A Geometric Description of the Iterative Least-Squares Decoding of Analog Block Codes

Fangning Hu and Werner Henkel
International University Bremen (IUB)
Campus Ring 1
D-28759 Bremen, Germany
Email: {f.hu, w.henkel}@iu-bremen.de

Abstract

This paper outlines that when decoding an arbitrary analog block codes, i.e., a block codes over the complex or real numbers, in an iterative fashion by splitting the H-Matrix in two, leads to a least-squares estimate. Such a Turbo-like algorithm represents iterative projections in Euclidean space. A step size controls the convergence speed. The paper generalized an earlier result based on array codes (product codes) with analog parity-check component codes. The results in here are considered to be an important step towards an intuitive understanding of iterative decoding schemes of conventional Turbo and LDPC codes.

1 Introduction

Codes over real or complex numbers had first been proposed in [1], [2]. These early studies, which were further refined in [3], [4], were focusing on analog RS codes, i.e., RS codes over complex numbers with almost traditional Hamming-metric decoding. More recent papers, e.g., [5], [6], [7] show applications in multicarrier transmission. One may note that an OFDM (Orthogonal Frequency Division Multiplex) or DMT (Discrete MultiTone) signal is nothing else than an analog RS code or BCH code, respectively, in case some cyclically consecutive carriers are not used for transmission. [6] also studies the use of pilot tones which are scattered over the OFDM/DMT symbol.

In [8], authors described an iterative Turbo-like decoding of analog product codes with parity-check component codes and showed that this algorithm leads to the least-squares solution. Their algorithm is slightly different from standard Turbo decoding. In here, we slightly modify their algorithm into standard Turbo decoding.

Our decoding algorithm for the analog product code can be regarded as a special case of the Gaussian message passing (or Kalman filtering/smoothing) on graphs. [9], [10] and [11] show that when the graph represents many Gaussian distributed variables, it leads to a least squares solution. In this paper, we prove the convergence and propose a geometric description of our algorithm. With this geometric description, the convergence process of Turbo decoding for analog product code can be clearly illustrated by iterative projections in Euclidean space. A similar geometric description can be applied to the algorithm of [8]. Based on this geometric description, we further apply the same decoding scheme to any analog block code by splitting the parity matrix in two.

We first start with the analog product code. Let $K = k^2$ analog information symbols ($\in \mathbb{R}$) be arranged as an $k \times k$ matrix. This information matrix is mapped to the $N = (k+1) \times (k+1)$ code matrix $\boldsymbol{X}$

$$\boldsymbol{X} = \left[\begin{array}{ccc|c} x_{1,1} & \cdots & x_{1,k} & x_{1,k+1} \\ \vdots & \ddots & \vdots & \vdots \\ x_{k,1} & \cdots & x_{k,k} & x_{k,k+1} \\ \hline x_{k+1,1} & \cdots & x_{k+1,k} & x_{k+1,k+1} \end{array}\right] , \quad (1)$$

such that

$$\begin{aligned} \boldsymbol{1}^T \cdot \boldsymbol{X} &= \boldsymbol{0} \\ \boldsymbol{X} \cdot \boldsymbol{1} &= \boldsymbol{0} \end{aligned} \quad (2)$$

hold, where $\boldsymbol{1}$ is an $(k+1)$-dimensional column vector of all ones. The equations represent column and row constraints, respectively, i.e., the sum along each column/row equals zero.

In order to simplify the analysis, we define a column vector $\boldsymbol{x}$ to contain the sequence of its columns. We write

$$\boldsymbol{x} = \text{vec}\ (\boldsymbol{X})\ .$$

2 Decoding Strategy

Given the received noisy codeword $\boldsymbol{R}_{(0)}$, two kinds of extrinsic informations are computed according to the column and row constraints, respectively, as follows:

$$\begin{aligned} r_{i,j}^{col} &= -\sum_{l=1,l\neq i}^{k+1} r_{l,j}\ , \quad i,j = 1,\cdots,k+1\ , \\ r_{i,j}^{row} &= -\sum_{l=1,l\neq j}^{k+1} r_{i,l}\ , \quad i,j = 1,\cdots,k+1\ , \end{aligned} \quad (3)$$

with $r_{i,j}$ denoting the $(i,j)th$ entry of $\boldsymbol{R}_{(\nu)}, \nu = 0, 1, \ldots$. Equations (3) can be rewritten as the entries of two update matrices

$$\bar{\boldsymbol{I}}\boldsymbol{R}_{(\nu)} \quad \text{and} \quad \boldsymbol{R}_{(\nu)}\bar{\boldsymbol{I}} \tag{4}$$

where

$$\bar{\boldsymbol{I}} = \boldsymbol{E} - \boldsymbol{I} = \begin{bmatrix} 0 & 1 & \cdots & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & \cdots & \cdots & 1 & 0 \end{bmatrix}, \tag{5}$$

$\boldsymbol{E}$ being an $(k+1)\times(k+1)$ matrix of all $1'$s. Since these two matrices are used to extract extrinsic information, we call them extrinsic matrices. The decoding method is an analog counterpart of the Turbo decoding for binary product codes in [12] where the decoding by column and row constraints is regarded as two independent decoders. At each iteration step, we decode it either according to the column constraints

$$\boldsymbol{R}_{(\nu)} = \frac{\boldsymbol{R}_{(\nu-1)} - w_1\bar{\boldsymbol{I}}\boldsymbol{R}_{(\nu-1)}}{w_1+1}, \tag{6}$$

or the row constraints

$$\boldsymbol{R}_{(\nu+1)} = \frac{\boldsymbol{R}_{(\nu)} - w_2\boldsymbol{R}_{(\nu)}\bar{\boldsymbol{I}}}{w_2+1}, \tag{7}$$

where ν denotes the iteration step index and w_1, w_2 are weighting factors.

Let $\boldsymbol{y} = \text{vec}\ (\boldsymbol{R})$ contain the sequence of its columns. The extrinsic matrices in Eqn. (4) can then be rewritten as

$$\boldsymbol{M}_1\boldsymbol{y} \quad \text{and} \quad \boldsymbol{M}_2\boldsymbol{y}$$

with

$$\boldsymbol{M}_1 = \boldsymbol{I} \otimes \bar{\boldsymbol{I}}, \boldsymbol{M}_2 = \bar{\boldsymbol{I}} \otimes \boldsymbol{I}, \tag{8}$$

where $\otimes$ denotes the Kronecker product. Then, the iterative algorithm in equations (6) and (7) can be reformulated as

$$\begin{aligned} \boldsymbol{y}_{(\nu)} &= (\boldsymbol{y}_{(\nu-1)} - w_1\boldsymbol{M}_1\boldsymbol{y}_{(\nu-1)})/(w_1+1), \\ \boldsymbol{y}_{(\nu+1)} &= (\boldsymbol{y}_{(\nu)} - w_2\boldsymbol{M}_2\boldsymbol{y}_{(\nu)})/(w_2+1), \end{aligned} \tag{9}$$

which can be further simplified as

$$\begin{aligned} &\boldsymbol{y}_{(\nu)} = \\ &= [\boldsymbol{I} - w_1(\boldsymbol{I} \otimes \bar{\boldsymbol{I}})]\boldsymbol{y}_{(\nu-1)}/(w_1+1) \\ &= [\boldsymbol{I} - w_1(\boldsymbol{I} \otimes (\boldsymbol{E} - \boldsymbol{I}))]\boldsymbol{y}_{(\nu-1)}/(w_1+1) \\ &= [\boldsymbol{I} - w_1(\boldsymbol{I} \otimes \boldsymbol{E} - \boldsymbol{I})]\boldsymbol{y}_{(\nu-1)}/(w_1+1) \\ &= [(w_1+1)\boldsymbol{I} - w_1(\boldsymbol{I} \otimes \boldsymbol{E})]\boldsymbol{y}_{(\nu-1)}/(w_1+1) \\ &= \boldsymbol{y}_{(\nu-1)} - \frac{w_1}{w_1+1}(\boldsymbol{I} \otimes \boldsymbol{E})\boldsymbol{y}_{(\nu-1)} \\ &= \boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}^{(1)}_{(\nu-1)} = \boldsymbol{A}_1\boldsymbol{y}_{(\nu-1)} \end{aligned} \tag{10}$$

and

$$\begin{aligned} \boldsymbol{y}_{(\nu+1)} &= [\boldsymbol{I} - w_2(\bar{\boldsymbol{I}} \otimes \boldsymbol{I})]\boldsymbol{y}_{(\nu)}/(w_2+1) \\ &= [\boldsymbol{I} - w_2((\boldsymbol{E} - \boldsymbol{I}) \otimes \boldsymbol{I})]\boldsymbol{y}_{(\nu)}/(w_2+1) \\ &= [\boldsymbol{I} - w_2(\boldsymbol{E} \otimes \boldsymbol{I} - \boldsymbol{I})]\boldsymbol{y}_{(\nu)}/(w_2+1) \\ &= [(w_2+1)\boldsymbol{I} - w_2(\boldsymbol{E} \otimes \boldsymbol{I})]\boldsymbol{y}_{(\nu)}/(w_2+1) \\ &= \boldsymbol{y}_{(\nu)} - \frac{w_2}{w_2+1}(\boldsymbol{E} \otimes \boldsymbol{I})\boldsymbol{y}_{(\nu)} \\ &= \boldsymbol{y}_{(\nu)} - \boldsymbol{y}^{(2)}_{(\nu)} = \boldsymbol{A}_2\boldsymbol{y}_{(\nu)}, \end{aligned} \tag{11}$$

with

$$\begin{aligned} \boldsymbol{y}^{(1)}_{(\nu-1)} &= \tfrac{w_1}{w_1+1}(\boldsymbol{I} \otimes \boldsymbol{E})\boldsymbol{y}_{(\nu-1)}, \\ \boldsymbol{y}^{(2)}_{(\nu)} &= \tfrac{w_2}{w_2+1}(\boldsymbol{E} \otimes \boldsymbol{I})\boldsymbol{y}_{(\nu)}), \end{aligned} \tag{12}$$

and

$$\begin{aligned} \boldsymbol{A}_1 &= \boldsymbol{I} \otimes (\boldsymbol{I} - \tfrac{w_1}{w_1+1}\boldsymbol{E}), \\ \boldsymbol{A}_2 &= (\boldsymbol{I} - \tfrac{w_2}{w_2+1}\boldsymbol{E}) \otimes \boldsymbol{I}. \end{aligned} \tag{13}$$

Substituting Eqn. (10) and Eqn. (13) into Eqn. (11), we obtain

$$\begin{aligned} \boldsymbol{y}_{(\nu+1)} &= \boldsymbol{A}_2\boldsymbol{A}_1\boldsymbol{y}_{(\nu-1)} \\ &= [(\boldsymbol{I} - \tfrac{w_2}{w_2+1}\boldsymbol{E}) \otimes \boldsymbol{I}] \\ &\quad \cdot[\boldsymbol{I} \otimes (\boldsymbol{I} - \tfrac{w_1}{w_1+1}\boldsymbol{E})] \cdot \boldsymbol{y}_{(\nu-1)} \\ &= (\boldsymbol{I} - \tfrac{w_2}{w_2+1}\boldsymbol{E}) \otimes (\boldsymbol{I} - \tfrac{w_1}{w_1+1}\boldsymbol{E}) \cdot \boldsymbol{y}_{(\nu-1)} \\ &= \boldsymbol{\Phi} \cdot \boldsymbol{y}_{(\nu-1)} = \boldsymbol{\Phi}^{(\nu+1)/2} \cdot \boldsymbol{y}_{(0)}. \end{aligned} \tag{14}$$

for $\nu+1$ even. The sufficient condition for the algorithm to converge is the eigenvalue of $\boldsymbol{\Phi}$ not to be greater than one. By selecting $\boldsymbol{T} = \boldsymbol{F} \otimes \boldsymbol{F}$ where $\boldsymbol{F}$ is an $k+1$ by $k+1$ DFT matrix, we diagonalize $\boldsymbol{\Phi}$ to $\boldsymbol{\Lambda} = \boldsymbol{T}^H\boldsymbol{\Phi}\boldsymbol{T}$ where the diagonal elements of $\boldsymbol{\Lambda}$ are the eigenvalues of $\boldsymbol{\Phi}$. The following provides the derivation:

$$\begin{aligned} \boldsymbol{\Lambda} &= \boldsymbol{T}^H\boldsymbol{\Phi}\boldsymbol{T} \\ &= (\boldsymbol{F}^H \otimes \boldsymbol{F}^H) \cdot [(\boldsymbol{I} - \tfrac{w_2}{w_2+1}\boldsymbol{E}) \otimes (\boldsymbol{I} - \tfrac{w_1}{w_1+1}\boldsymbol{E})] \\ &\quad \cdot(\boldsymbol{F} \otimes \boldsymbol{F}) \\ &= [\boldsymbol{F}^H(\boldsymbol{I} - \tfrac{w_2}{w_2+1}\boldsymbol{E})\boldsymbol{F}] \otimes [\boldsymbol{F}^H(\boldsymbol{I} - \tfrac{w_1}{w_1+1}\boldsymbol{E})\boldsymbol{F}] \\ &= (\boldsymbol{I} - \tfrac{w_2}{w_2+1}\begin{bmatrix} k+1 & 0 \\ 0 & \boldsymbol{0}_{k\times k} \end{bmatrix}) \\ &\otimes (\boldsymbol{I} - \tfrac{w_1}{w_1+1}\begin{bmatrix} k+1 & 0 \\ 0 & \boldsymbol{0}_{k\times k} \end{bmatrix}) \\ &= \begin{bmatrix} \frac{1-kw_2}{w_2+1} & 0 \\ 0 & \boldsymbol{I}_{k\times k} \end{bmatrix} \otimes \begin{bmatrix} \frac{1-kw_1}{w_1+1} & 0 \\ 0 & \boldsymbol{I}_{k\times k} \end{bmatrix}. \end{aligned} \tag{15}$$

Thus, we have

$$\begin{aligned} \text{diag}(\Lambda) = &[\frac{(1-kw_1)(1-kw_2)}{(w_1+1)(w_2+1)}, \underbrace{\frac{1-kw_2}{w_2+1}, \ldots, \frac{1-kw_2}{w_2+1}}_{k}, \\ &\underbrace{\underbrace{\frac{1-kw_1}{w_1+1}, 1, \ldots, 1}_{k+1}, \ldots\ldots \underbrace{\frac{1-kw_1}{w_1+1}, 1, \ldots, 1}_{k+1}}_{k}] \end{aligned} \tag{16}$$

$$\begin{array}{lll} 1 & \text{eigenvalue equal to} & \frac{(1-kw_1)(1-nw-2)}{(w_1+1)(w_2+1)} \\ k & \text{eigenvalues equal to} & \frac{1-kw_1}{w_1+1} \\ k & \text{eigenvalues equal to} & \frac{1-kw_2}{w_2+1} \\ k^2 & \text{eigenvalues equal to} & 1 \end{array}. \tag{17}$$

For convergence, we require

$$\begin{cases} |\frac{(1-kw_1)(1-kw_2)}{(w_1+1)(w_2+1)}| & < 1 \\ |\frac{1-kw_1}{w_1+1}| & < 1 \\ |\frac{1-kw_2}{w_2+1}| & < 1 \end{cases}, \tag{18}$$

which yields the conditions

$$0 < w_1 < \frac{2}{k-1}, \quad 0 < w_2 < \frac{2}{k-1}. \tag{19}$$

3 Geometric Description

In this section, we will give a geometric description of this iterative algorithm. The column and row constraints described in (2) can be rewritten as

$$\begin{aligned} \text{vec } (\mathbf{1}^T \cdot \boldsymbol{X}) &= \mathbf{0} \Rightarrow (\boldsymbol{I} \otimes \mathbf{1}^T) \cdot \boldsymbol{x} = \mathbf{0} \,, \\ \text{vec } (\boldsymbol{X} \cdot \mathbf{1}) &= \mathbf{0} \Rightarrow (\mathbf{1}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{x} = \mathbf{0} \,. \end{aligned} \quad (20)$$

A parity-check matrix $\boldsymbol{H}, \boldsymbol{H}\boldsymbol{x} = \mathbf{0}$ for such analog codes can be constructed from (20) as

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_1 \\ \boldsymbol{H}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} \otimes \mathbf{1}^T \\ \mathbf{1}^T \otimes \boldsymbol{I} \end{bmatrix} . \quad (21)$$

$\boldsymbol{H}_1$ and $\boldsymbol{H}_2$ are column and row constraints, respectively. The parity-check matrix uniquely defines the code space $\mathcal{X}$.

$$\mathcal{X} = \{\boldsymbol{x} : \boldsymbol{H}\boldsymbol{x} = \mathbf{0}\} .$$

The least-squares solution is the projection of a received vector $\boldsymbol{y}$ onto the code space $\mathcal{X}$. Furthermore, if we define two superspaces corresponding to the column and row constraints as

$$\begin{aligned} \mathcal{G}_1 &= \{\boldsymbol{x} : \boldsymbol{H}_1\boldsymbol{x} = \mathbf{0}\} \,, \\ \mathcal{G}_2 &= \{\boldsymbol{x} : \boldsymbol{H}_2\boldsymbol{x} = \mathbf{0}\} \,, \end{aligned} \quad (22)$$

then $\mathcal{X} = \mathcal{G}_1 \cap \mathcal{G}_2$.

Lemma 3.1: $\boldsymbol{y}^{(1)}_{(\nu-1)}$ is orthogonal to space $\mathcal{G}_1$ and $\boldsymbol{y}^{(2)}_{(\nu)}$ is orthogonal to space $\mathcal{G}_2$.

Proof: From (22), we know that any column of $\boldsymbol{H}_1^T$ is orthogonal to the space $\mathcal{G}_1$. Then, any linear combination of columns of $\boldsymbol{H}_1^T$ is also orthogonal to $\mathcal{G}_1$. We show that $\boldsymbol{y}^{(1)}_{(\nu-1)}$ is a linear combination of columns of $\boldsymbol{H}_1^T$ as

$$\boldsymbol{y}^{(1)}_{(\nu-1)} = \boldsymbol{H}_1^T \boldsymbol{\alpha} \,,$$

where

$$\boldsymbol{\alpha} = \frac{w_1}{w_1+1} \boldsymbol{H}_1 \boldsymbol{y}_{(\nu-1)} \,.$$

We verify this by

$$\begin{aligned} \boldsymbol{H}_1^T \boldsymbol{\alpha} &= \tfrac{w_1}{w_1+1} \boldsymbol{H}_1^T \boldsymbol{H}_1 \boldsymbol{y}_{(\nu-1)} \\ &= \tfrac{w_1}{w_1+1} (\boldsymbol{I} \otimes \mathbf{1}^T)^T (\boldsymbol{I} \otimes \mathbf{1}^T) \boldsymbol{y}_{(\nu-1)} \\ &= \tfrac{w_1}{w_1+1} (\boldsymbol{I} \otimes \mathbf{1})(\boldsymbol{I} \otimes \mathbf{1}^T) \boldsymbol{y}_{(\nu-1)} \\ &= \tfrac{w_1}{w_1+1} (\boldsymbol{I} \otimes (\mathbf{1} \cdot \mathbf{1}^T)) \boldsymbol{y}_{(\nu-1)} \\ &= \tfrac{w_1}{w_1+1} (\boldsymbol{I} \otimes \boldsymbol{E}) \boldsymbol{y}_{(\nu-1)} \\ &= \boldsymbol{y}^{(1)}_{(\nu-1)} \,. \end{aligned} \quad (23)$$

Similarly, $\boldsymbol{y}^{(2)}_{(\nu)} = \boldsymbol{H}_2^T \boldsymbol{\beta}$ where $\boldsymbol{\beta} = \frac{w_2}{w_2+1} \boldsymbol{H}_2 \boldsymbol{y}_{(\nu)}$. ■

Lemma 3.2: When $w_1 = w_2 = \dfrac{1}{k}$, then $\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}^{(1)}_{(\nu-1)}$ is the projection of $\boldsymbol{y}_{(\nu-1)}$ onto space $\mathcal{G}_1$ and $\boldsymbol{y}_{(\nu)} - \boldsymbol{y}^{(2)}_{(\nu)}$ is the projection of $\boldsymbol{y}_{(\nu)}$ onto space $\mathcal{G}_2$.

Proof: Since $\boldsymbol{y}^{(1)}_{(\nu-1)}$ is orthogonal to $\mathcal{G}_1$ by Lemma 3.1, we only need to prove that $\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}^{(1)}_{(\nu-1)}$ lies in $\mathcal{G}_1$ (see Fig. 1). With $\boldsymbol{H}_1 = \boldsymbol{I} \otimes \mathbf{1}^T$ and (12),

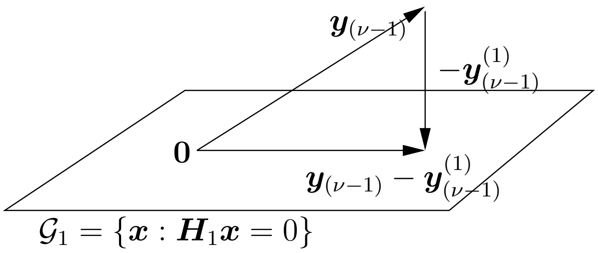

Fig. 1. The projection of $\boldsymbol{y}_{(\nu-1)}$ onto the space $\mathcal{G}_1$

we obtain

$$\begin{aligned} &\boldsymbol{H}_1 \cdot (\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}^{(1)}_{(\nu-1)}|_{w_1=\frac{1}{k}}) \\ &= \boldsymbol{H}_1\boldsymbol{y}_{(\nu-1)} - \tfrac{1}{k+1} \boldsymbol{H}_1 (\boldsymbol{I} \otimes \boldsymbol{E}) \boldsymbol{y}_{(\nu-1)} \\ &= \boldsymbol{H}_1\boldsymbol{y}_{(\nu-1)} - \tfrac{1}{k+1} (\boldsymbol{I} \otimes \mathbf{1}^T)(\boldsymbol{I} \otimes \boldsymbol{E}) \boldsymbol{y}_{(\nu-1)} \\ &= \boldsymbol{H}_1\boldsymbol{y}_{(\nu-1)} - \tfrac{1}{k+1} (\boldsymbol{I} \otimes (\mathbf{1}^T \cdot \boldsymbol{E})) \boldsymbol{y}_{(\nu-1)} \\ &= \boldsymbol{H}_1\boldsymbol{y}_{(\nu-1)} - \tfrac{k+1}{k+1} (\boldsymbol{I} \otimes \mathbf{1}^T) \boldsymbol{y}_{(\nu-1)} \\ &= \boldsymbol{H}_1\boldsymbol{y}_{(\nu-1)} - \boldsymbol{H}_1\boldsymbol{y}_{(\nu-1)} = \mathbf{0} \,. \end{aligned} \quad (24)$$

This means, $\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}^{(1)}_{(\nu-1)}$ lies in $\mathcal{G}_1$ by the definition of $\mathcal{G}_1$ (cf. (22)). Similarly, $\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}^{(2)}_{(\nu)}$ lies in $\mathcal{G}_2$. ■

The dimension should be at least $(k+1)^2 = 4$. However, in order to get an intuitive insight to this projection process, we first consider a two-dimensional case.

It is clear from Lemma 3.2 that w_1, w_2 control the length of the vectors $\boldsymbol{y}^{(1)}_{(\nu-1)}$ and $\boldsymbol{y}^{(2)}_{(\nu)}$, respectively. The larger w_1, w_2, the longer are the vectors $\boldsymbol{y}^{(1)}_{(\nu-1)}$ and $\boldsymbol{y}^{(2)}_{(\nu)}$.

In the following, we show four cases with different values of w_1, w_2, i.e., different projection lengths. The first case is when $w_1 = w_2 = 1/k$. In this case, exact projections occur according to Lemma 3.2. This procedure can be described by projections in Euclidean space illustrated in Fig. 2. At each iteration step ν, the previous vector is projected onto $\mathcal{G}_1$ and $\mathcal{G}_2$ in turn. From Fig. 2, we see that this process makes $\boldsymbol{y}_{(\infty)}$ converge to $\mathcal{G}_1 \cap \mathcal{G}_2$, the overall least-squares solution.

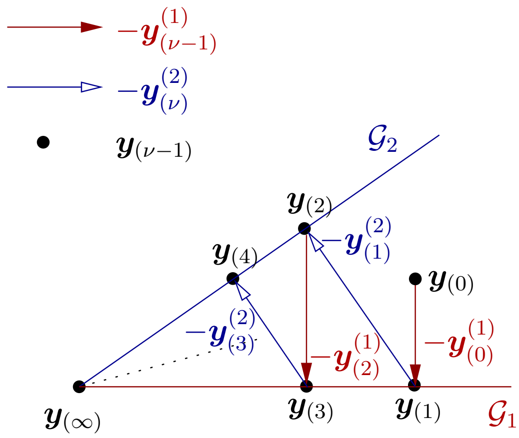

Fig. 2. The convergence process by iterative projections in a two-dimensional Euclidean space when $w_1 = w_2 = 1/k$

When w_1, w_2 achieve the convergence bound at $w_1 = w_2 = \frac{2}{k-1}$ as shown in Eqn. (19), we find that the length of the projection vectors is exactly twice the projection length and an oscillatory phenomenon occurs. To simplify the description, we use two orthogonal spaces $\mathcal{G}_1, \mathcal{G}_2$ to illustrate the oscillatory phenomenon as shown in Fig. 3

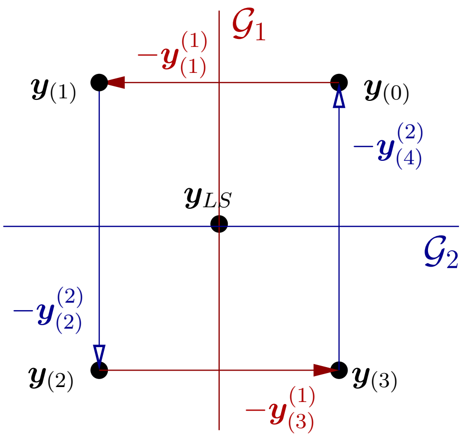

Fig. 3. The oscillatory phenomenon at the converging bound in a two-dimensional Euclidean space when $w_1 = w_2 = 2/(k-1)$

When $w_1, w_2 < \frac{2}{k-1}$, we see in Fig. 4 that the process finally slowly converges to the least-squares solution.

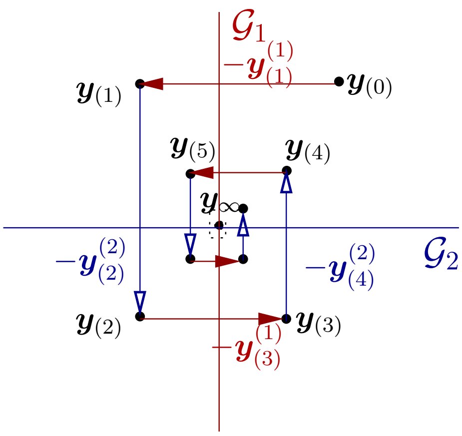

Fig. 4. The convergence process by iterative projections in a two-dimensional Euclidean space when $w_1 = w_2 < 2/(k-1)$

In the last case with $w_1, w_2 > \frac{2}{k-1}$, the iterative algorithm will diverges as shown in Fig. 5.

Now consider the decoding algorithm in [8] with

$$\boldsymbol{R}_{(\nu)} = \frac{\boldsymbol{R}_{(\nu-1)} - w\bar{\boldsymbol{I}}\boldsymbol{R}_{(\nu-1)} - w\boldsymbol{R}_{(\nu-1)}\bar{\boldsymbol{I}}}{1+2w} . \quad (25)$$

It computes the current matrix as the weighted sum of the intrinsic and extrinsic matrices. Their algorithm differs from the Turbo scheme in updating the current matrix by using both column and row extrinsic matrices at the same time.

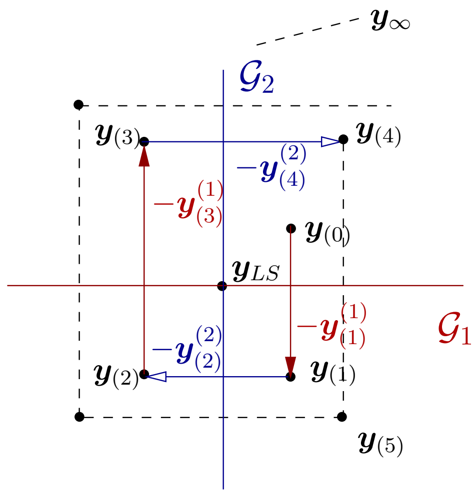

Fig. 5. The divergence process by iterative projections in a two-dimensional Euclidean space when $w_1 = w_2 > 2/(k-1)$

By a vector representation, we obtain:

$$\boldsymbol{y}_{(\nu)} = (\boldsymbol{y}_{(\nu-1)} - w\boldsymbol{M}_1\boldsymbol{y}_{(\nu-1)} - w\boldsymbol{M}_2\boldsymbol{y}_{(\nu-1)})/(1+2w) . \quad (26)$$

It has been proven in [8] that $\boldsymbol{y}_{(\infty)}$ converges to the least-squares solution when $0 < w < 1/k$. However we found there is a slight error on their convergence condition and the correct one should be $0 < w < 1/(k-1)$. In the following, we will show how this Turbo-like iterative method can also be illustrated by projections in Euclidean space.

Substituting (8) into (26), we see that Iteration (26) is equivalent to the following equations:

$$\begin{aligned}
&\boldsymbol{y}_{(\nu)} = \\
&= [\boldsymbol{I} - w(\boldsymbol{I} \otimes \bar{\boldsymbol{I}}) - w(\bar{\boldsymbol{I}} \otimes \boldsymbol{I})]\boldsymbol{y}_{(\nu-1)}/(1+2w) \\
&= [\boldsymbol{I} - w(\boldsymbol{I} \otimes (\boldsymbol{E} - \boldsymbol{I})) \\
&\quad - w((\boldsymbol{E} - \boldsymbol{I}) \otimes \boldsymbol{I})]\boldsymbol{y}_{(\nu-1)}/(1+2w) \\
&= [\boldsymbol{I} - w(\boldsymbol{I} \otimes \boldsymbol{E} - \boldsymbol{I}) \\
&\quad - w(\boldsymbol{E} \otimes \boldsymbol{I} - \boldsymbol{I})]\boldsymbol{y}_{(\nu-1)}/(1+2w) \\
&= [(1+2w)\boldsymbol{I} \\
&\quad - w(\boldsymbol{I} \otimes \boldsymbol{E}) - w(\boldsymbol{E} \otimes \boldsymbol{I})]\boldsymbol{y}_{(\nu-1)}/(1+2w) \\
&= \boldsymbol{y}_{(\nu-1)} - \frac{w}{1+2w}(\boldsymbol{I} \otimes \boldsymbol{E})\boldsymbol{y}_{(\nu-1)} \\
&\quad - \frac{w}{1+2w}(\boldsymbol{E} \otimes \boldsymbol{I})\boldsymbol{y}_{(\nu-1)} .
\end{aligned} \quad (27)$$

In order to simplify the description, we define

$$\begin{aligned}
\boldsymbol{y}_{(\nu-1)}^{(1)} &= \tfrac{w}{1+2w}(\boldsymbol{I} \otimes \boldsymbol{E})\boldsymbol{y}_{(\nu-1)} , \\
\boldsymbol{y}_{(\nu-1)}^{(2)} &= \tfrac{w}{1+2w}(\boldsymbol{E} \otimes \boldsymbol{I})\boldsymbol{y}_{(\nu-1)} .
\end{aligned} \quad (28)$$

With a similar proof as in Lemma 3.1 and Lemma 3.2, we found that $\boldsymbol{y}_{(\nu-1)}^{(1)}, \boldsymbol{y}_{(\nu-1)}^{(2)}$ are orthogonal to $\mathcal{G}_1, \mathcal{G}_2$, respectively. w controls the length of the two projection vectors $-\boldsymbol{y}_{(\nu-1)}^{(1)}$ and $-\boldsymbol{y}_{(\nu-1)}^{(2)}$. When $w = 1/(k-1)$, $\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}_{(\nu-1)}^{(1)}$ and $\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}_{(\nu-1)}^{(2)}$ are the projections of $\boldsymbol{y}_{(\nu-1)}$ onto spaces $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. A geometric illustration of Eqn. (27) for a two-dimensional case with $w = \frac{1}{k-1}$ is given in Fig. 6. At each iteration step ν, the previous vector

$\boldsymbol{y}_{(\nu-1)}$ is projected onto $\mathcal{G}_1$ and $\mathcal{G}_2$ in parallel which delivers two projection vectors $-\boldsymbol{y}^{(1)}_{(\nu-1)}$ and $-\boldsymbol{y}^{(2)}_{(\nu-1)}$. Both projection vectors are added to $\boldsymbol{y}_{(\nu-1)}$ resulting in the current vector $\boldsymbol{y}_{(\nu)}$. From Fig. 6, we see that this process makes $\boldsymbol{y}_{(\infty)}$ converge to $\mathcal{G}_1 \cap \mathcal{G}_2$, the overall least-squares solution. However, instead of one projection in each step, there are two projections in parallel at each iteration step.

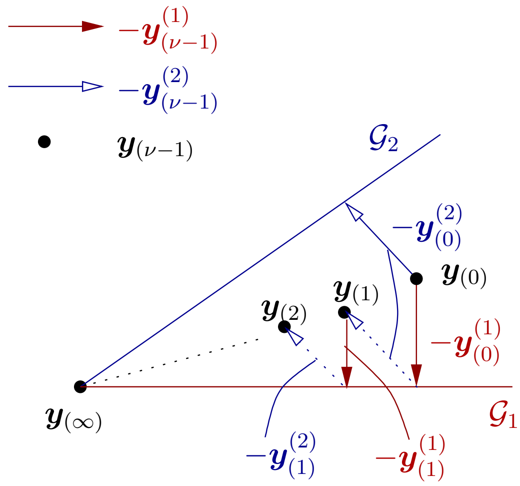

Fig. 6. The convergence process by iterative projections in a two-dimensional Euclidean space with two-fold projections and with $w = 1/(k-1)$.

4 Iterative Decoding of Arbitrary Linear Analog Block Codes

For any block analog code, its parity check matrix $\boldsymbol{H}$ can be expressed as

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_1 \\ \boldsymbol{H}_2 \end{bmatrix} . \tag{29}$$

Let $\mathcal{G}_1, \mathcal{G}_2$ be orthogonal spaces to $\boldsymbol{H}_1, \boldsymbol{H}_2$, respectively. These orthogonal spaces are, of course, spanned by the rows of the corresponding generator matrices. From the described geometric interpretation, our decoding approach is to find the projections of a vector $\boldsymbol{y}_{(\nu-1)}$ onto $\mathcal{G}_1, \mathcal{G}_2$.

Assume that $\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}^{(1)}_{(\nu-1)}$ is the projection of $\boldsymbol{y}_{(\nu-1)}$ onto space $\mathcal{G}_1$ and $\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}^{(2)}_{(\nu-1)}$ is the projection of $\boldsymbol{y}_{(\nu-1)}$ onto space $\mathcal{G}_2$. The iterative algorithm can be written as

$$\boldsymbol{y}_{(\nu)} = \boldsymbol{y}_{(\nu-1)} - \lambda \boldsymbol{y}^{(1)}_{(\nu-1)} - \lambda \boldsymbol{y}^{(2)}_{(\nu-1)} . \tag{30}$$

As long as $\lambda \leq 1$, $\boldsymbol{y}_{(\infty)}$ will converge to the least-squares solution.

According to Fig. 1, $\boldsymbol{y}^{(1)}_{(\nu-1)}$ is orthogonal to $\mathcal{G}_1$, thus can be expressed as a linear combination of columns of $\boldsymbol{H}_1^T$. Without loss of generality, suppose

$$\boldsymbol{y}^{(1)}_{(\nu-1)} = \boldsymbol{H}_1^T \boldsymbol{\alpha} .$$

Since $\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}^{(1)}_{(\nu-1)}$ lies in $\mathcal{G}_1$, we have

$$\boldsymbol{H}_1 \cdot (\boldsymbol{y}_{(\nu-1)} - \boldsymbol{y}^{(1)}_{(\nu-1)}) = \boldsymbol{H}_1 \cdot (\boldsymbol{y}_{(\nu-1)} - \boldsymbol{H}_1^T \boldsymbol{\alpha}) = \boldsymbol{0} .$$

The solution for $\boldsymbol{\alpha}$ is

$$\boldsymbol{\alpha} = (\boldsymbol{H}_1 \boldsymbol{H}_1^T)^{-1} \boldsymbol{H}_1 \boldsymbol{y}_{(\nu-1)} .$$

The prerequisite for this solution is that $\boldsymbol{H}_1$ is a row full-rank matrix, otherwise the inverse of $(\boldsymbol{H}_1 \boldsymbol{H}_1^T)$ would not exist. However, parity-checks can be considered to be linearly independent. Now, $\boldsymbol{y}^{(1)}_{(\nu-1)}$ can be expressed as

$$\boldsymbol{y}^{(1)}_{(\nu-1)} = \boldsymbol{H}_1^T (\boldsymbol{H}_1 \boldsymbol{H}_1^T)^{-1} \boldsymbol{H}_1 \boldsymbol{y}_{(\nu-1)} . \tag{31}$$

Similarly,

$$\boldsymbol{y}^{(2)}_{(\nu-1)} = \boldsymbol{H}_2^T (\boldsymbol{H}_2 \boldsymbol{H}_2^T)^{-1} \boldsymbol{H}_2 \boldsymbol{y}_{(\nu-1)} . \tag{32}$$

where $\boldsymbol{H}_2$ must also be a row full-rank matrix.

Equations (30), (31), and (32) represent the iterative least-squares decoding of an arbitrary linear analog code based on splitting the parity-check matrix in two.

5 Conclusions and outlook

Starting from a geometric illustration of the iterative decoding of analog product codes, we determined a Turbo-like iterative decoding procedure for arbitrary linear analog block codes by splitting its parity-check matrix in two and projecting received codewords onto the Null spaces of these two matrices in an iterative fashion. This provides us with an easy and intuitive understanding what iterative decoding of block codes over real or complex numbers is about, just iterative projections finally reaching the least-squares solution. We expect that further steps will guide us back to codes over discrete number fields in the hope of a more intuitive understanding of iterative decoding schemes there, as well.

Acknowledgment

This work is part of the FP6 / IST project M-Pipe and is co-funded by the European Commission.

References

[1] J.K. Wolf, “Analog Codes,” proc. *IEEE Int. Conf. on Comm. (ICC ’83)*, Boston, MA, USA, Vol. 1, pp. 310-12, June 19-22, 1983.

[2] J.K. Wolf, “Redundancy, the Discrete Fourier Transform, and Impulse Noise Cancellation,” *IEEE Tran. on Comm.*, Vol. COM-31, No. 3, pp. 458-461, March 1983.

[3] W. Henkel, “Multiple Error Correction with Analog Codes,” *AAECC-6*, Rome, and *Springer Lecture Notes in Computer Science* 357, pp. 239-249, 1989, (http://trsys.faculty.iu-bremen.de).

[4] W. Henkel, *Zur Decodierung algebraischer Blockcodes über komplexen Alphabeten*, VDI-Verlag, Düsseldorf, 1989. (in German), (http://trsys.faculty.iu-bremen.de).

[5] Z. Wang and G. B. Giannakis, “Complex-Field Coding for OFDM over Fading Wireless Channels,” *IEEE Trans. on Inform. Theory*, Vol. 49, No. 3, March 2003.

[6] F. Abdelkefi, P. Duhamel, and F. Albgerge, "Impulsive Noise Cancellation in Multicarrier Transmission," *IEEE Trans. on Comm.*, Vol. 53, No. 1, pp. 94-106, Jan. 2005.
[7] W. Henkel and F. Hu, "OFDM and Analog RS/BCH Codes," proc. *OFDM-Workshop 2005*, Hamburg, Aug. 31 - Sept. 1, 2005.
[8] M. Mura, W. Henkel, and L. Cottatellucci, "Iterative Least-Squares Decoding of Analog Product Codes," *IEEE Intern. Symp. on Inform. Theory (ISIT 2003)*, Yokohama, June 29 - July 4, 2003.
[9] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Proc. Mag.*, pp. 28-41, Jan. 2004.
[10] Y. Weiss and W. T. Freeman, "On the optimality of the max-product belief propagation algorithm in arbitrary graphs," *IEEE Trans. Inform. Theory*, Vol. 47, pp. 736-744, 2001.
[11] P. Rusmevichientong and B. Van Roy, "An analysis of belief propagation on the turbo decoding graph with Gaussian densities," *IEEE Trans. Inform. Theory*, Vol. 47, pp. 745-765, Feb. 2001.
[12] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. on Inform. Theory*, Vol. 42, No.2, March 1996.