A Noise-Adaptive Algorithm for First-Order Reed-Muller Decoding

Jon Feldman Ibrahim Abou-Faycal Matteo Frigo

***Abstract*— We consider the problem of decoding First-Order Reed-Muller codes efficiently. We give an algorithm that implicitly adapts to the noise conditions, runs significantly faster than known maximum-likelihood algorithms, and yields an error rate that is very close to optimal. When applied to CCK demodulation (used in the 802.11b standard for Wireless Local Area Networks), the algorithm runs up to 4 times faster than a decoder based on the Fast Hadamard Transform, with a loss of at most 0.2 dB in error rate. We show analytically that the error rate of our adaptive algorithm is $2^{-\Omega(n)}$, where n is the length of a codeword.**

I. Introduction

First-Order Reed-Muller (FORM) codes are widely used in communications applications ranging from the 1969 Mariner deep space probe [8] to the IEEE 802.11b standard for Wireless Local Area Networks (WLANs) [4]. While their good distance properties and simple structure make them attractive, soft-decision Maximum-Likelihood (ML) decoding of FORM codes requires computing the correlation between the received vector and all possible codewords. These operations are computationally expensive, especially at high data rate as is the case for Complementary Code Keying (CCK) demodulation in 802.11b [4]. The most efficient known algorithm for computing these correlations is based the Fast Hadamard Transform (FHT) [9]. We will refer to a decoder that uses the FHT to perform a correlation of all possible codewords as an *FHT decoder*. Even the FHT decoder may not be fast enough in certain systems such as software radios. In the particular case of CCK, the FHT decoder uses $O(n^2)$ operations to decode one codeword of block length n. Even for the simplest FORM code, the Hadamard code, the FHT decoder performs a superlinear $\Theta(n \log n)$ number of operations.

Alternatively, one could use *majority logic* or "threshold" decoding algorithms such as the ones described in [5], [2]. Intuitively speaking, majority logic decoders tally "votes" for the value of each information symbol based on simple calculations on the received codeword, and output the value with the most votes. Decoders of this kind are simple and fast, but they are suboptimal when used for soft-decoding FORM codes. When applied to CCK demodulation, we observed that the symbol error rate of majority logic decoding can be up to 2.4 dB worse than optimal.

Jon Feldman, NE43-309, Laboratory for Computer Science, M.I.T., Cambridge, MA, 02139. Email: `jonfeld@theory.lcs.mit.edu`. This work was done while the author was visiting Vanu Inc.

Ibrahim Abou-Faycal, Vanu Inc., 1 Porter Square, Suite 18, Cambridge, MA 02140. Email: `ibrahim@vanu.com`.

Matteo Frigo, Vanu Inc. Email: `athena@vanu.com`.

Nonetheless, the performance degradation of the majority-logic decoder is negligible when the noise is low. Ideally, one would like to use the fast majority-logic algorithm when the noise is low, and the slower maximum-likelihood algorithm when the noise is high. Unfortunately, the error conditions are not known in advance, and they tend to change over time anyway. Consequently, we design a decoding strategy that *implicitly* adapts its algorithm to the noise conditions without explicitly knowing what they are.

In this paper, we build such an adaptive strategy for decoding FORM codes, which we refer to as the *hybrid* algorithm. We build upon the work of Paterson and Jones [3], and Van Nee [6] who explored variations of majority logic algorithms for decoding this class of codes with applications to OFDM in mind. Both papers use a soft-decision version of majority logic, where instead of taking the majority of a group of hard-valued votes, the algorithm takes an average of a group of soft-valued votes, and makes a hard decision on that average. We extend their techniques by using this "soft average" as not only the value on which to make a hard decision, but also as a confidence measure in the decision. When this confidence measure dips below a certain level, we switch to the slower maximum-likelihood decoder.

Our experiments show that the decoding speed of the hybrid algorithm is significantly improved over an optimized implementation of an FHT decoder, at the expense of a negligible degradation in error-correcting performance. When applied to CCK demodulation, the algorithm runs up to 4 times faster than FHT decoder, with a loss of at most 0.2 dB in error rate. (See Figures 1 and 2.)

We obtain an analytical expression of the additive symbol error rate of our algorithm, assuming q-PSK modulation through an AWGN channel. We show that overall, the symbol error rate of our algorithm is at most an additive $2^{-\Omega(n)}$ worse than that of an optimal ML algorithm, where n is the length of a codeword, as long as the noise is above a certain threshold. (The notation $\Omega(n)$ denotes some function that grows *at least* linearly with n.) The threshold is quite reasonable; for example, if we are using 4-PSK, then as long as the SNR is at least 4 dB, we give an upper bound of $2^{-n/10+1}$ on the additive difference in error rate between our hybrid algorithm and an ML decoder.

A. Outline

We begin in Section II by showing how our hybrid algorithm applies to the problem of CCK demodulation, and give our experimental results. In Section III, we generalize the hybrid algorithm to decode any FORM code, and prove the $2^{-\Omega(n)}$ bound on its symbol error rate in Section IV.

II. CCK Demodulation for 802.11

The IEEE 802.11 standard for wireless local area networks has high data rates in order to operate at speeds comparable to Ethernet. Complementary Code Keying (CCK) was adopted by the IEEE as the modulation scheme to achieve this data rate [4]. For the purposes of demodulation, CCK is essentially isomorphic to a simple FORM code. In this section, we detail our decoding algorithm as it applies to CCK, and give experimental results that show a significant improvement in running time with only a negligible loss in error rate.

For the modulation step of CCK, an information sequence (c_0, c_1, c_2, c_3) is a block of four symbols, where $c_i \in \{0, 1, 2, 3\}$. These are modulated using QPSK to values $\phi_i = \omega^{c_i}$, where $\omega = e^{\pi j/2} = j = \sqrt{-1}$, and encoded into eight complex numbers $(y_0, \ldots, y_7)$ using the following encoding function:

$$
\begin{aligned}
y_0 &= \phi_0 & y_1 &= -\phi_0\phi_1 \\
y_2 &= \phi_0\phi_2 & y_3 &= \phi_0\phi_1\phi_2 \\
y_4 &= -\phi_0\phi_3 & y_5 &= \phi_0\phi_1\phi_3 \\
y_6 &= \phi_0\phi_2\phi_3 & y_7 &= \phi_0\phi_1\phi_2\phi_3
\end{aligned}
\tag{1}
$$

These eight symbols are then subject to an AWGN channel. We use $(r_0, \ldots, r_7)$ to denote the noisy symbols received at the other end of the channel. We have $r_i = y_i + N_i$, where N_i is a complex Gaussian random variable with mean 0 and variance $2\sigma^2$. Based on the received vector r, the decoder must output hard estimates $\hat{c}_i$ of the information symbols c_i, where $i \in \{0, 1, 2, 3\}$.

Two details in CCK make it slightly different than a FORM code. The negative signs in front of y_1 and y_4 are there to achieve better autocorrelation properties from the codewords, an important feature of CCK modulation [4]. Additionally, the information carried by ϕ_0 is differentially encoded; i.e., the actual information is the difference between the ϕ_0 symbol of two successive blocks. Neither of these two issues affects the underlying decoding problem directly.

A. Majority Logic Decoding

Consider the case where there is no noise in the channel, i.e., $N_i = 0$, and so $r_i = y_i$, for all i. Now the decoding problem is simple. For example, consider the expression $-r_1 r_0^*$. If there is no noise in the channel, then $-r_1 r_0^* = -y_1 y_0^* = \phi_1$. Similarly, $-r_4^* r_6 = -y_4 y_6 = \phi_2$, and $r_7 r_3^* = y_7 y_3^* = \phi_3$. Therefore, when there is no noise in channel, we can simply "read off" ϕ_1, ϕ_2 and ϕ_3 using simple operations between certain received symbols.

In reality, these computations will be corrupted by noise, and will not always yield the correct answer. For example, $-r_1 r_0^* = (-y_1 + N_1)(y_0 + N_0)^*$. However, in expectation, we still have $-r_1 r_0^* = \phi_1$, and if the noise is low, then $-r_1 r_0^*$ will be close to ϕ_1.

The principle behind majority logic decoding is to use simple computations on the received bits to produce "votes" for the value of each information symbol. In hard decision majority logic, the value that receives the most votes becomes the decoded information symbol. In soft decision majority logic, the votes are "soft" values, and they are averaged to form a "soft estimate" for each information symbol. Ideally, these votes should involve as many code bits as possible so that local noise cannot drastically affect our estimate.

In CCK, we use $(\hat{\phi}_1, \hat{\phi}_2, \hat{\phi}_3)$ to denote the soft estimates for ϕ_1, ϕ_2, ϕ_3 (we will cover ϕ_0 as a special case later), and compute each of them based on four votes as follows:

$$
\begin{aligned}
\hat{\phi}_1 &= (-r_1 r_0^* + r_3 r_2^* - r_4^* r_5 + r_7 r_6^*) \;/\; 4 \\
\hat{\phi}_2 &= (r_2 r_0^* - r_1^* r_3 - r_4^* r_6 + r_7 r_5^*) \;/\; 4 \\
\hat{\phi}_3 &= (-r_4 r_0^* - r_1^* r_5 + r_6 r_2^* + r_7 r_3^*) \;/\; 4
\end{aligned}
$$

Ideally, if $\hat{\phi}_i$ is a good estimate of ϕ_i, then $|\phi_i - \hat{\phi}_i|$ should be small. The majority logic decoders of Paterson and Jones [3], and Van Nee [6] commit to a hard decision $\hat{c}_i$ for each information symbol c_i, based on $\hat{\phi}_i$. By a hard decision based on $\hat{\phi}_i$, we mean that $\hat{c}_i = \arg\min_{c \in \{0,1,2,3\}} |\omega^c - \hat{\phi}_i|$.

B. Switching to an optimal algorithm

Our hybrid algorithm first computes the values $\hat{c}_i$, $i \in \{1, 2, 3\}$, as in majority logic. However, before committing to the hard estimates $\hat{c}_i$, the hybrid algorithm checks how close the hard estimates are to their soft counterparts. We establish a global "sensitivity" parameter θ. If $|\arg(\hat{\phi}_i) - \arg(\omega^{\hat{c}_i})| > \theta$, for some $i \in \{1, 2, 3\}$, we discard all the estimates $\hat{c}_i$, and revert to the optimal FHT decoder for the entire block; otherwise, we commit to the hard estimates $\hat{c}_i$. Since in practice the channel amplifies the signal by some unknown gain, we choose to use the difference in phase as a reliability measure instead of the difference in magnitude.

We now address how compute the estimate $\hat{c}_0$, once we have committed to $(\hat{c}_1, \hat{c}_2, \hat{c}_3)$. We set $\phi_i = \omega^{\hat{c}_i}$, for all $i \in \{1, 2, 3\}$, and then use the equations in (1) to compute eight votes for ϕ_0. Specifically, we set $\hat{\phi}_0 = \frac{1}{8}(r_0 - r_1\phi_1^* + r_2\phi_2^* + r_3\phi_1^*\phi_2^* - r_4\phi_3^* + r_5\phi_1^*\phi_3^* + r_6\phi_2^*\phi_3^* + r_7\phi_1^*\phi_2^*\phi_3^*)$, and make a hard decision $\hat{c}_0$ based on $\hat{\phi}_0$. Otherwise, if we are not confident in the estimates $\hat{c}_i$, we throw them out and revert to the optimal FHT decoder for this block.

The hybrid algorithm can be optimized in many ways, some of which we now describe. One simple optimization is to compute each $\hat{\phi}_i$ separately, and perform the confidence check before computing the next one. In this way, we save computation if the check fails. Also, we do not need to set each $\hat{\phi}_i$ to the *average* of its four votes, but simply to the *sum* of the votes, since we are only interested in the phase of $\hat{\phi}_i$. We need to compare each phase difference $|\arg(\hat{\phi}_i) - \arg(\omega^{\hat{c}_i})|$ against θ, which is a non-trivial task, since $\arg()$ is an expensive procedure. To overcome this problem, we do not actually compute $\arg()$. We set a constant $s = \tan\theta$, and just compare the ratio of the real and imaginary parts of $\hat{\phi}_i$ with s. This is logically equivalent to comparing $|\arg(\hat{\phi}_i) - \arg(\omega^{\hat{c}_i})| \le \theta$ (with perhaps some sign changes depending on the value of $\hat{c}_i$), and only requires one multiplication.

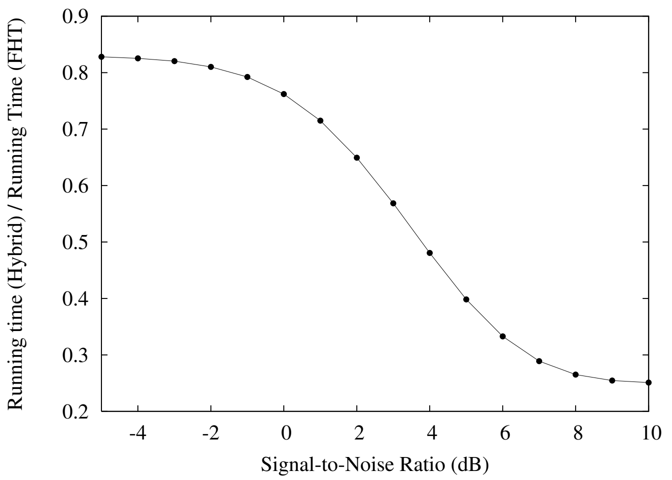

Fig. 1. Running time of Hybrid Decoding Algorithm as a function of SNR for CCK Demodulation. Running time is given as a fraction of the running time for the FHT decoder

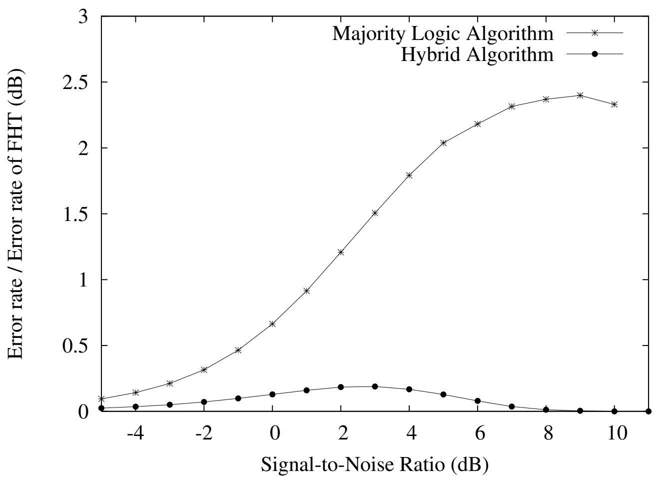

Fig. 2. Loss in block error rate of the majority logic and hybrid algorithms vs. optimal algorithm as a function of SNR. The y-axis is E_a/E_o in dB, where E_a is the block error rate of the plotted algorithm, and E_o is the block error rate of the optimal FHT algorithm.

C. Experimental Results

We ran our hybrid algorithm against an optimized version of the FHT decoder for SNR from -5 to 10, and measured the running time and block error rate of both at each SNR. By block error rate, we mean the number of blocks in which at least one of the four ϕ's is decoded incorrectly over the total number of blocks. We used a value of θ such that $\tan\theta = 2/3$. This offered the best trade-off between running time and error correcting ability, since comparing to the ratio $2/3$ is computationally simple. All experiments were performed on 1 billion encoded blocks of data subject to a simulated AWGN channel with varying SNR. All algorithms ran on a Pentium III 1 GHz processor.

When the SNR is high, the hybrid algorithm runs approximately four times faster than the FHT decoder. As the SNR decreases, the frequency with which the hybrid algorithm switches to the slower FHT decoder increases, and thus the running time increases. Figure 1 shows this relationship in detail. We remark that the hybrid algorithm is *always* faster than the FHT decoder, regardless of the noise level.

The error performance of the hybrid algorithm is quite good. Figure 2 shows the block error rate as a function of SNR of the hybrid algorithm and the majority logic algorithm (without switching), as compared to the optimal FHT algorithm. Here we see that the majority logic algorithm can perform a full $2.4\,$dB worse than FHT, whereas the hybrid algorithm is never more than $.2\,$dB worse, making it quite close to an optimal decoder.

III. MAJORITY-LOGIC DECODING OF FORM CODES

In this section we show how the hybrid algorithm can be generalized to arbitrary FORM codes [5]. We first review FORM codes, and establish a formal notation for them. We then go through the same steps as we did for CCK. We define simple computations on the received symbols that act as "votes" for each information symbol, show how to produce "soft estimates" using those votes, and explain how these soft estimates can be used as confidence measures in the same way as in CCK demodulation.

Consider an information word $c \in \mathbb{Z}_q^k$. The individual symbols $c_i \in \mathbb{Z}_q$ can be viewed as coefficients of a first-order polynomial $P(x) = c^T x$, where $x \in \{0, \ldots, p-1\}^k$ for some $p \leq q$. A codeword consists of $n = p^k$ symbols from $\mathbb{Z}^q$ and is obtained by evaluating $P(x) \mod q$ for all possible values of x. We denote this code by $FORM_q(k, p)$. For simplicity, we assume in the remainder of this paper that p is even. In classic Reed-Muller codes, $p = 2$, as it does for most such codes used in practice.

We note that CCK is essentially isomorphic to the the code $FORM_4(3, 2)$, apart from the negations used for autocorrelation, and the fact that ϕ_0 is differentially encoded. In fact, as we defined $FORM_4(3, 2)$, ϕ_0 does not exist at all. Such a "phase shift bit" can modeled in FORM codes by having an additional information symbol c' act as an additive constant to the polynomial P, so $P(x) = c^T x + c'$.

To derive "votes" for information symbols from a received codeword, we use a technique similar to the one used in Reed's algorithm [5] for decoding binary Reed-Muller codes of arbitrary order (See also [3], [6]). We decode one information symbol at a time, and produce an estimate $\hat{c} = (\hat{c}_1, \ldots, \hat{c}_k)$ of the original information word c.

For all $\ell \in \{1, 2, \ldots, k\}$, let F_ℓ be the set of all pairs (x, y), $x, y \in \{0, \ldots, p-1\}^k$, such that:

- $x_i = y_i$ for all $i \neq \ell$,
- x_ℓ is even, and
- $y_\ell = x_\ell + 1$.

The equality $P(y) - P(x) = c_\ell \pmod q$ holds for all $(x, y) \in F_\ell$, and therefore each pair in F_ℓ can be seen as casting a "vote" for c_ℓ. The cardinality of F_ℓ is $|F_\ell| = n/2$, and therefore any number $m \leq n/2$ of independent votes for c_ℓ may be selected from F_ℓ.

A. Soft-decision Decoding

Using q-PSK modulation, a codeword is sent through the channel as the set $\{\omega^{P(x)} : x \in \{0,\ldots,p-1\}^k\}$, where $\omega = e^{2\pi j/q}$, $j=\sqrt{-1}$. Denote a received symbol by $r_x = \omega^{P(x)} + N_x$, where N_x is a complex Gaussian random variable with mean 0 and variance $2\sigma^2$. We choose a set $V_\ell \subseteq F_\ell$ of m votes for each information symbol, and compute the following quantity:

$$A_\ell = \sum_{(x,y)\in V_\ell} r_y r_x^* \,.$$

Because $P(y)-P(x) = c_\ell \pmod q$ holds for all $(x,y)\in V_\ell$, we have $\omega^{P(y)}(\omega^{P(x)})^* = \omega^{P(y)-P(x)} = \omega^{c_\ell}$, and therefore we have $E[r_y r_x^*] = \omega^{c_\ell}$ and $E[A_\ell] = m\omega^{c_\ell}$. The majority-logic algorithm now makes a hard decision by setting $\hat{c}_\ell$ to the most likely value in $\mathbb{Z}_q$ given the value for A_ℓ. For all $i \in \mathbb{Z}_q$, let $\phi_i = \left|\arg(A_\ell) - \arg(\omega^i)\right|$. The hard decision is made by setting $\hat{c}_\ell = \arg\min_{i\in\mathbb{Z}_q} \phi_i$. This step can be made computationally efficient, especially for some values of q such as $q=4$.

B. Switching to an optimal algorithm

Not only does A_ℓ provide a value on which to make a hard decision, it also provides us with a measure of how reliable the decision is. If A_ℓ is close to its expectation, we are confident that the error is small and the decision correct.

The Hybrid algorithm accepts the decision when $\phi_{\hat{c}_\ell} \le \theta$, for all ℓ, for some fixed threshold angle $\theta < \pi/q$. Otherwise, it discards the decision and reverts to a ML algorithm for the entire block. A small θ improves the error correcting performance, because the optimal algorithm runs more often, but it increases the computational load.

IV. Bound on the Symbol Error Rate

In this section we give a theoretical bound on the symbol error rate H_e of the Hybrid algorithm under the AWGN channel. Our bound holds for any FORM code. We show that H_e is at most an additive $B_e = e^{-\Omega(m)}$ larger than the symbol error rate O_e of an optimal ML algorithm, where m is the number of votes we choose to compute for each information symbol, as long as the noise does not exceed a certain threshold. Since m can be made as large as $n/2$, where n is the block length of the code, this shows that the additive difference B_e in error rate between the hybrid algorithm and an optimal ML algorithm can be made exponentially small in the block length. We now state our main theorem.

Theorem 1: For all parameters α, θ and t, such that $0\le\alpha\le 1$, $0\le\theta<\pi/q$, and $0\le t<1$, we have $H_e \le O_e + B_e$, where

$$B_e \le e^{-\left(\frac{(1-\alpha)^2\sin^2(2\pi/q-\theta)}{4\sigma^2}\right)m} + e^{-\left(\frac{t\alpha\sin(2\pi/q-\theta)}{\sigma^2} - \ln\left(\frac{t\arccos(-t)}{(1-t^2)^{3/2}} + \frac{1}{1-t^2}\right)\right)m}.$$

Theorem 1 is proven in Section IV-B. For fixed q, θ and σ^2, the theorem gives $B_e = e^{-\Omega(m)}$, as long as there exist t and α such that $0\le t<1$, $0\le t<1$, and

$$\frac{t\alpha\sin(2\pi/q-\theta)}{\sigma^2} > \ln\left(\frac{t\ \arccos(-t)}{(1-t^2)^{3/2}} + \frac{1}{1-t^2}\right).$$

Many reasonable settings of the parameters give us this condition. For example, let $q=4$ (QPSK) and $\theta=\arctan(2/3)$ (as we chose for CCK). If we set $t=1/2$, then as long as $\sigma^2 \ge 1/5$, which corresponds to an SNR greater than 4 dB, Theorem 1 bounds B_e by approximately $2\cdot 2^{-m/5}$.

The remainder of this section is devoted to proving Theorem 1. In the majority-logic algorithm, each soft vote contributes to the soft symbol estimate a noise term that is the product of two independent Gaussian variables. Consequently, before proving Theorem 1, we first need to prove two technical lemmas to analyze the distribution of the product of two Gaussian random variables.

A. Probability distribution of the product of two Gaussians

In the following, K_ν denotes the modified Bessel functions of the second kind of index ν ([1, 9.6]). For brevity, we define the following function, which appears in the calculations in Lemma 3:

$$g(t) = \frac{t\ \arccos(-t)}{(1-t^2)^{3/2}} + \frac{1}{1-t^2}\,.$$

Lemma 2: Let a and b be independent identically distributed (IID) complex Gaussian variables with mean 0 and variance 2. The probability density function of $|ab|$ is $f(c) = cK_0(c)$.

Proof: Let $F(c) = Pr[|ab|<c]$ be the probability distribution function of $|ab|$, so that $f(c) = \frac{d}{dc}F(c)$. We have:

$$\begin{aligned}
1-F(c) &= \Pr[|ab|\ge c] \\
&= \frac{1}{(2\pi)^2}\int_{\mathbb{C}} da \int_{|b|\ge c/|a|} e^{-(|a|^2+|b|^2)/2}\, db \\
&= \frac{1}{2\pi}\int_{\mathbb{C}} e^{-|a|^2/2} da \int_0^{2\pi}\frac{d\theta}{2\pi}\int_{c/|a|}^{\infty} re^{-r^2/2}\, dr \\
&= \frac{1}{2\pi}\int_{\mathbb{C}} e^{-\frac{1}{2}\left(|a|^2+\frac{c^2}{|a|^2}\right)}\, da \\
&= \int_0^\infty re^{-\frac{c}{2}\left(\frac{r^2}{c}+\frac{c}{r^2}\right)}\, dr\,.
\end{aligned}$$

The substitution $e^z = r^2/c$ yields

$$1-F(c) = \int_{-\infty}^{\infty} \frac{c}{2}e^z\, e^{-c\cosh z}\, dz$$

The Bessel function K_1 obeys the following identity [1, 9.6.24]:

$$\begin{aligned}
K_1(c) &= \int_0^\infty e^{-c\cosh z}\cosh z\, dz \\
&= \int_{-\infty}^{\infty}\frac{1}{2}e^z e^{-c\cosh z}\, dz\,.
\end{aligned}$$

Therefore, $1-F(c) = cK_1(c)$ holds. From [1, 9.6.28], we have $\frac{d}{dc}[-cK_1(c)] = cK_0(c)$, and therefore the equality $f(c) = \frac{d}{dc}F(c) = \frac{d}{dc}\left[1-cK_1(c)\right] = cK_0(c)$ holds. ∎

Lemma 3: Let $a_1,\ldots,a_m$ and $b_1,\ldots,b_m$ be IID complex Gaussian variables with mean 0 and variance 2. Then, for all t such that $0\le t<1$, we have

$$\Pr\left[\sum_{i=1}^m |a_i b_i| \ge D\right] \le e^{\left(\ln g(t) - \frac{tD}{m}\right)m}\,.$$

Proof: Let $\mathcal{L}_s[f]$ be the Laplace transform of f, where f is the probability density function of an arbitrary $|a_i b_i|$. From Lemma 2, we have $f(c) = cK_0(c)$. From [1, 29.2.9], we have that $\mathcal{L}_s[-xF(x)] = (\mathcal{L}_s[F(x)])'$. Consequently, we have $\mathcal{L}_s[f(c)] = \mathcal{L}_s[c\, K_0(c)] = -(\mathcal{L}_s[K_0(c)])'$. From [7, p. 388], we have $\mathcal{L}_s[K_0(c)] = (1-s)^{-1/2} \arccos s$, for $s > -1$. Finally, we get:

$$\begin{aligned}
\mathcal{L}_s[f] &= -\frac{d}{ds}\left[(1-s^2)^{-1/2} \arccos s\right] \\
&= -\frac{s\ \arccos s}{(1-s^2)^{3/2}} + \frac{1}{1-s^2}, \\
&= g(-s), \qquad s > -1.
\end{aligned}$$

By the Chernoff bound, for all $0 \geq s > -1$,

$$\Pr\left[\sum_{i=1}^{m} |a_i b_i| \ \geq D\right] \leq \frac{(\mathcal{L}_s[f])^m}{e^{-sD}}$$

The lemma follows from the substitution $t = -s$. ■

B. Proof of Theorem 1

If $|\arg(A_\ell) - \arg(\omega^{c_\ell})| < 2\pi/q - \theta$, then either the algorithm decodes correctly, and does not switch to the optimal algorithm, or it switches to the optimal algorithm. If this event does not occur, we upper bound the probability of error by one. Let $B_e = \Pr[|\arg(A_\ell) - \arg(\omega^{c_\ell})| \geq 2\pi/q - \theta]$. Then, the inequality $H_e \leq O_e + B_e$ holds.

Let d be the distance between the point $m\omega^{c_\ell}$ and the closest point such that $|\arg(A_\ell) - \arg(\omega^{c_\ell})| \geq 2\pi/q - \theta$. We have $d = m \sin(2\pi/q - \theta)$. In order to cause an error, the distance A_ℓ must deviate from its mean by at least d. Consequently, we have

$$\begin{aligned}
B_e &\leq \Pr\left[|A_\ell - m\omega^{c_\ell}| \geq d\right] \\
&= \Pr\left[\left|\sum_{(x,y)\in V_\ell} N_x^* \omega^{P(y)} + N_y \omega^{-P(x)} + N_x^* N_y\right| \geq d\right],
\end{aligned}$$

where the N_x's are the IID complex Gaussian additive noise terms with zero mean and variance $2\sigma^2$.

For the i^{th} pair (x, y) in V_ℓ, define new random variables a_i, b_i, where $a_i = N_x^* \omega^{P(y)}$ and $b_i = N_y \omega^{-P(x)}$. These new variables are phase-shifted versions of IID Gaussians, and therefore they are IID Gaussian with the same variance as the elements of N. Using the new variables, we have

$$B_e \leq \Pr\left[\left|\sum_{i=1}^{m} a_i + b_i + \omega^{-c_\ell} a_i b_i\right| \geq d\right].$$

We break this event down into two events, which we treat independently. Formally, for $0 \leq \alpha \leq 1$, we have $B_e \leq B_e^{(1)} + B_e^{(2)}$, where

$$\begin{aligned}
B_e^{(1)} &= \Pr\left[\left|\sum_{i=1}^{m} a_i + b_i\right| \geq (1-\alpha)d\right], \\
B_e^{(2)} &= \Pr\left[\left|\sum_{i=1}^{m} \omega^{-c_\ell} a_i b_i\right| \geq \alpha d\right].
\end{aligned}$$

The probability $B_e^{(1)}$ is equivalent to $\Pr[|A| \geq (1-\alpha)d]$, where A is a complex Gaussian random variable with variance $4m\sigma^2$. We can compute this probability exactly:

$$\begin{aligned}
B_e^{(1)} &= \Pr\left[|A| \geq (1-\alpha)d\right] \\
&= e^{-\frac{(1-\alpha)^2 d^2}{4m\sigma^2}} \\
&= e^{-\left(\frac{(1-\alpha)^2 \sin^2(2\pi/q-\theta)}{4\sigma^2}\right)m}
\end{aligned} \tag{2}$$

To bound $B_e^{(2)}$, we note that the sum of the magnitudes of the products is no smaller than the magnitude of the sum of the products. We have:

$$\begin{aligned}
B_e^{(2)} &= \Pr\left[\left|\sum_{i=1}^{m} \omega^{-c_\ell} a_i b_i\right| \geq \alpha d\right] \\
&\leq \Pr\left[\sum_{i=1}^{m} |\omega^{-c_\ell} a_i b_i| \geq \alpha d\right] \\
&= \Pr\left[\sum_{i=1}^{m} |a_i b_i| \geq \alpha d\right]
\end{aligned} \tag{3}$$

(We can drop the ω^{-c_ℓ} factor since its magnitude is one.) Now let $a_i' = a_i/\sigma, b_i' = b_i/\sigma$, for all i. We have:

$$B_e^{(2)} \leq \Pr\left[\sum_{i=1}^{m} |a_i' b_i'| \geq \frac{\alpha d}{\sigma^2}\right].$$

By Lemma 3, we have

$$\begin{aligned}
B_e^{(2)} &\leq e^{\left(\ln g(t) - \frac{t\alpha d}{m\sigma^2}\right)m} \\
&= e^{-\left(\frac{t\alpha \sin(2\pi/q-\theta)}{\sigma^2} - \ln g(t)\right)m}
\end{aligned} \tag{4}$$

Combining (2) and (4) proves the theorem. ■

We note that our bound would be tighter if we could avoid the approximation in (3). Perhaps a tighter analysis of the sum of Gaussian products could help improve this bound further.

REFERENCES

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*. Dover, New York, 1965.

[2] J. L. Massey. Threshold decoding. Technical Report 410, MIT, Cambridge, Mass., 1963.

[3] K. Paterson and A. Jones. Efficient decoding algorithms for generalised Reed-Muller codes. Technical report, Hewlett-Packard Labs, Nov. 1998.

[4] Bob Pearson. Complementary code keying made simple, application note 9850, http://www.intersil.com/data/an/an9/an9850/an9850.pdf, May 2000.

[5] I. Reed. A class of multiple-error-correcting codes and the decoding scheme. *IRE Transactions on Information Theory*, PGIT-4:38–49, September 1954.

[6] R. van Nee. OFDM codes for peak-to-average power reduction and error correction. In *Proc. IEEE Globecom '96, London, England*, pages 740–744, Nov. 1996.

[7] G. N. Watson. *A Treatise on the Theory of Bessel Functions, Second Edition*. Cambridge University Press, 1996.

[8] S. Wicker. *Error Control Systems for Digital Communication and Storage*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[9] R. K. Yarlagadda and J. E. Hershey. *Hadamard Matrix Analysis and Synthesis*. Kluwer, Dordrecht, 1997.