Convergence Analysis of Turbo Decoding of Product Codes

Assaf Sella and Yair Be'ery, *Senior Member, IEEE*

***Abstract*—Geometric interpretation of turbo decoding has founded an analytical basis, and provided tools for the analysis of this algorithm. In this paper, we focus on turbo decoding of product codes, and based on the geometric framework, we extend the analytical results and show how analysis tools can be practically adapted for this case. Specifically, we investigate the algorithm's stability and its convergence rate. We present new results concerning the structure and properties of stability matrices of the algorithm, and develop upper bounds on the algorithm's convergence rate. We prove that for any 2×2 (information bits) product codes, there is a unique and stable fixed point. For the general case, we present sufficient conditions for stability. The interpretation of these conditions provides an insight to the behavior of the decoding algorithm. Simulation results, which support and extend the theoretical analysis, are presented for Hamming $[(7, 4, 3)]^2$ and Golay $[(24, 12, 8)]^2$ product codes.**

***Index Terms*—Convergence, fixed points, geometry, product codes, stability, turbo codes.**

I. Introduction

TURBO codes, first introduced in 1993 [1], are considered one of the most important developments in coding theory in recent years. Although simulation and practical results generally show excellent performance, there is a lack of theoretical basis for explaining the results and providing tools for their analysis.

The intense research put into this field has managed to explain the excellent coding gains of the encoding scheme [3], present a unified framework for the decoding of both convolution and block turbo codes [5], and analyze simple cases [8]. But still an analysis framework was missing.

Recently, a new approach [2] based on a geometric interpretation of the decoding algorithm has managed to reveal interesting features of the decoding process. The work of Richardson in [2] formalizes the geometric representation, develops tools and conditions for analyzing the stability of the fixed points of the algorithm, their uniqueness and the proximity to maximum-likelihood decoding.

In this paper, we investigate the stability and the convergence rate of turbo decoding of product codes (without "checks on checks" [4]). We extend the analytical results and show how the analysis tools of [2] can be used in practice when applied to this scheme. We study the special structure of the stability matrices and investigate their properties.

In Section II, we present the basics of Richardson's geometric representation, and quote its main results concerning the algorithm's stability, in which a certain Jacobian matrix plays a major role.

In Section III, we present the product code turbo encoder/decoder scheme and develop practical analysis tools for this case. We show that the Jacobian and the stability matrices have a special structure, and investigate their properties. This enables an analytical study of the stability and convergence rate of the algorithm. In particular, we prove that any 2×2 (information bits) product code turbo decoder has a unique and stable fixed point. For the general case, we develop upper bounds for the magnitude of the eigenvalues of the stability matrices. These bounds provide an insight to the behavior of the decoding algorithm, and establish sufficient conditions for stability.

In Section IV, simulation results, which support and extend the theoretical analysis, are presented for Hamming $[(7, 4, 3)]^2$ and Golay $[(24, 12, 8)]^2$ product codes. We present the behavior of the maximal eigenvalues of the stability matrices for different signal-to-noise ratios (SNRs). Further analysis of the results is made using distribution histograms of the complete eigenvalues spread.

II. The Geometric Perspective

A. Notations

Let $b^0, b^1, \ldots, b^{2^k-1}$ denote the set of all possible combinations $(x_1, \ldots, x_k)$ of k information bits at the encoder's input. The sequence is enumerated as follows:

$$\begin{aligned} b^0 &= (0, 0, \ldots, 0)^T \\ b^1 &= (1, 0, \ldots, 0)^T \\ b^2 &= (0, 1, 0, \ldots, 0)^T, \ldots, b^k = (0, \ldots, 0, 1)^T \\ b^{k+1} &= (1, 1, 0, \ldots, 0)^T, \ldots, b^{2^k-1} = (1, \ldots, 1)^T. \end{aligned}$$

B is a $2^k \times k$ matrix containing all input sequences

$$B = \begin{pmatrix} (b^0)^T \\ (b^1)^T \\ \vdots \\ (b^{2^k-1})^T \end{pmatrix}.$$

A *density* p assigns a nonnegative value to each of the b^i's, proportional to its *probability density*. For convenience, we will assume that densities are strictly positive. Densities p and q are

Manuscript received February 1, 2000; revised October 19, 2000. The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Sorrento, Italy, June 2000.

The authors are with the Department of Electrical Engineering–Systems, Tel-Aviv University, Ramat-Aviv 69978, Israel (e-mail: asella@eng.tau.ac.il; ybeery@eng.tau.ac.il).

Communicated by R. J. McEliece, Guest Editor.
Publisher Item Identifier S 0018-9448(01)00725-8.

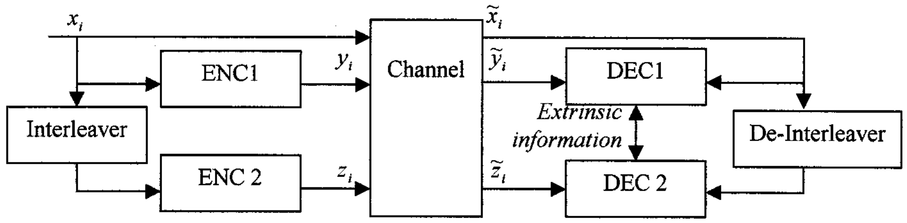

Fig. 1. The turbo decoder/encoder scheme.

equivalent (and thus belong to the same *equivalence class*) if they determine the same probability density.

Since turbo decoding (with maximum-likelihood component decoders) uses only the ratios between (probability) densities, it is invariant under equivalence. Therefore, we can choose a particular representative from each equivalence class. Richardson chose to use the density with $p(b^0) = p(0, 0, \ldots, 0) = 1$. By taking the logarithm of the representative densities, we define Φ to be the set of *log-densities* P, such that $P(b^0) = 0$ (in the sequel, upper case letters will denote log-densities, and lower case letters will denote densities).

Let H_i denote the set of all binary strings b whose ith bit is 1, and $\overline{H}_i$ the strings whose ith bit is 0. For each log-density P, we can calculate its bitwise log-likelihood values, by using the map $\tilde{\pi}(P)\colon \Re^{2^k-1} \mapsto \Re^k$

$$\tilde{\pi}(P)(i) = \log \frac{\sum_{b \in H_i} e^P(b)}{\sum_{b \in \overline{H}_i} e^P(b)}, \qquad 1 \le i \le k.$$

The map $\pi(\cdot)$ is defined as

$$\pi(P)(i) = \begin{cases} 0, & i = 0 \\ \tilde{\pi}(P)(i), & 1 \le i \le k \\ \sum_{\{j=1,\ldots,k:\, b^i(j)=1\}} \tilde{\pi}(P)(j), & k < i \le 2^k - 1. \end{cases}$$

Richardson interprets $\pi(\cdot)$ as the intersection of the surface of all log-densities having the same bitwise marginal distributions, with the space of bitwise independent log-densities.

B. Turbo Decoding

Fig. 1 shows the basic structure of a turbo decoding system.

The turbo decoder depends on the equivalence classes of $p(\tilde{x}|x)$, $p(\tilde{y}|x)$, $p(\tilde{z}|x)$. Let P_x, P_y, P_z represent these equivalence classes in Φ, respectively. Using this notation, the mth iteration of the decoding algorithm can be described by the following two update equations [with $Q_z^{(0)} = 0$]:

$$Q_y^{(m)} \leftarrow \pi\left(P_x + P_y + Q_z^{(m-1)}\right) - \left(P_x + Q_z^{(m-1)}\right) \quad (1)$$

$$Q_z^{(m)} \leftarrow \pi\left(P_x + Q_y^{(m)} + P_z\right) - \left(P_x + Q_y^{(m)}\right). \quad (2)$$

$Q_y^{(m)}$, $Q_z^{(m)}$ represent the extrinsic information from DEC1 and DEC2, respectively. If the algorithm converges, i.e., the pair $(Q_y^{(m)}, Q_z^{(m)})$ converges to (Q_y, Q_z), then the algorithm's estimation of the posterior log-likelihood is

$$L = P_x + Q_y + Q_z \quad (3)$$

and (Q_y, Q_z) is a *fixed point* of the decoding algorithm.

C. Stability of Turbo Decoding

Assuming (Q_y, Q_z) is a fixed point, we can linearize the update (1) and (2) by perturbing Q_z to $Q_z + \varepsilon_z$ and Q_y to $Q_y + \varepsilon_y$. After one decoding iteration, Q_z will be updated with $Q_z + \varepsilon_z'$, and Q_y with $Q_y + \varepsilon_y'$

$$\varepsilon_z' \cong (J_{P_x+Q_y+P_z} - I)(J_{P_x+P_y+Q_z} - I)\varepsilon_z \quad (4)$$

$$\varepsilon_y' \cong (J_{P_x+P_y+Q_z} - I)(J_{P_x+Q_y+P_z} - I)\varepsilon_y \quad (5)$$

where J_P denotes the *Jacobian* of the map π_P at 0, where $\pi_P\colon \Re^k \to \Re^k$ and $\pi_P(q) = \tilde{\pi}(P + Bq)$.

We conclude that the stability of the fixed point is determined by the stability of the matrices S_y and S_z

$$S_z = (J_{P_x+Q_y+P_z} - I)(J_{P_x+P_y+Q_z} - I) \quad (6)$$

$$S_y = (J_{P_x+P_y+Q_z} - I)(J_{P_x+Q_y+P_z} - I) \quad (7)$$

i.e., the fixed point of the turbo decoding will be stable if all the eigenvalues of S_y and S_z lie inside the unit disc.

If A and B are square matrices of the same dimensions, then AB and BA have the same eigenvalues, therefore, the stability of (6) is equivalent to the stability of (7), and it is sufficient to investigate the eigenvalues of S_y or S_z. In the sequel, we will use S to denote either S_y or S_z.

Let $\varepsilon_y^{(m)}$, $\varepsilon_z^{(m)}$ denote the distance between the extrinsic information from DEC1 and DEC2 after the mth iteration, and their fixed point values

$$\varepsilon_z^{(m)} = Q_z^{(m)} - Q_z$$
$$\varepsilon_y^{(m)} = Q_y^{(m)} - Q_y.$$

If we are in the vicinity of a fixed point, the distance after l more iterations is given by

$$\varepsilon_z^{(m+l)} \cong U \begin{pmatrix} \lambda_1^l & & & \\ & \lambda_2^l & & \\ & & \ddots & \\ & & & \lambda_k^l \end{pmatrix} U^{-1} \varepsilon_z^{(m)} \quad (8)$$

where $\lambda_1, \ldots, \lambda_k$ are the eigenvalues of S, and U is its eigenvectors matrix (note that we do not know that S is diagonalizable in general). As (8) shows, the *convergence rate* is determined by the magnitude of the eigenvalues. The most dominant is the one with the largest magnitude: $\lambda_{\max}$ ($|\lambda_{\max}| \ge |\lambda_i|$, $i = 1, \ldots, k$). The same applies to ε_y by replacing z with y.

The Jacobian of $\tilde{\pi}(\cdot)$ at P is a $k \times k$ matrix, whose elements are given by

$$(J_P)_{i,j} = \frac{\sum_{b \in H_i \cap H_j} p(b)}{\sum_{b \in H_i} p(b)} - \frac{\sum_{b \in \overline{H}_i \cap H_j} p(b)}{\sum_{b \in \overline{H}_i} p(b)}. \quad (9)$$

$(J_P)_{i,j}$ measures the dependence of the likelihood value of bit j on bit i.

$$\begin{array}{ccc|ccc}
x_1^1 & \cdots & x_{k_c}^1 & y_{k_c+1}^1 & \cdots & y_{n_y}^1 \\
\vdots & & \vdots & \vdots & & \vdots \\
x_1^{k_r} & \cdots & x_{k_c}^{k_r} & y_{k_c+1}^{k_r} & \cdots & y_{n_y}^{k_r} \\
\hline
z_{k_r+1}^1 & \cdots & z_{k_r+1}^{k_c} & & & \\
\vdots & & \vdots & & & \\
z_{n_z}^1 & \cdots & z_{n_z}^{k_c} & & &
\end{array}$$

Fig. 2. Construction of product code.

Using (9), we can calculate the stability matrix, and estimate the turbo decoding convergence rate. In order to do it, we have to calculate Jacobian matrices of $k \times k$ elements, where each element requires the calculation of four probability measures of the type $\sum_{b \in H_i} p(b)$. Brute-force calculation includes the estimation and summation of 2^{k-1} elements (all input sequences whose ith bit is 1). Even for relatively moderate values of k, this calculation may not be feasible.

III. Product Codes Turbo Decoding

We will analyze the turbo decoding algorithm in the case where *product codes* are used. Product code turbo encoders use block encoders, and a rows-to-columns interleaver.

Fig. 2 shows the code structure. The information bits are arranged in k_r rows and k_c columns $(k = k_r \cdot k_c)$. The ith row (x^{r_i}) enters a (systematic) (n_y, k_c, d_r) block encoder (C_R) and forms (the nonsystematic portion of) a row codeword

$$y^i = (y^i_{k_c+1}, \ldots, y^i_{n_y})^T, \qquad i = 1, \ldots, k_r.$$

The ith column (x^{c_i}) enters a (systematic) (n_z, k_r, d_c) block encoder (C_C) and forms (the nonsystematic portion of) a column codeword

$$z^i = (z^i_{k_r+1}, \ldots, z^i_{n_z})^T, \qquad i = 1, \ldots, k_c$$

(where d_r and d_c are the minimal distances of the row and column codes, respectively).

A. Computational Complexity of the Jacobian

Let us calculate $\sum_{x \in H_i} p(x)$for $P = P_x + P_y + Q_z$, assuming $x_i \in x^{r_a}$

$$\begin{aligned}
\sum_{x \in H_i} p(x) &= \sum_{x: x_i = 1} p_y(x) p_x(x) q_z(x) \\
&= \sum_{x: x_i=1} p(\tilde{y}/x) p_x(x) q_z(x) \\
&= \sum_{x: x_i=1} \prod_{m=1}^{k_r} p(\tilde{y}^m / x^{r_m}) p_x(x^{r_m}) q_z(x^{r_m}) \\
&= \left(\sum_{x^{r_a}: x_i=1} p(\tilde{y}^a / x^{r_a}) p_x(x^{r_a}) q_z(x^{r_a}) \right) \\
&\quad \cdot \prod_{m=1, m \neq a}^{k_r} \sum_{x^{r_m}} p(\tilde{y}^m / x^{r_m}) p_x(x^{r_m}) q_z(x^{r_m}).
\end{aligned} \tag{10}$$

Since each row is separately encoded, we can calculate the posterior probability of receiving $\tilde{y}$ by multiplying the posterior probabilities of receiving each row separately. A brute-force calculation requires the estimation and summation of 2^{k_c-1} elements in the first sum, and 2^{k_c} elements in the subsequent $k_r - 1$ sums. Therefore, the complexity is $o(k_r 2^{k_c})$ (the complexity of calculating the elements of the sum is ignored). Recall that in the general case the complexity was $o(2^{k_r k_c - 1})$. Also, note that similar complexity may be achieved by modifying the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm [6].

The same method can be used in the calculation of elements of the type $\sum_{x \in H_i \cap H_j} p(x)$

$$\begin{aligned}
&\sum_{x \in H_i \cap H_j} p(x) \\
&= \sum_{x:\, x_i=1,\, x_j=1} p_y(x) p_x(x) q_z(x) \\
&= \begin{cases}
\left(\displaystyle\sum_{x^{r_a}:\, x_i=1,\, x_j=1} p(\tilde{y}^a / x^{r_a}) p_x(x^{r_a}) q_z(x^{r_a}) \right) \\
\quad \cdot \displaystyle\prod_{m=1,\, m \neq a}^{k_r} \sum_{x^{r_m}} p(\tilde{y}^m / x^{r_m}) p_x(x^{r_m}) q_z(x^{r_m}), \\
\hfill x_i,\, x_j \in x^{r_a} \\
\left(\displaystyle\sum_{x^{r_a}:\, x_i=1} p(\tilde{y}^a / x^a) p_x(x^{r_a}) q_z(x^{r_a}) \right) \\
\quad \cdot \left(\displaystyle\sum_{x^{r_b}:\, x_j=1} p(\tilde{y}^b / x^{r_b}) p_x(x^{r_b}) q_z(x^{r_b}) \right) \\
\quad \cdot \displaystyle\prod_{m=1,\, m \neq a,b}^{k_r} \sum_{x^{r_m}} p(\tilde{y}^m / x^{r_m}) p_x(x^{r_m}) q_z(x^{r_m}), \\
\hfill x_i \in x^{r_a},\, x_j \in x^{r_b}
\end{cases}
\end{aligned} \tag{11}$$

By substituting (10) and (11) in (9), we get the general expression for a Jacobian element

$$\begin{aligned}
&(J_{P_x + P_y + Q_z})_{i,j} \\
&= \begin{cases}
\dfrac{\displaystyle\sum_{x^{r_a}:\, x_i=1,\, x_j=1} p(\tilde{y}^a / x^{r_a}) p_x(x^{r_a}) q_z(x^{r_a})}{\displaystyle\sum_{x^{r_a}:\, x_i=1} p(\tilde{y}^a / x^{r_a}) p_x(x^{r_a}) q_z(x^{r_a})} \\
\quad - \dfrac{\displaystyle\sum_{x^{r_a}:\, x_i=0,\, x_j=1} p(\tilde{y}^a / x^{r_a}) p_x(x^{r_a}) q_z(x^{r_a})}{\displaystyle\sum_{x^{r_a}:\, x_i=0} p(\tilde{y}^a / x^{r_a}) p_x(x^{r_a}) q_z(x^{r_a})}, \\
\hfill x_i \in x^{r_a},\, x_j \in x^{r_b} \\
0, \qquad x_i \in x^{r_a},\, x_j \in x^{r_b},\, a \neq b.
\end{cases}
\end{aligned} \tag{12}$$

Thus, the calculation complexity of a Jacobian element is $o(2^{k_c-1})$. We also note that the Jacobian of the rows decoding (J^R) is a block-diagonal matrix, whose ith block $(J^{R,i})$ is the $k_c \times k_c$ Jacobian matrix of the ith row

$$J^R = J_{P_x + P_y + Q_z} = \begin{pmatrix} J^{R,1} & & & \\ & J^{R,2} & & \\ & & \ddots & \\ & & & J^{R,k_r} \end{pmatrix}. \tag{13}$$

Note that J^R has only $k_r k_c^2$ nonzero elements, compared to $k_r^2 k_c^2$ elements in the general case.

The same arguments apply to the Jacobian matrix of the columns decoding (J^C), i.e., for $P = P_x + Q_y + P_z$. If the encoder's input bits x_i were numbered column after column (i.e., x_1 is the first bit of the first column, x_{k_r+1} is the first bit of the second column, etc.), the Jacobian would have a similar structure to J^R

$$\tilde{J}^C = \begin{pmatrix} \tilde{J}^{C,1} & & & \\ & \tilde{J}^{C,2} & & \\ & & \ddots & \\ & & & \tilde{J}^{C,k_c} \end{pmatrix}. \tag{14}$$

But, since we choose to number x_i row after row, J^C will have a different structure after applying the rows-to-columns permutation

$$\left(\tilde{J}^C\right)_{i,j} = \left(J^C\right)_{\substack{((i-1)\bmod k_r)k_c+\lfloor (i-1)/k_r\rfloor+1,\\ ((j-1)\bmod k_r)k_c+\lfloor (j-1)/k_r\rfloor+1}}. \tag{15}$$

We will denote with $a_{i,j}$, $a^l_{i,j}$, $b_{i,j}$, $\tilde{b}_{i,j}$, and $\tilde{b}^l_{i,j}$ the elements of J^R, $J^{R,l}$, J^C, $\tilde{J}^C$, and $\tilde{J}^{C,l}$, respectively [note that $a^l_{i,j}$ ($\tilde{b}^l_{i,j}$) measures the dependence between the ith and jth bits of the lth row (column)]. The case of $k_r = k_c = 3$ is as shown in (16) at the bottom of the page, and the Jacobian of the columns decoding is as shown in (17) also at the bottom of this page.

B. Stability Matrices

Analysis of (9) shows that each element in the Jacobian matrix is the difference between two probability ratios. In each ratio, the numerator is the sum of positive probability measures, which is strictly contained (except when $i = j$) in the denominator. Therefore, each ratio is a positive number smaller than 1, and the absolute value of their difference is less than 1

$$\begin{aligned} |(J_P)_{i,j}| &< 1, \qquad i \neq j \\ (J_P)_{i,i} &= 1. \end{aligned} \tag{18}$$

We will investigate the relations between S^R, S^C—the stability matrices of the rows and columns decoders, and S—the stability matrix of the turbo decoder. Let

$$\begin{aligned} S^R &= J^R - I \\ S^C &= J^C - I \\ S_y &= S^R S^C \\ S_z &= S^C S^R. \end{aligned} \tag{19}$$

The case of $k_r = k_c = 3$ is as shown in (20) and (21) at the bottom of the following page.

$$J^R = J_{P_x+P_y+Q_z} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & & & & & & \\ a_{2,1} & a_{2,2} & a_{2,3} & & & & & & \\ a_{3,1} & a_{3,2} & a_{3,3} & & & & & & \\ & & & a_{4,4} & a_{4,5} & a_{4,6} & & & \\ & & & a_{5,4} & a_{5,5} & a_{5,6} & & & \\ & & & a_{6,4} & a_{6,5} & a_{6,6} & & & \\ & & & & & & a_{7,7} & a_{7,8} & a_{7,9} \\ & & & & & & a_{8,7} & a_{8,8} & a_{8,9} \\ & & & & & & a_{9,7} & a_{9,8} & a_{9,9} \end{pmatrix}$$
$$= \begin{pmatrix} a^1_{1,1} & a^1_{1,2} & a^1_{1,3} & & & & & & \\ a^1_{2,1} & a^1_{2,2} & a^1_{2,3} & & & & & & \\ a^1_{3,1} & a^1_{3,2} & a^1_{3,3} & & & & & & \\ & & & a^2_{1,1} & a^2_{1,2} & a^2_{1,3} & & & \\ & & & a^2_{2,1} & a^2_{2,2} & a^2_{2,3} & & & \\ & & & a^2_{3,1} & a^2_{3,2} & a^2_{3,3} & & & \\ & & & & & & a^3_{1,1} & a^3_{1,2} & a^3_{1,3} \\ & & & & & & a^3_{2,1} & a^3_{2,2} & a^3_{2,3} \\ & & & & & & a^3_{3,1} & a^3_{3,2} & a^3_{3,3} \end{pmatrix}. \tag{16}$$

$$J^C = J_{P_x+Q_y+P_z} = \begin{pmatrix} b_{1,1} & & & b_{1,4} & & & b_{1,7} & & \\ & b_{2,2} & & & b_{2,5} & & & b_{2,8} & \\ & & b_{3,3} & & & b_{3,6} & & & b_{3,9} \\ b_{4,1} & & & b_{4,4} & & & b_{4,7} & & \\ & b_{5,2} & & & b_{5,5} & & & b_{5,8} & \\ & & b_{6,3} & & & b_{6,6} & & & b_{6,9} \\ b_{7,1} & & & b_{7,4} & & & b_{7,7} & & \\ & b_{8,2} & & & b_{8,5} & & & b_{8,8} & \\ & & b_{9,3} & & & b_{9,6} & & & b_{9,9} \end{pmatrix}. \tag{17}$$

For general values of k_r and k_c, S is a $k_r \times k_r$ block matrix, where each block $(S^{i,j})$ is a $k_c \times k_c$ matrix. The main diagonal of each block matrix $S^{i,j}$ is zero, and the main diagonal of S is the zero matrix $(S^{i,i} = 0)$. Equation (22) shows the structure

$$S_y = \begin{pmatrix} & & & & a_{1,2}b_{2,5} & a_{1,3}b_{3,6} & & a_{1,2}b_{2,8} & a_{1,3}b_{3,9} \\ & & & a_{2,1}b_{1,4} & & a_{2,3}b_{3,6} & a_{2,1}b_{1,7} & & a_{2,3}b_{3,9} \\ & & & a_{3,1}b_{1,4} & a_{3,2}b_{2,5} & & a_{3,1}b_{1,7} & a_{3,2}b_{2,8} & \\ & a_{4,5}b_{5,2} & a_{4,6}b_{6,3} & & & & & a_{4,5}b_{5,8} & a_{4,6}b_{6,9} \\ a_{5,4}b_{4,1} & & a_{5,6}b_{6,3} & & & & a_{5,4}b_{4,7} & & a_{5,6}b_{6,9} \\ a_{6,4}b_{4,1} & a_{6,5}b_{5,2} & & & & & a_{6,4}b_{4,7} & a_{6,5}b_{5,8} & \\ & a_{7,8}b_{8,2} & a_{7,9}b_{9,3} & & a_{7,8}b_{8,5} & a_{7,9}b_{9,6} & & & \\ a_{8,7}b_{7,1} & & a_{8,9}b_{9,3} & a_{8,7}b_{7,4} & & a_{8,9}b_{9,6} & & & \\ a_{9,7}b_{7,1} & a_{9,8}b_{8,2} & & a_{9,7}b_{7,4} & a_{9,8}b_{8,5} & & & & \end{pmatrix}$$

$$= \begin{pmatrix} & & & & a^1_{1,2}\tilde{b}^2_{1,2} & a^1_{1,3}\tilde{b}^3_{1,2} & & a^1_{1,2}\tilde{b}^2_{1,3} & a^1_{1,3}\tilde{b}^3_{1,3} \\ & & & a^1_{2,1}\tilde{b}^1_{1,2} & & a^1_{2,3}\tilde{b}^3_{1,2} & a^1_{2,1}\tilde{b}^1_{1,3} & & a^1_{2,3}\tilde{b}^3_{1,3} \\ & & & a^1_{3,1}\tilde{b}^1_{1,2} & a^1_{3,2}\tilde{b}^2_{1,2} & & a^1_{3,1}\tilde{b}^1_{1,3} & a^1_{3,2}\tilde{b}^2_{1,3} & \\ & a^2_{1,2}\tilde{b}^2_{2,1} & a^2_{1,3}\tilde{b}^3_{2,1} & & & & & a^2_{1,2}\tilde{b}^2_{2,3} & a^2_{1,3}\tilde{b}^3_{2,3} \\ a^2_{2,1}\tilde{b}^1_{2,1} & & a^2_{2,3}\tilde{b}^3_{2,1} & & & & a^2_{2,1}\tilde{b}^1_{2,3} & & a^2_{2,3}\tilde{b}^3_{2,3} \\ a^2_{3,1}\tilde{b}^1_{2,1} & a^2_{3,2}\tilde{b}^2_{2,1} & & & & & a^2_{3,1}\tilde{b}^1_{2,3} & a^2_{3,2}\tilde{b}^2_{2,3} & \\ & a^3_{1,2}\tilde{b}^2_{3,1} & a^3_{1,3}\tilde{b}^3_{3,1} & & a^3_{1,2}\tilde{b}^2_{3,2} & a^3_{1,3}\tilde{b}^3_{3,2} & & & \\ a^3_{2,1}\tilde{b}^1_{3,1} & & a^3_{2,3}\tilde{b}^3_{3,1} & a^3_{2,1}\tilde{b}^1_{3,2} & & a^3_{2,3}\tilde{b}^3_{3,2} & & & \\ a^3_{3,1}\tilde{b}^1_{3,1} & a^3_{3,2}\tilde{b}^2_{3,1} & & a^3_{3,1}\tilde{b}^1_{3,2} & a^3_{3,2}\tilde{b}^2_{3,2} & & & & \end{pmatrix} \quad (20)$$

$$S_z = \begin{pmatrix} & & & & b_{1,4}a_{4,5} & b_{1,4}a_{4,6} & & b_{1,7}a_{7,8} & b_{1,7}a_{7,9} \\ & & & b_{2,5}a_{5,4} & & b_{2,5}a_{5,6} & b_{2,8}a_{8,7} & & b_{2,8}a_{8,9} \\ & & & b_{3,6}a_{6,4} & b_{3,6}a_{6,5} & & b_{3,9}a_{9,7} & b_{3,9}a_{9,8} & \\ & b_{4,1}a_{1,2} & b_{4,1}a_{1,3} & & & & & b_{4,7}a_{7,8} & b_{4,7}a_{7,9} \\ b_{5,2}a_{2,1} & & b_{5,2}a_{2,3} & & & & b_{5,8}a_{8,7} & & b_{5,8}a_{8,9} \\ b_{6,3}a_{3,1} & b_{6,3}a_{3,2} & & & & & b_{6,9}a_{9,7} & b_{6,9}a_{9,8} & \\ & b_{7,1}a_{1,2} & b_{7,1}a_{1,3} & & b_{7,4}a_{4,5} & b_{7,4}a_{4,6} & & & \\ b_{8,2}a_{2,1} & & b_{8,2}a_{2,3} & b_{8,5}a_{5,4} & & b_{8,5}a_{5,6} & & & \\ b_{9,3}a_{3,1} & b_{9,3}a_{3,2} & & b_{9,6}a_{6,4} & b_{9,6}a_{6,5} & & & & \end{pmatrix}$$

$$= \begin{pmatrix} & & & & \tilde{b}^1_{1,2}a^2_{1,2} & \tilde{b}^1_{1,2}a^2_{1,3} & & \tilde{b}^1_{1,3}a^3_{1,2} & \tilde{b}^1_{1,3}a^3_{1,3} \\ & & & \tilde{b}^2_{1,2}a^2_{2,1} & & \tilde{b}^2_{1,2}a^2_{2,3} & \tilde{b}^2_{1,3}a^3_{2,1} & & \tilde{b}^2_{1,3}a^3_{2,3} \\ & & & \tilde{b}^3_{1,2}a^2_{3,1} & \tilde{b}^3_{1,2}a^2_{3,2} & & \tilde{b}^3_{1,3}a^3_{3,1} & \tilde{b}^3_{1,3}a^3_{3,2} & \\ & \tilde{b}^1_{2,1}a^1_{1,2} & \tilde{b}^1_{2,1}a^1_{1,3} & & & & & \tilde{b}^1_{2,3}a^3_{1,2} & \tilde{b}^1_{2,3}a^3_{1,3} \\ \tilde{b}^2_{2,1}a^1_{2,1} & & \tilde{b}^2_{2,1}a^1_{2,3} & & & & \tilde{b}^2_{2,3}a^3_{2,1} & & \tilde{b}^2_{2,3}a^3_{2,3} \\ \tilde{b}^3_{2,1}a^1_{3,1} & \tilde{b}^3_{2,1}a^1_{3,2} & & & & & \tilde{b}^3_{2,3}a^3_{3,1} & \tilde{b}^3_{2,3}a^3_{3,2} & \\ & \tilde{b}^1_{3,1}a^1_{1,2} & \tilde{b}^1_{3,1}a^1_{1,3} & & \tilde{b}^1_{3,2}a^2_{1,2} & \tilde{b}^1_{3,2}a^2_{1,3} & & & \\ \tilde{b}^2_{3,1}a^1_{2,1} & & \tilde{b}^2_{3,1}a^1_{2,3} & \tilde{b}^2_{3,2}a^2_{2,1} & & \tilde{b}^2_{3,2}a^2_{2,3} & & & \\ \tilde{b}^3_{3,1}a^1_{3,1} & \tilde{b}^3_{3,1}a^1_{3,2} & & \tilde{b}^3_{3,2}a^2_{3,1} & \tilde{b}^3_{3,2}a^2_{3,2} & & & & \end{pmatrix} \quad (21)$$

of S_y

$$S_y = \begin{pmatrix} 0 & S_y^{1,2} & \cdots & S_y^{1,k_r} \\ S_y^{2,1} & 0 & & \\ \vdots & & \ddots & \\ S_y^{k_r,1} & S_y^{k_r,2} & \cdots & 0 \end{pmatrix}. \tag{22}$$

The elements of $S_y^{i,j}$ are the posterior dependence between two bits in the ith row multiplied by the posterior dependence between a bit in the ith row and its corresponding (same column) bit in the jth row

$$\begin{aligned}(S_y^{i,j})_{m,n} &= \big(a_{(i-1)\cdot k_c+m,\,(i-1)\cdot k_c+n} \cdot b_{(i-1)\cdot k_c+n,\,(j-1)\cdot k_c+n} \\ &\quad - \delta(m-n)\big)\,(1-\delta(i-j)) \\ &= \left(a^i_{m,n}\cdot \tilde{b}^n_{i,j} - \delta(m-n)\right)(1-\delta(i-j)) \end{aligned}\tag{23}$$

where

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0. \end{cases}$$

A similar analysis can be applied to S_z.

C. Properties of Stability Matrices

In the following we will present some properties of the stability matrices.

Claim 1: The set of the eigenvalues of $J^R(J^C)$ is the union of the set of the eigenvalues of the Jacobian matrices of each row (column)

$$eig\{J^R\} = \bigcup_{i=1}^{k_r} \mathrm{eig}\{J^{R,i}\} \tag{24a}$$

$$\mathrm{eig}\{J^C\} = \mathrm{eig}\left\{\tilde{J}^C\right\} = \bigcup_{i=1}^{k_c} eig\left\{\tilde{J}^{C,i}\right\} \tag{24b}$$

Proof: A known linear algebra lemma states that the eigenvalues of a block-diagonal matrix is the union of the eigenvalues of its blocks. This proves (24a).

Under a similarity transform, J^C is a block diagonal like J^R (the similarity transform is the rows-to-columns permutation). Using it with (24a) proves (24b). □

Denote with $S^{R,i} = J^{R,i} - I$ and $S^{C,i} = \tilde{J}^{C,i} - I$ the decoding stability matrices of the ith row and the ith column, respectively. Let $p^{R,i}(x)$, $p^{C,i}(x)$, $p^R(x)$, $p^C(x)$ represent the characteristic polynomials of $S^{R,i}$, $S^{C,i}$, S^R, S^C, respectively

$$\begin{aligned} p^{R,i}(x) &= \prod_{l=1}^{k_c}(x-\lambda_l^{R,i}) \\ &= x^{k_c} + c^{R,i}_{k_c-1}x^{k_c-1} + c^{R,i}_{k_c-2}x^{k_c-2} \\ &\quad + \cdots + c_1^{R,i}x + c_0^{R,i} \end{aligned}\tag{25}$$

$$\begin{aligned} p^R(x) &= \prod_{l=1}^{k_rk_c}(x-\lambda_l^R) \\ &= x^{k_rk_c} + c^R_{k_rk_c-1}x^{k_rk_c-1} + c^R_{k_rk_c-2}x^{k_rk_c-2} \\ &\quad + \cdots + c_1^R x + c_0^R \end{aligned}\tag{26}$$

where $\lambda_l^{R,i}$, λ_l^R are the eigenvalues of $S^{R,i}$, S^R, respectively. Than, using Claim 1, we get that the characteristic polynomial of S^R is the product of the characteristic polynomials of the stability matrix of each row—$S^{R,i}$ (the same holds for S^C and $S^{C,i}$)

$$p^R(x) = \prod_{i=1}^{k_r} p^{R,i}(x) \tag{27}$$

Claim 2: The sum of the eigenvalues of each of the stability matrices—S^R, S^C, and S is zero.

Proof: The sum of the diagonal elements of a square matrix is the sum of its eigenvalues. Using (13)–(15), (18), and the definition in (19), it can be easily shown that $(S^R)_{i,i} = (S^C)_{i,i} = 0$. Equation (22) shows that the diagonal elements of S are zero as well. □

Note that $S^{R,i}$ and $S^{C,i}$ have an all-zero main diagonal, and therefore possess the same property.

Using Claim 2 we get:

$$c^R_{k_rk_c-1} = -\sum_{l=1}^{k_rk_c}\lambda_l^R = 0. \tag{28}$$

Note that the same holds for the characteristic polynomials of S^C and S.

Claim 3: $c^R_{k_rk_c-2} = \sum_{l\neq m}\lambda_l^R\lambda_m^R \leq 0$.

Proof: Consider the characteristic polynomial of the decoding stability matrix of the first row (for $k_r = k_c = 3$)

$$\begin{aligned} p^{R,1}(\lambda) &= |\lambda I - S^{R,1}| \\ &= \begin{vmatrix} \lambda & -a_{1,2} & -a_{1,3} \\ -a_{2,1} & \lambda & -a_{2,3} \\ -a_{3,1} & -a_{3,2} & \lambda \end{vmatrix} \\ &= \lambda^3 - \lambda(a_{1,2}a_{2,1} + a_{1,3}a_{3,1} + a_{2,3}a_{3,2}) \\ &\quad + a_{1,2}a_{2,3}a_{3,1} + a_{1,3}a_{3,2}a_{2,1}. \end{aligned}$$

The coefficient $c^{R,1}_{k_c-2} = c_1^{R,1}$ is minus the sum of all products of two symmetric (with respect to the main diagonal) elements in $S^{R,i}$. By induction, in the general case we get

$$\begin{aligned} c^{R,i}_{k_c-2} &= \sum_{\substack{l,m=1\\ l\neq m}}^{k_c} \lambda_l^{R,i}\cdot\lambda_m^{R,i} \\ &= -\sum_{\substack{l,m=1\\ l\neq m}}^{k_c} a_{(i-1)k_c+l,\,(i-1)k_c+m} \\ &\quad \cdot a_{(i-1)k_c+m,\,(i-1)k_c+l}. \end{aligned}\tag{29}$$

Starting from (9) we will develop an alternative expression for the elements of the Jacobian

$$\begin{aligned} (J_P)_{i,j} &= \frac{\sum_{b\in H_i\cap H_j} p_r(b)}{\sum_{b\in H_i} p_r(b)} - \frac{\sum_{b\in \overline{H}_i\cap H_j} p_r(b)}{\sum_{b\in \overline{H}_i} p_r(b)} \\ &= \frac{p_r(H_i,H_j)}{p_r(H_i)} - \frac{p_r\left(\overline{H}_i,H_j\right)}{p_r\left(\overline{H}_i\right)} \\ &= \frac{p_r(H_i,H_j) - \left[p_r(H_i,H_j) + p_r\left(\overline{H}_i,H_j\right)\right]p_r(H_i)}{p_r(H_i)p_r\left(\overline{H}_i\right)} \\ &= \frac{p_r(H_i,H_j) - p_r(H_i)p_r(H_j)}{p_r(H_i)p_r\left(\overline{H}_i\right)}. \end{aligned}\tag{30}$$

Here, we used the probability density as the representative of the equivalence class of densities. In the third equality we used the normalization $\sum_b p_r(b) = 1$, to substitute $p_r(\overline{H}_i) = 1 - p_r(H_i)$.

Using (30) we get

$$a_{l,m}a_{m,l} = \frac{[p_r(H_l, H_m) - p_r(H_l)p_r(H_m)]^2}{p_r(H_l)p_r\left(\overline{H}_l\right)p_r(H_m)p_r\left(\overline{H}_m\right)} \geq 0. \quad (31)$$

Substituting (31) in (29) we prove that

$$c^{R,i}_{k_c-2} \leq 0, \qquad 1 \leq i \leq k_r \quad (32)$$

Using (25), (26), and substituting (32), we get

$$c^{R}_{k_r k_c-2} = \sum_{i=1}^{k_r} c^{R,i}_{k_c-2} \leq 0. \quad (33)$$

This proves the claim. □

Note that the same holds for the coefficients of $p^{C,i}(x)$, $p^C(x)$. Also note that the properties in Claims 2 and 3 hold for the stability matrices (and their corresponding characteristic polynomials) of DEC1 and DEC2 (Fig. 1) in the general turbo decoding scheme.

Equation (30) can be used to get a compact expression for the characteristic polynomial in the general case. For example, for $k_r = k_c = 3$ we have

$$c^{R,1}_0 = a_{1,2}a_{2,3}a_{3,1} + a_{1,3}a_{3,2}a_{2,1}.$$

Using (30) we get

$$a_{1,2}a_{2,3}a_{3,1} = a_{1,3}a_{3,2}a_{2,1} \Rightarrow c^{R,1}_0 = 2a_{1,2}a_{2,3}a_{3,1}.$$

D. Analysis of the Stability Matrices for $k_r = k_c = 2$

For $k_r = k_c = 2$ the stability matrices are as follows:

$$J^R - I = \begin{pmatrix} 0 & a_{1,2} & 0 & 0 \\ a_{2,1} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{3,4} \\ 0 & 0 & a_{4,3} & 0 \end{pmatrix} \quad (34)$$

$$J^C - I = \begin{pmatrix} 0 & 0 & b_{1,3} & 0 \\ 0 & 0 & 0 & b_{2,4} \\ b_{3,1} & 0 & 0 & 0 \\ 0 & b_{4,2} & 0 & 0 \end{pmatrix} \quad (35)$$

$$S^R S^C = \begin{pmatrix} & & & a_{1,2}b_{2,4} \\ & & a_{2,1}b_{1,3} & \\ & a_{3,4}b_{4,2} & & \\ a_{4,3}b_{3,1} & & & \end{pmatrix}. \quad (36)$$

The characteristic equations of S^R, S^C and their roots are given by

$$\begin{aligned} &((\lambda^R)^2 - a_{1,2}a_{2,1})((\lambda^R)^2 - a_{3,4}a_{4,3}) = 0 \\ &\qquad \Rightarrow \lambda^R = \pm\sqrt{a_{1,2}a_{2,1}}, \pm\sqrt{a_{3,4}a_{4,3}} \\ &((\lambda^C)^2 - b_{1,3}b_{3,1})((\lambda^C)^2 - b_{2,4}b_{4,2}) = 0 \\ &\qquad \Rightarrow \lambda^C = \pm\sqrt{b_{1,3}b_{3,1}}, \pm\sqrt{b_{2,4}b_{4,2}}. \end{aligned} \quad (37)$$

A straightforward calculation gives the eigenvalues of S

$$\begin{aligned} &(\lambda^2 - a_{1,2}b_{2,4}a_{4,3}b_{3,1})(\lambda^2 - a_{2,1}b_{1,3}a_{3,4}b_{4,2}) = 0 \\ &\qquad \Rightarrow \lambda = \pm\sqrt{a_{1,2}b_{2,4}a_{4,3}b_{3,1}}, \pm\sqrt{a_{2,1}b_{1,3}a_{3,4}b_{4,2}}. \end{aligned} \quad (38)$$

Theorem 1: The fixed point(s) of *any* product code turbo decoder with $k_r = k_c = 2$ is (are) always stable.

Proof: Equation (38) shows that the eigenvalues of S are the square root of the product of four Jacobian elements, whose absolute value, according to (18), is smaller than 1. Therefore, the eigenvalues will be smaller than 1, i.e., S is stable. □

Note that the fixed-point's stability does not depend on the SNR, nor on the row or column encoders. Also note that (31) and (37) imply that the eigenvalues of S^R and S^C are real.

Assuming all Jacobian elements are of the same order of magnitude, we will get

$$\lambda \approx (\lambda^R)^2 \approx (\lambda^C)^2. \quad (39)$$

Therefore, the (turbo) combination of the row decoder followed by the column decoder doubles the convergence rate.

We will now discuss the uniqueness of the fixed point in this case. In [2], Richardson defines a function $C_2 = C_2(f^*, P_x, P_y, P_z)$ as the radius of the largest open ball R, centered at a fixed point f^*, such that if $f \in R$ then

$$P_x + P_y + Q_z(f) \in T$$

and

$$P_x + Q_y(f) + P_z \in T$$

(T is the set of P such that $J_P - I$ is strictly stable), where

$$Q_z(f) = B\pi^{-1}_{P_x+P_y}(f)$$

and

$$Q_y(f) = B\pi^{-1}_{P_x+P_z}(f)$$

That is, if a point f in the ball R was a fixed point of the decoding algorithm, then it would have been a stable one. Richardson proves that in this ball with radius C_2, f^* is a unique (and stable) fixed point.

Theorem 2: Any product code turbo-decoder with $k_r = k_c = 2$ has a unique (and stable) fixed point.

Proof: For $k_r = k_c = 2$, we proved that $J^R - I$ and $J^C - I$ are strictly stable *regardless* of the point f in which they are calculated. Thus, for every f, $P_x + P_y + Q_z(f) \in T$ and $P_x + Q_y(f) + P_z \in T$. Therefore, $C_2 = +\infty$, and the turbo decoder possess a unique fixed point. □

There is a different perspective on belief-propagation in graphs with a single loop. For a 2 × 2 product code (refer to Fig. 2), x^1_1, x^1_2, x^2_1, x^2_2 denote the information bits, y^1, y^2 denote the independent codes of the first and second rows, respectively, and z^1, z^2 denote the independent codes of the first and second columns, respectively. The corresponding belief network [9] has a single loop

$$x^1_1 \rightarrow y^1 \rightarrow x^1_2 \rightarrow z^2 \rightarrow x^2_2 \rightarrow y^2 \rightarrow x^2_1 \rightarrow z^1 \rightarrow x^1_1.$$

Since the unobserved variables are binary, it follows from [10] that there is a global convergence to the maximum *a posteriori* (MAP) bitwise decision.

E. Analysis of the Stability Matrices for the General Case

In order to analyze the behavior in the general case, we will use the following known lemma [11, pp. 344–346].

Lemma 1: For a given matrix A, the maximal sum of magnitudes of each row's elements is an upper bound for the magnitude of the matrix eigenvalues, i.e.

$$|\lambda_{\max}| \le \max_i \sum_j |(A)_{i,j}| \tag{40}$$

[note that $\max_j \sum_i |(A)_{i,j}|$ is also an upper bound of $|\lambda_{\max}|$, hence a similar analysis, based on the maximal sum of magnitudes of each column's elements, may also be applied].

We will start the analysis with the case $k_r = k_c = 3$. Without loss of generality, let us assume that the first row of S^R is the one in which the sum of the magnitudes of its elements is the largest. The largest eigenvalues of S^R and S^C are bounded by this sum

$$\begin{aligned} |\lambda_{\max}^R| &\le |a_{1,2}| + |a_{1,3}| \\ |\lambda_{\max}^C| &\le |a_{1,2}| + |a_{1,3}| \end{aligned} \tag{41}$$

where $\lambda_{\max}^R$, $\lambda_{\max}^C$ denote the maximal eigenvalues of S^R, S^C, respectively.

We will also assume that the upper bound is smaller than 1

$$|a_{1,2}| + |a_{1,3}| = \overline{\lambda} < 1. \tag{42}$$

Let us calculate the sum of magnitudes of a certain row in S. For example, the fourth row of (20)

$$\begin{aligned} &|a_{4,5}b_{5,2}| + |a_{4,6}b_{6,3}| + |a_{4,5}b_{5,8}| + |a_{4,6}b_{6,9}| \\ &\quad = |a_{4,5}|(|b_{5,2}| + |b_{5,8}|) + |a_{4,6}|(|b_{6,3}| + |b_{6,9}|) \\ &\quad \le (|a_{4,5}| + |a_{4,6}|)(|a_{1,2}| + |a_{1,3}|) \\ &\quad \le (|a_{1,2}| + |a_{1,3}|)^2 = \overline{\lambda}^2. \end{aligned} \tag{43}$$

A similar calculation can be applied to any row of S. Using it with Lemma 1, we conclude that the eigenvalues of S are bounded by $\overline{\lambda}^2$. Thus, the convergence rate of S, as predicted by the upper bound, is twice as fast as S^R or S^C rates (see also simulation results in Section IV).

Let us assume now that the bound is larger than 1

$$|a_{1,2}| + |a_{1,3}| > 1. \tag{44}$$

$a_{1,2}$, $a_{1,3}$ appear only in the first row of S_y. The sum of magnitudes of the elements in this row is

$$|a_{1,2}|(|b_{2,5}| + |b_{2,8}|) + |a_{1,3}|(|b_{3,6}| + |b_{3,9}|). \tag{45}$$

Therefore, if $(|b_{2,5}| + |b_{2,8}|)$ and $(|b_{3,6}| + |b_{3,9}|)$ are small enough, the expression in (45) can be smaller than 1. In order to understand this condition, we will arrange the information bits in a 3×3 matrix as follows:

$$\begin{matrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{matrix}. \tag{46}$$

$b_{2,5}$ and $b_{2,8}$ ($=\tilde{b}_{1,2}^2$ and $\tilde{b}_{1,3}^2$, respectively) measure the posterior dependence between the second bit in the first row and the rest of the bits in its column. If the second column was successfully decoded, these values will be small. The same applies to $b_{3,6}$, $b_{3,9}$ and the decoding of the third column.

Therefore, if DEC1 did not decode the first row successfully, but DEC2 managed to decode the second and third columns sufficiently well, the overall decoding algorithm will converge.

The same arguments hold for the general case. For arbitrary values of k_r and k_c, we can use (23) to calculate the sum of magnitudes of elements in row $(i-1)k_c + m$ of S_y

$$\begin{aligned} \sum_{j=1}^{k_r} \sum_{n=1}^{k_c} |(S_y^{i,j})_{m,n}| &= \sum_{\substack{n=1 \\ n\ne m}}^{k_c} \left(\left|a_{(i-1)\cdot k_c + m,\, (i-1)\cdot k_c + n}\right| \right. \\ &\qquad \left. \cdot \sum_{\substack{j=1 \\ j\ne i}}^{k_r} \left|b_{(i-1)\cdot k_c + n,\, (j-1)\cdot k_c + n}\right| \right) \\ &= \sum_{\substack{n=1 \\ n\ne m}}^{k_c} \left(\left|a_{m,n}^i\right| \cdot \sum_{\substack{j=1 \\ j\ne i}}^{k_r} \left|\tilde{b}_{i,j}^n\right| \right). \end{aligned} \tag{47}$$

The basic element in the sum is the product of the posterior dependence between two bits in a row, and the sum of the posterior dependencies between one of these bits and all the bits in its column. Small column posterior dependencies can compensate for a large value of inter-row bit dependence and *vice versa* (a similar analysis can be done for the sum of magnitudes of row elements of S_z).

Equation (47) can be used to establish a sufficient condition for stability

$$\begin{aligned} &\sum_{\substack{n=1 \\ n\ne m}}^{k_c} \left(\left|a_{(i-1)\cdot k_c + m,\, (i-1)\cdot k_c + n}\right| \right. \\ &\qquad \left. \cdot \sum_{\substack{j=1 \\ j\ne i}}^{k_r} \left|b_{(i-1)\cdot k_c + n,\, (j-1)\cdot k_c + n}\right| \right) < 1 \\ &\qquad\qquad \forall\, 1 \le i \le k_r,\ 1 \le m \le k_c \end{aligned} \tag{48a}$$

$$\begin{aligned} &\sum_{\substack{n=1 \\ n\ne m}}^{k_c} \left(\left|a_{m,n}^i\right| \cdot \sum_{\substack{j=1 \\ j\ne i}}^{k_r} \left|\tilde{b}_{i,j}^n\right| \right) < 1 \\ &\qquad\qquad \forall\, 1 \le i \le k_r,\ 1 \le m \le k_c. \end{aligned} \tag{48b}$$

Equation (47) can also be used to develop a set of upper bounds to the convergence rate of the algorithm

$$\begin{aligned}\lambda_{\max} &\le \max_{1\le i\le k_r,\,1\le m\le k_c} \sum_{\substack{n=1\\ n\ne m}}^{k_c}\left(\left|a_{(i-1)\cdot k_c+m,\,(i-1)\cdot k_c+n}\right| \cdot \sum_{\substack{j=1\\ j\ne i}}^{k_r}\left|b_{(i-1)\cdot k_c+n,\,(j-1)\cdot k_c+n}\right|\right)\\ &\le \max_{1\le i\le k_r}\left\{\max_{1\le m\le k_c}\left\{\sum_{\substack{n=1\\ n\ne m}}^{k_c}\left|a_{(i-1)\cdot k_c+m,\,(i-1)\cdot k_c+n}\right|\right\} \cdot \max_{1\le n\le k_c}\left\{\sum_{\substack{j=1\\ j\ne i}}^{k_r}\left|b_{(i-1)\cdot k_c+n,\,(j-1)\cdot k_c+n}\right|\right\}\right\}\\ &\le \max_{1\le i\le k_r,\,1\le m\le k_c}\left\{\sum_{\substack{n=1\\ n\ne m}}^{k_c}\left|a_{(i-1)\cdot k_c+m,\,(i-1)\cdot k_c+n}\right|\right\} \cdot \max_{1\le i\le k_r,\,1\le n\le k_c}\left\{\sum_{\substack{j=1\\ j\ne i}}^{k_r}\left|b_{(i-1)\cdot k_c+n,\,(j-1)\cdot k_c+n}\right|\right\}\end{aligned} \tag{49a}$$

or in its equivalent representation

$$\begin{aligned}\lambda_{\max} &\le \max_{1\le i\le k_r,\,1\le m\le k_c} \sum_{\substack{n=1\\ n\ne m}}^{k_c}\left(\left|a^{i}_{m,n}\right| \cdot \sum_{\substack{j=1\\ j\ne i}}^{k_r}\left|\tilde{b}^{n}_{i,j}\right|\right)\\ &\le \max_{1\le i\le k_r}\left\{\max_{1\le m\le k_c}\left\{\sum_{\substack{n=1\\ n\ne m}}^{k_c}\left|a^{i}_{m,n}\right|\right\} \cdot \max_{1\le n\le k_c}\left\{\sum_{\substack{j=1\\ j\ne i}}^{k_r}\left|\tilde{b}^{n}_{i,j}\right|\right\}\right\}\\ &\le \max_{1\le i\le k_r,\,1\le m\le k_c}\left\{\sum_{\substack{n=1\\ n\ne m}}^{k_c}\left|a^{i}_{m,n}\right|\right\} \cdot \max_{1\le i\le k_r,\,1\le n\le k_c}\left\{\sum_{\substack{j=1\\ j\ne i}}^{k_r}\left|\tilde{b}^{n}_{i,j}\right|\right\}.\end{aligned} \tag{49b}$$

The first bound follows from a direct use of Lemma 1 with the general expression for the stability matrix elements given in (23): we search for the maximal sum of absolute values of row elements of S. To calculate a row element we add the absolute values of $k_r - 1$ elements of S^C (corresponding to the posterior dependencies between one of the bits in the row and all the bits in its column), and multiply it by an element of S^R. Thus, to compute a row of S we need $(k_r-1)(k_c-1)$ additions and $k_c - 1$ multiplications. This should be carried out for each of the $k_r k_c$ rows of S. Then, to calculate the bound, we have to add the $k_c - 1$ nonzero elements of each row, and find the maximal value between all the rows. To conclude, the computational complexity of the first bound consists of $k_r^2 k_c(k_c - 1)$

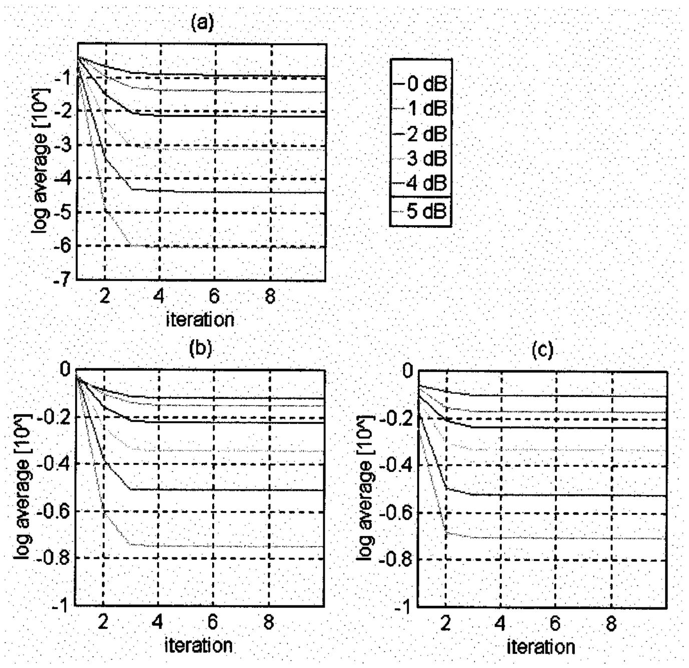

Fig. 3. Averaged maximal eigenvalue of (a) S, (b) S^R, and (c) S^C for Hamming $[(7,4,3)]^2$ code.

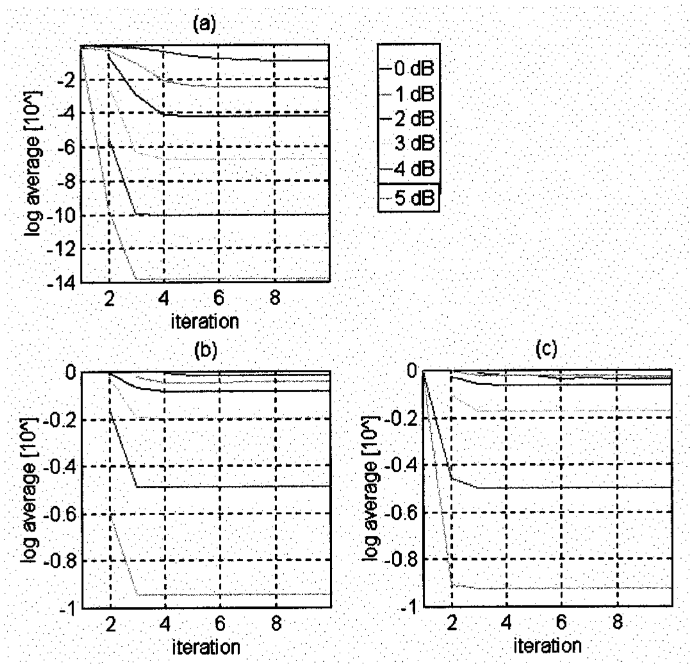

Fig. 4. Averaged maximal eigenvalue of (a) S, (b) S^R, and (c) S^C for Golay $[(24,12,8)]^2$ code.

TABLE I
PERCENTAGE OF STABLE MATRICES FOR HAMMING $[(7,4,3)]^2$ CODE

Percentage of stable S^R matrices										
iter # / SNR [dB]	1	2	3	4	5	6	7	8	9	10
0	14	48	54	55	56	56	56	56	56	56
1	10	60	71	71	72	72	72	72	72	72
2	10	77	84	85	86	86	86	86	86	86
3	8	90	94	95	95	95	95	95	95	95
4	10	95	98	98	98	98	98	98	98	98
5	8	98	99	99	99	99	99	99	99	99
Percentage of stable S^C matrices										
Iter # / SNR [dB]	1	2	3	4	5	6	7	8	9	10
0	37	52	56	57	57	57	57	57	57	57
1	48	66	71	72	72	72	72	72	72	72
2	59	81	85	86	86	86	86	86	86	86
3	76	92	94	95	95	95	95	95	95	95
4	87	97	98	98	98	98	98	98	98	98
5	93	99	99	99	99	99	99	99	99	99
Percentage of stable S matrices										
Iter # / SNR [dB]	1	2	3	4	5	6	7	8	9	10
0	98	99	99	99	99	99	99	99	99	99
1	96	98	99	99	99	99	99	99	99	99
2	94	99	99	99	99	99	99	100	99	100
3	93	99	100	100	100	100	100	100	100	100
4	94	99	100	100	100	100	100	100	100	100
5	95	100	100	100	100	100	100	100	100	100

additions, $k_r k_c(k_c-1)$ multiplications, and finding the maxima of $k_r k_c$ elements.

The second bound is looser than the first one: for each row (r_i), we upper-bound the sum of the posterior dependencies between one bit from the r_ith row and all the bits in its column [$k_c(k_r-1)$ additions, and finding the maximum of k_c elements]. We then upper-bound the sum of the posterior dependencies between each pair of bits of the r_ith row [$k_c(k_c-1)$ additions, and finding the maximum of k_c elements], and multiply the results. Thus, for calculating the second bound, we perform $k_r k_c(k_r+k_c-2)$ additions, k_r multiplications, and $2k_r$ maxima search of k_c elements.

The third bound is the loosest one: we upper-bound the sum of the posterior dependencies between one bit from any row and all the bits in its column [$k_r k_c(k_r-1)$ additions, and finding the maximum of $k_r k_c$ elements]. We then upper-bound the sum of the posterior dependencies between each pair of bits of any row [$k_r k_c(k_c-1)$ additions, and finding the maximum of $k_r k_c$ elements], and multiply the results. The computational complexity here is $k_r k_c(k_r+k_c-2)$ additions, one multiplication, and two maxima search of $k_r k_c$ elements.

The third bound also shows that the convergence rate of the whole decoding algorithm is the product of upper bounds of the convergence rate of the rows and columns decoding. Hence, in logarithmic scale, according to the bound, the algorithm converges at least twice as fast as the (slowest of the) rows or columns decoding.

IV. SIMULATION RESULTS

When the stability matrix is calculated at the fixed point of the algorithm $(Q_y,\, Q_z)$, it measures how fast will the algorithm

TABLE II
PERCENTAGE OF STABLE MATRICES FOR GOLAY $[(24,12,8)]^2$ CODE

Percentage of stable S^R matrices										
iter # / SNR [dB]	1	2	3	4	5	6	7	8	9	10
0	0	0	0	9	18	21	23	25	25	25
1	0	0	28	52	59	61	65	65	65	66
2	0	6	82	90	90	90	90	90	90	90
3	0	61	99	99	99	99	99	99	99	99
4	0	93	100	100	100	100	100	100	100	100
5	0	99	100	100	100	100	100	100	100	100

Percentage of stable S^C matrices										
Iter # / SNR [dB]	1	2	3	4	5	6	7	8	9	10
0	0	0	4	13	20	22	24	24	25	25
1	0	4	40	55	61	65	66	68	68	68
2	0	49	89	91	92	92	92	92	92	92
3	0	94	99	99	99	99	99	99	99	99
4	9	100	100	100	100	100	100	100	100	100
5	45	100	100	100	100	100	100	100	100	100

Percentage of stable S matrices										
Iter # / SNR [dB]	1	2	3	4	5	6	7	8	9	10
0	69	39	45	63	77	81	82	82	85	88
1	8	15	76	82	89	92	94	97	98	96
2	0	76	99	99	100	100	100	100	100	100
3	0	100	100	100	100	100	100	100	100	100
4	0	100	100	100	100	100	100	100	100	100
5	5	100	100	100	100	100	100	100	100	100

converge to this point, assuming it is already in its vicinity. In our simulations, we used (12) to calculate S after every iteration [i.e., using $Q_y^{(m)}$, $Q_z^{(m)}$].

For a given SNR (additive white Gaussian noise channel), we have simulated the transmission of w blocks (each one with $k_r k_c$ bits). For each block, we ran t decoding iterations, and for each iteration, we computed the averaged maximal eigenvalue of S, S^R, and S^C according to the logarithmic convergence slope criteria. Denote the maximal eigenvalues with magnitude smaller than 1 by $\lambda_1, \ldots, \lambda_u (u \leq w)$. According to (8), the convergence function (f) of the ith block acts like $f(l) = \lambda_i^l$; taking the logarithm we get $\log f(l) = l \log \lambda_i$, hence $\log \lambda_i$ denotes the slope of the convergence function in logarithmic scale. The averaged logarithmic convergence slope will be $\frac{1}{u} \sum_{i=1}^{u} \log \lambda_i$, and the equivalent eigenvalue is

$$\lambda_{\text{av}} = 10^{\frac{1}{u} \sum_{i=1}^{u} \log \lambda_i}.$$

This averaging takes into account only the eigenvalues with magnitudes smaller than 1. Therefore, it enables us to compute the average logarithmic convergence slope when the decoding of the blocks (for a given SNR) is stable. The percentage of these stable blocks decoding out of the total number of blocks (i.e., u/w) will also be calculated.

Simulation results for Hamming $[(7,4,3)]^2$ and Golay $[(24,12,8)]^2$ product codes are presented in Figs. 3 and 4, for $w_H = 1000$ and $w_G = 100$ blocks (for the Hamming and Golay codes, respectively) and $t = 10$ iterations.

The general behavior shown by the figures is a monotonic decrease in the eigenvalues magnitude as the iteration number increases, until it reaches its final value. It also shows that a larger SNR has a faster convergence rate, moreover, as the SNR gets larger, it has more influence on the convergence rate.

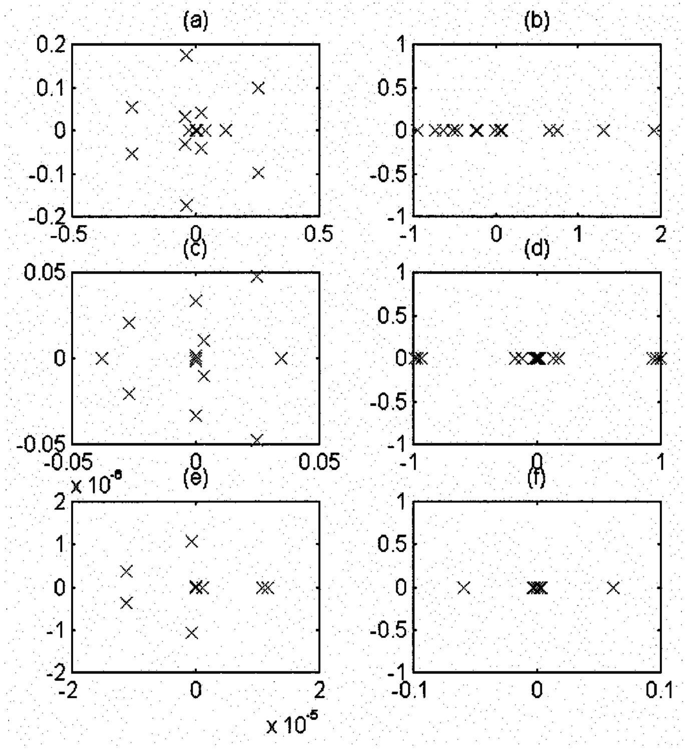

Fig. 5. Hamming $[(7,4,3)]^2$—eigenvalues spread of (a) S and (b) S^R for SNR of -3 [dB], 0 [dB] (c) and (d), and 3 [dB] (e) and (f).

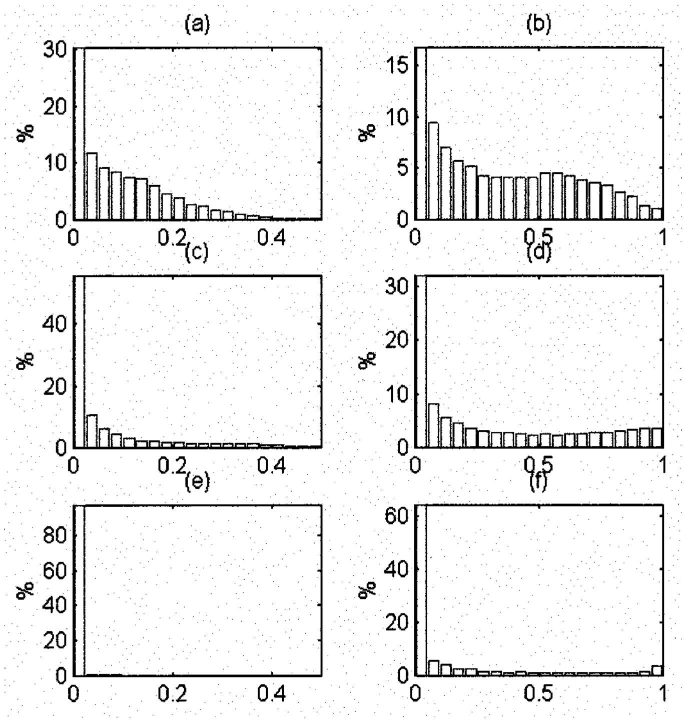

Fig. 7. Hamming $[(7,4,3)]^2$—histograms of the magnitude of the eigenvalues of (a) S and (b) S^R for SNR of -3 [dB], 0 [dB] (c) and (d), and 3 [dB] (e) and (f) (eigenvalues of S larger than 0.5, and eigenvalues of S^R larger than 1 were truncated).

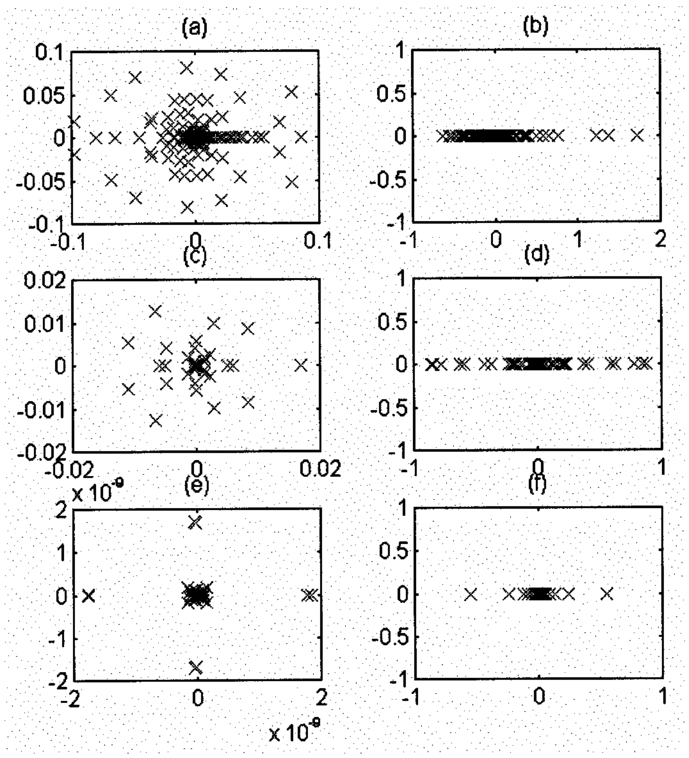

Fig. 6. Golay $[(24,12,8)]^2$—eigenvalues spread of (a) S and (b) S^R for SNR of -3 [dB], 0 [dB] (c) and (d), and 3 [dB] (e) and (f).

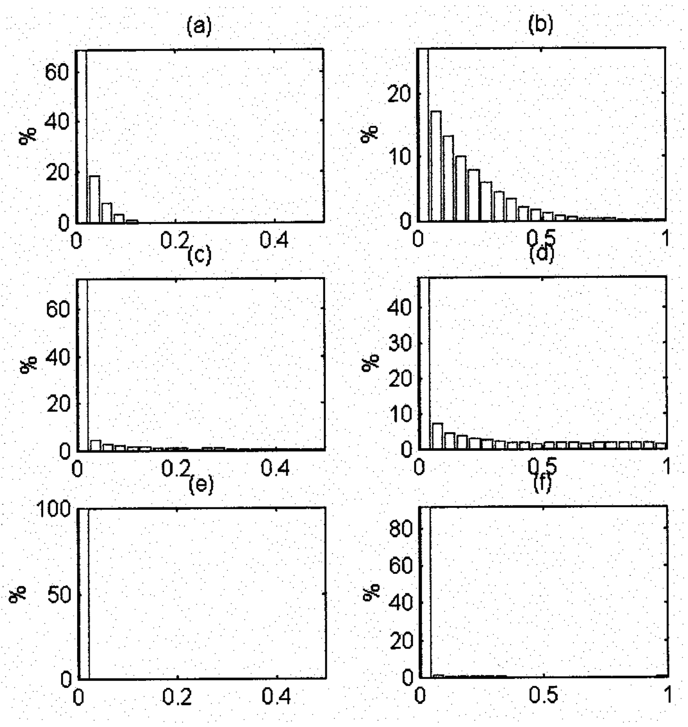

Fig. 8. Golay $[(24,12,8)]^2$—histograms of the magnitude of the eigenvalues of (a) S and (b) S^R for SNR of -3 [dB], 0 [dB] (c) and (d), and 3 [dB] (e) and (f) (eigenvalues of S larger than 0.5, and eigenvalues of S^R larger than 1 were truncated).

Tables I and II show the percentage of stable stability matrices for both codes, i.e., percentage of matrices in which all the eigenvalues are inside the unit disk. For example, for the Golay $[(24,12,8)]^2$ code, at an SNR of 0 [dB], only 25% of the stability matrices of the rows and columns decoders are stable, but when they are combined to form S, the stability percentage in close to 90%. This demonstrates how successful column decoding can compensate for unsuccessful row decoding, and *vice versa* (refer to the discussion in Section III-E).

It is interesting to observe that the averaged maximal eigenvalue of S is orders of magnitudes smaller than the eigenvalues of S^R and S^C. This becomes more obvious as the SNR is larger. For example, for SNR of 5 [dB], the averaged eigenvalues of the rows and columns stability matrices are in the order of 10^{-1}, while the averaged eigenvalues of the total stability matrix are in the order of 10^{-7} (Hamming) to 10^{-14} (Golay). Note that the upper bound in (43) predicts only a factor of 2 (in logarithmic scale).

Another phenomenon is the large difference between the Hamming and Golay maximal eigenvalues of the total stability matrix, compared to the small difference between the maximal eigenvalues of the rows and columns stability matrices for these codes.

In Figs. 5 and 6, we have plotted representative eigenvalues spreads for three different SNRs: -3, 0, and 3 [dB]. This spread is obtained by calculating the eigenvalues of S and S^R for a single information block decoding (the eigenvalues spread of S^C is similar to that of S^R and was therefore omitted) at the algorithm's fixed point. The algorithm's (practical) fixed point is defined as $(Q_y^{(M)}, Q_z^{(M)})$, where M is the first iteration for which the following holds (we used $\varepsilon = 10^{-5}$):

$$\left| \frac{\left(Q_y^{(M)}\right)_{i,j} - \left(Q_y^{(M-1)}\right)_{i,j}}{\left(Q_y^{(M)}\right)_{i,j}} \right| < \varepsilon \qquad \forall\, i,\, j$$
$$\left| \frac{\left(Q_z^{(M)}\right)_{i,j} - \left(Q_z^{(M-1)}\right)_{i,j}}{\left(Q_z^{(M)}\right)_{i,j}} \right| < \varepsilon \qquad \forall\, i,\, j.$$

These eigenvalue spreads were selected to represent the general behavior observed at the various SNRs. A more quantitative measure is presented in Figs. 7 and 8.

Note that the eigenvalues of S^R and S^C are always real. We proved this property for any 2×2 (information bits) codes (see Section III-D). For the general case, it was proved in [2] that (in general) there exists a basis in which J^R and J^C, and hence S^R and S^C, are simultaneously symmetric. It follows that they have real eigenvalues.

Figs. 7 and 8 show the histograms of the magnitudes of the eigenvalues of S and S^R. From these results, we learn that as the SNR increases, the magnitudes of vast majority of the eigenvalues of the rows (and columns) stability matrix become more concentrated around the origin. Therefore, observing the maximal eigenvalue in this case (as we did in Figs. 3 and 4) does not reveal the whole picture.

Recall that from Claim 1 (Section III-C), the set of eigenvalues of S^R is the union of the set of eigenvalues of each row stability matrix $(S^{R,i})$. Thus, a large eigenvalue will indicate a large uncertainty in the decoding of some bits in its corresponding row. If some of the corresponding column bits (see Section III-E) also have large uncertainties, we would expect large eigenvalues in the total stability matrix. But, as indicated by simulation results, at large SNR, the number of "problematic" row bits becomes smaller, and therefore their chances to meet corresponding "problematic" column bits becomes significantly smaller. This can explain why the eigenvalues of S are so small compared to these of S^R and S^C, and gives us a deeper understanding of the convergence of the turbo decoding algorithm.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewer for his helpful comments.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. of Int. Communications Conf. (ICC'93)*, pp. 1064–1070.
[2] T. Richardson, "The geometry of turbo-decoding dynamics," *IEEE Trans. Inform. Theory*, vol. 46, pp. 9–23, Jan. 2000.
[3] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–427, Mar. 1996.
[4] R. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, pp. 1003–1010, Aug. 1998.
[5] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.
[6] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.
[7] L. Duan and B. Rimoldi, "The turbo decoding algorithm has fixed points," in *Proc. IEEE Int. Symp. Information Theory (ISIT'98)*, Cambridge, MA, Aug. 1998, p. 277.
[8] R. J. McEliece, E. R. Rodemich, and J. Cheng, "The turbo decision algorithm," in *Proc. 33rd Allerton Conf. Communication, Control and Computing*, Aug. 1995.
[9] R. J. McEliece, D. J. C. MacKay, and J. F. Cheng, "Turbo decoding as an instance of Pearl's 'Belief Propagation' algorithm," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 140–152, Feb. 1998.
[10] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Comput.*, vol. 12, pp. 1–41, 1998.
[11] R. A. Horn and C. R. Johnson, *Matrix Analysis*. U.K.: Cambridge Univ. Press, 1991.