shaded (green) spheres correspond to pseudocodewords that predict $c_k = 1$.

by a plane that goes through $(-7,-8,-8)$ and $(7,8,8)$. The decision boundary made for c_1 by iterative max-product decoding is given by the plane that passes through the origin and is tangent to the two spheres. Iterative max-product decoding outputs $\hat{c}_1 = 0$ for any signal on the same side of this plane as $(7,8,8)$ and outputs $\hat{c}_1 = 1$ for any signal on the opposite side. Iterative sum-product decoding will average these two decisions using weights determined from the correlations, as described in Theorem 2. In this case it is clear that the decision made by iterative max-product decoding will be the same as the decision made by iterative sum-product decoding, since in the latter case the weights are equal when the signal lies on the decision plane used by max-product decoding. ■

Example 11: *Pseudosignal spheres for the iteratively decoded (3,2,2) code.* Fig. 8 and Fig. 9 show the spheres that contribute to iterative sum-product decoding and iterative max-product decoding for 2 and 3 iterations in the factor graph of Fig. 5a. For iterative sum-product decoding, the distances given by the intersections of *all* spheres with the signal vector are used to obtain the weights used for averaging. For iterative max-product decoding, only the prediction given by the sphere that gives the largest distance is used. ■

A. Max-Product Decoding Decodes to Vertices of the Convex Hull of Pseudosignals

Many of the pseudosignal spheres in Figs. 8 and 9 are contained within the spheres of other pseudosignals. These pseudosignals will not affect the output of max-product decoding. Generally, we have the following theorem.

Theorem 3: *With probability 1 on a Gaussian channel, max-product decoding decodes to vertices of the convex hull of all pseudosignals. Every pseudosignal corresponding to a codeword is a vertex of this convex hull.*

Proof: Let the set of unique pseudosignals be $\mathbf{s}_1, \ldots, \mathbf{s}_M$ and assume the decoder decodes to just one of these pseudosignals, $\mathbf{s}_j$. Then,

$$\mathbf{s}_j^{\mathrm{T}}\mathbf{y} > \max_{m=1,\ldots,M,m\neq j} \mathbf{s}_m^{\mathrm{T}}\mathbf{y} \geq \sum_{m=1,\ldots,M,m\neq j} a_m \mathbf{s}_m^{\mathrm{T}}\mathbf{y},$$
$$\forall a_1,\ldots,a_M : a_m \geq 0 \ \ \forall m, \sum_{m=1,\ldots,M,m\neq j} a_m \leq 1.$$

Since $\sum_{m=1,\ldots,M,m\neq j} a_m \mathbf{s}_m^{\mathrm{T}}$ is a point in the convex hull of the other pseudosignals, $\mathbf{s}_j$ is outside the convex hull of the other pseudosignals. So, $\mathbf{s}_j$ is a vertex of the convex hull of all pseudosignals. The decoder decodes to more than one pseudosignal when the correlations between the channel output and 2 or more pseudosignals are equal. This happens with probability 0, proving the first part of the theorem.

Since the pseudosignals corresponding to codewords sit on the corners of the skewed hypercube, $\bigotimes_{k=1}^{N}[-m_k, m_k]$, within which all pseudosignals are contained, the pseudosignals corresponding to codewords are vertices of the convex hull. ■

Theorem 3 allows us to discard most pseudosignals for the (3,2,2) code of Fig. 2 in the context of max-product decoding. However, not all spurious pseudosignals are contained in the convex hull of the codewords. Also, all pseudosignals contribute to the final decision made by the sum-product algorithm.

B. The Solid Angle Tessellation of Max-Product Decoding

Since max-product decoding picks the pseudosignal that is most highly correlated with the channel output, only the direction of the channel output in signal space determines the decoder output. So, instead of the Voronoi tessellation used by maximum-likelihood decoding, iterative max-product decoding uses a solid angle tessellation, or, equivalently, a tessellation of the surface of the unit hypersphere in signal space.

To determine the boundary of the solid angle claimed by a pseudosignal, we take the intersection of the solid angles claimed by that pseudosignal when compared with each and every other pseudosignal. In fact, only pseudosignals at the vertices of the convex hull need to be considered, since the decoder will not decode to other pseudosignals (see Theorem 3).

The boundary between pseudosignals $\mathbf{s}_1$ and $\mathbf{s}_2$ is given by all channel outputs $\mathbf{y}$ that satisfy $\mathbf{s}_1^{\mathrm{T}}\mathbf{y} = \mathbf{s}_2^{\mathrm{T}}\mathbf{y}$, or

$$(\mathbf{s}_1 - \mathbf{s}_2)^{\mathrm{T}}\mathbf{y} = 0.$$

So, the boundary is given by the hyperplane that goes through the origin and is orthogonal to the line intersecting $\mathbf{s}_1$ and $\mathbf{s}_2$, as shown in Fig. 10.

C. Why This Interpretation?

The goal of this paper is to interpret pseudocodes in signal space. The pseudocodewords sit in a very high dimensional space, whose dimensionality depends on the number of decoding iterations. In pseudocode space, max-product decoding is equivalent to minimum distance decoding. As described above, the pseudocodewords can be mapped to the signal space of fixed

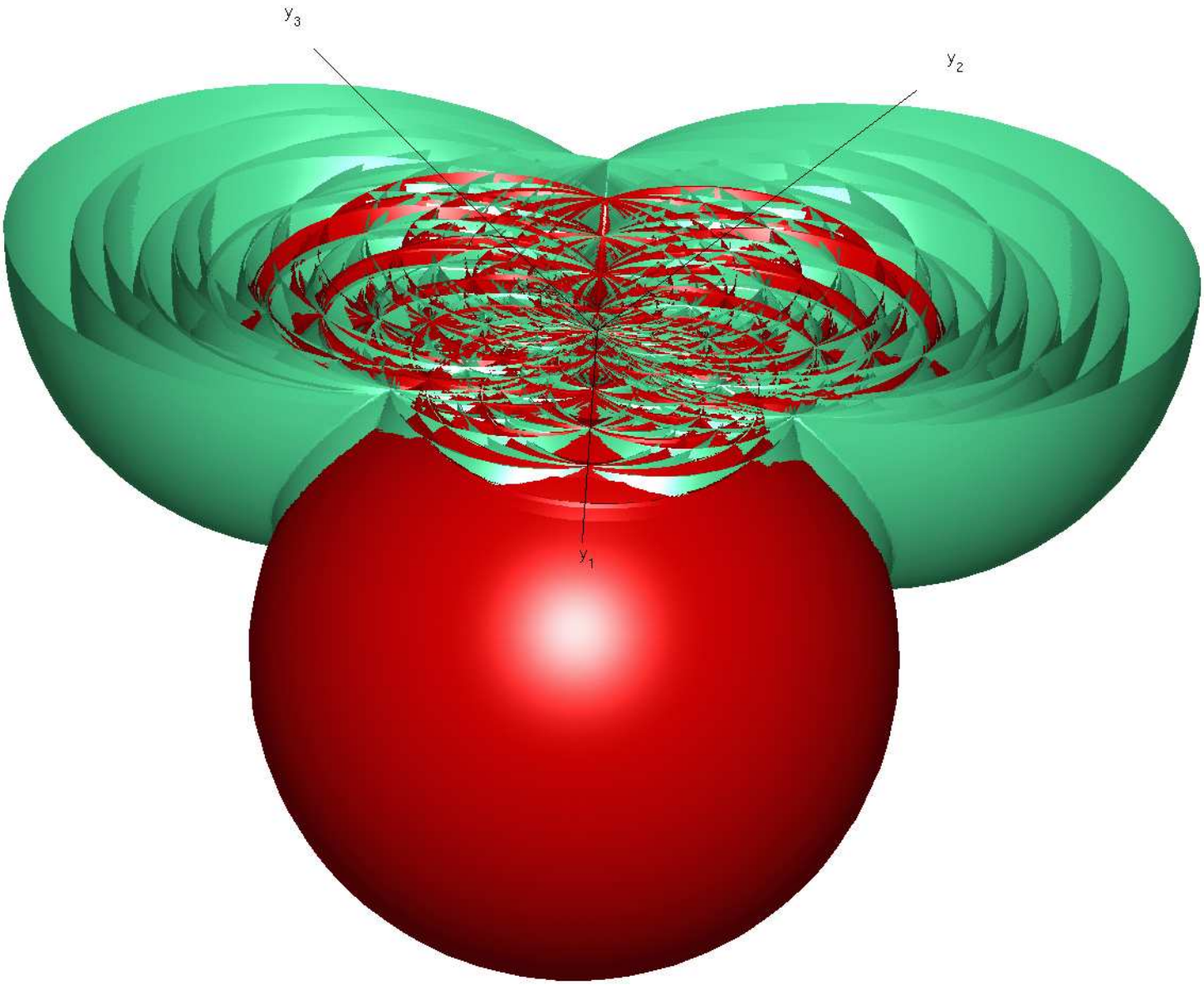

Fig. 9. A signal space view of 3 decoding iterations in the factor graph from Fig. 5 that describes the (3,2,2) code. The dark-shaded (red) spheres correspond to pseudocodewords that predict $c_k = 0$, whereas the light-shaded (green) spheres correspond to pseudocodewords that predict $c_k = 1$.

dimension, at the cost of switching from a minimum-distance geometric interpretation to a maximum-correlation interpretation. A question that arises is what new insights are gained by this description.

In pseudocode space, max-product decoding is equivalent to minimum distance decoding, so it may seem preferable to work in pseudocode space. However, isotropic noise in signal space becomes extremely anisotropic in pseudocode space. In fact, the noise in pseudocode space sits in a vanishingly low-dimensional linear subspace. So, classical characteristics of the pseudocode like weight distributions and minimum distance are mute concepts in the analysis of iterative decoding.

In order to combat this problem, Wiberg introduced the notion of pseudodistance [14], but did not associate with it a geometric interpretation. By projecting the pseudocodewords into signal space, we give pseudodistance a *geometric interpretation* as shown in Fig. 10. The pseudodistance of s_2 from a s_1 is simply twice the distance of s_1 from the marked hyperplane. A consequence that one can immediately derive from this geometric description is that any pseudosignal that lies on the line through s_1 and s_2 has the same pseudodistance from s_1. These geometric insights are much more difficult to realize in pseudocode space and we consider this geometric interpretation to be a significant advantage of the signal space description.

The projection to signal space yields a geometric characterization of the decoding regions in a space where the *noise* is isotropic and easy to describe. The moderate price one has to pay for this conversion is that minimum distance decoding is replaced by the closely related maximum correlation decoding.

Further, as described below, the interpretation of iterative decoding in signal space reveals a number of characteristics of iterative decoding like skewness, and spurious pseudosignals. Also, it is possible to give the exact shape of the decoding region for any codeword, which allows the computation of exact expres-

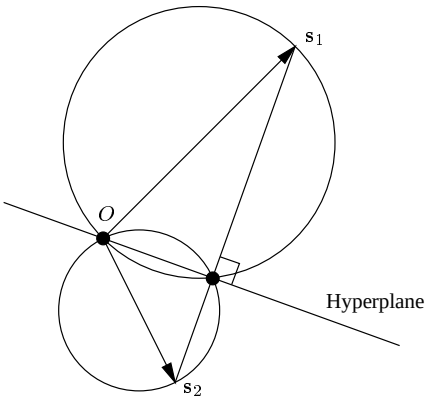

Fig. 10. The solid angle boundary between pseudosignals $\mathbf{s}_1$ and $\mathbf{s}_2$ is given by the hyperplane that goes through the origin O and is orthogonal to the line intersecting $\mathbf{s}_1$ and $\mathbf{s}_2$.

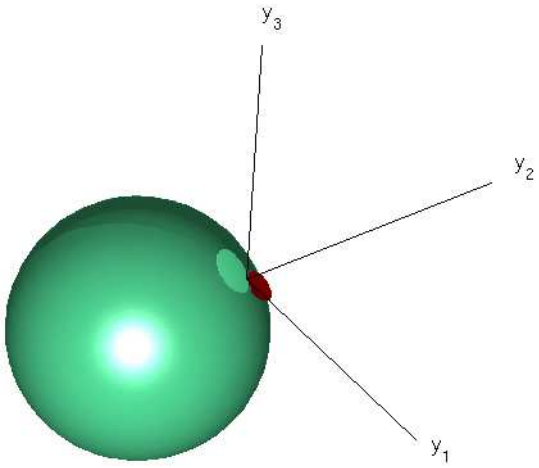

Fig. 11. When the spheres shown in Fig. 7 are cut by the correct decision plane, we see that the iterative decoder is suboptimal.

sions for the probability of decoding error.

While most of these concepts can also be derived abstractly in the context of a pseudocode alone, we find the geometric interpretations in signal space considerably helpful for interpreting the concepts.

V. Skewness in Signal Space

We might hope that if we could remove the spurious pseudocodewords (the pseudocodewords that do not correspond to codewords), the resulting decoder would perform the same as MAP word decoding or MAP bit decoding. However, it turns out that iterative decoders scale the dimensions of each signal point unequally and this "skewness" leads to decoding errors.

Let $r_j(\bar{\mathbf{c}})$ be the *fraction* of times $c_j = 1$ in the pseudocodeword $\bar{\mathbf{c}}$. So,

$$r_j(\bar{\mathbf{c}}) = \ell_j(\bar{\mathbf{c}})/m_j. \tag{10}$$

¿From Theorem 2, the iterative max-product decoder for bit c_k outputs the value of the root bit in the pseudocodeword $\bar{\mathbf{c}}^*$ satisfying

$$\bar{\mathbf{c}}^* = \operatorname*{argmax}_{\bar{\mathbf{c}} \in \bar{C}_k} (\mathbf{1} - 2\mathbf{r}(\bar{\mathbf{c}}))^{\mathrm{T}} \mathbf{M}\mathbf{y},$$

where $\mathbf{M}$ a diagonal matrix whose diagonal is the vector of multiplicities and $\mathbf{1}$ is a vector of ones with length N.

Let $\bar{C}'_k$ be the pseudocode obtained by removing all spurious pseudocodewords. Suppose for $\bar{\mathbf{c}} \in \bar{C}'_k$, $\mathbf{c}$ is the corresponding codeword. Then, $r_j(\bar{\mathbf{c}}) = 1$ if $c_j = 1$ and $r_j(\bar{\mathbf{c}}) = 0$ if $c_j = 0$. and so $\mathbf{1} - 2\mathbf{r}(\bar{\mathbf{c}}) = \boldsymbol{\psi}(\mathbf{c})$, the true signal for codeword $\mathbf{c}$. It follows that iterative max-product decoding for bit c_k outputs the value of c_k in

$$\begin{aligned}
\mathbf{c}^* &= \operatorname*{argmax}_{\mathbf{c} \in C} \boldsymbol{\psi}(\mathbf{c})^{\mathrm{T}} \mathbf{M}\mathbf{y} \\
&= \operatorname*{argmin}_{\mathbf{c} \in C} (\boldsymbol{\psi}(\mathbf{c}) - \mathbf{y})^{\mathrm{T}} \mathbf{M}(\boldsymbol{\psi}(\mathbf{c}) - \mathbf{y}) - \boldsymbol{\psi}(\mathbf{c})^{\mathrm{T}} \mathbf{M}\boldsymbol{\psi}(\mathbf{c}) \\
&= \operatorname*{argmin}_{\mathbf{c} \in C} (\boldsymbol{\psi}(\mathbf{c}) - \mathbf{y})^{\mathrm{T}} \mathbf{M}(\boldsymbol{\psi}(\mathbf{c}) - \mathbf{y}).
\end{aligned} \tag{11}$$

Clearly, this decoder is a MAP word decoder only if $\mathbf{M} \propto \mathbf{I}$, *i.e.*, the multiplicities are equal.

Example 12: *Skewed decision surface for the iteratively decoded 3-bit repetition code.* Since none of the pseudocodewords for the 3-bit repetition code are spurious, this code clearly illustrates the effect of skewness. In Example 10, we found that iterative max-product decoding uses a decision surface that intersects the origin and is tangent to the two spheres shown in Fig. 7. A normal vector for this plane is $(7, 8, 8)$. However, since the code is a repetition code, the correct decision plane has a normal vector $(1, 1, 1)$, which is not perfectly aligned with the vector used by the iterative decoder. If we cut the picture shown in Fig. 7 by the correct decision plane, we get the picture shown in Fig. 11. Incorrect decisions will be made for those signal vectors that pass through the dark-shaded (red) "cap". ■

By prescaling the channel outputs, we can attempt to compensate for skewness. Let the scaled channel output vector be

$$\tilde{\mathbf{y}} = \rho \mathbf{M}^{-1} \mathbf{y},$$

where ρ is a positive scalar. Replacing $\mathbf{y}$ with $\tilde{\mathbf{y}}$ in the derivation of (11), we see that the skewness-compensated decoder will be optimal when there aren't any spurious pseudocodewords.

Example 13: *Compensating for skewness in the 3-bit repetition code.* It was shown in Example 12 that the decision plane used by the iterative max-product decoder is skewed with respect to the optimal decision plane (see Fig. 11). By scaling the channel outputs, we can try to align the decoder's decision plane with the optimal decision plane. Here, we consider 20 iterations. From Table I, we have $\mathbf{M}^{-1} = \mathrm{diag}(1/2697817, 1/245139, 1/245139)$. A choice of ρ gives

$$\tilde{\mathbf{y}} = \rho \mathbf{M}^{-1} \mathbf{y} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.9156 & 0 \\ 0 & 0 & 0.9156 \end{pmatrix} \mathbf{y}.$$

Fig. 12 shows the simulated BER for bit 1 as a function of the value of the scale applied to y_2 and y_3 for 3 different noise levels. The empirical optimal scale factors match the theoretical prediction of 0.9156. ■

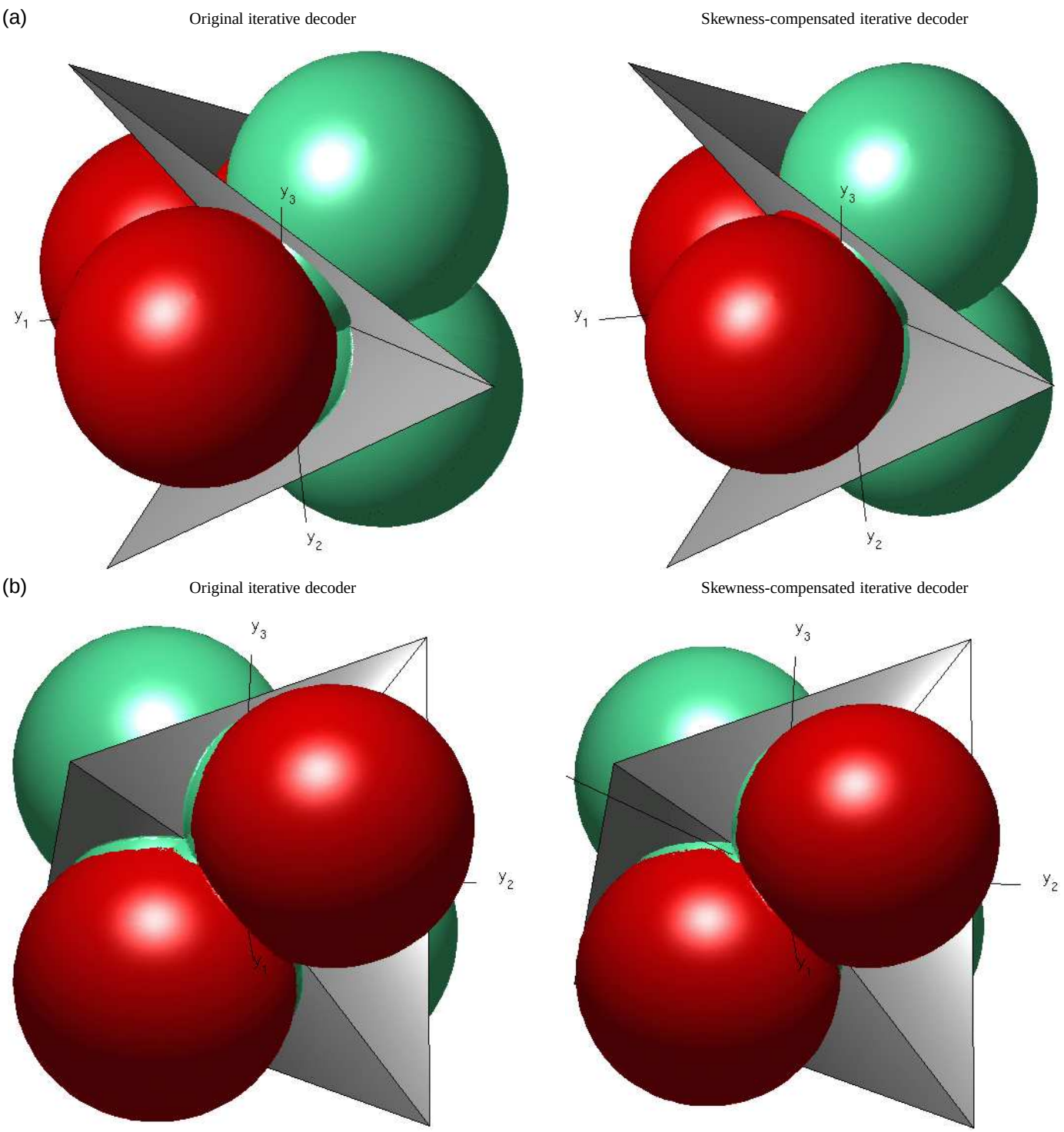

Fig. 13. Skewness compensation moves the decision boundary for 2 iterations (a) and 3 iterations (b) of max-product decoding in the (3,2,2) code closer to the optimal decision boundary. The piece-wise planar surface is the optimal decision surface for the (3,2,2) code. The dark-shaded (red) spheres correspond to pseudocodewords that predict $c_k = 0$, whereas the light-shaded (green) spheres correspond to pseudocodewords that predict $c_k = 1$.

Example 14: *Compensating for skewness in the (3,2,2) code.* ¿From Fig. 5, the multiplicity vectors for 2 and 3 iterations of iterative decoding in the (3,2,2) code are $\mathbf{m} = (5, 4, 4)^{\mathrm{T}}$ and $\mathbf{m} = (10, 9, 9)^{\mathrm{T}}$. Figs. 13a and 13b show that the skewness-compensated iterative decoders produce decision boundaries that are closer to the optimal (MAP) decision boundaries. ■

Example 15: *"Compensating" for skewness in the (7,4) Hamming code.* Now we compensate for skewness in the signal space corresponding to 10 iterations of max-product decoding for bit c_4 in the factor graph shown in Fig. 3a. Using the recur-

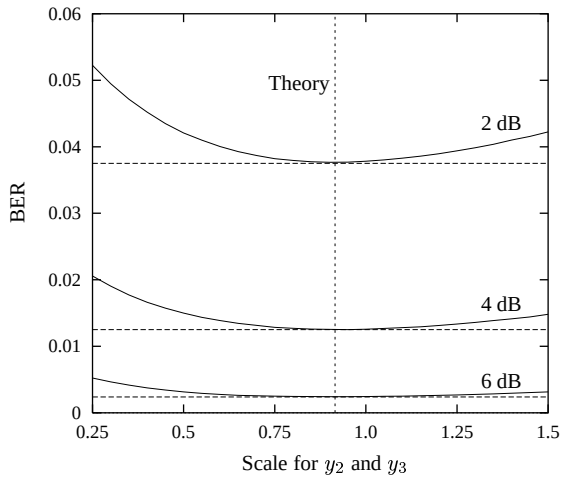

Fig. 12. Simulated BER showing the effect of scaling the channel outputs for bits 2 and 3 on iterative max-product decoding in the 3-bit repetition code factor graph of Fig. 4a. The horizontal lines give the BERs for optimal decoding; the vertical line gives the theoretical prediction of the scale factor needed to compensate for skewness.

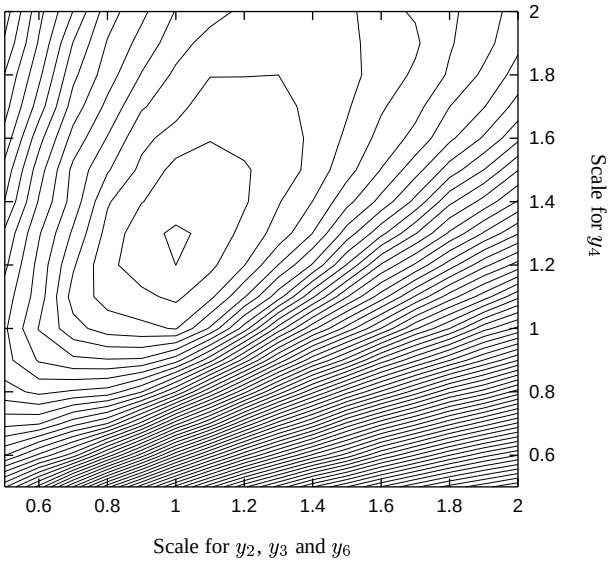

Fig. 14. Contour plot of the fraction of times the decision for c_4 made after 10 iterations of max-product decoding in Fig. 3 *disagrees* with the decision for c_4 obtained by MAP codeword decoding, as a function of two channel output scale factors. The outermost contour corresponds to a disagreement rate of 0.26 and the innermost contour corresponds to a disagreement rate of 0.13.

sion formula from Theorem 1, the multiplicity vector after 10 iterations is $\mathbf{m} = (6723, 9652, 9652, 11383, 6723, 9652, 6723, 6723, 6723, 6723)^{\mathrm{T}}$. A matrix that is proportional to $\mathbf{M}^{-1}$ is $\mathrm{diag}(1.0, 0.70, 0.70, 0.59, 1.0, 0.70, 1.0)$. So, to compensate for skewness, we ought to scale the channel output for bit 4 by 0.59 and the channel outputs for bits 2, 3, and 6 by 0.70. Fig. 14 shows a contour plot of the fraction of times the decision for c_4 made by the max-product decoder *disagrees* with the decision for c_4 obtained by MAP codeword decoding, as a function of the two scale factors. Signals were drawn from an isotropic distribution in signal space. Clearly, the best scale factors are quite different from the scale factors needed to compensate for skewness. ■

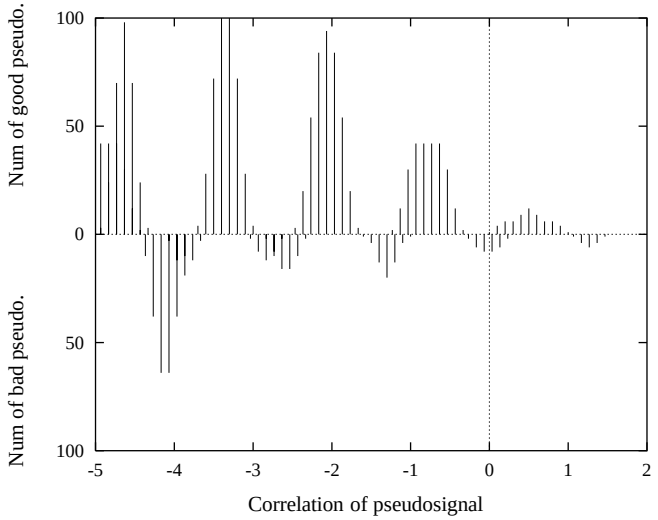

Fig. 15. Histogram of the correlations of pseudosignals with a particular channel output, after 3 iterations of decoding in the (3,2,2) code in Fig. 5. The bars above the horizontal axis give the number of pseudosignals that give the correct bit decision ("good" pseudosignals), whereas the bars below the horizontal axis give the number of pseudosignals that give the wrong bit decision ("bad" pseudosignals). The correlations are normalized by subtracting off the correlation of the pseudosignal corresponding to the MAP codeword. The decoding decision is determined by the pseudosignal with maximal corellation, which corresponds to the rightmost bar in the figure. In this case, max-product decoding makes the wrong decision.

This example shows that the effects of skewness and spurious pseudosignals cannot, in general, be treated independently.

VI. Spurious Pseudosignals

In the repetition code, there is a one-to-one correspondence between codewords and pseudocodewords, *i.e.*, there aren't any spurious pseudocodewords. So, the skewness-compensated max-product decoder finds the MAP codeword. In general, max-product decoding will decode to spurious pseudocodewords. Even for the "simple" (3,2,2) code, the iterative decoding tessellation leaks onto the wrong side of the optimal decision boundary, even after compensating for skewness (see Fig. 13a and Fig. 13b).

Fig. 15 shows a histogram of the correlations of pseudosignals with a particular channel output, after 3 iterations of decoding in the (3,2,2) code in Fig. 5. The bars above the horizontal axis give the number of pseudosignals that give the correct bit decision ("good" pseudosignals), whereas the bars below the horizontal axis give the number of pseudosignals that give the wrong bit decision ("bad" pseudosignals). The correlations are normalized by subtracting off the correlation of the pseudosignal corresponding to the MAP *codeword*. So, correlations greater than 0 (to the right of the vertical dashed line) correspond to spurious pseudosignals introduced by the iterative decoder that cause the iterative decoder to deviate from MAP decoding. In this case, we see that the pseudosignal with the highest correlation gives the wrong bit decision.

In sum-product decoding, the decision is based on a weighted average of the counts of pseudosignals, where the weights are the exponentials of the variance-normalized correlations (see

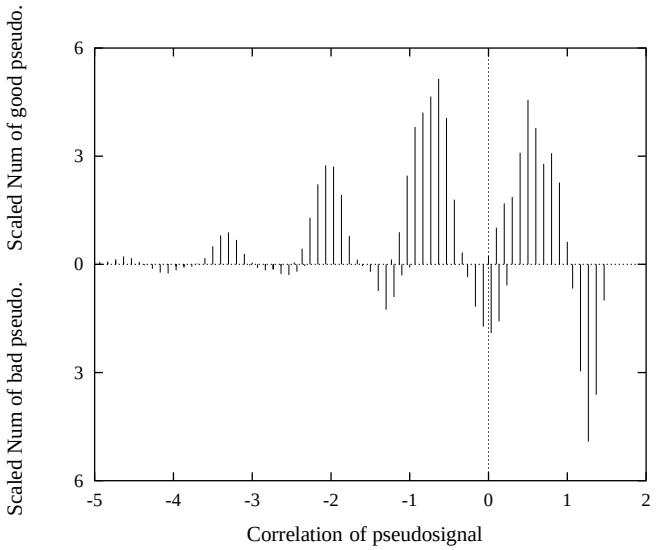

Fig. 16. Histogram from Fig. 15, with each bar scaled by the exponential of the variance-normalized correlation. For sum-product decoding, the total mass above the horizontal axis competes with the mass below the axis. In this case, sum-product decoding makes the correct decision.

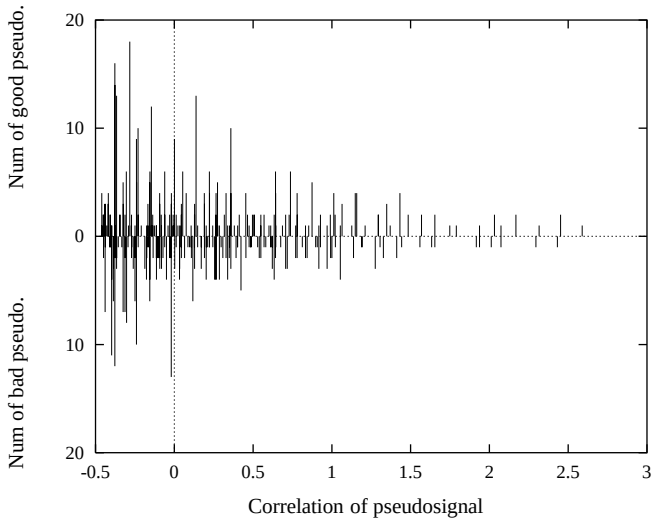

Fig. 17. Histogram of the correlations of pseudosignals with a particular channel output, after 3 iterations of decoding in the Hamming (7,4,3) code in Fig. 3a – a good pseudosignal wins.

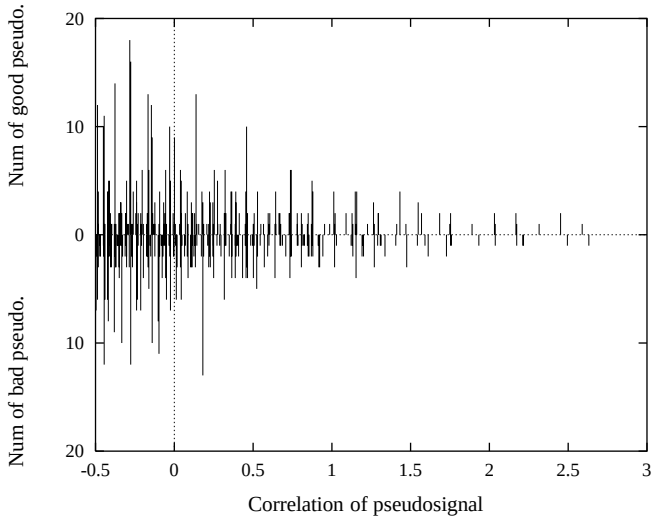

Fig. 18. Histogram of the correlations of pseudosignals with a particular channel output, after 3 iterations of decoding in the Hamming (7,4,3) code in Fig. 3a – a bad pseudosignal wins.

Theorem 2). Fig. 16 was produced by scaling each bar in Fig. 15 by the exponential of the variance-normalized correlation. So, the sum product algorithm picks the decision that has highest overall weight in this plot. Clearly, the scaled bars above the horizontal axis have more weight than the bars below the axis, so the sum-product algorithm will make a correct bit decision. In fact, even if we just consider the scaled bars for the pseudosignals that have higher correlation than the MAP codeword pseudosignal, the sum-product algorithm correctly decodes.

Fig. 17 shows the correlation histogram for a particular channel output, after 3 iterations of decoding in the (7,4,3) Hamming code in Fig. 3a. There are many bad pseudosignals that are more correlated with the channel output than the pseudosignal corresponding to the MAP codeword. Nonetheless, a good pseudosignal leads to a correct decision. The union bound on the error probability introduced in [14] is given by the probability that there exists a bad pseudocodeword that is more highly correlated with the channel output than the pseudosignal corresponding to the MAP codeword. This example directly challenges the validity of the union bound, since the competition is between good and bad pseudosignals, not the MAP codeword pseudosignal and bad pseudosignals.

Fig. 18 shows the histogram for a different channel output, where a bad pseudocodeword has the highest correlation, causing an incorrect max-product decision. If we apply the sum-product algorithm instead, we obtain the scaled histogram shown in Fig. 19, where the good pseudosignals clearly win the day and give a correct decoding decision.

VII. Attenuating Spurious Pseudosignals

In this section, we show how to attenuate the contributions of spurious pseudosignals for max-product decoding. One of the main problems in the analysis of iterative decoding is that the *leaves* of the computation tree largely determine the correlation of the corresponding pseudosignal with the channel output. Even simple statements about the behavior of the algorithm are very difficult to state rigorously. In particular, even if max-product decoding converges to a codeword, there is no guarantee that the codeword is the MAP codeword [32]. This is because the part of the computation tree that favors a certain codeword decision only contributes a small fraction of the total correlation of the corresponding pseudosignal.

We can attempt to combat the influence of the leaves by "attenuating" the messages that are passed from the leaves in the computation tree, so that when the decoder converges to a codeword, the correlation is mostly determined by the portion of the computation tree that corresponds to a codeword.

We say that an iterative max-product decoder has converged to a codeword if, for every codeword bit c_k, $k = 1, \ldots, N$ and corresponding max-product decision $\hat{c}_k$, we have

$$\underset{c_k}{\operatorname{argmax}} \mathcal{F}_{jk}(c_k) = \hat{c}_k, \quad \forall j : f_j \text{ is a neighbor of } c_k,$$

and $\hat{\mathbf{c}} \in C$. $\mathcal{F}_{jk}(c_k)$ is the message passed from function (parity

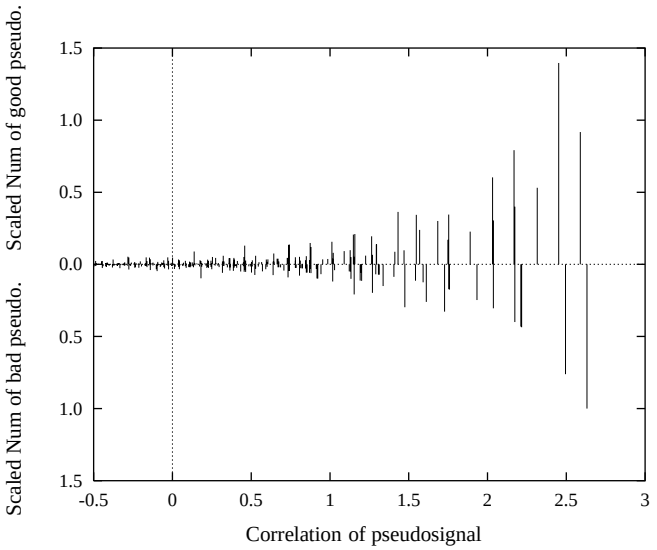

Fig. 19. Histogram from Fig. 18, with each bar scaled by the exponential of the variance-normalized correlation. For sum-product decoding, the total mass above the horizontal axis competes with the mass below the axis. In this case, sum-product decoding makes the correct decision.

check) f_j to bit c_k.

Suppose that all messages passed in each iteration are exponentiated by the same factor, β. Let $n_k^{(j)}$ denote the number of occurrences of bit c_k at distance j from the root of the computation tree, and let $p_k^{(j)}(\bar{\mathbf{c}})$ denote the number of times that $c_k = 1$ among the bits of pseudocodeword $\bar{\mathbf{c}}$ at distance j from the root.

For i iterations, the correlation of the channel output with the pseudosignal corresponding to pseudocodeword $\bar{\mathbf{c}}$ is

$$\sum_{j=0}^{i} \beta^j \sum_{k=1}^{N} p_k^{(j)}(\bar{\mathbf{c}})(y_k+1)^2 + (n_k^{(j)} - p_k^{(j)}(\bar{\mathbf{c}}))(y_k-1)^2$$
$$= 2\sum_{j=0}^{i}\sum_{k=1}^{N} y_k \beta^j (n_k^{(j)} - 2p_k^{(j)}(\bar{\mathbf{c}})) + F, \qquad (12)$$

where F is constant for all pseudocodewords $\bar{\mathbf{c}}$. So, the formula from Theorem 2 is modified such that the attenuated max-product decoder finds the pseudocodeword,

$$\bar{\mathbf{c}}^* = \operatorname*{argmax}_{\bar{\mathbf{c}}\in\bar{C}_k} \sum_{j=0}^{i} \beta^j (\mathbf{n}^{(j)} - 2\mathbf{p}^{(j)}(\bar{\mathbf{c}}))^{\mathrm{T}}\mathbf{y},$$

where $\mathbf{n}^{(j)}$ is the vector with elements $n_k^{(j)}$, and $\mathbf{p}^{(j)}(\bar{\mathbf{c}})$ is the vector with elements $p_k^{(j)}(\bar{\mathbf{c}})$.

Let $r_k^{(j)}(\bar{\mathbf{c}})$ be the attenuation-weighted fraction of times that $c_k = 1$ among the bits of pseudocodeword $\bar{\mathbf{c}}$ at distance j from the root:

$$r_k^{(j)}(\bar{\mathbf{c}}) = \beta^j p_k^{(j)}(\bar{\mathbf{c}})/n_k^{(j)}. \qquad (13)$$

Then, the attenuated max-product decoder finds the pseudocodeword,

$$\bar{\mathbf{c}}^* = \operatorname*{argmax}_{\bar{\mathbf{c}}\in\bar{C}_k} \sum_{j=0}^{i} \beta^j (\mathbf{1} - 2\mathbf{r}^{(j)}(\bar{\mathbf{c}}))^{\mathrm{T}}\mathbf{N}^{(j)}\mathbf{y}, \qquad (14)$$

where $\mathbf{N}^{(j)}$ is a diagonal matrix with diagonals $n_k^{(j)}$.

By including the effect of β, the skewness in the attenuated decoder can be compensated for in a similar way as it was in the unattenuated case (previous section):

$$\tilde{y}_k = \rho\Big(\sum_{j=0}^{\infty} \beta^j n_k^{(j)}\Big)^{-1} y_k.$$

By choosing β judiciously, we can hope to focus the above maximization on the part of the pseudocodeword that has converged to a codeword, making the above decision optimal.

Theorem 4: Attenuation Theorem. *If the attenuated, skewness-compensated max-product decoder converges to a codeword for some $\beta < \exp(-\lambda_{\max})$, then this codeword is the MAP codeword.*

Proof: Suppose that the decoder converges to a codeword after i' iterations, so that for $i \geq i'$ iterations, the bit decisions are constant and correspond to a codeword. For such a pseudocodeword $\bar{\mathbf{c}}$, denote the corresponding codeword by $\mathbf{h}(\bar{\mathbf{c}})$. So, the codeword in the converged part of the pseudocodeword (14) can be written

$$\mathbf{c}^* = \operatorname*{argmax}_{\mathbf{c}\in C} \max_{\substack{\bar{\mathbf{c}}\in\bar{C}:\\ \mathbf{h}(\bar{\mathbf{c}})=\mathbf{c}}} \Bigg(\sum_{j=0}^{i-i'} \beta^j \sum_{k=1}^{N} \psi(c_k) n_k^{(j)} \tilde{y}_k + \sum_{j=i-i'+1}^{i} \beta^j \sum_{k=1}^{N} (1 - 2r_k^{(j)}(\bar{\mathbf{c}})) n_k^{(j)} \tilde{y}_k \Bigg).$$

In the first line we used the fact that all pseudocodeword bits at distance $j \leq i - i'$ from the root, and corresponding to c_k, have the same value as c_k, so $1 - 2r_k^{(j)}(\bar{\mathbf{c}}) = \psi(c_k)$.

For any $\beta < \exp(-\lambda_{\max})$, there is a finite $\delta > 0$, such that $\beta = \exp(-(\lambda_{\max} + \delta))$. Making this substitution and using lemma 1 to bound $n_k^{(j)}$, the second term is bounded thus:

$$\sum_{j=i-i'+1}^{i} \exp(-j(\lambda_{\max}+\delta)) \sum_{k=1}^{N} (1 - 2r_k^{(j)}(\bar{\mathbf{c}})) n_k^{(j)} \tilde{y}_k$$
$$\leq \sum_{j=i-i'+1}^{i} \exp(-j(\lambda_{\max}+\delta)) \sum_{k=1}^{N} n_k^{(j)} |\tilde{y}_k|$$
$$< \sum_{j=i-i'+1}^{i} \exp(-j(\lambda_{\max}+\delta)) \sum_{k=1}^{N} \exp(j(\lambda_{\max} + o(1))) |\tilde{y}_k|$$
$$= \Big(\sum_{j=i-i'+1}^{i} \exp(-j(\delta - o(1))) \Big) \Big(\sum_{k=1}^{N} |\tilde{y}_k| \Big),$$

As $i \to \infty$, $\sum_{j=i-i'+1}^{i} \exp(-j(\delta - o(1))) \to 0$, but $\sum_{k=1}^{N} |\tilde{y}_k|$ can be kept finite (by choosing ρ appropriately), so this term

goes to 0. Thus, we have

$$\begin{aligned}\mathbf{c}^* &= \operatorname*{argmax}_{\mathbf{c}\in C} \sum_{j=0}^{\infty} \beta^j \sum_{k=1}^{N} \psi(c_k) n_k^{(j)} \tilde{y}_k \\ &= \operatorname*{argmax}_{\mathbf{c}\in C} \sum_{k=1}^{N} \psi(c_k) \Big(\sum_{j=0}^{\infty} \beta^j n_k^{(j)}\Big) \rho\Big(\sum_{j=0}^{\infty} \beta^j n_k^{(j)}\Big)^{-1} y_k. \\ &= \operatorname*{argmax}_{\mathbf{c}\in C} \sum_{k=1}^{N} \psi(c_k) y_k \\ &= \operatorname*{argmin}_{\mathbf{c}\in C} \sum_{k=1}^{N} (\psi(c_k) - y_k)^2, \end{aligned} \tag{15}$$

which is the MAP codeword. ■

From a small number of experiments, we have some anecdotal evidence for cases where β was required to be so small that the attenuated decoder did not converge. However, we are still examining the practical implications of this theorem and cannot make general statements.

VIII. Conclusions

Using the computation tree [5, 11–14], we derived a signal space description of iterative decoding. While impressive headway has been made recently in the analysis of maximum likelihood decoding of codes on graphs [6–8] and iterative decoding in infinite-length codes [8–10, 33], we focus on the difficult but important case of message passing algorithms in graphs for codes of finite length.

Using the notion of a pseudocode and a computation tree [14], we characterized the signal space of iterative decoding. In the case of the max-product algorithm, these regions are given by a solid-angle tessellation of signal space with boundaries given by hyperplanes. To our knowledge this is the first time that the decoding regions of iterative decoding have been characterized.

The signal space description of iterative decoding offers a number of insights into modes of failure and success of iterative decoding. One consequence is the effect of skewness which partially explains why certain symbols in a received word may influence a decoding decision to a larger degree than others. Also we show that the *density* of pseudocodewords plays an important roll for the decision regions of the sum-product algorithm.

The signal space characterization is an exact framework to describe the inner workings of iterative message passing algorithms. However, as expected, for non-trivial codes and a reasonable number of iterations, our characterization reflects the analytic complexity of iterative decoding. In particular the number of codewords in the pseudocode grows very fast (it can be shown that under mild conditions on the underlying factor graph the number of pseudocodewords grows doubly exponential in the number of iterations). For the future development of the signal space characterization, it will be crucial to find descriptions that allow for the efficient treatment of pseudocodes.

References

[1] E. A. Lee and D. G. Messerschmitt, *Digital Communication*, Kluwer Academic Publishers, Norwell MA., 1994.

[2] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Transactions on Information Theory*, vol. 28, no. 1, pp. 55–67, January 1982.

[3] A. R. Calderbank and N. J. A. Sloane, "New trellis codes based on lattices and cosets," *IEEE Transactions on Information Theory*, vol. 33, no. 2, pp. 177–195, March 1987.

[4] G. David Forney, Jr., "Coset codes - Part I: Introduction and geometrical classification," *IEEE Transactions on Information Theory*, vol. 34, pp. 1123–1151, September 1988.

[5] B. J. Frey, R. Koetter, and A. Vardy, "Skewness and pseudocodewords in iterative decoding," in *Proceedings of IEEE International Symposium on Information Theory*, 1998.

[6] S. Benedetto and G. Montorsi, "Unveiling turbo-codes: Some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, pp. 409–428, March 1996.

[7] D. Divsalar and R. J. McEliece, "Effective free distance of turbocodes," *Electronics Letters*, vol. 32, no. 5, pp. 445–446, February 1996.

[8] D. J. C MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, March 1999.

[9] T. J. Richardson, "Turbodecoding from a geometric perspective," in *Proceedings of IEEE International Symposium on Information Theory*, 1998.

[10] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low density parity check codes," Submitted to *IEEE Transactions on Information Theory*, July 1999.

[11] S. M. Aji, G. B. Horn, and R. J. McEliece, "On the convergence of iterative decoding on graphs with a single cycle," in *Proceedings of IEEE International Symposium on Information Theory*, 1998.

[12] G. David Forney, Jr., F. R. Kschischang, and B. Marcus, "Iterative decoding of tail-biting trellises," in *Proceedings of the 1998 Information Theory Workshop*, 1998.

[13] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Computation*, vol. 12, no. 1, pp. 1–42, 2000.

[14] N. Wiberg, *Codes and Decoding on General Graphs*, Department of Electrical Engineering, Linköping University, Linköping Sweden, 1996, Doctoral dissertation.

[15] R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge MA., 1963.

[16] B. J. Frey, *Graphical Models for Machine Learning and Digital Communication*, MIT Press, Cambridge MA., 1998, See `http://www.cs.utoronto.ca/∼frey`.

[17] B. J. Frey, F. R. Kschischang, H. A. Loeliger, and N. Wiberg, "Factor graphs and algorithms," in *Proceedings of the 35^{th} Allerton Conference on Communication, Control and Computing 1997*, 1998.

[18] G. David Forney, Jr., "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–277, March 1973.

[19] F. R. Kschischang and V. Sorokine, "On the trellis structure of block codes," *IEEE Transactions on Information Theory*, vol. 41, pp. 1924–1937, 1995.

[20] R. J. McEliece, "On the BJCR trellis for linear block codes," *IEEE Transactions on Information Theory*, vol. 42, 1996.

[21] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, pp. 533–547, 1981.

[22] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 219–230, February 1998.

[23] R. J. McEliece, D. J. C. MacKay, and J. F. Cheng, "Turbo-decoding as an instance of Pearl's 'belief propagation' algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, February 1998.

[24] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," Submitted to *IEEE Transactions on Information Theory*, available at `http://www.cs.utoronto.ca/∼frey`, July 1998.

[25] S. M. Aji and R. J. McEliece, "A general algorithm for distributing information in a graph," in *Proceedings of IEEE International Symposium on Information Theory*, 1997.

[26] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, August 1996, Reprinted in *Electronics Letters*, vol. 33, March 1997, 457–458.

[27] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Transactions on Communications*, vol. 44, pp. 1261–1271, October 1996.

[28] R. J. McEliece, D. Divsalar, and H. Jin, "Coding theorems for 'turbo-like' codes," in *Proceedings of the 35^{th} Allerton Conference on Communication, Control and Computing 1997*, 1998.

[29] J. Lodge, R. Young, P. Hoeher, and J. Hagenauer, "Separable MAP 'filters' for the decoding of product and concatenated codes," in *Proceedings*

of IEEE International Conference on Communications, 1993, pp. 1740–1745.

[30] S. Benedetto and G. Montorsi, "Iterative decoding of serially concatenated convolutional codes," *Electronics Letters*, vol. 32, pp. 1186–1188, 1996.

[31] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, March 1996.

[32] R. J. McEliece, E. R. Rodemich, and J.-F. Cheng, "The turbo decision algorithm," in *Proceedings of the* 33^{rd} *Allerton Conference on Communication, Control and Computing 1996*, 1996.

[33] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs and belief propagation," in *Proceedings of IEEE International Symposium on Information Theory*, 1998.