Sub-Gaussian Model Based LDPC Decoder for SαS Noise Channels

Iulian Topor
Acoustic Research Laboratory,
Tropical Marine Science Institute,
National University of Singapore,
Singapore 119227.
iulian@arl.nus.edu.sg

Mandar Chitre
Acoustic Research Laboratory,
Tropical Marine Science Institute,
National University of Singapore,
Singapore 119227.
mandar@arl.nus.edu.sg

Mehul Motani
Department of Electrical
and Computer Engineering,
National University of Singapore,
Singapore 117576.
motani@nus.edu.sg

***Abstract*—OFDM communication through warm shallow water acoustic channel (WSWA) is affected by snapping shrimp noise. Due to its impulsive broadband nature, the snapping shrimp noise is able to affect a large number of OFDM sub-carriers at once. The effect on the sub-carriers can be modeled as a symmetric α-stable ($S\alpha S$) noise vector with dependent components. Although the performance of the conventional LDPC soft decoder is poorer in non-Gaussian $S\alpha S$ noise than in Gaussian noise, the performance can be improved by employing the dependency between the noise components while decoding. The multivariate probability densities and marginals are required in the computation of symbol a posteriori probabilities. For binary codes, the complexity of the algorithm using marginals is $O(2^dN)$, where d is the number of dependent components and N is the length of the code. Practical implementation of a decoder employing dependency through marginals is infeasible for high number of dependent components. In order to address this problem we develop a lower complexity algorithm using the sub-Gaussian model of $S\alpha S$ vectors.**

I. Introduction

Coded OFDM was shown to have good performance in warm shallow water acoustic (WSWA) channels [1], [2]. The WSWA channel is one of the more challenging underwater acoustic communication channels. Besides an extensive time-varying multipath, it is characterized by high levels of non-Gaussian ambient noise due to snapping shrimp. Earlier research showed that the snapping shrimp noise can be modeled using symmetric α-stable ($S\alpha S$) distributions. The snapping shrimp noise is impulsive and its energy is spread over a wide frequency band. Therefore, snapping shrimp noise is able to affect a high number of OFDM sub-carriers at once [3]. At the decoder, this is equivalent to a noise vector distorting all the symbols transmitted on the OFDM sub-carriers. The components of this vector are dependent because they are affected by same impulse. The structure of the dependency depends of the implementation details of the OFDM receiver, e.g., the ratio between the carrier frequency and the sampling frequency used in the digital system [4], [5]. This paper focuses on the particular case in which the noise vector is an isotropic $S\alpha S$ vector.

The conventional LDPC soft decoder is known to perform well in Gaussian channels with independent noise components. Its performance drops in channels with non-Gaussian stable noise with independent components, due to the heavy tails of the stable distribution. When the components of the non-Gaussian stable noise vector are dependent, the performance of the decoder improves if the dependency is employed in the decoding process. The improvement is the result of the extra information available in the dependency. In the case of conventional LDPC soft decoder, the dependency is employed into the decoding process by computing the symbol a posterior probabilities from the underlying multivariate distribution through computation of the marginals. For a binary code, the complexity of computation of marginals is $O(2^dN)$, where d is the number of dependent components and N is the length of the code. The computational requirements increase exponentially with the number of dependent components. It is therefore infeasible to employ the component dependency through marginals when the number of dependent components is large. This situation arises in the case of OFDM transmissions with hundreds of carriers.

In order to use LDPC coded OFDM for communications through the WSWA channel, we designed an LDPC soft decoder that is able to employ the dependency without computing marginals. The decoder uses the sub-Gaussian representation of the $S\alpha S$ noise vector to factor the noise vector into a scalar random variable and an a noise vector with independent components. We present the details and the performance of this decoder.

II. Mathematical Preliminaries

This section briefly introduces stable distributions and α-stable random vectors. Mathematical details and proofs of the properties and definitions in this section are available in [7].

A stable distribution is characterized by four parameters:

- characteristic exponent, $\alpha \in (0, 2]$
- scale parameter, $\sigma \geq 0$
- skewness parameter, $\beta \in [-1, 1]$
- shift parameter, $\mu \in \mathbb{R}$

and denoted by $S_\alpha(\sigma, \beta, \mu)$. A symmetric stable distribution is obtained when $\beta = \mu = 0$ and is characterized only by characteristic exponent α and scale parameter σ. A random variable X is called *standard* $S\alpha S$ if $\sigma = 1$. A standard $S\alpha S$

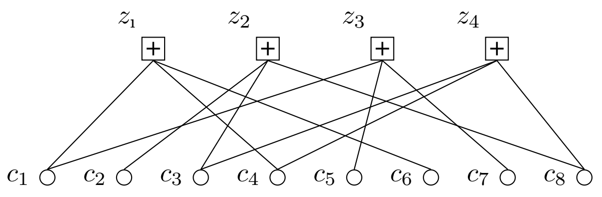

Fig. 1. Tanner graph of the code defined by (2).

random variable X with $\alpha = 2$ is normally distributed with variance $\sigma_G^2 = 2\sigma^2$.

Property II.1. *Let $A \sim S_{\alpha/2}((cos\frac{\pi\alpha}{4})^{2/\alpha}, 1, 0)$, be an $\alpha/2$-stable random variable totally skewed to the right with Laplace transform*

$$E\ e^{-\xi A} = e^{-\xi^{\alpha/2}},\ \xi > 0,$$

Let $\mathbf{G} = (G_1, G_2, ..., G_d)$ be a zero mean Gaussian vector in $\mathbb{R}^d$ independent of A. Then the random vector

$$\mathbf{X} = A^{1/2}\mathbf{G} \tag{1}$$

has an $S\alpha S$ distribution in $\mathbb{R}^d$.

Definition II.2. *Any vector $\mathbf{X}$ distributed as in (1) is called* sub-Gaussian $S\alpha S$ random vector in $\mathbb{R}^d$ *with underlying vector* $\mathbf{G}$.

The components of a sub-Gaussian random vector are dependent. Every isotropic $S\alpha S$ random vector with $\alpha < 2$ is sub-Gaussian, with the components of the underlaying vector $\mathbf{G}$ being i.i.d. zero-mean Gaussian random variables.

III. Conventional LDPC Soft Decoder

A. Overview

This section presents a short overview of the conventional belief-propagation based LDPC decoder. More details can be found in [6, chap. 15].

LDPC codes are block codes often defined by their parity-check matrices. An LDPC code of length N and dimension K is defined by a parity-check matrix $\mathbf{H}$ of size $M \times N$, where $M = N - K$. The Tanner graph is a graphical representation of the code used by the decoders employing algorithms on graphs. An example of a parity-check matrix of a very short LDPC code is shown in (2) and its Tanner graph is shown in Fig. 1.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The conventional LDPC soft decoder is an iterative decoder that employs the sum-product algorithm [8] on the Tanner graph. Before the decoding process starts, the decoder needs to compute the symbol a posteriori probabilities. After the initialization stage, the sum-product algorithm estimates the symbol *pseudo-posterior* probabilities, by combining the information from the symbol a posteriori probabilities with the information about the code structure captured into the Tanner graph.

The probabilities used by the sum-product algorithm, for LDPC decoding, are:

- $p_n(x)$ - the symbol a posteriori probability, initialized at node c_n.

$$p_n(x) = P(c_n = x|\mathbf{r}). \tag{3}$$

- $r_{mn}(x)$ - computed at node z_m and passed to symbol node c_n.

$$r_{mn}(x) = \sum_{\{\{x_{n'}, n' \in \mathcal{N}_{m,n}\}: x = \sum_l x_l\}} \prod_{l \in \mathcal{N}_{m,n}} q_{ml}(c_l), \tag{4}$$

  where, $\mathcal{N}_{m,n}$ is the set of indexes of the symbols that participate in check z_m, except symbol c_n.

- $\gamma_n(x)$ - the *extrinsic probability* - computed at node c_n.

$$\gamma_n(x) = \prod_{m' \in \mathcal{M}_n} r_{m'n}(x), \tag{5}$$

  where, $\mathcal{M}_n$ is the set of indexes of the checks in which symbol c_n participates.

- $q_n(x)$ - is the pseudo-posterior probability of symbol c_n.

$$q_n(x) = \alpha_{\text{norm}} \cdot p_n(x) \cdot \gamma_n(x), \tag{6}$$

  where, α_{norm} is a normalization factor.

- $q_{mn}(x)$ - computed at node c_n and is passed to node z_m.

$$q_{mn}(x) = \alpha_{\text{norm}} \cdot p_n(x) \cdot \frac{\gamma_n(x)}{r_{mn}(x)} \tag{7}$$

  where, α_{norm} is a normalization factor.

B. Symbol a posteriori probabilities computation

For a transmission system that generates codewords $\mathbf{c}$ with equal probability, the a posteriori probability of a codeword, for a received vector $\mathbf{r}$, is the normalized value of the probability density of the noise vector $\mathbf{w}(\mathbf{c})$, corresponding to the codeword:

$$P(\mathbf{c}|\mathbf{r}) = \frac{p(\mathbf{w}(\mathbf{c}))}{\sum_{\mathbf{c}} p(\mathbf{w}(\mathbf{c}))}. \tag{8}$$

The a posteriori probability of a symbol c_n is obtained as marginal of codeword a posteriori probabilities:

$$P(c_n = x|\mathbf{r}) = \sum_{\sim\{c_n = x\}} P(\mathbf{c}|\mathbf{r}), \tag{9}$$

where, '$\sim \{.\}$' is the *summary* notation from [8].

For a binary code of length N and noise with d dependent components, the complexity of the algorithm that computes the symbol a posteriori probabilities through marginals is $O(2^d N)$. Therefore, exact computation of symbol a posteriori probabilities becomes unfeasible for high values of d.

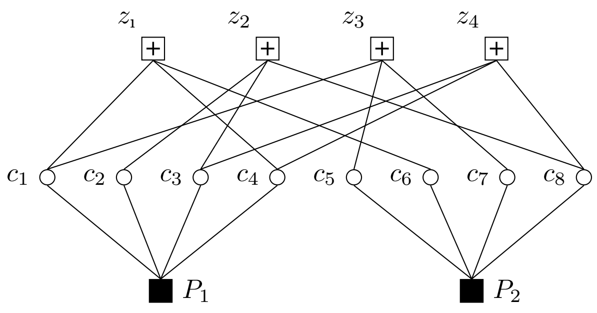

Fig. 2. Factor graph of LDPC code, for transmission through noise with 4 dependent components.

C. Algorithm summary

The steps of conventional LDPC soft decoder are presented in the listing **'Algorithm A'**, where step (1.i.) is the step with complexity $O(2^d N)$.

```
Algorithm A
Inputs: $\mathbf{H}$, $\mathbf{r}$, $L$, PDF of the noise distribution.
1. Initialization:
     i. Compute $p_n(x)$, for each $c_n$.
    ii. Set $q_{mn}(x) = p_n(x)$, for all $(m, n)$ with $\mathbf{H}(m, n) = 1$.
2. Horizontal step:
     i. Compute $r_{mn}(x)$, for all $(m, n)$ with $\mathbf{H}(m, n) = 1$.
3. Vertical step:
     i. Compute $\gamma_n(x)$ and $q_n(x)$, for each $c_n$.
    ii. Codeword estimation:
        • Set $\hat{c}_n = 1$ if $q_n(1) > 0.5$, else set $\hat{c}_n = 0$.
        • If $\mathbf{H}\hat{\mathbf{c}}$ = 0, then STOP and RETURN $\hat{\mathbf{c}}$;
        • Otherwise,
          – If the number of iterations $< L$, CONTINUE;
          – Otherwise, STOP and RETURN 'FAILURE'.
   iii. Compute $q_{mn}(x)$ for all $(m, n)$ with $\mathbf{H}(m, n) = 1$.
    iv. Loop to Horizontal Step.
```

IV. Sub-Gaussian Decoder

A. Overview

Our decoder is an extension of the conventional LDPC soft decoder and is designed to be used in channels with isotropic $S\alpha S$ noise. The decoder makes use of the sub-Gaussian model of $S\alpha S$ noise vectors to avoid computation of marginals for symbol a posteriori probabilities. The main characteristic of the decoder is that the decoding process starts with rough estimates of symbol a posteriori probabilities, and their values are updated after each sum-product algorithm iteration. This is achieved by exchanging information between the symbol nodes affected by the same noise vector. The information exchange happens through a posteriori probability nodes attached to the symbol nodes of the Tanner graph. Each a posteriori probability node represents to a noise vector. Therefore, all symbol nodes affected by the same noise vector are connected to same a posteriori probability node. Fig. 2 shows the Tanner graph from Fig. 1 augmented with the a posteriori probability nodes P_j, when the number of dependent components is 4.

The update of symbol a posteriori probabilities involves only message passing between the symbol nodes and a posteriori probability nodes. Each symbol node c_n passes to its a posteriori probability node neighbor the probabilities $\lambda_n(A^i)$. At the a posteriori probability nodes, probabilities $\lambda_n(A^i)$ are combined together to compute the probabilities $\pi_n(A^i)$ that are distributed to the symbol nodes. The equation used to compute $\pi_n(A^i)$ ensures that the information originated at a symbol node does not return back to the souce, but only reaches all other symbol nodes connected to the a posteriori node. Finally, at each node c_n, probabilities $\pi_n(A^i)$ are used to update the symbol a posteriori probabilities $p_n(x)$, using

$$p_n(x) = \sum_{A^i} P(c_n = x | r_n, A^i) \cdot \pi_n(A^i). \tag{10}$$

Probabilities $\lambda_n(A^i)$ are computed at node c_n from the extrinsic probabilities $\gamma_n(x)$, using

$$\lambda_n(A^i) = \sum_{x} P(c_n = x | r_n, A^i) \cdot \gamma_n(x). \tag{11}$$

Probabilities $\pi_n(A^i)$ are computed at a node P_j and are sent to node c_n. They are computed using

$$\pi_n(A^i) = \pi(A^i) \cdot \prod_{k \neq n} \lambda_k(A^i), \tag{12}$$

where, probabilities $\pi(A^i)$ are the a priori probabilities of A^i, defined by the discrete distribution A_d.

B. Algorithm summary

The steps of sub-Gaussian decoder are presented in the listing **'Algorithm B'**. The decoder features same main steps as the conventional LDPC soft decoder, with new sub-steps added to the 'Vertical' and 'Initialization' steps. The sub-steps added to 'Initialization' step are (1.i.), (1.ii.), (1.iii.) and (1.iv.), while 'Vertical' step adds sub-step (3.iii).

Step (1.v.) in **'Algorithm B'** is step (1.i.) in **'Algorithm A'**, but this time it is performed using (10), which has a much lower complexity.

```
Algorithm B
Inputs: $\mathbf{H}$, $\mathbf{r}$, $L$, $\alpha$, $\sigma$, $A_d$.
1. Initialization:
     i. $\sigma_G = \sqrt{2}\sigma$.
    ii. $\pi_n(A^i) = \pi(A^i)$.
   iii. $w_n^G(x, A^i) = \frac{1}{(A^i)^{1/\alpha}}[r_n - t_n(x)]$.
    iv. $p(c_n = x | r_n, A^i) = \frac{1}{\sqrt{2\pi}\sigma_G} \cdot e^{-\frac{1}{2\sigma_G^2} \cdot [w_n^G(x, A^i)]^2}$.
     v. $P(c_n = x | r_n, A^i) = \alpha_{\text{norm}} \cdot p(c_n = x | r_n, A^i)$.
    vi. Compute $p_n(x)$, for each $c_n$.
   vii. Set $q_{mn}(x) = p_n(x)$, for all $(m, n)$ with
        $\mathbf{H}(m, n) = 1$.
```

2. **Horizontal step:**
   - Compute $r_{mn}(x)$, for all (m,n) with $\mathbf{H}(m,n) = 1$.
3. **Vertical step:**
   i. Compute $\gamma_n(x)$ and $q_n(x)$, for each c_n.
   ii. Codeword estimation:
      - Set $\hat{c}_n = 1$ if $q_n(1) > 0.5$, else set $\hat{c}_n = 0$.
      - If $\mathbf{H\hat{c}} = 0$, then STOP and RETURN $\mathbf{\hat{c}}$;
      - Otherwise,
        – If the number of iterations $< L$, CONTINUE;
        – Otherwise, STOP and RETURN 'FAILURE'.
   iii. Symbol a posteriori probabilities update:
      - Compute $\lambda_n(A^i)$, for all nodes c_n.
      - Compute $\pi_n(A^i)$, for each node P_j and each c_n connected to P_j.
      - Compute $p_n(x)$, for each c_n.
   iv. Compute $q_{mn}(x)$ for all (m,n) with $\mathbf{H}(m,n) = 1$.
   v. Loop to **Horizontal Step**.

C. *Symbol a posteriori probabilities computation*

In order to avoid using marginals, our design employs an indirect approach for symbol a posteriori probabilities computation. The approach is similar to a decoder developed for channels with Gaussian noise and channel fading [9]. This decoder combines LDPC soft decoding with the estimation of fading. Our decoder does not estimate fading levels; instead we estimate a scalar random variable that is factored out from the random vector using the sub-Gaussian model of $S\alpha S$ vectors.

The sub-Gaussian model makes possible to express the component of an isotropic $S\alpha S$ noise vector $\mathbf{w}$ as

$$w_n^{\mathrm{S\alpha S}} = A^{1/\alpha} \cdot w_n^G, \tag{13}$$

where, A is a totally skewed to right α-stable random variable and w_n is a zero mean normal random variable. For transmissions through additive white $S\alpha S$ noise (AWSαSN) channel, the amplitude of the noise component corresponding to symbol c_n of codeword $\mathbf{c}$ is

$$A^{1/\alpha} \cdot w_n^G = r_n - t_n, \tag{14}$$

where, r_n is the sample received from the channel and t_n is the signal amplitude that is considered to have been transmitted for symbol c_n. Conditional on the knowledge of A, the amplitudes of the components of the underlying Gaussian vector can be computed as

$$w_n^G = \frac{1}{A^{1/\alpha}}(r_n - t_n). \tag{15}$$

If the isotropic $S\alpha S$ noise is characterized by $\alpha < 2$, then the components w_n are independent. Therefore, the symbol a posteriori probabilities computation does not involve multivariate PDF and marginals.

Using $r_n^{'} = \frac{r_n}{A^{1/\alpha}}$ and $s^{'} = \frac{1}{A^{1/\alpha}}$ in (15) results in

$$w_n^G = r_n^{'} - s^{'} \cdot t_n, \tag{16}$$

which is similar with the equation of a noise component for a transmission through a channel with Gaussian noise and fading:

$$w_n^G = r_n - s \cdot t_n, \tag{17}$$

where s is the fading factor. Therefore, a fading level estimation algorithm can be employed to estimate the value of A.

The algorithm we implemented does not identify the exact value of A; instead it estimates the a posteriori probability $\pi_n(A^i)$ of each possible value A^i of A. The estimated probabilities are further used to compute the symbol a posteriori probabilities, using (10).

For random variable A with continuous distribution, $\pi_n(A^i)$ are probability densities, and the symbol a posteriori probability of symbol c_n is computed as

$$p_n(x) = \int_0^\infty P(c_n = x | r_n, A^i) \cdot \pi_n(A^i) \cdot dA^i, \tag{18}$$

where, x is a value from code's alphabet and

$$P(c_n = x | r_n, A^i) = \alpha_{\text{norm}} \cdot p(c_n = x | r_n, A^i). \tag{19}$$

In (19), α_{norm} is a normalization factor and

$$p(c_n = x | r_n, A^i) = \frac{1}{\sqrt{2\pi}\sigma_G} \cdot e^{-\frac{1}{2\sigma_G^2}\cdot[w_n^G(x,A^i)]^2}, \tag{20}$$

with $w_n^G(x, A^i)$ being defined by (15).

Although A has a continuous stable distribution, we discretize it to have a finite number of discrete values A^i. The symbol a posteriori probability equation in this case is

$$p_n(x) = \alpha_{norm} \cdot \sum_{A^i} P(c_n = x | r_n, A^i) \cdot \pi_n(A^i). \tag{21}$$

Our algorithm uses a priori probabilities $\pi(A^i)$ to estimate the a posteriori probabilities $\pi_n(A^i)$, for each symbol c_n. Probabilities $\pi(A^i)$ are the probabilities of a discrete distribution that approximates the continuous distribution $S_{\alpha/2}((cos\frac{\pi\alpha}{4})^{2/\alpha}, 1, 0)$. That is, the cumulative probability function (CDF) of the discrete distribution approximates the CDF of the continuous distribution. Therefore, each probability $\pi(A^i)$ is the accumulated probability of a segment of the PDF of the continuous distribution. We generated the discrete distribution by splitting the support of PDF into a finite number

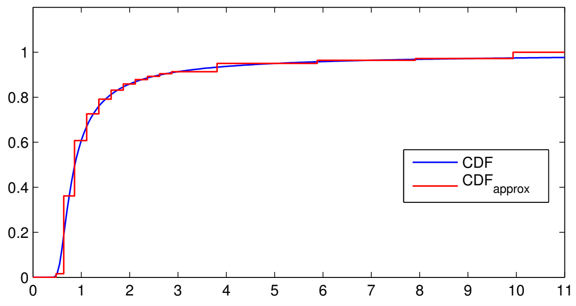

Fig. 3. Approximation of continuous distribution of A.

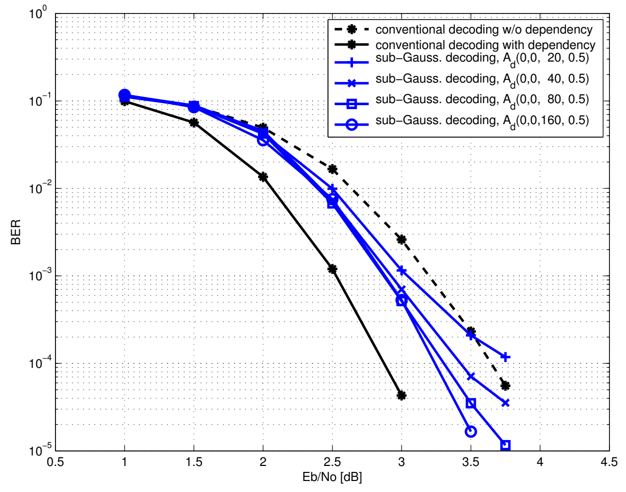

Fig. 4. Decoder performance for 10-dimensional isotropic $S\alpha S$ noise.

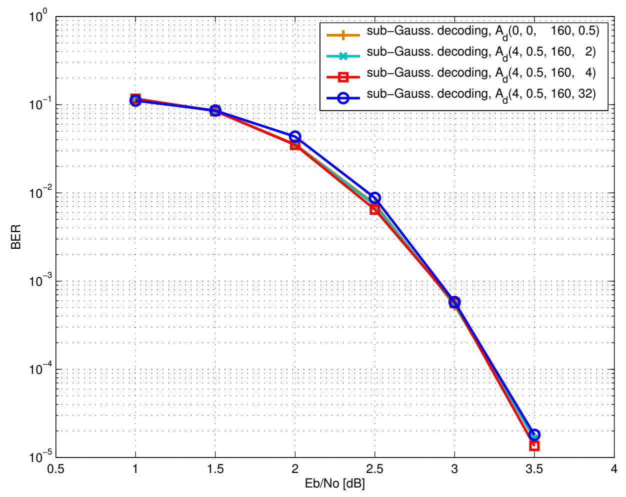

Fig. 5. Performance of sub-Gaussian decoder with $A_{max} = 160$, for 10-dimensional isotropic $S\alpha S$ noise.

of segments and define the distance from the origin to the weight center of segment i as A^i, with $\pi(A^i)$ taking the value of the cumulated probability of segment i. Because the PDF is defined on the interval $[0, \infty)$, one of the segments we define is (A_{max}, ∞), where A_{max} is a chosen value. The rest of the segments can be obtained by splitting the interval $[0, A_{max}]$ in equal segments. As the PDF of $S_{\alpha/2}((cos\frac{\pi\alpha}{4})^{2/\alpha}, 1, 0)$ is characterized by a narrow peak next to origin, we use two different segment lengths (δ_1 and δ_2) for the interval next to the origin and for the interval away from the origin. For this, we split the interval $[0, Ar]$ in segments of length δ_1 and the interval $(Ar, A_{max}]$ in segments of length δ_2. We denote the distribution obtained in this way by $A_d(A_r, \delta_1, A_{max}, \delta_2)$, and the particular case $\delta_1 = \delta_2$ by $A_d(0, 0, A_{max}, \delta_2)$. At the end, the accumulated probability of segment (A_{max}, ∞) is added to the probability of the highest value A^i, in order to satisfy $\sum_i \pi(A^i) = 1$. The CDF of $S_{\alpha/2}((cos\frac{\pi\alpha}{4})^{2/\alpha}, 1, 0)$, for $\alpha = 1.7$, is shown in Fig. 3, together with the CDF of the approximating distribution $A_d(3, 0.25, 10, 2)$.

D. *Algorithm complexity*

The number of operations required to initialize the symbol a posteriori probabilities and to update them over the entire period of a codeword decoding session is the sum of the number of operations required to initialize them and the number of operations required to update them after each sum-product iteration. The complexity of the algorithm is $O(N_{tier}N_AN)$, where N_{tier} is the number of iterations, N_A is the number of A^i values of distribution A_d and N is the length of the code. Note that the complexity of this algorithm is independent of the number of dependent noise components.

V. Simulations results

A. *Simulations setup*

The goals of the simulations we performed was to compare the performance of the proposed decoder with the performance of the conventional LDPC decoder, and to find the relation between its design parameters and performance. We simulated LDPC encoded BPSK transmissions through isotropic $AWS\alpha SN$ with $\alpha = 1.7$, which is the characteristic exponent of $S\alpha S$ distribution that models the snapping shrimp noise. The LDPC code was defined by a 1000×2000 randomly generated parity-check matrix and the maximum number of decoding iterations was $L = 30$. The signal-to-noise ratio (SNR) was computed as in [2] with

$$N_0 = 4\sigma^2. \tag{22}$$

B. *Performance vs. A_{max}*

In order to understand how A_{max} value influences the performance of the decoder, we simulated transmissions through 10-dimensional isotropic $S\alpha S$ noise, decoded with sub-Gaussian decoder using distribution $A_d(0, 0, A_{max}, 0.5)$ with A_{max} = 20, 40, 80 and 160. The results of the simulations are presented in Fig. 4. The plots show that, for constant δ_2, the performance increases when A_{max} increases.

Fig. 4 shows simulation results with conventional LDPC soft decoder, when the dependency was employed through marginals as well as when it was not employed. When the dependency was not employed, the symbol a posteriori probabilities were computed individually using univariate PDF, as if the noise components were independent. The drop in performance in this case, as compared to the case when the dependency was employed through marginals, is due to errors in symbol a posterior probabilities computations resulting from the PDF mismatch. The results show that the performance of the sub-Gaussian decoder is sub-optimal. Although the performance is poorer than that of the conventional decoder employing the dependency through marginals, it is significantly better than the conventional decoder that ignores the dependency. The algorithm therefore provides a trade-off between computational complexity and optimality.

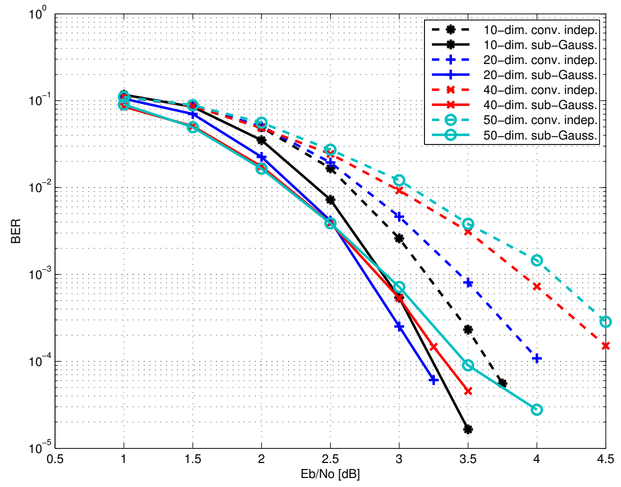

Fig. 6. Sub-Gaussian decoder vs. conventional decoder without dependency for $A_d(4, 0.5, 160, 2)$

C. *Performance vs. δ_2*

We found that the complexity of the proposed algorithm depends of N_A, the number of elements of A_d. This implies that a faster decoder requires A_d with a small N_A. Nevertheless, the expectation was that the structure of A_d would influence the performance of the decoder. Series of simulations in which only the segment length δ_2 varied shows that in some cases the number of elements of A_d can be reduced considerably without a significant change in decoder's performance. Such a case is presented in Fig. 5. The figure shows that for $A_{max} = 160$ the decoder performance is not significantly affected as δ_2 varies from 0.5 to 32, although the number of elements of A_d decreases from 320 to 12. The number of A^i elements in each case is:

- $A_d(0, 0, 160, 0.5)$: $N_A = 320$
- $A_d(4, 0.5, 160, 2)$: $N_A = 86$
- $A_d(4, 0.5, 160, 4)$: $N_A = 47$
- $A_d(4, 0.5, 160, 32)$: $N_A = 12$

D. *Performance vs. number of dependent components*

Fig. 6 shows the performance of the proposed decoder for 10, 20, 40 and 50 of dependent noise components, together with the performance of the conventional decoder that does not employ the dependency into the decoding process. The performance of conventional decoder which employs the dependency through marginals computation could not be assessed for these cases, because the high computation power requirements made the simulations unfeasible. The results show that, with one exception, the performance of sub-Gaussian decoder decreases when the number of dependent components increases, but it is always higher then the performance of the conventional decoder. They show also that the performance gain of sub-Gaussian decoder, over the classical decoder, increases when the number of dependent components increases.

VI. Conclusions

A sub-Gaussian decoder was developed to exploit the dependency between noise experienced by OFDM sub-carriers in presence of impulsive snapping shrimp noise. We showed that the sub-Gaussian decoder is able to employ the dependency successfully. The number of computations required by the sub-Gaussian decoder is independent of the number of dependent noise components. Therefore, although sub-optimal, the sub-Gaussian decoder can replace a conventional LDPC soft decoder when the number of dependent noise components is too large to compute the marginals.

References

[1] M. Chitre, S.-H. Ong, and J. Potter, “Performance of coded OFDM in very shallow water channels and snapping shrimp noise”, OCEANS, 2005. Proceedings of MTS/IEEE, pp.996-1001, Vol. 2, 17-23 Sept. 2005.

[2] M. Chitre, J. Potter, and S.-H. Ong, “Viterbi Decoding of Convolutional Codes in Symmetric α-Stable Noise”, Communications, IEEE Transactions on, vol.55, no.12, pp.2230-2233, Dec. 2007.

[3] H. Rohling, “OFDM, Concepts for Future Communication Systems”, Springer-Verlag Berlin Heidelberg, 2011.

[4] A. Mahmood, M. Chitre, and M. Armand, “PSK communication with passband additive symmetric a-stable noise”, Communications, IEEE Transactions on, 2012 (under review).

[5] M. Chitre, A. Mahmood, and M. Armand, “Coherent communications in snapping-shrimp dominated ambient noise environments”, Acoustics 2012 Hong Kong Conference and Exhibition, May 2012.

[6] T. K. Moon, “Error Correction Coding - Mathematical Methods and Algorithms”, John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.

[7] G. Samorodnitsky, M. Taqqu, “Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance”, Chapman and Hall/CRC, 1994.

[8] F. Kschischang, B. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm”, Information Theory, IEEE Transactions on, 47(2):498519, Feb 2001.

[9] A. P. Worthen and W. E. Stark, “Low-Density Parity Check Codes for Fading Channels with Memory”, Proceedings of the 36th Annual Allerton Conference on Communication, Control, and Computing, 1998.