Tensor Reed-Muller Codes: Achieving Capacity with Quasilinear Decoding Time

Emmanuel Abbe* Colin Sandon* Oscar Sprumont*†

*EPFL
†University of Washington

Abstract

Define the codewords of the Tensor Reed-Muller code $\mathsf{TRM}(r_1, m_1; r_2, m_2; \ldots; r_t, m_t)$ to be the evaluation vectors of all multivariate polynomials in the variables $\{x_{ij}\}_{i=1,\ldots,t}^{j=1,\ldots m_i}$ with degree at most r_i in the variables $x_{i1}, x_{i2}, \ldots, x_{im_i}$. The generator matrix of $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ is thus the tensor product of the generator matrices of the Reed-Muller codes $\mathsf{RM}(r_1, m_1), \ldots, \mathsf{RM}(r_t, m_t)$.

We show that for any constant rate R below capacity, one can construct a Tensor Reed-Muller code $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ of rate R that is decodable in quasilinear time. For any blocklength n, we provide two constructions of such codes:

- Our first construction (with $t = 3$) has error probability $n^{-\omega(\log n)}$ and decoding time $O(n \log\log n)$.
- Our second construction, for any $t \geq 4$, has error probability $2^{-n^{\frac{1}{2} - \frac{1}{2(t-2)} - o(1)}}$ and decoding time $O(n \log n)$.

One of our main tools is a polynomial-time algorithm for decoding an arbitrary tensor code $C = C_1 \otimes \ldots \otimes C_t$ from $\frac{d_{\min}(C)}{2\max\{d_{\min}(C_1),\ldots,d_{\min}(C_t)\}} - 1$ adversarial errors. Crucially, this algorithm does not require the codes $C_1, \ldots, C_t$ to themselves be decodable in polynomial time.

1 Introduction

Reed-Muller (RM) codes, which were introduced by Reed and Muller in 1954 [Ree54, Mul54], are one of the simplest and most widely used families of codes. Their codewords can be viewed as the evaluation vectors (over $\mathbb{F}_2^m$) of all polynomials of degree at most r in m variables. Although RM codes were recently shown to achieve capacity on the erasure channel [KKM+17] as well as all BMS channels for both the bit error [RP24] and the block error [AS23], we do not know of any polynomial-time algorithm for decoding them in the constant-rate regime. In this paper, we introduce a variant of RM codes called *Tensor Reed-Muller codes*, where the m variables are split into groups and the degree requirement is applied to each group separately[1]. We prove that Tensor Reed-Muller codes achieve capacity efficiently:

Theorem 1. *Consider any noise probability $p > 0$ and any rate $R < 1 - h(p)$. Then for any integers $n \in \mathbb{N}$ and $t \geq 4$, we can construct a Tensor Reed-Muller code $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ of length $n^{1 \pm o(1)}$ and rate $R \pm o(1)$ such that:*

1. *If $t = 3$, there exists a decoder D for $\mathsf{TRM}(r_1, m_1; r_2, m_2; r_3, m_3)$ with worst-case runtime $O(n \log\log n)$ and decoding success probability $1 - n^{-\omega(\log n)}$ under p-noisy errors.*

[1] See Section 2.2 for the formal definition.

2. *If $t > 3$, there exists a decoder D for $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ with worst-case runtime $O(n \log n)$ and decoding success probability $1 - 2^{-n^{\frac{1}{2} - \frac{1}{2(t-2)} - o(1)}}$ under p-noisy errors.*

Our analysis makes use of the following result for adversarial errors, which is of independent interest:

Theorem 2. *Consider any integers $r_1 \leq m_1, \ldots, r_t \leq m_t$ and define $n = 2^{m_1 + \ldots + m_t}$. Then there is an $O(n \log n)$-time algorithm for decoding the code $C := \mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ from*

$$\left\lceil \frac{d_{\min}(C)}{2 \max_i \left\{ 2^{m_i - r_i} \right\}} \right\rceil - 1$$

adversarial errors.

One can modify our algorithm to decode any tensor code $C := C_1 \otimes \ldots \otimes C_t$ from $\left\lceil \frac{d_{\min}(C)}{2 \max\{d_{\min}(C_1), \ldots, d_{\min}(C_t)\}} \right\rceil - 1$ adversarial errors, but in that case the algorithm runs in time $O(n \sum_i n_i)$, for n_i the length of code C_i (see Section 4.)

1.1 Previous Work

Reed-Muller codes have in recent years attracted a lot of attention for their decoding performances under random noise. By bounding their weight distribution appropriately, [ASW15, SS20] first showed that Reed-Muller codes achieve capacity on the erasure channel (BEC) and the symmetric channel (BSC) in the regimes where the rate of the code is either very close to 0 or very close to 1. [KKM+17] then leveraged the double transitivity of their permutation group to show that Reed-Muller codes of constant rates achieve capacity on the BEC.

[Sam20] combined the results of [KKM+17] with new ℓ_q-norm inequalities to obtain better bounds on the weight distributions of doubly transitive codes. This allowed [HSS21] to prove that Reed-Muller codes of constant rates can decode a constant fraction of random errors, although the maximum code rate allowed was below the capacity of the channel. The first capacity results were obtained in [RP24], which proved that for any $R \in (0, 1)$, the asymptotic bit error probability of a rate-R Reed-Muller code vanishes on any memoryless symmetric channel (BMS) whose capacity is greater than R. That still left open the question of showing that the block error probability vanishes as well, which was finally proven in [AS23, ASSV24]:

Theorem 3 ([AS23])**.** *Consider any error parameter $p \in (0, \frac{1}{2})$ and any rate $R < 1 - h(p)$. Then any sequence of Reed-Muller codes $\left\{ \mathsf{RM}(r_i, m_i) \right\}_i$ of asymptotic rate R satisfies*

$$\Pr_{z \sim p} \left[D_{ML}(c + z) = c \right] \geq 1 - 2^{-2^{\Omega(\sqrt{m_i})}}$$

for every $c \in \mathsf{RM}(r_i, m_i)$, where $z \sim p$ denotes a p-noisy error string and D_{ML} denotes the maximum-likelihood decoder for the code $\mathsf{RM}(r_i, m_i)$.

The line of work described above established that Reed-Muller codes achieve capacity on all BMS channels. The most natural next challenge is then to understand whether or not they can be decoded efficiently. Reed provided the first known algorithm in [Ree54], allowing for the correction of half the minimum distance many adversarial errors. For random errors, efficient algorithms for decoding $\mathsf{RM}(r, m)$ are known in the case where r is constant. The first algorithms [VMS92, Sak05] focused on the regime where $r \in \{1, 2\}$, but algorithms for all values of $r = O(1)$ have since been obtained. They exploit either the recursive structure of Reed-Muller codes [Dum04, DS06, Dum06, YA20] or

their minimum-weight parity checks [SHP18]. In the regime $r = \omega(1)$, the only known algorithm with proven polynomial runtime is the algorithm of [SSV17], which successfully decodes any error pattern in $\mathsf{RM}(m-2t, m)$ for which the same erasure pattern can be corrected in $\mathsf{RM}(m-t, m)$.

Theorem 4 ([SSV17])**.** *For any integers m and $t \leq m$, there exists an $O\left(2^m \cdot \mathrm{poly}(\binom{m}{\leq t})\right)$ - time decoder D for the code $\mathsf{RM}(m-2t, m)$ with the following property: for every $z \in \{0,1\}^{2^m}$, if*

$$\left\{x \in \mathsf{RM}(m-t, m) : z_i \geq x_i \text{ for all } i\right\} = \{0\},$$

then every $c \in \mathsf{RM}(m-2t, m)$ satisfies

$$D(c+z) = c.$$

In particular, there are currently no known polynomial-time algorithms for reliably decoding Reed-Muller codes of constant rates. In this paper, we leverage the results of [SSV17] to obtain efficient decoding algorithms for Tensor Reed-Muller codes of constant rates (see Theorem 1).

Our general approach shares some high-level similarities with Forney's concatenated codes [For66], as it also involves combining two codes to create a new one. However, tensoring presents two significant advantages. First, the tensor product of two Reed-Muller codes $\mathsf{RM}(r_1, m_1)$ and $\mathsf{RM}(r_2, m_2)$ has a simple and intuitive description: its codewords are the evaluation vectors of all multilinear polynomials in the variables $x_1, \ldots, x_{m_1}, y_1, \ldots, y_{m_2}$ with degree at most r_1 in the variables $x_1, \ldots, x_{m_1}$ and degree at most r_2 in the variables $y_1, \ldots, y_{m_2}$, and codewords can further be viewed as tensors (see Section 2). We do not know of a similarly simple description for the concatenation of $\mathsf{RM}(r_1, m_1)$ and $\mathsf{RM}(r_2, m_2)$. Second, concatenation usually requires the alphabet of the outer code to be large. One can try to mimic concatenation by arranging the entries of each codeword of a binary code C_{out} into groups of size k_1 and encoding each group of coordinates using an inner code C_{in} of dimension k_1, but this can have a significant impact on the minimum distance of the final code.

1.2 Proof Techniques

There are two regimes in which we currently have efficient decoding algorithms for Reed-Muller codes:

- Regime 1: when the noise is smaller than $1 - \frac{\binom{m}{\leq m-t}}{2^m}$, one can use the work of [SSV17] to decode a code $\mathsf{RM}(m-2t, m)$ of length $n = 2^m$.
- Regime 2: when the blocklength of the code is very small, brute-force decoding, which runs in time $O(2^n)$, may have reasonable runtime.

This work combines the two regimes above to obtain an efficient decoder for Tensor Reed-Muller codes. We take a short code $\mathsf{RM}(r_1, m_1)$ of rate $R - o(1)$ and tensor it with a longer code $\mathsf{RM}(r_2, m_2)$ of rate $1 - o(1)$. The codewords of the resulting code $\mathsf{TRM}(r_1, m_1; r_2, m_2)$ are all the matrices $A \in \{0,1\}^{2^{m_2} \times 2^{m_1}}$ such that each row of A is a codeword of $\mathsf{RM}(r_1, m_1)$ and each column of A is a codeword of $\mathsf{RM}(r_2, m_2)$. To decode our tensor code, we first use the brute-force algorithm to decode each row of A independently. After this first step, which takes polynomial time as long as $2^{m_1} \approx \log n$, only a $o(1)$ fraction of A's entries will have been incorrectly decoded. We then use the high-rate algorithm of [SSV17] to decode each column of A independently. At the end of this second decoding step, the fraction of incorrectly decoded entries will have dropped below

$n^{-\omega(\log n)}$, allowing us to take a union bound over all coordinates. To further reduce the decoding error probability, we can take the tensor product of $\mathsf{TRM}(r_1, m_1; r_2, m_2)$ with an even longer RM code $\mathsf{RM}(r_3, m_3)$ of rate $1 - o(1)$ and repeat the same argument.

The ideas outlined above, further iterated, would allow us to decode t-Tensor Reed-Muller codes (for some t) with a decoding failure probability of about $2^{-n^{1/4}}$. To bring this error rate closer to the distance-optimal $2^{-\Omega(\sqrt{n})}$, we introduce a new algorithm for decoding tensor codes from adversarial errors. This algorithm works for any tensor code $C_1 \otimes \ldots \otimes C_t$ and relies on the fact that erasures are generally easier to decode than errors.[2] First, replace each row that is not a codeword of C_1 by an all-erasures row. Then, go through each column and determine whether or not there is a unique codeword $c \in C_2$ compatible with the (now partially-erased) column. If so, replace the column by c; otherwise, replace every entry in the column by an erasure symbol. For $t > 2$, repeat this process with every additional axis. By adding additional checks that ensure we never return subtensors that are too far away from the corresponding input, we obtain in Theorem 5 an algorithm for decoding any arbitrary tensor code $C = C_1 \otimes \ldots \otimes C_t$ from $\frac{d_{\min}(C)}{2 \max_i \{d_{\min}(C_i)\}} - 1$ adversarial errors. For Reed-Muller codes, this algorithm runs in time $O(n \log n)$.

Our final construction combines the ideas of the above two paragraphs: the first two Reed-Muller codes $\mathsf{RM}(r_1, m_1)$ and $\mathsf{RM}(r_2, m_2)$ are of subpolynomial lengths and taken as in the first paragraph. As mentioned above, this allows us to bring the error probability down to about $n^{-\omega(\log n)}$. The remaining Reed-Muller codes $\{\mathsf{RM}(r_i, m_i)\}_{i=3,\ldots,t}$ all have $m_i = \frac{\log n - m_1 - m_2}{t-2}$ and $r_i = \frac{m_i + m_i^{3/4}}{2}$. By the arguments we outlined in the second paragraph, we can recover any sent codeword of $\mathsf{TRM}(r_3, m_3; \ldots; r_t, m_t)$ with fewer than about $n^{\frac{1}{2} - \frac{1}{2(t-2)}}$ errors. But since the first pass on $\mathsf{RM}(r_1, m_1)$ and $\mathsf{RM}(r_2, m_2)$ brought the error rate down to $n^{-\omega(\log n)}$, by the Chernoff bound, the probability that there are more than $n^{\frac{1}{2} - \frac{1}{2(t-2)}}$ errors is bounded by $2^{-n^{\frac{1}{2} - \frac{1}{2(t-2)} - o(1)}}$.

2 Notation and Preliminaries

Throughout this paper, we will use $\mathbb{N} = \{1, 2, 3, \ldots\}$ to denote the set of positive integers and $\log(x)$ to denote the logarithm of x in base 2. We define the entropy function $h : [0, 1] \to [0, 1]$ to be

$$h(x) := -x \log(x) - (1 - x) \log(1 - x).$$

For any positive real number a, we define $\lceil a \rceil$ to be the *ceiling* of a, i.e. the smallest integer $n \in \mathbb{N}$ such that $n \geq a$. For any $n \in \mathbb{N}$, we define the set $[n] := \{1, 2, \ldots, n\}$. For any $n \in \mathbb{N}$ and any $p \in [0, 1]$, we denote by $z \sim_n p$ the Boolean random vector of length n whose entries are independent and identically distributed Bernoulli variables of probability p. When n is clear from context, we will drop the subscript and write $z \sim p$. We will need the following two very standard results (see e.g. [BLM13] and [BHS80] respectively):

Lemma 1 (The Chernoff bound)**.** *Let $X_1, X_2, \ldots, X_n$ be independent random variables taking values in $\{0, 1\}$ and define $E := \sum_{i=1}^{n} \mathbb{E}[X_i]$. Then for any $\alpha \geq 1$, we have*

$$\Pr\left[\sum_{i=0}^{n} X_i \geq \alpha E\right] \leq \left(\frac{e^{\alpha - 1}}{\alpha^{\alpha}}\right)^{E}.$$

[2] For any linear code $C \subseteq \{0, 1\}^n$, given a partially-erased codeword of C, one can use the parity-check matrix to obtain a system of $n - \dim C$ linear equations in $e \leq n$ unknowns, where e is the number of erased coordinates. This can be solved by Gaussian elimination in time $O(n^3)$. For Reed-Muller codes, we improve this decoding time to $O(n \log n)$ whenever the number of erasures is below the minimum distance - see Lemma 3.

Lemma 2 (The Master theorem)**.** *Suppose $T(n)$ denotes the running time of an algorithm on an input of size n, and suppose $T(n)$ can be expressed recursively as*

$$T(n) \leq aT\left(\frac{n}{a}\right) + O(n)$$

for some constant $a > 0$. Then if $T(1) = O(1)$, we have $T(n) \leq O(n \log n)$.

2.1 Reed-Muller Codes

We will denote by $\mathsf{RM}(r, m)$ the Reed-Muller code with m variables and degree r. The codewords of the Reed-Muller code $\mathsf{RM}(r, m)$ are the evaluation vectors (over all points in $\mathbb{F}_2^m$) of all multivariate polynomials of degree $\leq r$ in m variables. We refer the reader to the survey [ASSY23] for a more thorough exposition to Reed-Muller codes.

2.2 Tensor Reed-Muller Codes

For any choice of Reed-Muller codes $\mathsf{RM}(r_1, m_1)$, $\mathsf{RM}(r_2, m_2), \ldots, \mathsf{RM}(r_t, m_t)$, we define the Tensor Reed-Muller code $\mathsf{TRM}(r_1, m_1; r_2, m_2; \ldots; r_t, m_t)$ as follows: Consider $m := \sum_i m_i$ variables $\{x_{ij}\}_{i=1,\ldots,t}^{j=1,\ldots m_i}$ and define the set

$$\mathcal{S} := \Big\{ S_1 \cup S_2 \cup \cdots \cup S_t : \text{ for all } i,\ S_i \subseteq \{x_{i1}, \ldots, x_{im_i}\} \text{ and } |S_i| \leq r_i \Big\}.$$

Abusing notation, we say that a monomial is in $\mathcal{S}$ if the set of its constituent variables is in $\mathcal{S}$. Then the evaluation vector of a polynomial $f(x_{11}, \ldots, x_{tm_t})$ over all points in $\{0,1\}^m$ is in $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ if and only if all the monomials of f are in $\mathcal{S}$. We note that the generator matrix of the code $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ is the tensor product of the generator matrices of the Reed-Muller codes $\mathsf{RM}(r_1, m_1), \ldots, \mathsf{RM}(r_t, m_t)$. We also note that the codewords of $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ can be seen as the t-dimensional tensors $A \in \{0,1\}^{2^{m_1} \times \cdots \times 2^{m_t}}$ satisfying the condition that for every $i \in [t]$, every i-axis vector of A is a codeword of $\mathsf{RM}(r_i, m_i)$.[3] Finally, we note that the rate of $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ is equal to the product of the rates of the codes $\{\mathsf{RM}(r_i, m_i)\}_{i=1}^t$.

3 Helpful Lemmas

In this section, we will prove performance guarantees for two decoding algorithms that will be used as subroutines throughout this paper. We start with an algorithm for efficiently testing and correcting Reed-Muller codes from adversarial erasures.

Lemma 3. *For any nonnegative integers $r \leq m$, there is an $O(m2^m)$-time algorithm which, given an input string $y \in \{0, 1, *\}^{2^m}$ with fewer than 2^{m-r} erasure symbols, determines whether or not there exists a codeword $c \in \mathsf{RM}(r, m)$ such that $y_i \in \{c_i, *\}$ for all $i \in [2^m]$. The algorithm returns such a codeword c if it exists and an error message otherwise.*

Proof. Our decoder is given in Algorithm 1. We first prove by induction on m that it always finds the desired codeword $c \in \mathsf{RM}(r, m)$ if such a codeword exists. Note that the base case $m = 1$ holds trivially, since Algorithm 1 always succeeds when $r = 0$ or $r = m$.

[3] We say that a vector $v \in \{0,1\}^{n_i}$ is an i-axis vector of a tensor $A \in \{0,1\}^{n_1 \times \cdots \times n_t}$ if it is a "row" of A along the i^{th} axis - formally, if there exist indices $\{j_k \in [n_k]\}_{k \in [t] \setminus i}$ such that for all $s \in [n_i]$, we have $v_s = A_{j_1, \ldots, j_{i-1}, s, j_{i+1}, \ldots, j_t}$.

Algorithm 1: Codeword testing and erasure correction for Reed-Muller codes

Input: Two integers $0 \leq r \leq m$ and a vector $y \in \{0,1,*\}^{2^m}$ with fewer than 2^{m-r} erasure entries.

Output: A codeword $c \in \mathsf{RM}(r,m)$ with $y_i \in \{c_i, *\}$ for all $i \in [2^m]$, if such a c exists; an error message otherwise.

if $r = 0$ **then**
  If there exists $b \in \{0,1\}$ such that $y_i \in \{b,*\}$ for all i, output $(b,b,\ldots,b)$. Otherwise, output an error message.
end
else if $r = m$ **then**
  Output y.
end
else
  Define $y^0 := (y_1,\ldots,y_{2^{m-1}})$ and $y^1 := (y_{2^{m-1}+1},\ldots,y_{2^m})$ and let $y^{\text{sum}} := y^0 + y^1$ (defining $y_i^{\text{sum}} = *$ whenever either $y_i^0 = *$ or $y_i^1 = *$). Run Algorithm 1 on input $(r-1, m-1, y^{\text{sum}})$ and denote the output you receive by c^{sum}. If c^{sum} is an error message, abort and output an error.
  if y^0 *contains fewer erasure symbols than* y^1 **then**
    Run Algorithm 1 on input $(r, m-1, y^0)$, denoting the output you receive by c^0. If c^0 is an error message, abort and output an error. Otherwise, define c to be the concatenation $c := (c^0, c^0 + c^{\text{sum}})$. If $y_i \in \{c_i, *\}$ for all $i \in [2^m]$, output c; otherwise, output an error.
  end
  else
    Run Algorithm 1 on input $(r, m-1, y^1)$, denoting the output you receive by c^1. If c^1 is an error message, abort and output an error. Otherwise, define c to be the concatenation $c := (c^1 + c^{\text{sum}}, c^1)$. If $y_i \in \{c_i, *\}$ for all $i \in [2^m]$, output c; otherwise, output an error.
  end
end

For the inductive case, suppose there exists a codeword $c \in \mathsf{RM}(r,m)$ such that y agrees with c on all non-erased entries. Let $f(x_1,\ldots,x_m)$ be the unique polynomial whose evaluation vector is c and express f as

$$f(x_1,\ldots,x_m) = f_0(x_2,\ldots,x_m) + x_1 \cdot f_1(x_2,\ldots,x_m). \tag{1}$$

We make the following two observations:

(i) Define $c^0 \in \{0,1\}^{2^{m-1}}$ to be the vector containing the first half of c's entries. Then c^0 is the evaluation vector of the polynomial $f_0(x_2,\ldots,x_m)$.

(ii) Define $c^1 \in \{0,1\}^{2^{m-1}}$ to be the vector containing the second half of c's entries. Then c^1 is the evaluation vector of $f_0(x_2,\ldots,x_m) + f_1(x_2,\ldots,x_m)$.

Both (i) and (ii) follow immediately from the fact that the indices of c are ordered lexicographically (and thus an index $v \in \mathbb{F}_2^m$ is in the first half if and only if $v_1 = 0$).

Now, since $y^0 := (y_1, \ldots, y_{2^m})$ is a corrupted evaluation vector for the polynomial f_0 and $y^1 := (y_{2^m+1}, \ldots, y_{2^m})$ is a corrupted evaluation vector for the polynomial $f_0 + f_1$, the vector $y^{\text{sum}} := y^0 + y^1$ must be an evaluation vector for f_1 corrupted with fewer than 2^{m-r} erasures. By induction, since f_1 has degree $\leq r-1$, running Algorithm 1 on input $(r-1, m-1, y^{\text{sum}})$ will then return the correct evaluation vector c^{sum} for the polynomial $f_1(x_2, \ldots, x_m)$. (See line 8.)

Furthermore, since y contains fewer than 2^{m-r} erasures, one of y^0 or y^1 must contain fewer than 2^{m-r-1} erasures. Without loss of generality, we assume that y^0 contains fewer erasures[4]. By induction, running Algorithm 1 on input $(r-1, m, y^0)$ will then return the correct evaluation vector c^0 for the polynomial f_0. (See line 10.) Since c^1 is the evaluation vector for $f_0 + f_1$ and we have obtained the evaluation vectors c^0, c^{sum} for f_0 and f_1, setting $c^1 = c^0 + c^{\text{sum}}$ will successfully recover c^1. Thus the algorithm indeed outputs the correct codeword $c = (c_0, c_1)$. (See line 10.)

We have proven that whenever there exists a (by our theorem's requirement, unique) codeword $c \in \mathsf{RM}(r, m)$ that is consistent with y, our algorithm returns it. Note also that Algorithm 1 never outputs a codeword $c \in \mathsf{RM}(\mathsf{r}, \mathsf{m})$ that is not consistent with y; this is because before returning c, the algorithm verifies that $y_i \in \{c_i, *\}$ for all $i \in [2^m]$ (see lines 10 and 13). Thus Algorithm 1 always succeeds. As for the runtime analysis, we note that at each step, the algorithm spends $O(2^m)$ time performing basic computations and then makes 2 recursive calls on instances of length 2^{m-1}. By the Master theorem (Lemma 2), the total runtime will thus be $O(m2^m)$.

□

Our second lemma is essentially a special case of the work of [SSV17] (Theorem 4), which we will use in the following form to bound the running time and error probability of a decoder for high-rate Reed-Muller codes.

Lemma 4. *Consider any integers $m > t > 0$ and any $p \leq \frac{2^{-m+\frac{t}{2}}}{5}$. Then there exists a decoder $\tilde{D}$ for the Reed-Muller code $\mathsf{RM}(m-t, m)$ with the following two properties.*

1. *$\tilde{D}$ runs in time $O\left(2^m \cdot \text{poly}\left(\binom{m}{\leq \frac{t}{2}}\right)\right)$.*
2. *Under random errors of probability p, $\tilde{D}$ succeeds in decoding any sent codeword $c \in \mathsf{RM}(m-t, m)$ with probability*

$$\Pr_{z \sim p}\left[\tilde{D}(c+z) = c\right] \geq 1 - 2^{-2^{t/2}}.$$

Proof. By Theorem 4, it will suffice to show that the Reed-Muller code $\mathsf{RM}(m - \frac{t}{2}, m)$ can recover from p-noisy erasures with success probability $1 - 2^{-2^{t/2}}$. Since the code $\mathsf{RM}(m - \frac{t}{2}, m)$ has minimum distance $2^{t/2}$, it is enough to prove that for independent Bernoulli variables $X_1, X_2, \ldots, X_{2^m}$ with Bernoulli parameter $p \leq \frac{2^{-m+\frac{t}{2}}}{5}$, we have

$$\Pr\left[X_1 + X_2 + \ldots X_{2^m} \geq 2^{t/2}\right] \leq 2^{-2^{t/2}}.$$

But this follows immediately from the Chernoff bound (Lemma 1). □

[4]The proof is identical in the other case, with the roles of y^0 and y^1 reversed.

4 Decoding Arbitrary Tensor Codes from Adversarial Errors

In this section, we will prove a generalization of Theorem 2, which we state in Theorem 5. For any code $C \subseteq \{0,1\}^n$, define the function $f_C : \{0,1,*\}^n \to \{0,1,*\}^n$ to be

$$f_C(x) := \begin{cases} c & \text{if } c \in C,\ x_i \in \{c_i, *\} \text{ for all } i, \text{ and } x \text{ has fewer than } d_{\min}(C) \text{ erasures} \\ (*,\ldots,*) & \text{otherwise.} \end{cases} \tag{2}$$

The function f_C essentially tells us whether or not a partially-erased binary string with fewer than $d_{\min}(C)$ erasures is consistent with any codeword of C. It can always be computed in time $O(n^3)$ (see Note 2), and its runtime dictates the runtime of our following decoder for adversarial errors.

Theorem 5. *Consider any linear codes $C_1 \subseteq \{0,1\}^{n_1}, \ldots, C_t \subseteq \{0,1\}^{n_t}$ and define $n := \prod_{i=1}^t n_i$. Suppose there exists a function $T : \mathbb{N} \to \mathbb{N}$ such that $T(m) \geq m$ for all $m \in \mathbb{N}$ and such that for all $i \in [t]$, there is a $T(n_i)$-time algorithm for computing the function f_{C_i} defined in (2). Then there is an $O\left(\sum_{i=1}^t \frac{n}{n_i} \cdot T(n_i)\right)$-time algorithm for decoding the tensor code $C := C_1 \otimes \ldots \otimes C_t$ from*

$$\left\lceil \frac{d_{\min}(C)}{2 \max\left\{d_{\min}(C_1), \ldots, d_{\min}(C_t)\right\}} \right\rceil - 1$$

adversarial errors.

Remark 1. *By Note 2, for any linear codes $C_1, \ldots, C_t$ we can take $T(n) = O(n^3)$, which gives us a runtime of $O\left(n \sum_i n_i^2\right)$. But for Reed-Muller codes, we can do better: by Lemma 3, we can take $T(n) = O(n \log n)$, which gives a runtime of $O\left(\sum_i n \log n_i\right) = O(n \log n)$.*

Remark 2. *Note that in Algorithm 2, at each layer $i = 2, \ldots, t$ of the decoding process, the i-axis erasure pattern is the same for all i-axis vectors within the same i-subtensor of A. One could thus use Gaussian elimination to express the erased entries in this erasure pattern as a linear combination of the non-erased entries, then go through each i-axis vector of A and correct the erasures accordingly. If C_1 is taken to be the code of maximum length among $C_1, \ldots, C_t$, this will give a running time of $O(n \sum_i n_i)$ for decoding the tensor product of any linear codes $C_1, \ldots, C_t$ of lengths $n_1, \ldots, n_t$.*

Proof. Note that we may assume that each n_i is greater than 1, as otherwise C trivially reduces to a tensor product of $t-1$ codes. Our algorithm for the case where each n_i is greater than 1 is given in Algorithm 2. We first show by induction that it always outputs either a codeword of $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ or the all-erasures tensor. The base case $t = 1$ is trivial. For the case $t > 1$, we note that by induction, after the line-5 for loop completes, each $(t-1)$-subtensor of A is either a valid codeword of $\mathsf{TRM}(r_1, m_1; \ldots; r_{t-1}, m_{t-1})$ or a tensor filled with erasure symbols. Thus the erasure pattern of each t-axis vector in the line 8-for loop is identical. In particular, if there is a unique consistent codeword for each of these t-axis vectors, then the erased entries of each t-axis vector can be expressed as the same linear combination of non-erased coordinates. This means that the erased $(t-1)$-subtensors can be expressed as linear combinations of the non-erased $(t-1)$-subtensors; since the code is linear, the newly recovered subtensors must then be codewords of $\mathsf{TRM}(r_1, m_1; \ldots; r_{t-1}, m_{t-1})$. Combining this with the fact that by line 9, each t-axis vector is a codeword of $\mathsf{RM}(r_t, m_t)$ (otherwise by line 12, we would output an all-erasures tensor), we get that any Boolean output must indeed be a codeword of $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$.

Now that we have proven that our algorithm always outputs either a valid codeword or a tensor filled with erasures, we turn to showing that the algorithm correctly ouputs the sent codeword as long as there are fewer than $\frac{d_{\min}(C)}{2 \max\left\{d_{\min}(C_1), \ldots, d_{\min}(C_t)\right\}}$ errors. We again proceed by induction. The

Algorithm 2: A polynomial-time decoder for arbitrary tensor codes

Input: An $n_1 \times \ldots \times n_t$ Boolean tensor A.

Output: Either a codeword of the tensor code $C_1 \otimes \ldots \otimes C_t$ or an $n_1 \times \ldots \times n_t$ tensor filled with erasure symbols.

1 **if** $t = 1$ **then**

2 If A has fewer than $d_{\min}(C_1)$ erasures and there is a codeword $c \in C_1$ with $A_i \in \{c_i, *\}$ for all $i \in [n_1]$, replace A by c. Otherwise replace every entry in A by an erasure symbol.

3 **end**

4 **else**

5 **for** $i = 1, 2, \ldots, n_t$ **do**

6 Run Algorithm 2 on the $n_1 \times \ldots \times n_{t-1}$ tensor A^i whose entries are given by $A^i_{j_1 \ldots j_{t-1}} = A_{j_1 \ldots j_{t-1} i}$. Replace the entries $\{A_{j_1 \ldots j_{t-1} i}\}$ of A by the output entries.

7 **end**

8 **for** *every* $t-$*axis vector* $v \in \{0, 1, *\}^{n_t}$ *of* A *(see definition 3)* **do**

9 If v has fewer than $d_{\min}(C_t)$ erasures and there is a codeword $c \in C_t$ with $v_i \in \{c_i, *\}$ for all i, replace v by c. Otherwise, replace v by $(*, *, \ldots, *)$.

10 **end**

11 **if** *the updated tensor* A *contains erasure symbols or its Hamming distance from the original input is at least* $\frac{d_{\min}(C_1) \cdot \ldots \cdot d_{\min}(C_t)}{2}$ **then**

12 Replace every entry of A by an erasure symbol.

13 **end**

14 **end**

15 Output the updated tensor A.

base case $t = 1$ is trivial. For the general case, we note that in order for our Algorithm 2 to fail in decoding a noisy codeword containing fewer than $\frac{d_{\min}(C_1) \cdot \ldots \cdot d_{\min}(C_t)}{2}$ errors, one of the following two statements must hold:

(i) After we decode all the $(t-1)$-dimensional subtensors $\{A^i\}_{i \in [n_t]}$ (see line 6), there is a non-erasure erroneous entry in at least one of the updated subtensors A^i.

(ii) After we decode all the $(t-1)$-dimensional subtensors $\{A^i\}_{i \in [n_t]}$, there are at least $d_{\min}(C_t)$ values of $i \in [n_t]$ for which the updated subtensor A^i contains one or more erasures.

Indeed, if neither of these occur, then our line 9 will allow us to recover every entry of A correctly. We now show that neither point (i) nor point (ii) can occur. Suppose for contradiction that there exists a subtensor A^i as described in point (i). Note that by line 12, the Hamming distance between A^i and its corresponding input must be less than $\frac{d_{\min}(C_1) \cdot \ldots \cdot d_{\min}(C_{t-1})}{2}$. Since A^i is a codeword of $C_1 \times \ldots \times C_{t-1}$ (we proved in the first paragraph that any output of our algorithm is a codeword) and since $C_1 \times \ldots \times C_{t-1}$ has minimum distance $d_{\min}(C_1) \cdot \ldots \cdot d_{\min}(C_{t-1})$, there must have been at least $\frac{d_{\min}(C_1) \cdot \ldots \cdot d_{\min}(C_{t-1})}{2}$ corrupted entries in the i^{th} subtensor to begin with. This contradicts our theorem's requirement on the total number of errors.

Suppose instead that there are $d_{\min}(C_t)$ subtensors $\{A^{i_1}, \ldots, A^{i_{d_{\min}(C_t)}}\}$ satisfying point (ii) above. Since each of these subtensors is decoded independently, by our inductive hypothesis there

must have been at least $d_{\min}(C_t) \cdot \frac{d_{\min}(C_1)\cdot\ldots\cdot d_{\min}(C_{t-1})}{2\max\left\{d_{\min}(C_1),\ldots,d_{\min}(C_{t-1})\right\}} \geq \frac{d_{\min}(C)}{2\max\left\{d_{\min}(C_1),\ldots,d_{\min}(C_t)\right\}}$ errors. But this again contradicts our theorem's requirement on the total number of errors.

This concludes the proof of correctness. We now turn to the runtime analysis. Define $R(n_1,\ldots,n_t)$ to be the maximal runtime of Algorithm 2 on any code $C' = C'_1 \otimes \ldots \otimes C'_t$ with $C'_i \subseteq \{0,1\}^{n_i}$ for all i. Note that for $t > 1$, we have

$$R(n_1,\ldots,n_t) \leq n_t \cdot R(n_1,\ldots,n_{t-1}) + \frac{n}{n_t} T(n_t) + \alpha n,$$

where the first and second terms correspond to the bulk of the runtime of Algorithm 2 for the for-loops 5 and 8 respectively, whereas the constant α is chosen to be big enough that the αn-term covers all the other operations needed throughout the algorithm. We claim that

$$R(n_1,\ldots,n_t) \leq (\alpha+1)\sum_{i=1}^{t} \frac{n}{n_i} \cdot T(n_i).$$

For $t = 1$, the statement is obvious. For $t > 1$, by induction we have

$$\begin{aligned} R(n_1,\ldots,n_t) &\leq (\alpha+1) n_t \sum_{i=1}^{t-1} \frac{n/n_t}{n_i} T(n_i) + \frac{n}{n_t} T(n_t) + \alpha n \\ &\leq (\alpha+1)\sum_{i=1}^{t} \frac{n}{n_i} T(n_i), \end{aligned}$$

where in the last line we used the fact that by our theorem's requirement on T, we have $\alpha n = \frac{\alpha n}{n_t} n_t \leq \frac{\alpha n}{n_t} T(n_t)$.

□

5 Decoding Tensor Reed-Muller Codes from Random Errors

In this section, we leverage our Algorithm 2 to prove the following formal version of Theorem 1. Note that it is sufficient to prove Theorem 1 for $t \leq \sqrt{\log n}$, since after that the $O\left(\frac{1}{t}\right)$ improvement in the error probability is subsumed by the $o(1)$ term (one can always artificially increase t by taking tensor products with the trivial Reed-Muller code $\{0,1\}$).

Theorem 6. *Consider any constants $p \in (0, \frac{1}{2})$ and $R < 1 - h(p)$. Let $n \in \mathbb{N}$ be some growing parameter and consider any corresponding integer $3 \leq t \leq \sqrt{\log n}$. Define*

- $m_1 := \lceil \log\log n - 3 \rceil$ *and* r_1 *to be any integer such that* $\frac{\binom{m_1}{\leq r_1}}{2^{m_1}} = R \pm o(1)$
- $m_2 := \lceil 10 \log\log n \rceil$ *and* $r_2 := \lceil \frac{m_2}{2} + \sqrt{m_2} \log m_2 \rceil$
- $m_3 = \ldots = m_t := \lceil \frac{\log n - m_1 - m_2}{t-2} \rceil$ *and* $r_3 = \ldots = r_t := \lceil \frac{m_3 + m_3^{3/4}}{2} \rceil$

Then the Tensor Reed-Muller code $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ *has length* $n^{1+o(n)}$ *and rate* $R \pm o(1)$. *Moreover, there exists a decoder D with the following properties:*

1. *D has runtime $O(n \log\log n)$ in the case $t = 3$ and runtime $O(n \log n)$ in the case $t > 3$.*

2. For every codeword $c \in \mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$, D has decoding failure probability

$$\Pr_{z\sim p}\left[D(c+z) \neq c\right] \leq \begin{cases} n^{-\omega(\log n)} & \text{if } t = 3, \\ 2^{-n^{\frac{1}{2}-\frac{1}{2(t-2)}-o(1)}} & \text{otherwise.} \end{cases}$$

Proof. Note that $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ has rate

$$\begin{aligned} \prod_{i=1}^{t} \frac{\binom{m_i}{\leq r_i}}{2^{m_i}} &\geq \big(R - o(1)\big) \prod_{i=3}^{t} \left(1 - \frac{2^{h(\frac{m_i - r_i}{m_i})m_i}}{2^{m_i}}\right) \\ &\geq (R - o(1)) \prod_{i=3}^{t} \left(1 - 2^{-\frac{\sqrt{m_3}}{2\ln 2}}\right) \\ &\geq (R - o(1))(1 - t2^{-\frac{\sqrt{m_3}}{2\ln 2}}), \end{aligned}$$

where the first inequality follows from the fact that $\binom{n}{\leq d} \leq 2^{h(\frac{d}{n})n}$ for all integers n and $d \leq \frac{n}{2}$, the second inequality follows from the fact that $h\left(\frac{1-\mu}{2}\right) \leq 1 - \frac{\mu^2}{2\ln 2}$ for any $\mu \in (0,1)$, and the third inequality follows from the fact that $(1+x)^r \geq 1 + rx$ for all $x \geq -1$ and $r \geq 1$. Since $t \leq \sqrt{\log n}$ and $m_3 = \Omega(\frac{\log n}{t})$, our code $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ indeed has rate $R \pm o(1)$. It also has length $2^{m_1+m_2} \cdot 2^{(t-2)\lceil \frac{\log n - m_1 - m_2}{t-2} \rceil} = n^{1+o(1)}$.

The decoder D we use for decoding our code is given in Algorithm 3. We represent each codeword of $\mathsf{TRM}(r_1, m_1; \ldots; r_t, m_t)$ as a $2^{m_1} \times \ldots \times 2^{m_t}$ Boolean tensor. For any tensor $A \in \{0,1\}^{2^{m_1} \times \ldots \times 2^{m_t}}$, we call any vector along the first axis of A a "row" of A and call any vector along the second axis of A a "column" of A. We first bound the probability that Algorithm 3 makes a decoding mistake. Note that the only way a decoding mistake can occur is if either:

(i) Our algorithm would fail even if the counter condition (last sentence of line 10) was disregarded.

(ii) The counter eventually exceeds $n2^{-2^{(\log\log n)^{1/4}}}$.

We first show that point (i) is very unlikely to occur. Note that by Theorem 3, the lookup table D_1 used in the row-for loop of our Algorithm 3 (line 6) satisfies that for any $c_1 \in \mathsf{RM}(r_1, m_1)$,

$$\begin{aligned} \Pr_{z\sim p}\left[D_1[c_1 + z] \neq c_1\right] &\leq 2^{-2^{\Omega(\sqrt{m_1})}} \\ &\leq 2^{-2^{\Omega(\sqrt{\log\log n})}}. \end{aligned} \tag{3}$$

Thus at the end of the row-for loop, every entry of A has probability $\varepsilon \leq 2^{-2^{\Omega(\sqrt{\log\log n})}}$ of being different from the corresponding entry of the sent codeword B. Furthermore, since D_1 was applied independently to every row of A, any coordinates of A that are not in the same row have uncorrelated probabilities of being correct.

In particular, at the end of the row-for loop, the entries of any given column of A have uncorrelated probabilities of being incorrect. Now, since $\varepsilon \leq \frac{2^{-m_2}}{5}$ for all n large enough, we get from Lemma 4 that for any $c_2 \in \mathsf{RM}(r_2, m_2)$, the decoder D_2 used in the column-for loop (line 10) of our Algorithm

Algorithm 3: An efficient decoder for t-tensor Reed-Muller codes

Input: A $2^{m_1} \times \ldots \times 2^{m_t}$ Boolean tensor A.

Output: A $2^{m_1} \times \ldots \times 2^{m_t}$ Boolean tensor.

1 Create a look-up table $D_1 : \{0,1\}^{2^{m_1}} \to \mathsf{RM}(r_1, m_1)$.

2 **for** *each vector* $s \in \{0,1\}^{2^{m_1}}$ **do**

3 | Use brute-force search to find the $c \in \mathsf{RM}(r_1, m_1)$ closest to s. Set $D_1[s] = c$.

4 **end**

5 **for** *each row* u *of* A **do**

6 | Replace u by $D_1[u]$.

7 **end**

8 counter $\leftarrow 0$.

9 **for** *each column* v *of our updated tensor* A **do**

10 | Use Lemma 3 to check if $v \in \mathsf{RM}(r_2, m_2)$. If not, increase counter by 1 and replace the column v by the codeword $D_2(v) \in \mathsf{RM}(r_2, m_2)$, where D_2 is the decoder from Lemma 4 for the code $\mathsf{RM}(r_2, m_2)$. If counter $> n2^{-2^{(\log\log n)^{1/4}}}$, abort the entire algorithm and return the $\vec{0}$ codeword.

11 **end**

12 If $t = 3$, output the updated tensor A. If $t > 3$, run Algorithm 2 on A and return the output (if Algorithm 2 outputs a tensor filled with erasure symbols, return the $\vec{0}$ codeword).

3 satisfies

$$\begin{aligned} \Pr_{z\sim\varepsilon}\left[D_2(c_2+z) \neq c_2\right] &\leq O\left(2^{-2^{\frac{m_2-r_2}{2}}}\right) \\ &= 2^{-2^{2.5\log\log n \pm o(\log\log n)}} \\ &\leq 2^{-\omega\left(\log^2 n\right)} \end{aligned} \tag{4}$$

Thus at the end of our column-for loop, if we disregard the counter condition when running the algorithm, then every entry of A has probability

$$\varepsilon' \leq n^{-\omega(\log n)} \tag{5}$$

of being incorrect. Taking a union bound over all coordinates then concludes the analysis of point (i) for the case $t = 3$. For the case $t > 3$, we define for every $k \in [2^{m_3}] \times \ldots \times [2^{m_t}]$ the Boolean random variable X_k that is 1 if and only if upon running Algorithm 3 without the counter condition, at the end of the column-for loop (line 11), there exists $(i,j) \in [2^{m_1}] \times [2^{m_2}]$ such that the entry A_{ijk} is incorrect. By (5), since D_1 was applied independently to every row and D_2 was applied independently to every column, the random variables $\{X_k\}$ are independent Bernoulli random variables with probability parameter at most $2^{m_1+m_2}\varepsilon' = n^{-\omega(\log n)}$. By the Chernoff bound (Lemma 1), we then have

$$\Pr\left[\sum_k X_k \geq \frac{d_{\min}\big(\mathsf{TRM}(r_1,m_1;\ldots;r_t,m_t)\big)}{2^{m_1+m_2+1}d_{\min}\big(\mathsf{RM}(r_3,m_3)\big)}\right] \leq 2^{-\frac{d_{\min}\left(\mathsf{TRM}(r_1,m_1;\ldots;r_t,m_t)\right)}{d_{\min}\left(\mathsf{RM}(r_3,m_3)\right)}\cdot n^{-o(1)}}$$
$$\leq 2^{-n^{\frac{t-3}{2(t-2)}-o(1)}}.$$

Thus if we disregard the counter condition when running the algorithm, we get that with probability $1-2^{-n^{\frac{t-3}{2(t-2)}-o(1)}}$, there are $\leq \frac{d_{\min}\left(\mathsf{TRM}(r_1,m_1;\ldots;r_t,m_t)\right)}{2d_{\min}\left(\mathsf{RM}(r_3,m_3)\right)}$ errors remaining after the line-9 for loop. By Theorem 5, line 12 will then succeed with probability at least $1-2^{-n^{\frac{1}{2}-\frac{1}{2(t-2)}-o(1)}}$. This concludes our analysis for point (i). We then turn to showing that point (ii) is very unlikely to occur. Note that by (3), at the end of the row-for loop (line 7) of our algorithm, for any $k \in [2^{m_3}]\times\ldots\times[2^{m_t}]$ we have

$$\Pr\left[\exists(i,j)\in[2^{m_1}]\times[2^{m_2}] \text{ such that } A_{i,j,k} \text{ is incorrect}\right] \leq 2^{m_2}\cdot 2^{-2^{\Omega(\sqrt{\log\log n})}}$$
$$\leq 2^{-2^{\Omega(\sqrt{\log\log n})}}.$$

Since Algorithm 3 processes each $k\in[2^{m_3}]\times\ldots\times[2^{m_t}]$ independently up to the end of the column-for loop (line 11), we get

$$\Pr\left[\mathsf{counter} \text{ exceeds } n2^{-2^{(\log\log n)^{1/4}}}\right] \leq \Pr\left[\sum_{k=1}^{n2^{-m_1-m_2}} 2^{m_1}Y_k \geq n2^{-2^{(\log\log n)^{1/4}}}\right]$$

for $\{Y_k\}$ independent Bernoulli variables of probability $2^{-2^{\Omega(\sqrt{\log\log n})}}$. By the Chernoff bound (Lemma 1), we then have

$$\Pr\left[\mathsf{counter} \text{ exceeds } n2^{-2^{(\log\log n)^{1/4}}}\right] \leq 2^{-\Omega\left(n2^{-2^{(\log\log n)^{1/4}}}/2^{m_1}\right)} \leq O\left(2^{-\sqrt{n}}\right).$$

Combining this bound for point (ii) with our previously established bound for point (i), we get

$$\Pr_{z\sim p}\left[D(c+z)\neq c\right] \leq n^{-\omega(\log n)} + O\left(2^{-\sqrt{n}}\right)$$

for the case $t=3$ and

$$\Pr_{z\sim p}\left[D(c+z)\neq c\right] \leq 2^{-n^{\frac{1}{2}-\frac{1}{2(t-2)}-o(1)}} + O\left(2^{-\sqrt{n}}\right)$$

for the case $t>3$, as desired. We now turn to bounding our algorithm's runtime. Since there are $2^{2^{m_1}}$ vectors in $\{0,1\}^{2^{m_1}}$, creating the look-up table D_1 takes time

$$2^{2^{m_1}}\cdot O(2^{2^{m_1}}2^{m_1}) = o(n). \tag{6}$$

Since there are $\frac{n}{2^{m_1}}$ rows in the tensor A and since looking up a value in the table D_1 takes time $O(2^{m_1})$, the row-for loop (line 5) then takes time

$$\frac{n}{2^{m_1}}\cdot O\left(2^{m_1}\right) = O\left(n\right). \tag{7}$$

For the column-for loop (line 9), since there are $\frac{n}{2^{m_2}}$ columns in the tensor A, by Lemmas 3 and 4 the algorithm takes time

$$\frac{n}{2^{m_2}} \cdot O(m_2 2^{m_2}) + n 2^{-2^{(\log\log n)^{1/4}}} \cdot 2^{O(m_2)} = O\left(n \log\log n\right). \tag{8}$$

Combining equations (6), (7) and (8), we get that our decoder D has total runtime $O\left(n \log\log n\right)$ when $t = 3$. When $t > 3$, the algorithm additionally has to process line 12, which takes time

$$O\left(n \log n\right) \tag{9}$$

by Theorem 5 and Lemma 3. This brings the total runtime for the case $t > 3$ to $O\left(n \log n\right)$.

□

References

[AS23] Emmanuel Abbe and Colin Sandon. A proof that Reed-Muller codes achieve Shannon capacity on symmetric channels. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 177–193. IEEE, 2023. 1, 2

[ASSV24] Emmanuel Abbe, Colin Sandon, Vladyslav Shashkov, and Maryna Viazovska. Polynomial Freiman-Ruzsa, Reed-Muller codes and Shannon capacity, 2024. 2

[ASSY23] Emmanuel Abbe, Ori Sberlo, Amir Shpilka, and Min Ye. Reed-Muller codes. *Foundations and Trends in Communications and Information Theory*, 20(1–2):1–156, 2023. 5

[ASW15] Emmanuel Abbe, Amir Shpilka, and Avi Wigderson. Reed-Muller codes for random erasures and errors. *IEEE Trans. Inf. Theory*, 61(10):5229–5252, 2015. 2

[BHS80] Jon Louis Bentley, Dorothea Haken, and James B. Saxe. A general method for solving divide-and-conquer recurrences. *SIGACT News*, 12(3):36–44, 1980. 4

[BLM13] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press, 2013. 4

[DS06] Ilya Dumer and Kirill Shabunov. Recursive error correction for general Reed-Muller codes. *Discret. Appl. Math.*, 154(2):253–269, 2006. 2

[Dum04] Ilya Dumer. Recursive decoding and its performance for low-rate Reed-Muller codes. *IEEE Trans. Inf. Theory*, 50(5):811–823, 2004. 2

[Dum06] Ilya Dumer. Soft-decision decoding of Reed-Muller codes: a simplified algorithm. *IEEE Trans. Inf. Theory*, 52(3):954–963, 2006. 2

[For66] George D. Forney. Concatenated codes. *MIT Press*, 1966. 3

[HSS21] Jan Hazla, Alex Samorodnitsky, and Ori Sberlo. On codes decoding a constant fraction of errors on the BSC. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1479–1488. ACM, 2021. 2

[KKM+17] Shrinivas Kudekar, Santhosh Kumar, Marco Mondelli, Henry D. Pfister, Eren Sasoglu, and Rüdiger L. Urbanke. Reed-Muller codes achieve capacity on erasure channels. *IEEE Trans. Inf. Theory*, 63(7):4298–4316, 2017. 1, 2

[Mul54] David E. Muller. Application of boolean algebra to switching circuit design and to error detection. *Trans. I R E Prof. Group Electron. Comput.*, 3(3):6–12, 1954. 1

[Ree54] I. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4(4):38–49, 1954. 1, 2

[RP24] Galen Reeves and Henry D. Pfister. Reed-Muller codes on BMS channels achieve vanishing bit-error probability for all rates below capacity. *IEEE Trans. Inf. Theory*, 70(2):920–949, 2024. 1, 2

[Sak05] Bassem Sakkour. Decoding of second order Reed-Muller codes with a large number of errors. In Michael J. Dinneen, Ulrich Speidel, and Desmond P. Taylor, editors, *Proceedings of the IEEE ITSOC Information Theory Workshop 2005 on Coding and Complexity, ITW 2005, Rotorua, New Zealand, August 29 - September 1, 2005*, pages 176–178. IEEE, 2005. 2

[Sam20] Alex Samorodnitsky. An upper bound on l_q norms of noisy functions. *IEEE Trans. Inf. Theory*, 66(2):742–748, 2020. 2

[SHP18] Elia Santi, Christian Häger, and Henry D. Pfister. Decoding Reed-Muller codes using minimum- weight parity checks. In *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*, pages 1296–1300. IEEE, 2018. 3

[SS20] Ori Sberlo and Amir Shpilka. On the performance of Reed-Muller codes with respect to random errors and erasures. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1357–1376. SIAM, 2020. 2

[SSV17] Ramprasad Saptharishi, Amir Shpilka, and Ben Lee Volk. Efficiently decoding Reed-Muller codes from random errors. *IEEE Trans. Inf. Theory*, 63(4):1954–1960, 2017. 3, 7

[VMS92] A. S. Pershakov V. M. Sidel'nikov. Decoding of Reed-Muller codes with a large number of errors. *Problemy Peredachi Informatsii*, 28(3):80–94, 1992. 2

[YA20] Min Ye and Emmanuel Abbe. Recursive projection-aggregation decoding of Reed-Muller codes. *IEEE Trans. Inf. Theory*, 66(8):4948–4965, 2020. 2