ERROR FLOOR PREDICTION WITH MARKOV MODELS FOR QC-MDPC CODES

SARAH ARPIN[1], JUN BO LAU[2], RAY PERLNER[3], ANGELA ROBINSON[3], JEAN-PIERRE TILLICH[4], AND VALENTIN VASSEUR [5]

Abstract. Quasi-cyclic moderate-density parity check (QC-MDPC) code-based encryption schemes under iterative decoders offer highly-competitive performance in the quantum-resistant space of cryptography, but the decoding-failure rate (DFR) of these algorithms are not well-understood. The DFR decreases extremely rapidly as the ratio of code-length to error-bits increases, then decreases much more slowly in regimes known as the waterfall and error-floor, respectively.

This work establishes three, successively more detailed probabilistic models of the DFR for iterative decoders for QC-MDPC codes: the simplified model, the refined model for perfect keys, and the refined model for all keys. The models are built upon a Markov model introduced by Sendrier and Vasseur [SV19b] that closely predicts decoding behavior in the waterfall region but does not capture the error floor behavior. The simplified model introduces a modification which captures the dominant contributor to error floor behavior which is convergence to near codewords introduced by [Vas21a]. The refined models give more accurate predictions taking into account certain structural features of specific keys.

Our models are based on the step-by-step decoder, also used in [SV19b], which is highly simplified and experimentally displays worse decoding performance than parallel decoders used in practice. Despite the use of the simplified decoder, we obtain an accurate prediction of the DFR in the error floor and demonstrate that the error floor behavior is dominated by convergence to a *near codeword* during a failed decoding instance. Furthermore, we have run this model for a simplified version of the QC-MDPC code-based cryptosystem BIKE to better ascertain whether the DFR is low enough to achieve IND-CCA2 security. Our model for a modified version of BIKE 1 gives a DFR which is below $2^{-129.5}$, using a block length $r = 13477$ instead of the BIKE 1 parameter $r = 12323$.

1. Introduction

1.1. Motivation.

The NIST PQC standardization process. The U.S. National Institute of Standards and Technology (NIST) Post-Quantum Cryptography (PQC) standardization process, which began with 82 submissions, has selected four algorithms for standardization while three algorithms remain under consideration in the fourth round. One of the remaining candidates is BIKE, a cryptosystem based on quasi-cyclic moderate density parity check (QC-MDPC) codes which are decoded by an iterative Black-Grey-Flip (BGF) decoder [ABB+21, DGK20].

The BIKE cryptosystem offers competitive performance but lacks a formal security claim in the static-key setting, unlike its code-based competitors.

Estimating the DFR of BIKE. Cryptosystems with a non-zero probability of decryption failures require careful analysis when proving IND-CCA2 security. In particular, the standard security proof [HHK17] requires the average failure probability to be below $2^{-\lambda}$ where λ is the security parameter. BIKE is one such cryptosystem so meeting IND-CCA2 security requires that the average Decoding Failure Rate (DFR) of the QC-MDPC code used to be below $2^{-\lambda}$. A low DFR for BIKE is not just a theoretical concern, since a DFR sufficiently higher than $2^{-\lambda}$ enables the GJS key-recovery attack [GJS16], which exploits decoding failures in an IND-CCA security model.

The target DFR of $2^{-\lambda}$ is too small to compute directly, so another approach is needed to prove that the DFR of BIKE is below $2^{-\lambda}$ for the given parameter sets. Previous work [SV19b, SV19a, DGK20] approximates the DFR by (i) directly computing the average DFR for smaller code sizes by running the decoder (which is doable since the DFR is high enough so that decoding failures can be observed) then (ii) extrapolating the behavior to estimate the DFR for larger parameters which are out of reach of experiments. However, the DFR falls into two regimes, making such extrapolations unreliable. The first regime is called the *waterfall region*, where the DFR decays more than exponentially in the block size n of the scheme (with other parameters fixed). The second regime is called the *error floor* which kicks in when the block size is sufficiently large and where the decay is much slower (Figure 1). The issue is that the experiments are done in the waterfall region and extrapolation is made under the hypothesis that for the BIKE parameters we are still in the waterfall region. This could lead to underestimation of the DFR and consequently, an underestimation of the block size needed to achieve a particular security parameter λ.

It remains an open problem to predict for which value of n the error floor begins in the case of BIKE. It is really a problem of being able to predict the iterative decoding performance of a QC-MDPC code. For LDPC codes, which have an even sparser parity-check matrix (rows and columns of weight $O(1)$ instead of $O(\sqrt{n})$ as for MDPC codes) and when using the same kind of iterative decoder, the error floor phenomenon is better understood. For LDPC codes, the error floor is either due to the existence of low-weight codewords which would fool any decoder (and not only the suboptimal iterative decoders), or of small *near codewords*. The latter are also called trapping sets and are errors of small weight that also have a syndrome of small weight [Ric03, HB18, VCN14]. Error vectors which have a large enough intersection with those codewords or near codewords are known to cause the error floor behavior in LDPC codes.

On the other hand, MDPC codes typically have no low-weight codewords or near codewords. However, BIKE is based on Quasi-Cyclic MDPC (QC-MDPC) codes, and this gives BIKE more structure than a random MDPC code. BIKE admits a parity-check matrix $\boldsymbol{H}$ formed by two circulant blocks. Such codes have codewords of weight equal to the row weight w of $\boldsymbol{H}$ (which is of order $O(\sqrt{n})$) [SV19a, Vas21b, BBC$^+$21]. Moreover, there are near codewords of half the row weight [Vas21b] which, due to the block-circulant structure,

is also equal to the column weight d of $\boldsymbol{H}$. The lower bound[1] on the DFR based on those moderate-weight codewords is too low to be of concern for the BIKE parameters [BBC$^+$21].

However, a clear explanation of the error floor for QC-MDPC has not been found to date. The effect of the near codewords put forward in [Vas21b] could be a factor since it has been shown in [Vas21b] that if the error vector has a large intersection with one of the near codewords, then the DFR conditioned on this event is non-negligible and can be observed experimentally for the BIKE parameters. However, the probability that the error has a large enough intersection with those near codewords so that the DFR conditioned on this event can be measured experimentally is too small to be of concern for the BIKE parameters, even if it is bigger than the contribution to the DFR coming from the moderate-weight codewords. The experimental study conducted in [ABH$^+$22] shows that for scaled-down BIKE parameters where the error floor can be observed experimentally, the errors that contribute to the DFR do not have large intersection with the near codewords of [Vas21b].

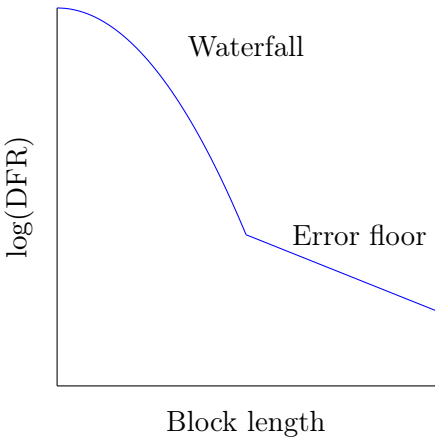


FIGURE 1. Waterfall and error floor regions for decoding failure rate.

Remarkably, the waterfall region is much better understood. Sendrier and Vasseur showed that an iterative decoder which works step by step can be analyzed by a Markov chain approach [SV19b, Vas21b]. Decoding step by step means that bits are flipped one at a time: at each step of the decoder a random position is considered and flipped according to whether the number of unsatisfied parity-checks involving this position (which is called the *counter* of the position) is greater than some threshold. The rationale behind this rule is as follows: Consider a bit which is not in error and denote by π_0 the probability that a parity-check involving this bit is not satisfied. Similarly we denote by π_1 the probability that a parity-check involving a bit in error is not satisfied. The crucial observation is that

$$\pi_0 < \pi_1. \tag{1}$$

Recall that due to the regular structure of $\boldsymbol{H}$ all positions are involved in the same number d of parity-checks. We therefore expect that the counter of a position which is not in error behaves as a binomial random variable $\mathrm{Bin}(d, \pi_0)$ for parameters d and π_0 [2] whereas the counter of a position which is in error behaves as a binomial random variable $\mathrm{Bin}(d, \pi_1)$. Therefore a bit in error tends to be involved in more unsatisfied parity-checks than a bit which is correct. For the standard iterative decoder, all positions are considered at once and flipped or not according to the same rule. The fact that the decoder is step by step makes it much more amenable to a Markov chain modeling where the states are the pairs

[1] It corresponds to the probability that the error covers at least half one of those codewords.

[2] The notation $\mathrm{Bin}(n, p)$ denotes a binomial random variable of parameters n and p; *i.e.* the sum of n i.i.d. Bernoulli variables X_i of parameter p, that is $\mathbf{P}(X_i = 1) = p$ for all i in $\{1, \cdots, n\}$.

(s,t), t being the number of positions which are in error and s the syndrome weight of the error. This Markov chain model closely predicts the DFR in the waterfall region but does not capture the error floor region. Roughly speaking, the DFR is predicted here by computing the resulting probability distributions of the pairs (s,t). The predicted DFR is the probability to attain a state where t is not zero at the end.

1.2. **Intuition behind our approach.**

Convergence to near codewords in the error floor regime. Our work started with a fundamental observation, namely that in the toy examples for which we could observe experimentally the error floor phenomenon, the remaining error at the end of the iterative decoding process when decoding failed covered in many cases one of the near codeword put forward in [Vas21b]. Interestingly enough, [ABH$^+$22, Fig 3,4 and 5] shows that in the error floor regime, errors which result in a decoding failure tend to have a bigger intersection with at least one of those near codewords than a random error of the same size. It was also experimentally shown in [ABH$^+$24] that the majority of decoding failures in the error floor of a scaled-down version of BIKE were due to these near codewords. Moreover, when we increase the block size resulting in the fact that we move away even further from the waterfall region, this phenomenon becomes even more prominent.

There is a good justification for this behavior as explained in §2.6. The point is that we actually do not have two kinds of behaviors for the counters depending on if the bit is in error or not as explained before, but rather four different behaviors of the counters. At each iteration of the iterative decoder, consider the near codeword that is the closest to the actual error (which incidentally as explained before, all near codewords are of size d, the column weight of $\boldsymbol{H}$) and denote by u the size of its intersection with the actual error. As we will explain in §2.6, we expect the following four kinds of behavior:

(1) The counters of the u bits that are at the same time in error and belong to the closest near codeword are distributed as the sum of two independent binomial variables $\mathrm{Bin}(u-1,\pi_0)+\mathrm{Bin}(d-u+1,\pi_1)$
(2) The counters of the $d-u$ bits that belong to the closest near codeword but are not in error are distributed as the sum of two binomial variables $\mathrm{Bin}(u,\pi_1)+\mathrm{Bin}(d-u,\pi_0)$.
(3) The counters of the rest of the $t-u$ bits in error behave "as usual" as a binomial variable $\mathrm{Bin}(d,\pi_1)$.
(4) The counters of the rest of the $n-d-t+u$ bits that are not in error also behave as expected, namely as a binomial variable $\mathrm{Bin}(d,\pi_0)$.

In other words, since $\pi_0 < \pi_1$, we expect an abnormal behavior from the d bits which belong to the closest near codeword. Indeed, those that are in error are less likely to be corrected by the iterative decoding process than the others which are in error: their counter is somewhat in between a counter of a bit which is in error and a counter of a bit which is not in error. Similarly, the $d-u$ bits of the near codeword that are not in error have a greater chance to be wrongly flipped by the iterative decoding process than the other bits in error. This is even worse, because the more bits are in error in the near codeword, the more the bits involved in this near codeword behave as their opposite: those that are still

not in error tend to look even more like bits which are in error. We experience in this case a snowball effect. As soon as enough bits are in error in the near codeword, there is a non negligible chance to end up at the end of the iterative decoding process with the whole near codeword being in error. It turns out that due to the peculiar structure of the near codeword, the iterative decoding process gets stuck there.

All this discussion suggests that errors which give rise to a decoding failure in the error floor region should be somewhat closer to one of those near codewords at the beginning and the intersection size only increases during the decoding process. This is the rationale of why we adapted the Markov model of [SV19b, Vas21b] to also keep track of this parameter u which is the biggest intersection of a near codeword with the residual error vector in the decoding process.

1.3. **Our approach.**

A new Markov model. We have followed two different but related approaches for keeping track of the size of the intersection of the error vector with the near codewords in a Markov model. A general Markov Model assumes there are N possible states for a given system. At each step, it is possible to move from one state to another with prescribed probabilities. These probabilities are assumed to be Markovian, i.e. it is assumed that transitioning to any state only depends on the current state and not on any additional information. .

- The first approach keeps track of the size u of the intersection of the error with a single, randomly selected near codeword $\boldsymbol{\nu}$. This allows to estimate the contribution to the DFR coming from $\boldsymbol{\nu}$. The difficulty here is to combine this contribution properly with the contributions coming from the other near codewords. We work with this approach in Section 4.
- The second approach keeps track of the size u of the intersection of the error with the closest near codeword. This alleviates the previous difficulty but introduces a new one, which is that during the decoding process we may switch to another near codeword. This is not easy to model by the Markov model. However, by making the assumption that in the error floor regime the decoding failure is dominated by errors which are close to a single near codeword, and that this does not change during the whole decoding process, we capture the DFR in the error floor regime with high accuracy. We work with this approach in Sections 5 and 6.

A Markov model taking into account the structure of the key. The first approach has been tested with scaled-down BIKE parameters and by taking a random key $\boldsymbol{H}$. To simplify the computation of π_0 and π_1 we made the heuristic assumption that the columns of $\boldsymbol{H}$ are drawn uniformly at random up to fixed column weight d. This new Markov model captures the error floor region and provides a much more conservative estimate than the DFR estimate based on the Markov model of [SV19b, Vas21b] which is too optimistic in the error floor regime. It turns out that the new DFR estimate is larger than the true estimate in both the waterfall and error floor regions (and not only in the waterfall regime as was the case for the Markov model of [SV19b, Vas21b]). This is quite encouraging since even

with this simple model there is a tool for choosing the parameters of BIKE in a conservative manner.

However, one of the drawback of this simple Markov model is that it does not take into account the structure of the key. It is indeed well known that there are weak keys [DGK20, Vas21b, NSP+23, WWW23] for which the decoding process behaves really badly. Moreover, there is a non-negligible difference between the best and worst key of the DFR in the few thousand of instances we took in the toy example we considered. This led us to refine the Markov model. We have tested Approach 1.3 and improved it in two ways : we took into account the structure of the key and we computed the transition probabilities in a more sophisticated way.

There is one case, where the structure of the key is much more amenable to a refined model - the case of perfect keys. Roughly speaking, this corresponds to a key where the near codewords have a very regular structure: the number of parity-check equations involving the positions in the support $\mathcal{S}$ of a given near codeword is maximal in this case. There are exactly $\binom{d+1}{2}$ such equations. They can be partitioned in a set of d equations each of them involving a different position in $\mathcal{S}$ and $\binom{d}{2}$ other equations each of them involving a different pair of the near codeword positions of $\mathcal{S}$. This structure is really helpful to set up a more accurate, but more involved, Markov model. The improvement we got with the new model is quite significant. We checked the model on a bigger toy example and it turns out that the model predicts rather accurately the error floor and this even when we chose different threshold rules for the step-by-step decoder.

One of the difficulties of having a Markov model working for all sorts of keys is that the near codewords split into two classes, those that have their support in the first half of the code positions and those having their support in the second half. When the keys are not perfect these two classes of near codewords might affect the DFR in a very different way . To circumvent this problem, we added a bit to the Markov state, *i.e.* getting a state (s, t, u, b) where $b = 0$ indicates that the closest near codeword has all its support in the first half of the positions and 1 otherwise. At the expense of increasing the state space size and adding more involved transition probability formulas, we get a Markov model which captures now all possible keys. Remarkably enough, in the toy examples we tried, this model is still accurate for predicting the error floor for the step-by-step decoder and other variants: the majority decoder (the threshold for flipping being half the total number of equations involving a bit), the BGF decoder, or a decoder with a new custom threshold rule which improves the DFR by a significant amount.

This Markov model can be run on the BIKE parameters and the new threshold rule displays remarkable performance despite the fact that it is based on the step by step decoder which is known to have worse performance than the standard decoder [Vas21b]. This improvement in the DFR can be traced back to the slight change in the decoding strategy put forward in [Sen24]. Rather than a fixed rule for the threshold, it is better to have a conservative threshold at the beginning of the decoding process and then gradually increase the threshold as the process continues. This lowers the chance of flipping bits which are not in error but in the near codeword at the begninning of the decoding procedure, thus

alleviating the phenomenon of convergence to a near codeword. We also verified what causes the error floor in the toy experiments: the experiments indicated a convergence to a near codeword, and so did the Markov model. This is also confirmed in the Markov model for the BIKE parameters. In this work, we conclude that the dominant contribution in the error floor for QC-MDPC codes under iterative decoders is due to convergence to a near codeword. We also verified what causes the error floor in the toy experiments: the experiments indicated a convergence to a near codeword, and so did the Markov model.

1.4. **Our contribution.** We present three Markovian models for the decoding behavior of QC-MDPC codes under iterative, hard-decision decoders:

(1) The Simplified Model which considers the distance of the error vector at each round of decoding from one fixed, randomly selected near codeword. This model captures the error floor regime and gives a more conservative estimate than experimental data and the [SV19b] model. It is discussed in Section 4.
(2) The Refined Model for Perfect Keys which considers the distance of the error vector at each round of decoding from the closest near codeword and applies only to perfect keys (Definition 4). This model is discussed in Section 5. It:
    - captures the error floor regime better than the first model
    - is studied with two different threshold rules
    - is helpful in understanding the final, generalized model.
(3) The Refined Model for All Keys which considers the distance of the error vector at each round of decoding from the closest near codeword and applies to all keys. This model computes degrees in a certain graph associated with the key to derive the model. This model is discussed in Section 6. We use this model to
    - give an extremely close approximation of the experimental data in both the waterfall and error floor regimes, further improving upon first two models and the work of [SV19b]
    - to estimate for the BIKE 1 parameters and with the previous BIKE 1 decoder using a single threshold function a DFR of about 2^{-91}
    - to show that for this decoding rule, there is an error floor behavior which appears already for slightly larger values of the block length than the one which was chosen and depending on the gap parameter δ of the decoder gives a contribution to the DFR which is between 2^{-100} and 2^{-120}
    - to improve this decoding rule by simply raising the minimum allowed threshold from 36 to 38 to show that for a slightly larger block length ($r = 13477$ instead of $r = 12323$) for obtaining a DFR which is below $2^{-129.5}$ for a typical key.

These models predict and experiments confirm that the dominating cause of failures in the error floor regime in a wide range of parameters is due to convergence to near codewords. All models use the step-by-step decoder which is experimentally shown to produce more decoding failures than parallel decoders. The most impactful model is the Refined Model for All Keys which matches the experimental data, also run using a step-by-step decoder, remarkably well throughout both the error floor and waterfall regimes. This model is applied to the BIKE cryptosystem parameter sets to give a DFR approximation; it is expected that

the DFR for the same parameter sets using a parallel bit-flipping decoder as specified in [ABB+21] and decoding rule based on iteration dependent threshold functions would result in a lower DFR than what is approximated by the model. This work is a strong indication that using only slightly larger BIKE parameters than those proposed, raising a little bit the minimum threshold, and filtering out atypical keys during key generation, should be enough to satisfy IND-CCA2 security.

2. Background

2.1. **Notation.** All codes considered are over $\mathbb{F}_2$.

Basic notation. Vectors and matrices are respectively denoted in bold letters and bold capital letters such as $\boldsymbol{a}$ and $\boldsymbol{A}$. Vectors are assumed to be row vectors and $\boldsymbol{x}^\intercal$ denotes the column vector which is the transpose of the row vector $\boldsymbol{x}$. The entry at index i of the vector $\boldsymbol{x}$ is denoted by x_i or $x(i)$. The Schur (componentwise) product $(x_i y_i)_{1\leqslant i\leqslant n}$ of two vectors $\boldsymbol{x} = (x_i)_{1\leqslant i\leqslant n}$ and $\boldsymbol{y} = (y_i)_{1\leqslant i\leqslant n}$ of the same length is denoted by $\boldsymbol{x} \star \boldsymbol{y}$. The hamming weight $|\boldsymbol{x}|$ of a vector $\boldsymbol{x}$ is the number of nonzero entries.

Probabilistic notation. $\mathscr{B}(n,p)$ denotes the binomial distribution of parameters n and p, that is the probability of the sum $\sum_{i=1}^{n} X_i$ of n independent identically distributed Bernoulli variables of parameter p. $\mathrm{Bin}(n,p)$ denotes a random variable distributed as $\mathscr{B}(n,p)$. Expressions like $\mathrm{Bin}(n_1,p_1) + \mathrm{Bin}(n_2,p_2)$ stand for the sum of the two independent binomial random variables $\mathrm{Bin}(n_1,p_1)$ and $\mathrm{Bin}(n_2,p_2)$. By abuse of notation we write $X \sim Y$ when two random variables X and Y have the same distribution.

2.2. **Iterative Decoders.**

Problem 1 (syndrome decoding problem (SDP))**.** *Given an $r \times n$ parity check matrix $\boldsymbol{H} \in \mathbb{F}_2^{r\times n}$, syndrome $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{e}^\intercal$, and target weight t, find an error vector $\boldsymbol{e}'$ that satisfies $\boldsymbol{H}\boldsymbol{e}'^\intercal = \boldsymbol{s}^\intercal$ and $|\boldsymbol{e}'| = t$.*

Iterative, bit-flipping syndrome decoders generally solve the SDP by guessing that columns of $\boldsymbol{H}$ with *many* bits in common with $\boldsymbol{s}$ are likely affected by errors. Let $\boldsymbol{H}_i$ denote the ith column of matrix $\boldsymbol{H}$, and define $\sigma_i(\boldsymbol{H},\boldsymbol{s}) = |\boldsymbol{H}_i \star \boldsymbol{s}|$, the number of unsatisfied parity check equations involving column $\boldsymbol{H}_i$ of $\boldsymbol{H}$. The bit-flipping decoders initialize $e' = \boldsymbol{0}$, compute $\sigma_j(\boldsymbol{H},\boldsymbol{s})$ for all or some $j \in \{0,\ldots,n-1\}$, then flip the jth bit e'_j whenever $\sigma_j \geq T$, where T is a given threshold function. We say a decoding failure has occurred whenever the output e' satisfies: $\boldsymbol{H}\boldsymbol{e}'^\intercal \neq \boldsymbol{s}^\intercal$ or $\boldsymbol{e}' \neq \boldsymbol{e}$.

The accuracy of bit-flipping decoders greatly depends on the threshold function. If the threshold is set too high, the decoder will cease to flip bits before arriving at the correct vector $\boldsymbol{e}'$. If the threshold is too low, more bits will be flipped than should be, again resulting in an incorrect vector $\boldsymbol{e}'$.

It is most common for iterative decoders to consider each column of $\boldsymbol{H}$ in comparison with $\boldsymbol{s}$ during each round of decoding. This can result in multiple bits in the guess $\boldsymbol{e}'$ being flipped in one round of decoding. The BIKE cryptosystem round 4 specification uses the

Black-Grey-Flip (BGF) iterative decoder [DGK20]. A closed form analysis of the DFR of the BGF decoder remains an open problem.

In this work we analyze the step-by-step decoder - an iterative decoder which randomly selects one column of $\boldsymbol{H}$ for comparison with $\boldsymbol{s}$ during each decoding round: At most one bit is flipped per round. This simplifies the computation of the Markov chain transition probabilities by restricting the number of possible states in the next step.

The step-by-step decoder is studied in detail and experimental evidence [Vas21b, Chap 7, Fig. 7.1] shows that for the classical bit flipping strategy, the step-by-step decoder performs significantly worse than the parallel version.

2.3. **QC-MDPC polynomial notation.** We assume that we have a quasi-cyclic moderate density parity check (QC-MDPC) code $C := C(n, r)$ with column weight d and $n = 2r$. Let $\boldsymbol{H} \in M_{r\times n}(\mathbb{F}_2)$ a parity-check matrix which describes the code C. The ring of circulant matrices of prime dimension r over $\mathbb{F}_2$ is isomorphic to the polynomial ring $\mathbb{F}_2[x]/(x^r - 1)$. Below, we show how to encode the information of $\boldsymbol{H}$ as two polynomials in $\mathbb{F}_2[x]/(x^r - 1)$.

Notation 1. *We denote the parity-check matrix $\boldsymbol{H} := (\boldsymbol{h}_0, \boldsymbol{h}_1)$ and overload the notation by introducing the polynomials*

$$h_i(x) = \sum_{j=0}^{r-1} h_{j,i}x^j \in \mathbb{F}_2[x]/(x^r - 1),$$

with $i \in \{0, 1\}$, $j \in \{0, \cdots, r - 1\}$. The polynomial $h_i(x)$ encodes the first column of the block $\boldsymbol{h}_i$ of the parity check matrix $\boldsymbol{H}$. The polynomial $xh_i(x)$ encodes the second column of the block $\boldsymbol{h}_i$, and so on. We will also write

$$h_0(x) = \sum_{j=1}^{d} x^{l_j} \quad \text{and} \quad h_1(x) = \sum_{j=1}^{d} x^{r_j},$$

where l_j and r_j run over the nonzero entries in the first column of $\boldsymbol{h}_0$ and $\boldsymbol{h}_1$, respectively.

We likewise represent error vectors as polynomials:

$$\boldsymbol{e} \in \mathbb{F}_2^n = (e_0, e_1, ...e_{r-1}, e_r, e_{n-1}) \rightsquigarrow$$
$$e(x) = e_0(x) \oplus e_1(x) = \sum_{j=0}^{r-1} e_j x^j \oplus \sum_{j=0}^{r-1} e_{r+j}x^j \in (\mathbb{F}_2[x]/(x^r - 1))^2.$$

We use "$\oplus$" to denote a formal sum, and "$+$" to denote addition in the ring $\mathbb{F}_2[x]/(x^r - 1)$.

The syndrome of an error $\boldsymbol{e} = (e_0, ..., e_{n-1})$ can be computed directly from the polynomials $e_0(x), e_1(x), h_0(x)$, and $h_1(x)$ as follows: Let $\boldsymbol{H}_i$ denote the columns of $\boldsymbol{H}$ for $i = 0, ..., n-1$ and recall that the polynomial $x^{i-rj}h_j(x)$ represents this column. In vector form, we have:

$$\boldsymbol{H}\boldsymbol{e}^T = \sum_{i=0}^{n-1} e_i \boldsymbol{H}_i.$$

Using the polynomial representation:

$$e_0(x)h_0(x) + e_1(x)h_1(x) = \sum_{i=0}^{r-1} e_i x^i h_0(x) + \sum_{j=r}^{n-1} e_j x^{j-r} h_1(x).$$

For $i \in \{0, ..., r-1\}$, $x^i h_0(x)$ is the polynomial representation for column $\boldsymbol{H}_i$ of $\boldsymbol{H}$ and for $j \in \{r, ..., n-1\}$, $x^{j-r} h_1(x)$ is the polynomial representation for column $\boldsymbol{H}_j$ of $\boldsymbol{H}$.

2.4. **Near codewords.** Near codewords were introduced in the literature of iterative decoding as a way to formalize a collection of error vectors with lower than expected syndrome weight, which are therefor difficult to decode. They are also known under the name "pseudo-codewords" or "trapping sets".

Definition 1 (near codeword of type (s,t))**.** *An error vector $\boldsymbol{e} \in \mathbb{F}_2^n$ is a near codeword of type (s,t) if $t = |\boldsymbol{e}|$ and $s = |\boldsymbol{H}\boldsymbol{e}^T|$.*

MDPC codes are known to have minimum distance which is typically linear in the codelength and are unlikely to have small near codewords. The situation is not the same for QC-MDPC codes. Here the minimum distance is of order $O(\sqrt{n})$ where n is the codelength. This is due to the fact that the word $h_1(x) \oplus h_0(x)$ is clearly a codeword of the code of parity-check matrix $\boldsymbol{H} = (\boldsymbol{h}_0, \boldsymbol{h}_1)$. This leads one to suspect that such codes also have small-weight near codewords. This is indeed the case and that this is most likely the main issue for decoding has been identified in V. Vasseur PhD thesis [Vas21b]. To define them, fix a parity check matrix $\boldsymbol{H} = (\boldsymbol{h}_0, \boldsymbol{h}_1)$.

Definition 2 (The set $\mathcal{N}$ of near codewords)**.** *The set $\mathcal{N}$ of near codewords is the union of all n-bit vectors with polynomial representations of the form $x^i h_0(x) \oplus \mathbf{0}$ and $\mathbf{0} \oplus x^i h_1(x)$, for all $i \in \{0, ..., r-1\}$.*

In this work, when we refer to a near codeword, we will mean an element of $\mathcal{N}$.

In addition to errors in $\mathcal{N}$ being difficult to decode, also errors which have a large overlap with vectors in $\mathcal{N}$ are likely to cause decoding failures. The elements of $\mathcal{N}$ are vectors in $\mathbb{F}_2^n$ which are (d,d)-near codewords in the sense of Definition 1. This can be read off directly from the polynomial representation of the syndrome: the near codeword $e(x) = x^i h_0(x) \oplus 0 \in \mathcal{N}$, for example, has syndrome

$$s(x) = (x^i h_0(x))h_0(x) = x^i h_0^2(x)$$

which is of weight d as $h_0(x)$ has precisely d nonzero coefficients, and in $\mathbb{F}_2[x]/(x^r - 1)$ squaring and multiplication by x preserve the number of nonzero coefficients.

Let $\boldsymbol{\nu} \in \mathcal{N}$ denote a fixed near codeword. For any vector $\boldsymbol{e} \in \mathbb{F}_2^n$ we let $u := |\boldsymbol{e} \star \boldsymbol{\nu}|$, the number of overlapping 1's in $\boldsymbol{e}$ and $\boldsymbol{\nu}$. Our Markov model simulates the decoding behavior of the step by step decoder on input $\boldsymbol{H}, \boldsymbol{s} = \boldsymbol{H}\boldsymbol{e}_{in}^T$. Each iteration of the decoder updates a guess $\boldsymbol{e}_{out}$ which should satisfy $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{e}_{out}^T$, and uses the value

$$\boldsymbol{H}\boldsymbol{e}_{in}^T - \boldsymbol{H}\boldsymbol{e}_{out}^T = \boldsymbol{H}(\boldsymbol{e}_{in} - \boldsymbol{e}_{out})^T.$$

By abuse of notation, we denote this value by $\boldsymbol{s}$ as well. When the decoder successfully decodes, it achieves $\boldsymbol{H}\boldsymbol{e}_{in}^T = \boldsymbol{H}\boldsymbol{e}_{out}^T$ and $\boldsymbol{s} = \mathbf{0}$.

During the decoding process, we let $\boldsymbol{e} := \boldsymbol{e}_{in} - \boldsymbol{e}_{out}$, $\boldsymbol{s} := \boldsymbol{H}\boldsymbol{e}^T$, $s = |\boldsymbol{s}|$, and $t := |\boldsymbol{e}|$. Define $\sigma_j = |\boldsymbol{H}_j \star \boldsymbol{s}|$, where $\boldsymbol{H}_j$ denotes the jth column of $\boldsymbol{H}$. The σ_j's are *counters* which keep track of which bits would be good to flip at a particular stage of the decoding process: the larger the value of σ_j, the more likely the bit j is a good bit to be flipped.

2.5. **Forced Cancellations.** Represent $\boldsymbol{H} = (\boldsymbol{h}_0, \boldsymbol{h}_1) = (h_0(x) = \sum_{j=0}^{r-1} h_{j,0}x^j, h_1(x) = \sum_{j=0}^{r-1} h_{j,1}x^j)$. Fix an error $\boldsymbol{e}$ with polynomial representation $e(x) = e_0(x) \oplus e_1(x)$. The syndrome is $s(x) = h_0(x) \cdot e_0(x) + h_1(x) \cdot e_1(x) \in \mathbb{F}_2[x]/(x^r - 1)$. As in Section 2.4, we let $\boldsymbol{\nu}$ denote an element of $\mathcal{N}$. Without loss of generality, we assume the nonzero entries of $\boldsymbol{\nu}$ are in the first block, so the polynomial representation of $\boldsymbol{\nu}$ is $\boldsymbol{\nu}(x) = x^k h_0(x)$. Let u denote $|\boldsymbol{\nu} \star \boldsymbol{e}|$, the number of nonzero entries shared by $\boldsymbol{e}$ and $\boldsymbol{\nu}$. The larger the value of u, the lower the expected syndrome weight: This is due to an effect we call *forced cancellations*.

The forced cancellations are terms with 0 coefficients in the products involved in the syndrome computation, caused overlaps between $e(x)$ and $\boldsymbol{\nu}(x)$. Since $\boldsymbol{e}$ has u overlaps with $\boldsymbol{\nu}$, we can decompose $e_0(x)$ into the parts which overlap with $\boldsymbol{\nu}(x)$ and the parts which do not overlap with $\boldsymbol{\nu}$:

$$e_0(x) = (x^k h_0 \cap e_0(x)) + e_{0,\sim\boldsymbol{\nu}}(x), \quad \text{where } |(x^k h_0 \cap e_0(x))| = u \text{ and } |e_{0,\sim\boldsymbol{\nu}}(x)| = t - u.$$

When we compute the syndrome of e, we find $\binom{u}{2}$ canceling pairs of terms coming from the product $h_0(x) \cdot (x^k h_0 \cap e_0(x))$:

$$\begin{aligned} s(x) &= e_0(x)h_0(x) + h_1(x)e_1(x) \\ &= h_0(x) \cdot ((x^k h_0 \cap e_0(x)) + e_{0,\sim\boldsymbol{\nu}}(x)) + h_1(x) \cdot e_1(x). \end{aligned} \tag{2}$$

The non-diagonal terms in the product $h_0(x) \cdot (x^k h_0 \cap e_0(x))$ have coefficients of 2, meaning the terms cancel. This is a property of the finite field $\mathbb{F}_2 : (a+b)^2 = a^2 + b^2$. There are $\binom{u}{2}$ such non-diagonal terms in the product (2), leading to the forced cancellations. Other cancellations may occur by chance, but we call the cancellations described here *forced cancellations*. When $u > 1$, there are $\binom{u}{2}$ forced cancellations, yielding a syndrome weight of at most $|e^0(x)| \cdot |h^0(x)| + |e^1(x)| \cdot |h^1(x)| - 2 \cdot \binom{u}{2}$. This is a smaller maximum syndrome weight than one would expect for a generic weight-$|\boldsymbol{e}|$ error vector, which makes it difficult for the iterative decoder to recognize and flip these bits.

Definition 3. *Given a fixed error vector $\boldsymbol{e}$ and a fixed near codeword $\boldsymbol{\nu}$, we classify the n error vector bits into the four categories:*

- u bad bits: *the u bits belonging to the supports of both $\boldsymbol{e}$ and $\boldsymbol{\nu}$;*
- $d - u$ suspicious bits: *the other $d - u$ bits of $\boldsymbol{\nu}$ (not in the support of $\boldsymbol{e}$);*
- $t - u$ normal bits in error: *the other $t - u$ bits of $\boldsymbol{e}$ (not in the support of $\boldsymbol{\nu}$);*
- $n - d - t + u$ *good bits: neither in $\boldsymbol{\nu}$ nor $\boldsymbol{e}$.*

2.6. **Tanner graphs.** A Tanner graph is a very handy tool for analyzing iterative decoding of LDPC or MDPC codes. For LDPC codes, this notion dates back to Gallager who explained and studied his iterative decoding algorithms [Gal63] by using them. In a more

general form, they have been defined in [Tan81]. It is a graph which represents the parity-check matrix of a code. For an $r \times n$ parity-check matrix $\boldsymbol{H} = (H_{i,j})_{\substack{0 \leqslant i < r \\ 0 \leqslant j < n}}$ it is a bipartite graph with two types of vertices, the *variable nodes* which are in bijection with the code positions $\{0, \cdots, n-1\}$, and the *check nodes* which are in bijection with the parity-checks, *i.e.* the rows of $\boldsymbol{H}$. There is an edge between a variable node associated to the code position j and the parity-check associated to the i-th row of $\boldsymbol{H}$ if and only if $H_{i,j} = 1$.

In our case, where $n = 2r$ and the parity-check matrix is formed by two circulant blocks, we can connect the polynomial notation established in Section 2.3 to Tanner graphs. More specifically, the variable node associated to the i-th code position is indexed by $x^i \oplus 0$ if $i < r$ and by $0 \oplus x^{i-r}$ if $i \geqslant r$. The polynomial expression $\lambda(x) \oplus \rho(x)$ represents the set of code positions $\{i_1, \cdots, i_\ell\}$ where $\lambda(x) = \sum_{i_\ell < r} x^{i_\ell}$ and $\rho(x) = \sum_{i_\ell \geqslant r} x^{i_\ell - r}$. Polynomial notation is also used to index the check nodes. The check node associated to the i-th row of $\boldsymbol{H}$ will be indexed by x^i. If $h_0(x), h_1(x)$ are the polynomials representing $\boldsymbol{H} = (\boldsymbol{h}_0, \boldsymbol{h}_1)$, the set of check nodes adjacent to the variable node $x^i \oplus 0$ is the set of monomials in $x^i h_0(x)$ and the set of check nodes adjacent to the variable node $0 \oplus x^i$ is the set of monomials appearing in $x^i h_1(x)$.

The subgraph $\mathscr{G}$ of the Tanner graph induced by a near codeword in $\mathcal{N}$ has a very special structure. For example, let $\boldsymbol{\nu} = h_0(x) \oplus 0$. The syndrome of $\boldsymbol{\nu}$ is $h_0^2(x) = \sum_{i=1}^{d} x^{2l_i}$ where $h_0(x) = \sum_{i=1}^{d} x^{l_i}$. The subgraph of the Tanner graph induced by $\boldsymbol{\nu}$ is formed by d variable nodes labeled $x^{l_i} \oplus 0$ for $i \in \{1, \cdots, d\}$, d parity check nodes labeled $x^{2l_1}, \cdots, x^{2l_d}$ where the variable node $x^{l_i} \oplus 0$ is adjacent to the parity-check node x^{2l_i}. These parity nodes are necessarily of odd degree (often 1 for "typical" keys) where equalities of the form $2l_i = l_j + l_k$ with j and k different from i, give rise to two additional edges linking the parity-check labeled x^{2l_i} to the variable nodes labeled $x^{l_j} \oplus 0$ and $x^{l_k} \oplus 0$ respectively. We have other parity-check nodes that are of even degree which are labeled by $x^{l_j + l_k}$ with $j \neq k$ and $l_j + l_k \neq 2l_i$ for any $i \in \{1, \cdots, d\}$. There are at most $\binom{d}{2}$ parity-checks of this kind. The peculiarity of this near codeword is that it is at the same time remarkably small and of very small syndrome (formed by the d parity-checks of the first group). This is due to the forced cancellations (Section 2.5) which explain that the check nodes $x^{l_i + l_j}$ (with $l_i \neq l_j$) do not contribute to the syndrome of the near codeword. This can be seen on the subgraph $\mathscr{G}$ because these nodes have even degree. Figure 2 illustrates this discussion.

An error which is equal to the support of a near codeword in $\mathcal{N}$ cannot be corrected by standard iterative decoders taking hard decisions: it is readily verified that no bit flipping could reduce the syndrome weight. But even errors which cover a non-negligible part of such a near codeword are problematic. Say that u positions among such a near codeword are in error. Let π_1 be the probability that a parity check of a position in error is not satisfied. Let π_0 denote the probability that a parity check of a position not in error is not satisfied. In essence, (hard decision) iterative decoding of LDPC/MDPC codes relies on the fact that $\pi_1 > \pi_0$. The counters of positions in error are generally higher than the counters of positions which are not in error. We expect that the counter of a position which is in error to be about $d\pi_1$ whereas the counter of a position which is not in error is expected to be about $d\pi_0$. Now let us look at the u positions which are in error in the near codeword: Their

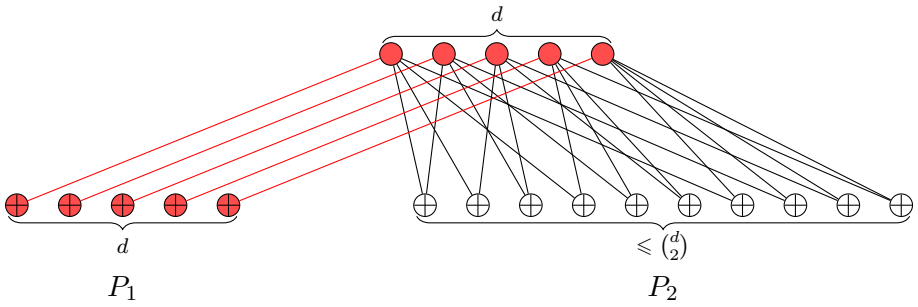


FIGURE 2. Illustration of the subgraph $\mathscr{G}$ induced by the near codeword $\boldsymbol{\nu}$.

counter is expected to be about $\pi_1+(u-1)\pi_0+(d-u)\pi_1 = (u-1)\pi_0+(d-u+1)\pi_1$ which is typically much less than $d\pi_1$. This is due to the fact that there are $(u-1)$ parity-checks adjacent to such a bit which are adjacent to another position which is in error, canceling the contribution of the error to this position (and these parity-checks behave roughly as a parity-check adjacent to a position which is not in error). These positions are therefore less likely to be corrected by the iterative decoding process than the other positions which are in error. But the situation is even worse for the $d-u$ positions which belong to the support of the near codeword, but which are not in error (i.e., the suspicious bits). For those positions a similar reasoning now conduces to model such a counter by the sum of $d-u$ Bernoulli random variables of parameter π_0 and u Bernoulli variables of parameter π_1. We therefore expect that the counter of such positions to be about $(d-u)\pi_0+u\pi_1$. Those positions are therefore more likely to be wrongly flipped than the other positions which are not in error. This is a big problem, since flipping such positions will only worsen what happens in the next iteration. This is typically what is observed in a failed decoding in the error floor region. The first iteration has a few positions of a near codeword in error and bad decisions in subsequent iterations end up with the whole near-codeword in error.

3. Convergence to a codeword in $\mathcal{N}$ in the error floor region

To verify the convergence to a near codeword in the error floor region we have tested parameters for which we can experimentally observe the error floor behavior. We have chosen a random QC-MDPC code with $r = 1723$, $d = 17$ and tested two decoding algorithms:

(1) The majority step by step decoder. The threshold is $\frac{d+1}{2} = 9$. It corresponds to flipping a bit every time this decreases the syndrome weight. For this decoder, we expect to hit the error floor faster than for other decoders because it has the most chance to flip suspicious bits, thus getting the snowball effect described in Section 2.6.
(2) An improved step by step decoder which uses a better (more conservative) decoding threshold. It starts with a high threshold value T_0 which favors to flip bits which are more likely to be in error. It is only when no bit can be flipped that a threshold

$T_1 \leqslant T_0$ is used. The values are given below.

$$T_0 = \begin{cases} \alpha \cdot s + \beta & \text{for } \alpha \cdot s + \beta \in \left[\frac{d+1}{2}, d\right] \\ \frac{d+1}{2} & \text{for } \alpha \cdot s + \beta < \frac{d+1}{2} \\ d & \text{otherwise,} \end{cases}$$

(3)

$$\text{where } \alpha \stackrel{\text{def}}{=} 0.006016213884791455$$

$$\beta \stackrel{\text{def}}{=} 8.797325112097532.$$

$$T_1 = \frac{d+1}{2}.$$

Figure 3 shows the experimental results obtained for both decoders. It is a striking illustration of the phenomenon of convergence to a near codeword in the error floor regime for both decoders. The curves corresponding to the "better threshold" decoder correspond to a threshold pair (T_0, T_1) given in (3). Each point corresponds to at least 200 decoding failures. Together with the experimental curve we have drawn two other curves: "Contrib. of ncw" is the contribution to the DFR coming solely from the convergence to a near codeword. This contribution dominates in the error floor regime since the DFR is almost identical to this curve in this region. This can also be seen by removing from the DFR the contribution from the near codewords. These are the curves referred to as "Contr. of other". These curves display both a waterfall phenomenon and this shows that as we move further in the error floor region, the DFR contribution coming from convergence to a near codeword dominates the DFR. Both decoders display this phenomenon.

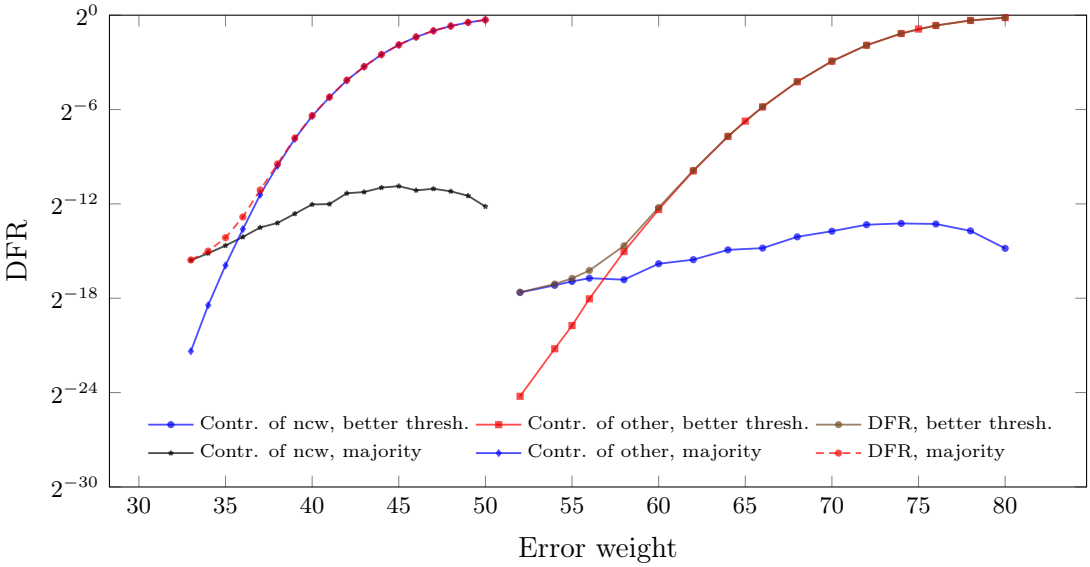


FIGURE 3. DFR vs. error weight ($r = 1723$, $d = 17$), experimental curves.

4. A Simple Markovian Model

4.1. **The simple model.** The aforementioned convergence to a near codeword in the decoding process in the error floor regime strongly suggests to track this phenomenon in the definition of the Markov state. We define the Markov state as the triple (s, t, u), where s is the syndrome weight, t is the error weight, and u is the size of the intersection of the support of the error with a *fixed, random near codeword* $\boldsymbol{\nu}$. To amplify this effect by the size of the set of near codewords, we post-process the DFR obtained by this model together with a DFR obtained from a variant model which does not take into account the effect of near codewords. The equations used to combine these DFRs are discussed in Section 4.5. This process is in contrast to the models used in subsequent sections (Sections 5 and 6), where only the effect of the closest near codeword is taken into account, and this remains the only near codeword which is assumed to have an effect on the iterative decoding process.

Model 1 (M1)**.** *Let e be an error of weight t with syndrome of weight s, and let $\boldsymbol{\nu}$ be a fixed random near codeword. Let $u := |e \star \boldsymbol{\nu}|$. This model follows the values (s, t, u) through the decoding process using a step by step decoder.*

Model 2 (M0)**.** *Let e be an error of weight t with syndrome of weight s. This model follows the same values as Model 1, but the value u is set to 0 in the initial vector, so this model does not take into account the effect of a near codeword.*

Our model, which combines Models 1 and 2 according to the formulae given in Section 4.5, is bound to see an error floor behavior: when the number of overlaps between an error vector and the near codeword becomes as large as possible (i.e., as u approaches d), decoding fails, and the probability that $u = d$ at the beginning of decoding is lower bounded by $\frac{t^d}{n^d}$ where t is the initial error weight. In the waterfall region, the DFR decays more than exponentially in n when the other parameters are fixed and therefore this $\frac{t^d}{n^d}$ term is going to dominate and provoke an error floor phenomenon.

The discussion given in Subsection 2.6 strongly suggests the following model for computing the transition probabilities of the Markov chain:

(1) We distinguish four kind of bits: bad, suspicious, normal, and good (Definition 3).
(2) We then use the approach brought forward to estimate the probabilities π_0 and π_1 to take into account all three of s, the syndrome weight, t the error weight and u the size of the intersection of the error with the near codeword. Here π_0 is as above, the probability that a parity-check adjacent to a given bit which is a good bit is not satisfied. π_1 is the probability that a parity-check adjacent to a given bit which is a normal bit in error is not satisfied.
(3) The transition probabilities are obtained by Counter Model 1 for the counters used in Markov Models 1 and 2.

Counter Model 1.

- *the counter of a bad bit is modeled by the sum of two independent binomial random variables* $\mathrm{Bin}(u-1, \pi_0) + \mathrm{Bin}(d-u+1, \pi_1)$

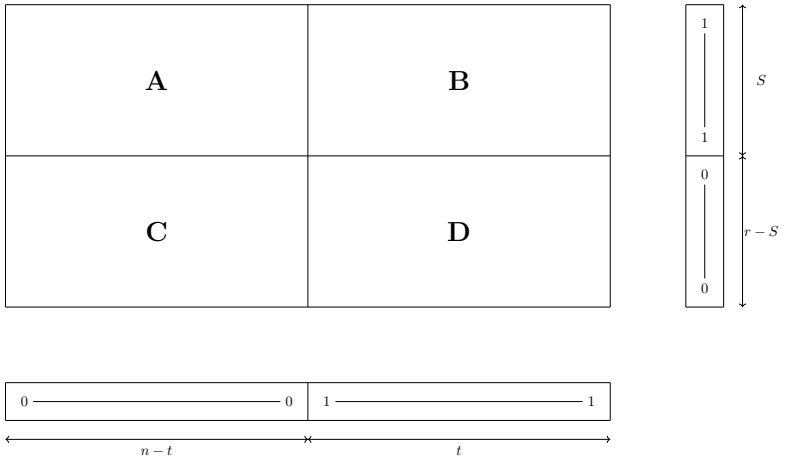


FIGURE 4. $\mathbf{H}'$ matrix, obtained by permuting rows and columns of $\mathbf{H}$. Below, we have $\boldsymbol{e}'$ from rearranging the entries of $\boldsymbol{e}$, and likewise $\boldsymbol{s}'$ to the right, computed from $\mathbf{H}'\boldsymbol{e}'^{\mathsf{T}}$.

- *the counter of a suspicious bit is the sum of two independent random variables* $\mathrm{Bin}(d-u,\pi_0)+\mathrm{Bin}(u,\pi_1)$
- *the counter of a normal bit in error is modeled by the binomial random variable* $\mathrm{Bin}(d,\pi_1)$*;*
- *the counter of a good bit is modeled by the binomial random variable* $\mathrm{Bin}(d,\pi_0)$*.*

To compute the relevant probabilities we will use the following probabilistic model for the parity-check matrix $\boldsymbol{H}$.

Assumption 1. *The columns of* $\boldsymbol{H}$ *are drawn uniformly at random with fixed column weight* d*.*

4.2. **Permutations of the parity check matrix.** For the purpose of the discussion of this model, we present the following permutations of the parity check matrix $\boldsymbol{H}$. Fix a parity check matrix $\boldsymbol{H}\in\mathbb{F}_2^{r\times n}$ Apply permutations $\mathbf{P}\in\mathbb{F}_2^{r\times r},\mathbf{Q}\in\mathbb{F}_2^{n\times n}$ to $\boldsymbol{H},\boldsymbol{e}$ and $\boldsymbol{s}$ so that $\boldsymbol{H}'=\mathbf{P}\boldsymbol{H}\mathbf{Q}^{-1}$, $\boldsymbol{e}'=\boldsymbol{e}\mathbf{Q}^{\mathsf{T}}$, and $\boldsymbol{s}'^{\mathsf{T}}=\mathbf{P}\boldsymbol{s}^{\mathsf{T}}$ corresponding to Figure 4. Let $S:=|\boldsymbol{s}|$. When $|\mathbf{B}|=S$, there is exactly one error per unsatisfied parity-check equation. This is the ideal case for error correction as flipping a bit in $\boldsymbol{e}$ from $1\to 0$ will not affect the other counters.

Let $X=\sum_{j\in e}|\boldsymbol{h}_j^{\mathsf{T}}\star\boldsymbol{s}|-S$. Note that $X=0$ corresponds to the ideal case above.

Construct the matrix $\mathbf{H}'_{\text{eff}}$ by permuting columns of $\boldsymbol{H}'$ according to the support of $\boldsymbol{\nu}$ as pictured in Figure 5. Since $\boldsymbol{H}'$ has been permuted, we likewise have permuted $\boldsymbol{\nu}$ and $\boldsymbol{e}'$, which have u nonzero overlaps in the $\mathbf{B}$ and $\mathbf{D}$ sections of $\mathbf{H}'$. In particular, there will be u columns in the $\mathbf{B},\mathbf{D}$ sections of $\mathbf{H}'$. In the rows of these columns, we will find the pairs of 1's which are forced to cancel by the algebraic structure of the circulant blocks described in Section 2.5. We flip these pairs from 1's to 0's. A total of $\binom{u}{2}$ pairs of 1's will be flipped:

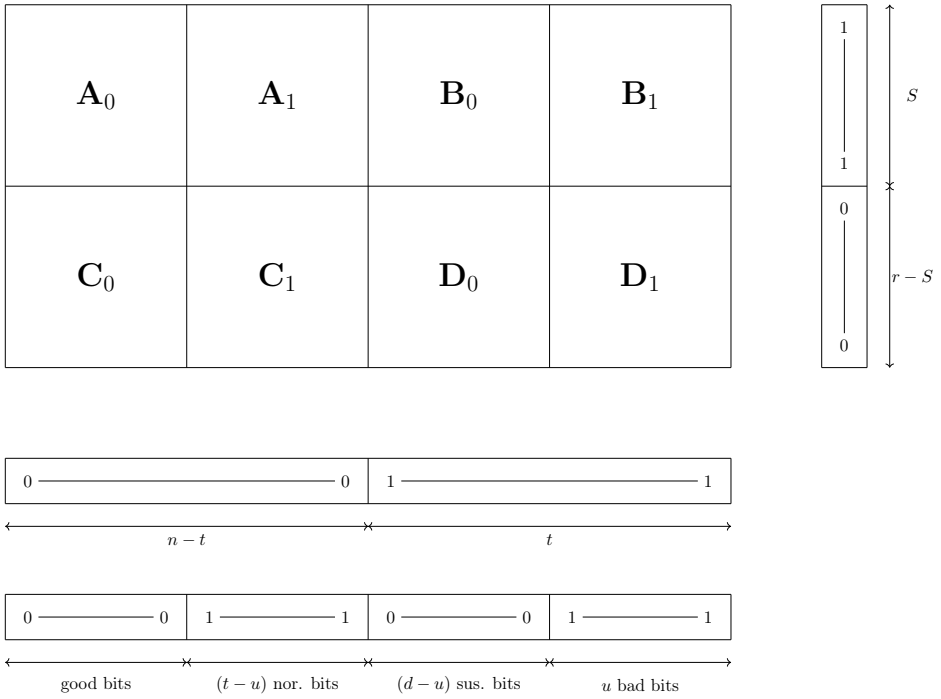


FIGURE 5. $\mathbf{H}_{\text{eff}}$ matrix, obtained by permuting rows and columns of $\mathbf{H}$ according to the intersection of $\boldsymbol{e}$ and $\boldsymbol{\nu}$.

Together, blocks $\mathbf{B}$ and $\mathbf{D}$ of $\mathbf{H}'$ have weight $d \cdot t$. After the cancellations, $\mathbf{H}'_{\text{eff}}$ will have blocks $\mathbf{B}_{\text{eff}}$ and $\mathbf{D}_{\text{eff}}$ which will be weight $S_{\max} := d \cdot t - 2\binom{u}{2}$.

4.3. **Assumptions.** The forced cancellations are a result of the quasi-cyclic structure of $\boldsymbol{H}$. For the remainder of this work, we make the heuristic assumption that the columns of $\boldsymbol{H}$ are drawn uniformly at random up to fixed column weight d.

Removing the forced cancellations from $\boldsymbol{H}'$, resulting in $\mathbf{H}'_{\text{eff}}$, allows us to use random variables to describe each column of $\mathbf{H}'_{\text{eff}}$.

Proposition 1. *Assuming that the random variables that indicate the number of errors $|\boldsymbol{h}_i^{\intercal} \star \boldsymbol{e}|$ for a given equation i are independent and if $\boldsymbol{e} \in \mathcal{E}_t$, the expectation of X_{eff} knowing S, $t = |\boldsymbol{e}|$, and $u = |\boldsymbol{e} \cap \boldsymbol{\nu}|$ is*

$$E[X_{\mathit{eff}}|S,t,u] = S \cdot \frac{\sum_l 2l\rho_{2l+1}}{\sum_l \rho_{2l+1}} \tag{4}$$

where

$$\rho_l = \sum_{j=0}^{l} \binom{t-u}{j}\binom{u}{l-j}\left(\frac{d}{r}\right)^j \left(\frac{r-d}{r}\right)^{t-u-j} \left(\frac{d+1-u}{r}\right)^{l-j} \left(\frac{r-d-1+u}{r}\right)^{u-l+j} \tag{5}$$

denotes the probability that any row in $\mathbf{H}'_{eff}$ *restricted to the support of* $\boldsymbol{e}$ *has weight* l*.*

Proof. For any row $i \in \{0, \ldots, r-1\}$ in $\mathbf{H}'_{\text{eff}}$, consider the t columns corresponding to the support of $\boldsymbol{e}$. There are $t-u$ columns corresponding to $\text{supp}(\boldsymbol{e}) \setminus \text{supp}(\boldsymbol{\nu})$ and u corresponding to $\text{supp}(\boldsymbol{e}) \cap \text{supp}(\boldsymbol{\nu})$ (which correspond to blocks $B_0 \cup D_0$ and $B_1 \cup D_1$ of Figure 5, respectively).

All columns in $\text{supp}(\boldsymbol{e}) \setminus \text{supp}(\boldsymbol{\nu})$ have weight d and all columns in $\text{supp}(\boldsymbol{e}) \cap \text{supp}(\boldsymbol{\nu})$ have weight $d+1-u$ due to forced cancellations. Any entry in row i on $\text{supp}(e) \setminus \text{supp}(\boldsymbol{\nu})$ thus has probability $\frac{d}{r}$ of being nonzero and probability $\frac{r-d}{r}$ of being zero. Any entry in row i on $\text{supp}(\boldsymbol{e}) \cap \text{supp}(\boldsymbol{\nu})$ has probability $\frac{d+1-u}{r}$ of being nonzero and probability $\frac{r-d-1+u}{r}$ of being zero.

Thus the probability that any row in $\mathbf{H}'_{\text{eff}}$ has weight l is given by Equation (5). □

Note that X_{eff} is equivalent to $|B_0| + |B_1| - |s|$. By using the probabilistic notation of Section 2 we will assume that

Assumption 2. *The counters* σ_j *follow binomial distributions according to Counter Model 1:*

$$\sigma_j \sim \begin{cases} \text{Bin}(d, \pi_0) & \text{if } j \text{ is a good bit} \\ \text{Bin}(d, \pi_1) & \text{if } j \text{ is a normal bit} \\ \text{Bin}(d-u, \pi_0) + \text{Bin}(u, \pi_1) & \text{if } j \text{ is a suspicious bit} \\ \text{Bin}(d-u+1, \pi_1) + \text{Bin}(u-1, \pi_0) & \text{if } j \text{ is a bad bit} \end{cases} \tag{6}$$

where

$$\pi_0 = \frac{S}{r}, \quad \pi_1 = \frac{S + \xi E[X_{eff}|S,t,u]}{S_{max}}, \quad S_{max} = d \cdot t - 2\binom{u}{2}$$

for some constant ξ*.*

Remark 1. *We note that the probability estimate* π_0 *in Assumption 2 is particularly pessimistic. Indeed, it is the probability that a random parity check is not satisfied, which is certainly an upper bound for the probability that a parity check corresponding to a bit which is not in error is unsatisfied. Models in Sections 5 and 6 use a more refined estimate of the probability* π_0 *that a parity check corresponding to a bit which is not in error is unsatisfied.*

Remark 2. *The constant* ξ *in Assumption 2 is inherited from a refinement given in* [Vas21a] *of the previous Markov model* [SV19b]*. Reducing the value of* ξ *from 1 was intended to compensate for Non-Markovian effects relative to the state information* (S,t) *used in that model, which would otherwise produce an overly optimistic DFR. The experimental data in Figure 7 assumes the same value* $\xi = 0.955$ *used by* [Vas21a]*, which further contributes to the pessimism of Assumption 2 as reflected in our experimental data. There is no analogous factor to* ξ *used in the models in Sections 5 and 6.*

Assumption 3. *We assume that the step by step bit flipping decoder is a time homogeneous Markov chain, i.e., for all* $i \geqslant 1$*, we have:*

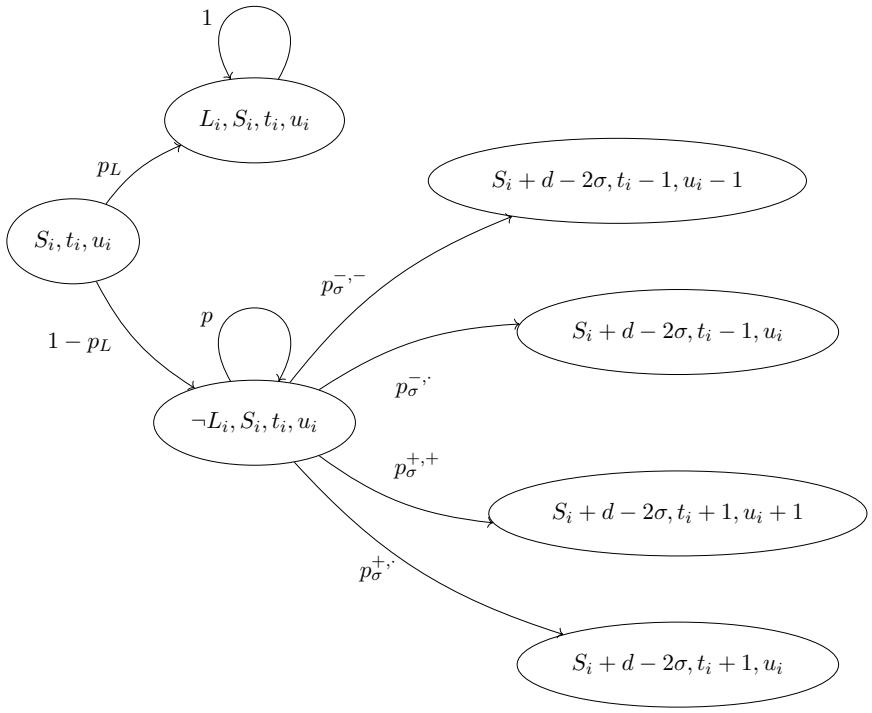


FIGURE 6. Transition diagram at the i-th step, starting with syndrome weight S_i, error weight t_i, and intersection u_i with a (d, d)-near codeword $\boldsymbol{\nu}$. Edges are transition probabilities calculated in Section 4.4.

$$\Pr[(S_{i+1}, t_{i+1}, u_{i+1}) = (a_{i+1}, b_{i+1}, c_{i+1}) \big| \neg L_i, (S_i, t_i, u_i) = (a_i, b_i, c_i), \ldots, \neg L_0, (S_0, t_0, u_0))$$
$$= \Pr[(S_{i+1}, t_{i+1}, u_{i+1}) = (a_{i+1}, b_{i+1}, c_{i+1}) \big| \neg L_i, (S_i, t_i, u_i) = (a_i, b_i, c_i)],$$

where $L_i := \{\sigma_j^{(i)} < T \ \ \forall j\}$ *corresponds to the blocked state at the* i*-th iteration.*

Markov end states. There are two possible Markov end states that dominate the experiment findings: $(0, 0, 0)$ when we have successful decoding or (d, d, d) when the state converges to a (d, d)-near codeword.

4.4. **Transition probabilities.** Consider the transition probabilities as presented in Figure 6. At the i-th iteration, suppose we have the state (S_i, t_i, u_i) and threshold T.

(1) For all positions $j \in \{0, 1, \ldots, 2r - 1\}$, we have $\sigma_j < T$. The decoder is in a blocked state (see Assumption 3) and will not flip any more bits. This happens with probability p_L.
(2) If there exists $j \in \{0, 1, \ldots, 2r-1\}$ such that $\sigma_j \geqslant T$, this happens with probability $1 - p_L$. Randomly sample position j. There are five possible outcomes:

(a) if $\sigma_j < T$, then $(S,t,u) \to (S,t,u)$ with probability p,
(b) if $\sigma_j \geqslant T, j \in e, j \in \boldsymbol{\nu}$, then $(S,t,u) \to (S+d-2\sigma, t-1, u-1)$ with probability $p_\sigma^{-,-}$,
(c) if $\sigma_j \geqslant T, j \in e, j \notin \boldsymbol{\nu}$, then $(S,t,u) \to (S+d-2\sigma, t-1, u)$ with probability $p_\sigma^{-,\cdot}$,
(d) if $\sigma_j \geqslant T, j \notin e, j \in \boldsymbol{\nu}$, then $(S,t,u) \to (S+d-2\sigma, t+1, u+1)$ with probability $p_\sigma^{+,+}$,
(e) if $\sigma_j \geqslant T, j \notin e, j \notin \boldsymbol{\nu}$, then $(S,t,u) \to (S+d-2\sigma, t+1, u)$ with probability $p_\sigma^{+,\cdot}$.

The superscript in the notation for the probabilities $p_\sigma^{-,-}, p_\sigma^{-,\cdot}, p_\sigma^{+,+}, p_\sigma^{+,\cdot}$ indicates the change to the values of t and u, respectively.

Counter probabilities. For $\sigma \in \{0, \ldots, d\}$, we write:

$$
\begin{aligned}
f_{S,t}^{11} &= \Pr[|h_j * s| = \sigma \big| |e| = t, |s| = S, j \in e, j \in \boldsymbol{\nu}] \\
&= \sum_{i=0}^{\sigma} \binom{d+1-u}{i} \pi_1^i (1-\pi_1)^{(d+1-u)-i} \binom{u-1}{\sigma-i} \pi_0^{\sigma-i} (1-\pi_0)^{(u-1)-(\sigma-i)}, \\
f_{S,t}^{10} &= \Pr[|h_j * s| = \sigma \big| |e| = t, |s| = S, j \in e, j \notin \boldsymbol{\nu}] \\
&= \binom{d}{\sigma} \pi_1^\sigma (1-\pi_1)^{d-\sigma}, \\
f_{S,t}^{01} &= \Pr[|h_j * s| = \sigma \big| |e| = t, |s| = S, j \notin e, j \in \boldsymbol{\nu}] \\
&= \sum_{i=0}^{\sigma} \binom{d-u}{i} \pi_0^i (1-\pi_0)^{(d-u)-i} \binom{u}{\sigma-i} \pi_1^{\sigma-i} (1-\pi_1)^{u-(\sigma-i)}, \\
f_{S,t}^{00} &= \Pr[|h_j * s| = \sigma \big| |e| = t, |s| = S, j \notin e, j \notin \boldsymbol{\nu}] \\
&= \binom{d}{\sigma} \pi_0^\sigma (1-\pi_0)^{d-\sigma}.
\end{aligned}
$$

To give an indication on how one might obtain such formulas, recall that if $X \sim \mathrm{Bin}(n,p), Y \sim \mathrm{Bin}(m,q)$ and $0 \leqslant k \leqslant n+m$, we have:

$$\Pr(X+Y=k) = \sum_{i=0}^{k} \Pr(X=i) \Pr(Y=k-i)$$

Using the probability mass function for sum of binomial distributions in Model 1, we obtain the formulae above.

Blocked state. The blocked state L_i happens when all the counters are below the threshold $\sigma_j < T$. This is given by the probability:

$$
\begin{aligned}
p_L &= \Pr[(S,t,u) \to (L,S,t,u)] \\
&= \Pr[\sigma_j < T \ \ \forall j | S,t,u] \\
&= \Big(\sum_{\sigma<T} f^{10}_{S,t}(\sigma)\Big)^{t-u}\Big(\sum_{\sigma<T} f^{11}_{S,t}(\sigma)\Big)^{u}\Big(\sum_{\sigma<T} f^{01}_{S,t}(\sigma)\Big)^{d-u}\Big(\sum_{\sigma<T} f^{00}_{S,t}(\sigma)\Big)^{(n-t)-(d-u)}.
\end{aligned}
$$

The formula is divided into two parts: one when we consider $j \in \boldsymbol{e}$ and the other $j \notin \boldsymbol{e}$. The arguments for both parts are identical. In the first part, $|\boldsymbol{e}| = t$ and there are two subcases $j \in \boldsymbol{\nu}$ or $j \notin \boldsymbol{\nu}$. The individual counter probabilities $f^{10}_{S,t}(\sigma)$ and $f^{11}_{S,t}(\sigma)$ computes the probability that the j-th counter is equal to σ with $j \in \boldsymbol{\nu}$ and $j \notin \boldsymbol{\nu}$ respectively. We sum all the individual probabilities up to the threshold T and take products in order to calculate the probability the blocked state.

Transition probabilities. Write $j \xleftarrow{\$} \{0, \dots 2r-1\}$ as random sampling j from $\{0, \dots 2r-1\}$ with uniform probability. Then we have the following transition probabilities:

$$
\begin{aligned}
p^{-,-}_{\sigma} &= \Pr[j \xleftarrow{\$} \{0,\dots,2r-1\}, j \in e, j \in \boldsymbol{\nu}, |h_j * s| = \sigma] \\
&= \frac{u}{2r} f^{11}_{S,t}(\sigma), \\
p^{-,\cdot}_{\sigma} &= \Pr[j \xleftarrow{\$} \{0,\dots,2r-1\}, j \in e, j \notin \boldsymbol{\nu}, |h_j * s| = \sigma] \\
&= \frac{t-u}{2r} f^{10}_{S,t}(\sigma), \\
p^{+,+}_{\sigma} &= \Pr[j \xleftarrow{\$} \{0,\dots,2r-1\}, j \notin e, j \in \boldsymbol{\nu}, |h_j * s| = \sigma] \\
&= \frac{d-u}{2r} f^{01}_{S,t}(\sigma), \\
p^{+,\cdot}_{\sigma} &= \Pr[j \xleftarrow{\$} \{0,\dots,2r-1\}, j \notin e, j \notin \boldsymbol{\nu}, |h_j * s| = \sigma] \\
&= \frac{(2r-t)-(d-u)}{2r} f^{00}_{S,t}(\sigma), \\
p &= \sum_{\sigma<T} \big(p^{-,-}_{\sigma} + p^{-,\cdot}_{\sigma} + p^{+,+}_{\sigma} + p^{+,\cdot}_{\sigma}\big).
\end{aligned}
$$

The last expression is obtained by calculating the probability of j not being flipped in a non-blocked state as a sum of probabilities when the counter is below the threshold.

4.5. **Markov Model Estimation of the DFR.** Our simple Markov model experiments yield the decoding failure rate under the assumptions of models M0 and M1, see Section 4.1. Model M0 (Model 2) assumes no effect of the near codewords $\mathcal{N}$ on the decoding failure rate and model M1(Model 1) records the effect of a single near codeword on the decoding failure rate. We modeled M0 using the same code by setting the value of u to 0 in the initial vector.

Let DFR(M0) and DFR(M1) denote the respective decoding failure rates. From these two values, we wish to estimate the true decoding failure rate, which we denote DFR, taking into account the existence of $|\mathcal{N}| = r$ near codewords.

We remark that there are (at least) two potential philosophies for taking into account the impact of $\mathcal{N}$ on the DFR:

(1) The difference $DFR(M1) - DFR(M0)$ accounts for the number of decoding failures due to the existence of one near codeword, so $|\mathcal{N}|(DFR(M1) - DFR(M0))$ gives the increase in DFR due to the existence of $|\mathcal{N}|$ near codewords. This philosophy yields the formula:

$$DFR = DFR(M0) + |\mathcal{N}|(DFR(M1) - DFR(M0)). \tag{7}$$

This assumes an additive effect of the elements of $\mathcal{N}$ on DFR, and we refer to Equation (7) as the *linear model*. See Figure 7.

(2) The proportion of M0-model successes which are still M1-successes will be amplified by each near codeword in $\mathcal{N}$. This philosophy yields the formula:

$$DFR = 1 - (1 - DFR(M0)) \left(\frac{1 - DFR(M1)}{1 - DFR(M0)} \right)^{|\mathcal{N}|}. \tag{8}$$

This assumes a multiplicative effect of the elements of $\mathcal{N}$ on the decoding success rate $(1 - DFR)$, and we refer to Equation (8) as the *multiplicative model*. See Figure 7.

We compare our data with experimental data obtained using [Vas24], as well as a previous Markov model [SV19b] which does not take into account the effect of the error having a large number of overlaps with a near codeword. Our Markov model gives a more conservative estimate for the DFR, in instances where models not taking into account the effect of near codewords underestimate the DFR.

5. A Refined Markovian Model for Perfect Keys

5.1. **Perfect keys.** The simple model in Section 4 has an advantage over the Markov model [SV19b, Vas21b] because it predicts a (conservative) error floor. However, the simplified model of the parity-check matrix 1 to compute the relevant probabilities is too crude and the whole computation does not take into account the key structure. In a further simplifying assumption, we let $\boldsymbol{\nu}$ denote not just any fixed near codeword, but the fixed codeword which maximizes $|\boldsymbol{e} \star \boldsymbol{\nu}|$ at the start of the decoding process. We assume that this $\boldsymbol{\nu}$ is the only near codeword that it is necessary to consider in the decoding process. The DFR may vary depending on the key structure (*i.e.* the particular form of $\boldsymbol{H}$). In this section, we give a more accurate computation of the transition probabilities of the Markov chain in the case where the key is as simple as possible, by assuming that the subgraph $\mathscr{G}$ of the Tanner graph induced by a near codeword is the simplest in some sense. More precisely, we focus on secret keys $(h_0(x), h_1(x))$ which are *perfect keys*:

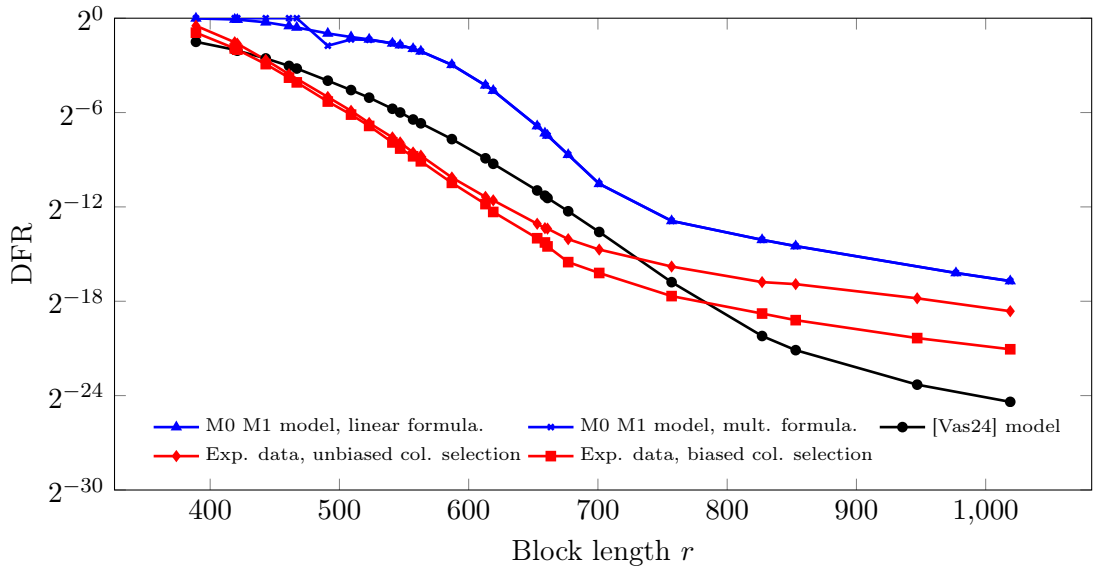


FIGURE 7. Plot of experimental DFR data, M0 M1 linear and multiplicative Markov model DFRs, and the DFR of the Markov model in described in [SV19b].

Definition 4 (perfect key)**.** *The secret key* $(h_0(x), h_1(x))$ *is perfect if and only if all sums* $l_i + l_j$ *for* i *and* j *in* $\{1, \cdots, d\}$ *and* $i \leqslant j$ *are different and so are the sums* $r_i + r_j$ *where* $h_0(x) = \sum_{i=1}^{d} x^{l_i}$ *and* $h_1(x) = \sum_{i=1}^{d} x^{r_i}$*.*

This condition is equivalent to any of the following conditions which should be met for all Tanner graphs induced by any of the near codewords of $\mathcal{N}$

i.) The number of parity-checks in $\mathscr{G}$ is exactly $\frac{d(d+1)}{2}$.
ii.) The number of parity-checks in $\mathscr{G}$ is maximal.
iii.) All parity-checks in $\mathscr{G}$ have either degree 1 or 2.
iv.) There are d parity-checks in $\mathscr{G}$ which have degree 1 and $\frac{d(d-1)}{2}$ parity-checks in $\mathscr{G}$ of degree 2.

This structure is helpful in mitigating the effect of the near codewords. The associated secret keys can probably be considered strong keys although the interplay between the left and the right part has also to be taken into account. In particular, it is very bad to choose $\boldsymbol{h}_0 = \boldsymbol{h}_1$ or take $\boldsymbol{h}_0$ and $\boldsymbol{h}_1$ which have a big intersection. This structure also helps in setting a Markov model which is slightly different from the previous one: the Markov states are now (s, t, u) where s and t are as before the syndrome and error weights, but u is now the maximal size of the intersection of the error with the near codewords in $\mathcal{N}$.

5.2. **A useful decomposition of the parity-checks and the syndrome.**

The Chaulet approach for taking into account the syndrome weight. We first review the Chaulet approach [Cha17, Chap. 7] to take into account the syndrome weight s in modeling the counters which themselves will be instrumental to compute the transition probabilities of the Markov chain. Intuitively we expect these counters to behave roughly as a linear function of s. Sendrier-Vasseur [SV19a] quantifies this intuition. They distinguish the counter of a bit in error σ_i^{err} from the counter of a bit which is correct σ_i^{good}. The first bit is involved in d parity-checks which we know contain an error in a given position, whereas the second one is involved in d parity-checks which we know involve a bit which is error-free in a given position. This biases the respective probabilities π_1 (in the error case) and π_0 (when the bit is correct) that the parity-check is unsatisfied. We will give here a discussion which simplifies [Cha17, Chap. 7] and [Vas21b, §12.3]. We color an edge of the Tanner graph in red if it is adjacent to a parity check which is unsatisfied. Let N^{err} be the number of red edges leaving the bits in error and let e_i be the number of bits that are in error in the parity-check labeled by i. We denote by s_i the syndrome of the parity-check labeled by i. By counting the number of edges leaving the bits in errors and counting the same number of red edges arriving at the unsatisfied parity-checks, we get

$$N^{\text{err}} = \sum_{i:s_i=1} e_i. \tag{9}$$

We have on one hand

$$\mathbb{E}(N^{\text{err}}|s,t) = dt \cdot \pi_1. \tag{10}$$

On the other hand, from (9) we have

$$\mathbb{E}(N^{\text{err}}|s,t) = s\mathbb{E}(e_1|s_1=1). \tag{11}$$

Putting these equations together we relate π_1 to the expected number of errors in a parity-check equation given that this parity-check is violated.

Lemma 1 (double counting lemma)**.**

$$\pi_1 = \frac{s}{dt}\mathbb{E}(e_1|s_1=1). \tag{12}$$

Note that the double counting argument (9) is at the heart of this fundamental equation. This will be the rationale of the more complicated computations we perform later in this section. Let $\rho_\ell \stackrel{\text{def}}{=} \frac{\binom{w}{\ell}\binom{n-w}{t-\ell}}{\binom{n}{t}}$ be the probability that a parity-check involves l bits in error.

$$\mathbb{E}(e_1|s_1=1) = \frac{\sum_\ell (2\ell+1)\rho_{2\ell+1}}{\sum_\ell \rho_{2\ell+1}}.$$

By substituting for $\mathbb{E}(e_1|s_1=1)$ in (12) we finally get

$$\pi_1 = \frac{s}{dt} \cdot \frac{\sum_\ell (2\ell+1)\rho_{2\ell+1}}{\sum_\ell \rho_{2\ell+1}}. \tag{13}$$

To compute π_0 we perform a similar reasoning by bringing in N^{cor} the number of red edges leaving the bits which are correct. We have

$$N^{\text{cor}} = w \cdot s - N^{\text{err}}.$$

By writing $\mathbb{E}(N^{\text{cor}}|s,t) = d(n-t)\cdot\pi_1$ and performing a similar reasoning we obtain

$$\pi_0 = \frac{s\left(w - \frac{\sum_\ell (2\ell+1)\rho_{2\ell+1}}{\sum_\ell \rho_{2\ell+1}}\right)}{d(n-t)}$$

All these computation are rigorous, what is not rigorous is the assumption that the parity-check equations behave independently from each other. This forms what we call Chaulet's model [Cha17, Chap. 7] for the counters.

Model 3 (Chaulet's model)**.**

$$\begin{aligned} \textit{(correct bit)}\ \sigma_i &\sim \text{Bin}(d,\pi_0) \\ \textit{(bit in error)}\ \sigma_i &\sim \text{Bin}(d,\pi_1) \end{aligned}$$

where

$$\begin{aligned} \pi_0 &\overset{def}{=} \frac{s\left(w - \frac{\sum_\ell (2\ell+1)\rho_{2\ell+1}}{\sum_\ell \rho_{2\ell+1}}\right)}{d(n-t)} \\ \pi_1 &\overset{def}{=} \frac{s}{dt}\cdot\frac{\sum_\ell (2\ell+1)\rho_{2\ell+1}}{\sum_\ell \rho_{2\ell+1}} \\ \rho_\ell &\overset{def}{=} \frac{\binom{w}{\ell}\binom{n-w}{t-\ell}}{\binom{n}{t}}. \end{aligned}$$

A useful partition of the parity-checks. The definition of a good/bad/normal/suspicious bit is basically the same as in the previous section with $\boldsymbol{\nu}$ being replaced by the closest near codeword to the error. If there are several near codewords which are as close, we just choose one of them. In order to obtain a probabilistic model for the counters it will be helpful to partition the parity-checks in 6 groups.

- P_1 : The set of parity-checks which are of degree 1 in the subgraph $\mathscr{G}$ induced by the closest near-codeword and which are adjacent to a bad bit.
- P_2 : The set of parity-checks in $\mathscr{G}$ of degree 2 and are adjacent to two bad bits.
- P_3 : The set of parity-checks in $\mathscr{G}$ of degree 2 and are adjacent to one bad bit and one suspicious bit.
- P_4 : The set of parity-checks in $\mathscr{G}$ of degree 2 and are adjacent to two suspicious bits.
- P_5 : The set of parity-checks which are of degree 1 in the subgraph $\mathscr{G}$ and which are induced by the dominant near codeword and which are adjacent to a suspicious bit.
- P_6 : The rest of the parity-checks (namely those that do not belong to $\mathscr{G}$).

In the case of a perfect key:

Fact 1.

$$|P_1| = u,\ |P_2| = \binom{u}{2},\ |P_3| = u(d-u),\ |P_4| = \binom{d-u}{2},\ |P_5| = d-u.$$

We define a red edge in the Tanner graph as an edge connecting a bit to a syndrome bit which is equal to 1. A syndrome bit equal to 1 in the Tanner is represented by a red parity-check equation. Figure 8 depicts this partition of bits and parity-checks.

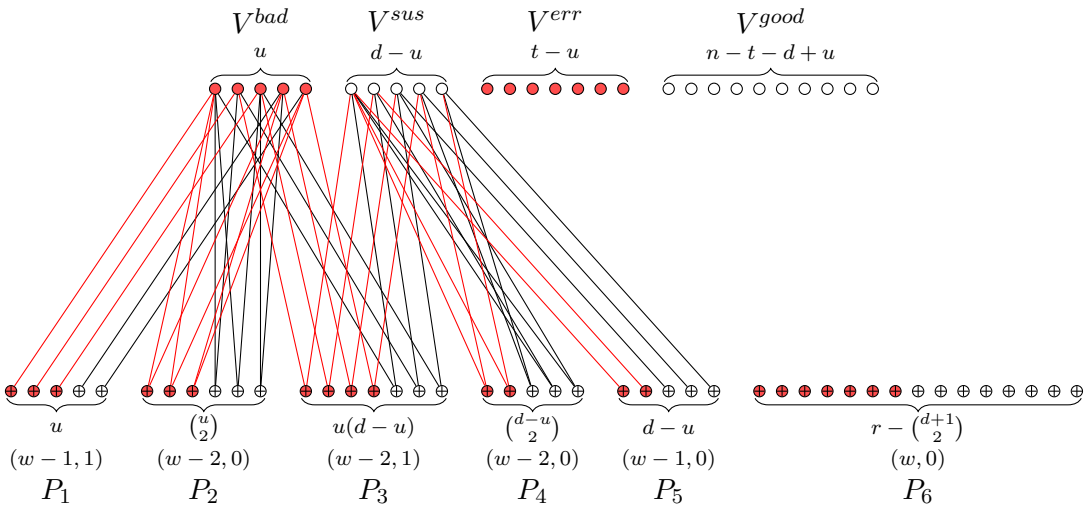


FIGURE 8. Illustration of the different groups. The size of the group is given just below the group and just below the size of the group, there is a pair (a, b) which gives the type of the parity-bit. a gives the number of additional edges arriving at this group coming from the set of bits which do not belong to the Tanner graph $\mathscr{G}$ induced by the closest near codeword, namely $V^{\text{err}} \cup V^{\text{good}}$. b indicates the contribution to the parity-check belonging to the group coming solely from the bits in $\mathscr{G}$. For instance for the first group P_1, the size is u and the corresponding pair is $(w-1, 1)$.

The reason why we decompose the parity-checks in several groups comes from the fact that they behave differently during decoding depending on their type. What we call *type* of a parity-check is the pair (a, b) where a is the number of edges arriving at this parity-check from bits not belonging to $\mathscr{G}$. b is a bit which indicates the contribution to the syndrome of parity-check coming from bits in the subgraph $\mathscr{G}$ of the Tanner graph induced by the closest near codeword. All groups of parity-checks have different types, with the exception of P_2 and P_4 which both have type $(w-2, 0)$. P_2 has type $(w-2, 0)$ because a parity-check belonging to it is adjacent to exactly 2 bits in $\mathscr{G}$ and the contribution to the syndrome is 0 because it is adjacent to 2 bits in error in $\mathscr{G}$. P_4 has the same type because it is adjacent to 2 bits which are correct in $\mathscr{G}$ and their contribution to the syndrome is also 0. We could

have merged P_2 with P_4 but prefer to keep them separate for the ease and clarity of the explanation.

5.2.1. *Decomposing the syndrome.* s_i denotes in all cases the number of unsatisfied parity-checks in group P_i. The syndrome weight s is therefore given by

$$s \quad = \quad s_1 + \cdots + s_6.$$

We also define for i in $\{1, \cdots, 6\}$ $\bar{s_i} = \mathbb{E}(s_i|s,t,u)$. To compute the relevant probabilistic model for our Markov model we will need to estimate the $\bar{s_i}$'s. For this purpose, we will make the following assumption

Assumption 4. *For all i in $\{1, \cdots, 6\}$ we have*

$$\bar{s_i} = s\frac{\mathbb{E}(s_i|t,u)}{\mathbb{E}(s|t,u)}.$$

Those expectations are readily computed

Proposition 2. *We have*

$$\mathbb{E}(s_1|t,u) = u\sum_\ell \rho^{(1)}_{2\ell+1}, \qquad \mathbb{E}(s_2|t,u) = \binom{u}{2}\sum_\ell \rho^{(2)}_{2\ell+1}$$

$$\mathbb{E}(s_3|t,u) = u(d-u)\sum_\ell \rho^{(3)}_{2\ell+1}, \quad \mathbb{E}(s_4|t,u) = \binom{d-u}{2}\sum_\ell \rho^{(4)}_{2\ell+1}$$

$$\mathbb{E}(s_5|t,u) = (d-u)\sum_\ell \rho^{(5)}_{2\ell+1}, \qquad \mathbb{E}(s_6|t,u) = \left(r - \binom{d+1}{2}\right)\sum_\ell \rho^{(6)}_{2\ell+1},$$

where

$$\rho^{(1)}_\ell = \frac{\binom{w-1}{\ell-1}\binom{n-d-w+1}{t-u-\ell+1}}{\binom{n-d}{t-u}}, \quad \rho^{(2)}_\ell = \frac{\binom{w-2}{\ell-2}\binom{n-d-w+2}{t-u-\ell+2}}{\binom{n-d}{t-u}}$$

$$\rho^{(3)} = \frac{\binom{w-2}{\ell-1}\binom{n-d-w+2}{t-u-\ell+1}}{\binom{n-d}{t-u}}, \quad \rho^{(4)}_\ell = \frac{\binom{w-2}{\ell}\binom{n-d-w+2}{t-u-\ell}}{\binom{n-d}{t-u}}$$

$$\rho^{(5)}_\ell = \frac{\binom{w-1}{\ell}\binom{n-d-w+1}{t-u-\ell}}{\binom{n-d}{t-u}}, \quad \rho^{(6)}_\ell = \frac{\binom{w}{\ell}\binom{n-d-w}{t-u-\ell}}{\binom{n-d}{t-u}}.$$

5.3. **The counter models.** We are ready now to give the counter models for the 4 groups of bits

Counter Model 2.

$$\begin{aligned}
\textit{bad bit : } \sigma_i &\sim \mathrm{Bin}(1,\pi_1) + \mathrm{Bin}(u-1,\pi_2) + \mathrm{Bin}(d-u,\pi_3)\\
\textit{suspicious bit : } \sigma_i &\sim \mathrm{Bin}(1,\pi_5) + \mathrm{Bin}(d-u-1,\pi_4) + \mathrm{Bin}(u,\pi_3)\\
\textit{normal bit in error : } \sigma_i &\sim \mathrm{Bin}(d,\pi^{err})\\
\textit{good bit : } \sigma_i &\sim \mathrm{Bin}(d,\pi^{good})
\end{aligned}$$

$$
\begin{aligned}
\pi_i &= \frac{\bar{s_i}}{|P_i|} \quad \textit{for all } i \in \{1, \cdots, 5\} \\
\pi^{err} &\stackrel{def}{=} \frac{1}{d(t-u)} \left(\sum_{i \in \{1,3\}} \bar{s_i} \frac{\sum_\ell 2\ell \rho^{(i)}_{2\ell+1}}{\sum_\ell \rho^{(i)}_{2\ell+1}} + \bar{s_2} \frac{\sum_\ell (2\ell-1) \rho^{(2)}_{2\ell+1}}{\sum_\ell \rho^{(2)}_{2\ell+1}} + \sum_{i \in \{4,5,6\}} \bar{s_i} \frac{\sum_\ell (2\ell+1) \rho^{(i)}_{2\ell+1}}{\sum_\ell \rho^{(i)}_{2\ell+1}} \right) \\
\pi^{good} &= \frac{1}{d(n-d-t+u)} \overline{N^{good}} \quad \textit{where} \\
\overline{N^{good}} &\stackrel{def}{=} w \cdot s - \bar{s_1} - \bar{s_5} - 2\bar{s_2} - 2\bar{s_3} - 2\bar{s_4} \\
&\quad - \sum_{i \in \{1,3\}} \bar{s_i} \frac{\sum_\ell 2\ell \rho^{(i)}_{2\ell+1}}{\sum_\ell \rho^{(i)}_{2\ell+1}} - \bar{s_2} \frac{\sum_\ell (2\ell-1) \rho^{(2)}_{2\ell+1}}{\sum_\ell \rho^{(2)}_{2\ell+1}} - \sum_{i \in \{4,5,6\}} \bar{s_i} \frac{\sum_\ell (2\ell+1) \rho^{(i)}_{2\ell+1}}{\sum_\ell \rho^{(i)}_{2\ell+1}}
\end{aligned}
$$

These formulas come from a double counting argument similar to the one leading to Chaulet's model. Let us start by explaining the formula for the π_i's.

The formula for the π_i's. As an illustration of the reasoning, we are going to explain the formula for π_2. Let e_2 be the number of edges leaving the group of bad bits and going to parity-check equations in P_2 that are not satisfied. Notice that the subgraph of the Tanner graph restricted to the set of bad bits and P_2 is bipartite and bi-regular. All bad bits have degree $u-1$ and all parity-checks have degree 2. Denote by s_2 the number of unsatisfied parity-checks in P_2 and by E_2 the total number of edges in this subgraph. Because all parity-checks have degree 2 in this graph we have the following equality

$$E_2 = 2|P_2| \tag{14}$$

The subgraph of this graph induced by all unsatisfied parity-checks has exactly e_2 edges by definition and is also 2-regular on the parity-check side. Therefore we also have

$$e_2 = 2|s_2|.$$

This implies that

$$\frac{s_2}{|P_2|} = \frac{e_2}{E_2}. \tag{15}$$

It is natural to define now π_2 as $\mathbb{E}\left[\frac{e_2}{E_2}|s,t,u\right]$, which leads to

$$\pi_2 = \frac{\bar{s_2}}{|P_2|}.$$

This kind of reasoning applies to all other cases, the corresponding subgraph being of constant degree 2 or 1 on the check node side. This leads to the aforementioned formulas for the π_i's.

The formula for π^{err}. This is obtained by a refinement of the double counting method of Julia Chaulet. Let us count in another way the number N^{err} of red edges leaving the normal bits in error. Recall that a red edge is an edge in the Tanner graph which is incident to a

check node which is unsatisfied. Let $n_\ell^{(i)}$ be the number of check nodes which have ℓ bits in error in P_i. We clearly have by counting the red edges from the check node perspective

$$N^{\text{err}} = \sum_{i\in\{1,3\}} \sum_{\ell} 2\ell \cdot n_{2\ell+1}^{(i)} + \sum_{\ell} (2\ell-1) n_{2\ell+1}^{(2)} + \sum_{i\in\{4,5,6\}} \sum_{\ell} (2\ell+1) n_{2\ell+1}^{(i)}. \tag{16}$$

We have on one hand

$$\mathbb{E}\left[N^{\text{err}} | s,t,u\right] = d.(t-u)\pi^{\text{err}}, \tag{17}$$

whereas on the other hand, because of (16), we approximate this expected value by

$$\mathbb{E}(N^{\text{err}}|s,t,u) \approx \sum_{i\in\{1,3\}} \bar{s_i}\mathbb{E}(e^{(i)}-1|s^{(i)}=1,t,u) + \bar{s_2}\mathbb{E}(e^{(2)}-2|s^{(2)}=1,t,u) + \sum_{i\in\{4,5,6\}} \bar{s_i}\mathbb{E}(e^{(i)}|s^{(i)}=1,t,u). \tag{18}$$

where s_i is the contribution to the syndrome of the check nodes which are in P_i, $s^{(i)}$ is the syndrome bit of any check node belonging to the group P_i, and $e^{(i)}$ is the number of bits in error adjacent to an arbitrary check node in P_i. The right-hand term is clearly equal to

$$\sum_{i\in\{1,3\}} \bar{s_i} \frac{\sum_\ell 2\ell \rho_{2\ell+1}^{(i)}}{\sum_\ell \rho_{2\ell+1}^{(i)}} + \bar{s_2} \frac{\sum_\ell (2\ell-1)\rho_{2\ell+1}^{(2)}}{\sum_\ell \rho_{2\ell+1}^{(2)}} + \sum_{i\in\{4,5,6\}} \bar{s_i} \frac{\sum_\ell (2\ell+1)\rho_{2\ell+1}^{(i)}}{\sum_\ell \rho_{2\ell+1}^{(i)}}.$$

Here we used that for any $i \in \{1,\cdots,6\}$, $\rho_\ell^{(i)}$ is exactly the probability that a parity-check belonging to group P_i contains ℓ errors. Substituting for this expression of $\mathbb{E}(N^{\text{err}}|s,t,u)$ in (17) gives the formula for π^{err} above. The formula for π^{good} follows a similar reasoning.

Once we have the models of the counters we set up the Markov chain model as explained in Appendix A. We also need to compute the initial distribution of states (s,t,u). This is explained in a slightly more general setting applying to general keys in Section B of the appendix.

5.4. **Results.** We have tested the model with a perfect key which is small enough so that we could observe the error-floor experimentally (see Figure 9). We checked two decoders, the step by step with a BGF style decoder and the step by step majority decoder (where the threshold is set at $\frac{d+1}{2}$). The model is in general conservative. In the waterfall region, we experience the same kind of pessimistic estimation of the DFR as in the original Sendrier and Vasseur Markov model [SV19b]. Interestingly enough, the model becomes significantly better in the error floor region where it follows amazingly well the experimental curve.

6. A Refined Markovian Model Applying to All Keys and Taking their Shape into Account

Even if the previous Markov model is impressively accurate in the error floor region, it has the big drawback to apply solely to perfect keys. If the density of perfect keys was for instance constant among the set of BIKE keys, this would not be a major issue. However, such keys tend to be extremely rare. We have never found one by looking for it at random but used instead algebraic constructions of planar difference sets, namely subsets $\mathscr{S}$ of $\mathbb{Z}_r$ such that all non zero elements v of $\mathbb{Z}_r$ can be expressed in a unique way as a

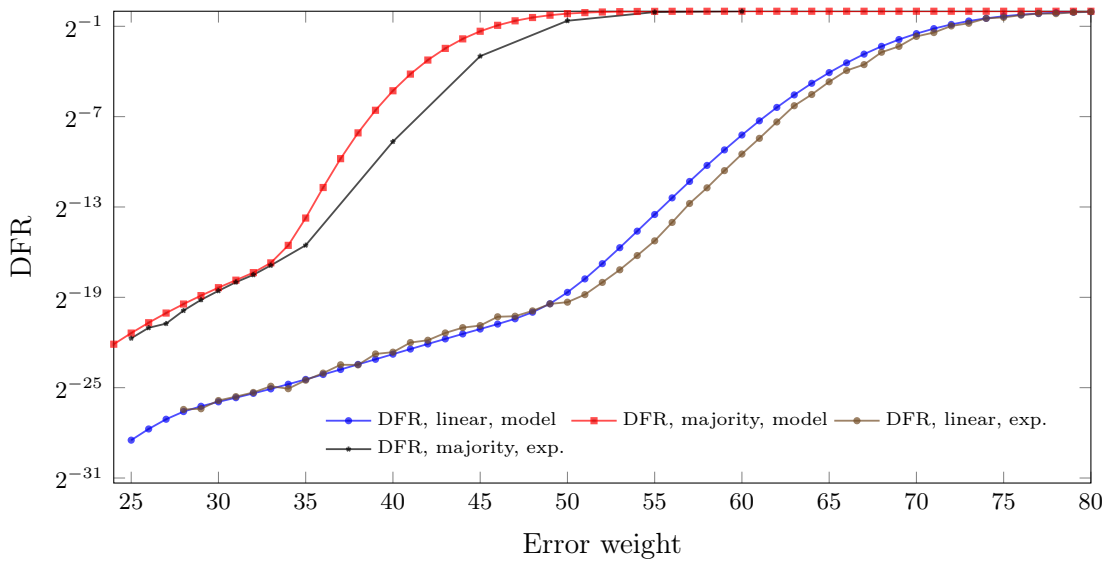


FIGURE 9. DFR vs. error weight ($r = 1723$, $d = 17$), perfect keys, experiments vs. model, step by step with majority threshold and step by step with threshold obtained by saturating the linear function $\alpha \cdot s + \beta$ to stay between $(d+1)/2$ and d for $\alpha \stackrel{\text{def}}{=} 0.0060162$, $\beta \stackrel{\text{def}}{=} 8.7973$.

difference of two elements x and y in $\mathscr{S}$: $v = x - y$. When we have such a difference set we choose two random subsets $\mathscr{S}_0$ and $\mathscr{S}_1$ of $\mathscr{S}$ and set $h_i(x) \stackrel{\text{def}}{=} \sum_{j \in \mathscr{S}_i} x^j$. Experimental evidence shows that perfect keys behave better than random keys with respect to iterative decoding. More generally the behavior of iterative decoding is definitely key dependent, there are in particular weak keys which behave significantly worse than typical keys. It is therefore desirable to have a way to predict the iterative decoding performance which is key dependent. This is what we are going to do here.

6.1. **Derivation of the Markov chain model.** The structure of the subgraph $\mathscr{G}$ of the Tanner graph associated to a near codeword depends in this case if it is of the form $x^a h_0(x) \oplus 0$ (which we call a *left near codeword*) or $0 \oplus x^b h_1(x)$ (which we call a *right near codeword*). It may have parity-check nodes of degree greater than 2. The following proposition which is straightforward to prove explains how parity-check nodes of a certain degree arise in $\mathscr{G}$.

Proposition 3. *The parity-checks of even degree $2m$ in the subgraph $\mathscr{G}$ corresponding to a left near codeword are associated to sets of m different pairs $\{l_{a_1}, l_{b_1}\}, \cdots, \{l_{a_m}, l_{b_m}\}$ such that*

$$l_{a_1} + l_{b_1} = \cdots = l_{a_m} + l_{b_m},$$

where the a_i's and the b_i's all belong to $\{0, \cdots, d-1\}$ and $a_i \neq b_i$ for all $i \in \{1, \cdots, m\}$. The parity-checks of odd degree $2m+1$ in this graph are associated to an $a \in \{0, \cdots, d-1\}$

and sets of m different pairs $\{l_{a_1}, l_{b_1}\}, \cdots, \{l_{a_m}, l_{b_m}\}$ such that

$$2l_a = l_{a_1} + l_{b_1} = \cdots = l_{a_m} + l_{b_m},$$

where the a_i's and the b_i's all belong to $\{0, \cdots, d-1\}$ and $a_i \neq b_i$ for all $i \in \{1, \cdots, m\}$.

A similar proposition holds for right near codewords where the r_i's replace the l_i's. As in the case of perfect keys we can partition the set of code positions in four sets, the bad bits, the suspicious bits, the normal bits in error and the good bits. We will simplify notation in what follows and label a parity-check x^a simply by a in what follows. We assume from now on that the closest near-codeword $\boldsymbol{n}$ to the error intersects it in exactly t positions and that the error is weight t. We also assume that $\boldsymbol{n} = h_0(x) \oplus 0$. The case when $\boldsymbol{n} = 0 \oplus h_1(x)$ is treated similarly.

The case of bad bits. Again, we have three kinds of parity-checks adjacent to a bad bit. If the label of the bad bit is say a_1, the labels of the other bad bits $a_2, \cdots, a_u$ and the labels of the suspicious bits is $a_{u+1}, \cdots, a_d$, then we can group the parity-check nodes adjacent to a_1 as

- the first group is formed by a single parity-check labeled $2a_1$. Note that it is necessarily of odd degree in $\mathscr{G}$;
- the second group is formed by $u-1$ parity-check parity nodes labeled $a_1+a_2, \cdots, a_1+a_u$;
- the third group is formed by $d-u$ parity-check parity nodes labeled $a_1+a_{u+1}, \cdots, a_1+a_d$.

We do not know the label $a_1, \cdots, a_u$, and we model the probability that the corresponding parity-checks are unsatisfied by using the following lemma

Lemma 2. *Let a be a bad bit and let*

- *$\rho^{b,1}_{\Delta,\ell}$ be the probability that the parity-check labeled $2a$ is adjacent to ℓ bits in error in total given that it is of degree Δ in $\mathscr{G}$ (this parity-check belongs to the first group);*
- *$\rho^{b,2}_{\Delta,\ell}$ be the probability that a parity-check labeled $a+b$ of degree Δ in $\mathscr{G}$ is adjacent to ℓ bits in error in total given that it b is another bad bit (this parity-check belongs to the second group);*
- *$\rho^{b,3}_{\Delta,\ell}$ be the probability that a parity-check labeled $a+b$ is adjacent to ℓ bits in error in total given that it is of degree Δ in $\mathscr{G}$, that b is a suspicious bit (this parity-check belongs to the third group).*

Then

$$\begin{aligned}
\rho^{b,1}_{\Delta,\ell} &= \frac{\sum_{j=1}^{\min(\Delta,\ell)} \binom{\Delta-1}{j-1}\binom{d-\Delta}{u-j}\binom{w-\Delta}{\ell-j}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d-1}{u-1}\binom{n-d}{t-u}} \\
\rho^{b,2}_{\Delta,\ell} &= \frac{\sum_{j=2}^{\min(\Delta,\ell)} \binom{\Delta-2}{j-2}\binom{d-\Delta}{u-j}\binom{w-\Delta}{\ell-j}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d-2}{u-2}\binom{n-d}{t-u}} \\
\rho^{b,3}_{\Delta,\ell} &= \frac{\sum_{j=1}^{\min(\Delta,\ell)} \binom{\Delta-2}{j-1}\binom{d-\Delta}{u-j}\binom{w-\Delta}{\ell-j}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d-2}{u-1}\binom{n-d}{t-u}}
\end{aligned}$$

Proof.

The formula for $\rho^{b,1}_{\Delta,\ell}$. We have to compute in this case the probability that an error of weight t which has weight u on the d bits of the near-codeword $\boldsymbol{n}$ and $t-u$ on the complement of the support of $\boldsymbol{n}$ intersects a given parity-check of weight w in exactly ℓ positions. Furthermore, we know that the parity-check $2a$ involves the bad bit a we are interested in, plus $\Delta-1$ other bits of $\boldsymbol{n}$. If we denote by $\rho^{b,1}_{\Delta,\ell,j}$ the probability that this error is also of weight j on the Δ bits which belong at the same time to the support of the parity-check labeled $2a$ and the support of $\boldsymbol{n}$, then we have

$$\rho^{b,1}_{\Delta,\ell,j} = \frac{\binom{\Delta-1}{j-1}\binom{d-\Delta}{u-j}\binom{w-\Delta}{\ell-j}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d-1}{u-1}\binom{n-d}{t-u}}$$

This can be verified by counting the number of ways the $\binom{d-1}{u-1}\binom{n-d}{t-u}$ configurations of $t-1$ errors where there are $u-1$ errors among the $d-1$ bits of $\boldsymbol{n}$ other than the bad bit a we are interested in satisfy all the needed constraints. Figure 10 gives a picture of the configuration we are interested in. This yields the formula for $\rho^{b,1}_{\Delta,\ell}$ by summing over all possible values of j.

The formula for $\rho^{b,2}_{\Delta,\ell}$. The reasoning is similar. We introduce $\rho^{b,2}_{\Delta,\ell,j}$ which is the probability that the error is also of weight j on the Δ bits which belong at the same time to the support of the parity-check labeled $a+b$ and the support of $\boldsymbol{n}$, then we have

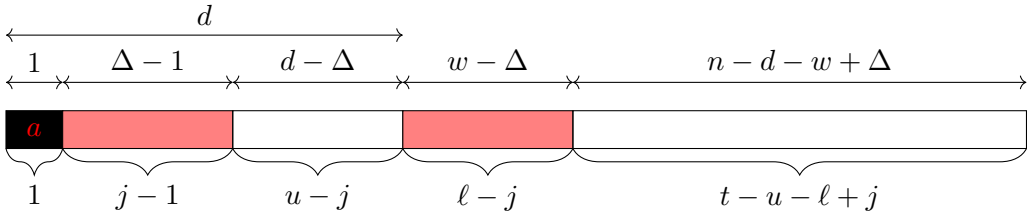
$$\rho^{b,2}_{\Delta,\ell,j} = \frac{\binom{\Delta-2}{j-2}\binom{d-\Delta}{u-j}\binom{w-\Delta}{\ell-j}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d-2}{u-2}\binom{n-d}{t-u}}$$

Figure 11 displays the corresponding configuration. The difference with the previous case is that now the parity-check labeled $a+b$ contains two bits which are in error by definition, namely a and b.

The formula for $\rho^{b,2}_{\Delta,\ell}$. This uses a similar reasoning, with the only difference that the parity-check labeled $a+b$ contains by definition a bit which is in error, namely a and a bit which is not in error, namely b. □ □

In the case of a perfect key:

FIGURE 10. This figure shows the configuration of the error and the parity-check. The positions in black and pink represent all the positions of the parity-check. The numbers on the top give the size of each block. The numbers on the bottom indicate the number of errors in each block. There is a first block of size 1 which is the bad bit a we are interested in. It is by definition a bit which is in error. The second block is of size $\Delta - 1$. It consists in the $\Delta - 1$ other positions involved in the parity-check which are also in the support of the near codeword $\boldsymbol{n}$. This block contains exactly $j - 1$ errors by assumption. The third block is of size $d - \Delta$. It consists in the $d - \Delta$ positions in the support of $\boldsymbol{n}$ which are not involved in the parity-check equation. It contains $u - j$ errors. The fourth block consists in the $w - \Delta$ positions of the parity-check which are not in the support of $\boldsymbol{n}$. It contains $\ell - j$ errors since the whole parity-check contains ℓ errors and j errors are already in the first two blocks of positions. The last block contains the remaining positions which belong neither to the parity-check equation nor to the support of $\boldsymbol{n}$. It contains the part of the t errors that remain, that is $t - u - l + j$.

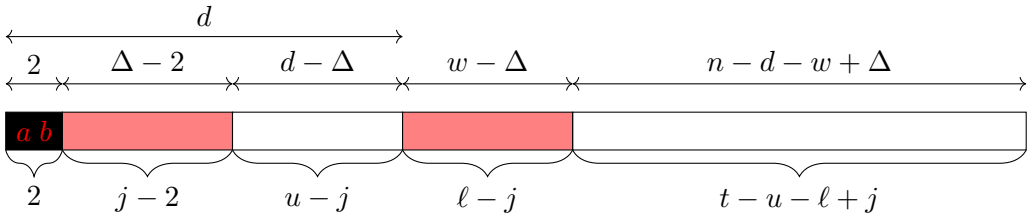


Lemma 3.

$$\begin{aligned}
\rho_{,\ell}^{b,1} &= \frac{\binom{w-1}{\ell-1}\binom{n-d-w+1}{t-u-\ell+1}}{\binom{n-d}{t-u}} \\
\rho_{,\ell}^{b,2} &= \frac{\binom{w-2}{\ell-2}\binom{n-d-w+2}{t-u-\ell+2}}{\binom{n-d}{t-u}} \\
\rho_{,\ell}^{b,3} &= \frac{\binom{w-2}{\ell-1}\binom{n-d-w+2}{t-u-\ell+1}}{\binom{n-d}{t-u}}
\end{aligned}$$

This coincides with the formula of the previous section.

FIGURE 11. This figure shows the configuration of the error and the parity-check. The positions in black and pink represent all the positions of the parity-check. The numbers on the top give the size of each block. The numbers on the bottom indicate the number of errors in each block. There is a first block of size 2 corresponding to bad bits a and b. By definition both bits are in error. The second block is of size $\Delta - 2$. It consists in the $\Delta - 2$ other positions involved in the parity-check which are also in the support of the near codeword $\boldsymbol{n}$. This block contains exactly $j-2$ errors by assumption. The third block is of size $d-\Delta$. It consists in the $d-\Delta$ positions in the support of $\boldsymbol{n}$ which are not involved in the parity-check equation. It contains $u - j$ errors. The fourth block consists in the $w - \Delta$ positions of the parity-check which are not in the support of $\boldsymbol{n}$. It contains $\ell - j$ errors since the whole parity-check contains ℓ errors and j errors are already in the first two blocks of positions. The last block contains the remaining positions which belong neither to the parity-check equation nor to the support of $\boldsymbol{n}$. It contains the part of the t errors that remain, that is $t - u - l + j$.



Proposition 4. *For $g \in \{1,2,3\}$, let $\pi^{\bar{b},g}$ be the average probability that a parity-check of group g associated to a bad bit is unsatisfied. Then*

$$\begin{aligned}\overline{\pi^{b,1}} &= \frac{1}{d}\sum_{a\in\boldsymbol{n}}\sum_{\ell}\rho^{b,1}_{\deg(2a),2\ell+1}\\ \overline{\pi^{b,g}} &= \frac{1}{d(d-1)}\sum_{\substack{a\in\boldsymbol{n}\\ b\in\boldsymbol{n},\, b\neq a}}\sum_{\ell}\rho^{b,g}_{\deg(a+b),2\ell+1},\ \textit{for } g\geqslant 2.\end{aligned}$$

Here $\deg(c)$ *refers to the degree of the parity-check node c in $\mathscr{G}$.*

Proof. The probability that the parity-check labeled $2a$ associated to a bad bit a is unsatisfied is given by $\sum_\ell \rho^{b,1}_{\deg(2a),2\ell+1}$. The average is then taken over all possible bits of the near codeword $\boldsymbol{n}$. The reasoning is the same for $\overline{\pi^{b,g}}$, the only difference being that we take the average over all possible edges leaving a variable node labeled a to a check node labeled $a+b$ for all possible b's different from a. □ □

In the case of a perfect key, this reduces to

Proposition 5. *For $g \in \{1,2,3\}$, let $\pi^{\overline{b},g}$ be the average probability that a parity-check of group g associated to a bad bit is unsatisfied. Then*

$$\overline{\pi^{b,g}} = \sum_{\ell} \rho^{b,g}_{,2\ell+1}.$$

Again, we recover the formula of th previous section.

The case of suspicious bits. The case of a suspicious bit is similar to the case of a bad bit and we also partition the parity-checks associated to it in three groups in exactly the same way as we did for a bad bit. There is a single parity-check in the first group associated to a suspicious bit a, namely the parity-check labeled $2a$. The second group contains $d-u-1$ other parity-checks labeled $a+b$ where b is another suspicious bit. The last group contains u parity-check bits labeled $a+b$ where b is a bad bit. Similarly to bad bits we have

Lemma 4. *Let a be a suspicious bit and let*

- *$\rho^{s,1}_{\Delta,\ell}$ be the probability that the parity-check labeled $2a$ in $\mathscr{G}$ is adjacent to ℓ bits in error in total given that it is of degree Δ in $\mathscr{G}$ (this parity-check belongs to the first group);*
- *$\rho^{s,2}_{\Delta,\ell}$ be the probability that a parity-check labeled $a+b$ is adjacent to ℓ bits in error in total given that it is of degree Δ in $\mathscr{G}$ and b is another suspicious bit (this parity-check belongs to the second group);*
- *$\rho^{s,3}_{\Delta,\ell}$ be the probability that a parity-check labeled $a+b$ is adjacent to ℓ bits in error in total given that it is of degree Δ in $\mathscr{G}$, that b is a bad bit (this parity-check belongs to the third group).*

Then

$$\begin{aligned}
\rho^{s,1}_{\Delta,\ell} &= \frac{\sum_{j=0}^{\min(\Delta,\ell)} \binom{\Delta-1}{j}\binom{d-\Delta}{u-j}\binom{w-\Delta}{\ell-j}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d-1}{u}\binom{n-d}{t-u}} \\
\rho^{s,2}_{\Delta,\ell} &= \frac{\sum_{j=0}^{\min(\Delta,\ell)} \binom{\Delta-2}{j}\binom{d-\Delta}{u-j}\binom{w-\Delta}{\ell-j}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d-2}{u}\binom{n-d}{t-u}} \\
\rho^{s,3}_{\Delta,\ell} &= \frac{\sum_{j=1}^{\min(\Delta,\ell)} \binom{\Delta-2}{j-1}\binom{d-\Delta}{u-j}\binom{w-\Delta}{\ell-j}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d-2}{u-1}\binom{n-d}{t-u}}
\end{aligned}$$

In the case of a perfect key:

Lemma 5. *Let a be a suspicious bit and let*

- *$\rho^{s,1}_{,\ell}$ be the probability that the parity-check labeled $2a$ in $\mathscr{G}$ is adjacent to ℓ bits in error in total (this parity-check belongs to the first group);*
- *$\rho^{s,2}_{,\ell}$ be the probability that a parity-check labeled $a+b$ is adjacent to ℓ bits in error in total (this parity-check belongs to the second group);*

- $\rho^{s,3}_{\Delta,\ell}$ *be the probability that a parity-check labeled* $a+b$ *is adjacent to* ℓ *bits in error in total given that* b *is a bad bit (this parity-check belongs to the third group).*

Then

$$\begin{aligned}
\rho^{s,1}_{,\ell} &= \frac{\binom{w-1}{\ell}\binom{n-d-w+1}{t-u-\ell}}{\binom{n-d}{t-u}} \\
\rho^{s,2}_{,\ell} &= \frac{\binom{w-2}{\ell}\binom{n-d-w+2}{t-u-\ell}}{\binom{n-d}{t-u}} \\
\rho^{s,3}_{\Delta,\ell} &= \frac{\binom{w-2}{\ell-1}\binom{n-d-w+2}{t-u-\ell+1}}{\binom{n-d}{t-u}}
\end{aligned}$$

Again, this coincides with the formula of the previous section. From lemma 4, we deduce

Proposition 6. *For* $g \in \{1,2,3\}$*, let* $\pi^{\overline{s},g}$ *be the average probability that a parity-check of group* g *associated to a suspicious bit is unsatisfied. Then*

$$\begin{aligned}
\overline{\pi^{s,1}} &= \frac{1}{d}\sum_{a\in\boldsymbol{n}}\sum_{\ell}\rho^{s,1}_{\deg(2a),2\ell+1} \\
\overline{\pi^{s,g}} &= \frac{1}{d(d-1)}\sum_{\substack{a\in\boldsymbol{n}\\ b\in\boldsymbol{n},\, b\neq a}}\sum_{\ell}\rho^{s,g}_{\deg(a+b),2\ell+1},\ \textit{for } g\geqslant 2.
\end{aligned}$$

Here $\deg(c)$ *refers to the degree of the parity-check node* c *in* $\mathscr{G}$*.*

In the case of a perfect key:

Proposition 7. *For* $g \in \{1,2,3\}$*, let* $\pi^{\overline{s},g}$ *be the average probability that a parity-check of group* g *associated to a suspicious bit is unsatisfied. Then*

$$\overline{\pi^{s,g}} = \sum_{\ell}\rho^{s,g}_{,2\ell+1}.$$

The case of normal bits is simpler, since it is not needed that we distinguish several groups of parity-checks.

The case of normal bits in error.

Proposition 8. *Let* $\rho^{e}_{\Delta,\ell}$ *be the probability that a parity-check node of degree* Δ *in the subgraph* $\mathscr{G}$ *of the Tanner graph induced by* $\boldsymbol{n}$ *and adjacent to a normal bit in error contains exactly* ℓ *erroneous bits. A parity-check which does not belong to* $\mathscr{G}$ *is considered to be of degree* 0*. We also let* $\overline{\pi^{e}}$ *be the average of these probabilities over all edges of the Tanner graph leaving the bits which are not in* $\boldsymbol{n}$*.*

We have

$$\rho^e_{\Delta,\ell} = \frac{\sum_{j=0}^{\min(\Delta,\ell-1)} \binom{\Delta}{j}\binom{d-\Delta}{u-j}\binom{w-\Delta-1}{\ell-j-1}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d}{u}\binom{n-d-1}{t-u-1}} \tag{19}$$

$$\overline{\pi^e} = \frac{1}{d(n-d)}\sum_{\Delta}(w-\Delta)n_\Delta \sum_{\ell}\rho^e_{\Delta,2\ell+1}, \tag{20}$$

where n_Δ is the number of check nodes in the subgraph $\mathscr{G}$ of the Tanner graph induced by $\boldsymbol{n}$ which are of degree Δ.

Proof. The reasoning behind the formula (19) is similar to what is done for Lemma 3 and is omitted. For computing the average of (19) we first count the edges in the Tanner graph leaving the $n-d$ positions which are not in $\boldsymbol{n}$. There are $n-d$ variable nodes of this kind and therefore $d(n-d)$ edges of this kind. $\overline{\rho^e_{\Delta,\ell}}$ is nothing but the average probability over all edges that such an edge is adjacent to a check node which is not satisfied. If such a check node is of degree Δ in $\mathscr{G}$, the subgraph of the Tanner graph induced by $\boldsymbol{n}$, then the probability that it is in error is exactly $\sum_\ell \rho^e_{\Delta,2\ell+1}$. There are exactly $(w-\Delta)n_\Delta$ edges of this kind that are adjacent to a check node of degree Δ in $\mathscr{G}$. This explains (20).

□

In the case of a perfect key:

Proposition 9. *Let $\rho^e_{\Delta,\ell}$ be the probability that a parity-check node of degree Δ in the subgraph $\mathscr{G}$ of the Tanner graph induced by $\boldsymbol{n}$ and adjacent to a normal bit in error contains exactly ℓ erroneous bits. A parity-check which does not belong to $\mathscr{G}$ is considered to be of degree 0. We also let $\overline{\pi^e}$ be the average of these probabilities over all edges of the Tanner graph leaving the bits which are not in $\boldsymbol{n}$. We have*

$$\rho^e_{\Delta,\ell} = \frac{\sum_{j=0}^{\min(\Delta,\ell-1)} \binom{\Delta}{j}\binom{d-\Delta}{u-j}\binom{w-\Delta-1}{\ell-j-1}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d}{u}\binom{n-d-1}{t-u-1}}$$

$$\overline{\pi^e} = \frac{1}{d(n-d)}\left\{ w\left[r-\frac{d(d+1)}{2}\right]\sum_\ell \rho^e_{0,2\ell+1} + (w-1)d\sum_\ell \rho^e_{1,2\ell+1} + (w-2)\frac{d(d-1)}{2}\sum_\ell \rho^e_{2,2\ell+1}\right\}$$

where n_Δ is the number of check nodes in the subgraph $\mathscr{G}$ of the Tanner graph induced by $\boldsymbol{n}$ which are of degree Δ.

It is here that the formula differs from the previous section. Experiments have shown that this new formula yields an improvement of the modeling predictions.

The case of good bits. The case of good bits is similar to the case of normal bits in error:

Proposition 10. *Let $\rho^g_{\Delta,\ell}$ be the probability that a parity-check node of $\mathscr{G}$ of degree Δ adjacent to a good bit contains exactly ℓ erroneous bits. We also let $\overline{\pi^g}$ be the average of*

these probabilities over all edges of the Tanner graph leaving the bits which are not in $\boldsymbol{n}$*. We have*

$$\rho^g_{\Delta,\ell} = \frac{\sum_{j=0}^{\min(\Delta,\ell)} \binom{\Delta}{j}\binom{d-\Delta}{u-j}\binom{w-\Delta-1}{\ell-j}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d}{u}\binom{n-d-1}{t-u}} \tag{21}$$

$$\overline{\pi^g} = \frac{1}{d(n-d)}\sum_{\Delta}(w-\Delta)n_\Delta \sum_{\ell}\rho^g_{\Delta,2\ell+1}, \tag{22}$$

In the case of a perfect key:

Proposition 11. *Let* $\rho^g_{\Delta,\ell}$ *be the probability that a parity-check node of* $\mathscr{G}$ *of degree* Δ *adjacent to a good bit contains exactly* ℓ *erroneous bits. We also let* $\overline{\pi^g}$ *be the average of these probabilities over all edges of the Tanner graph leaving the bits which are not in* $\boldsymbol{n}$*. We have*

$$\rho^g_{\Delta,\ell} = \frac{\sum_{j=0}^{\min(\Delta,\ell)} \binom{\Delta}{j}\binom{d-\Delta}{u-j}\binom{w-\Delta-1}{\ell-j}\binom{n-d-w+\Delta}{t-u-\ell+j}}{\binom{d}{u}\binom{n-d-1}{t-u}}$$

$$\overline{\pi^g} = \frac{1}{d(n-d)}\left\{ w\left[r-\frac{d(d+1)}{2}\right]\sum_{\ell}\rho^g_{0,2\ell+1} + (w-1)d\sum_{\ell}\rho^g_{1,2\ell+1} + (w-2)\frac{d(d-1)}{2}\sum_{\ell}\rho^g_{2,2\ell+1}\right\}.$$

This formula also differs from the formula given for perfect keys in the previous section. Together with the new formula of normal bits in error, they yield better predictions than using those of the previous section.

We will now analyze the structure of the subgraph $\mathscr{G}$ of the Tanner graph associated to the "first" near codeword $\boldsymbol{n} = (h_0(x), \mathbf{0})$. It is readily seen that the structure of this graph is given by

Notation 2. *Let* n_Δ *be the number of parity-check equations in* $\mathscr{G}$ *which are of degree* Δ*. For variable nodes of the form* $2l_a$*, there are* n_Δ *edges for each odd* Δ *connecting to check nodes of degree* Δ*. For all other nodes of the form* $l_a + l_b$*, there are* $\Delta n_\Delta - (\Delta \bmod 2)n_\Delta$ *edges for each* $\Delta \geqslant 1$ *connecting to check nodes of degree* Δ*. The remaining edges in the parity check equations amount to* $(w-\Delta)n_\Delta$ *for each* Δ*.*

We observe that:

- *The sum over all odd* Δ *of edges of the first kind equals* d
- *The sum over all* $\Delta \geqslant 1$ *of edges of the second kind equals* $d(d-1)$
- *The sum over all* Δ *of edges of the third kind equals* $d(n-d)$

Together these account for all dn *edges in the Tanner graph.*

We finally deduce the following model with a similar reasoning as what has been done in the case of perfect keys

Model 4.

$$\begin{aligned}
\textit{bad bit: } \sigma_i &\sim \mathrm{Bin}(1, \frac{s\overline{\pi^{b,1}}}{\mathbb{E}(s|t,u)}) + \mathrm{Bin}(u-1, \frac{s\overline{\pi^{b,2}}}{\mathbb{E}(s|t,u)}) + \mathrm{Bin}(d-u, \frac{s\overline{\pi^{b,3}}}{\mathbb{E}(s|t,u)}) \\
\textit{suspicious bit: } \sigma_i &\sim \mathrm{Bin}(1, \frac{s\overline{\pi^{s,1}}}{\mathbb{E}(s|t,u)}) + \mathrm{Bin}(d-u-1, \frac{s\overline{\pi^{s,2}}}{\mathbb{E}(s|t,u)}) + \mathrm{Bin}(u, \frac{s\overline{\pi^{s,3}}}{\mathbb{E}(s|t,u)}) \\
\textit{normal bit in error: } \sigma_i &\sim \mathrm{Bin}(d, \frac{s\overline{\pi^{e}}}{\mathbb{E}(s|t,u)}) \\
\textit{good bit: } \sigma_i &\sim \mathrm{Bin}(d, \frac{s\overline{\pi^{g}}}{\mathbb{E}(s|t,u)})
\end{aligned}$$

where where $\mathbb{E}(s|t,u)$ *is given by:*

$$\begin{aligned}
\mathbb{E}(s|t,u) = \frac{1}{w}\Big(&u\overline{\pi^{b,1}} + u(u-1)\overline{\pi^{b,2}} + u(d-u)\overline{\pi^{b,3}} + (d-u)\overline{\pi^{s,1}} \\
&+ (d-u)(d-u-1)\overline{\pi^{s,2}} + (d-u)u\overline{\pi^{s,3}} + (t-u)d\overline{\pi^{e}} + (n+u-t-d)d\overline{\pi^{g}}\Big)
\end{aligned}$$

6.2. **Results.** We have first checked this more general model on the same toy example of the previous section with a random key (see Figure 12). Note that the new model is not really a generalization of the one given in the previous section. The π_i's are estimated in this model applying to any key in a direct fashion by scaling the expected syndrome on each set of parity-checks according to the whole syndrome weight. It gives clearly a better behavior in the waterfall region than the previous model based on Chaulet's approach for estimating the π_i's. We have decomposed the DFR both in experiments and in the model in two parts, the one coming from the contribution from decodings that failed because they converged to a near codeword (this is called "Contr. of ncw") and another one which is the rest of the contribution to the DFR (called "Contr. of other"). The models and the experiments agree remarkably well. We have also tested two different step by step decoders on the same code. There is one which uses the majority rule for the threshold, and another one which uses the decoder with two thresholds T_0 and T_1 which is detailed in Figure 3. There is a first conservative threshold which serves to flip initially the bits and a second threshold (actually the majority rule here) which serves to flip the bits when there are no more bit flips to perform with the first rule. This decoder is much better than the majority decoder as can be expected.

In [Vas21b] which compared parallel decoders to similar step by step decoders, the performance of the parallel decoders were much better than those of the step by step decoders. However, the step by step decoder with two thresholds used to draw the previous graph is now rather competitive with the parallel BGF decoder of BIKE as shown in Figure 13. This seems to confirm that there is a significant gain in choosing conservative thresholds in the early iterations as already observed in [Sen24]. The comparison between both decoders show that the BGF decoder is slightly superior to the step by step decoder with a somewhat steeper waterfall and a better error floor.

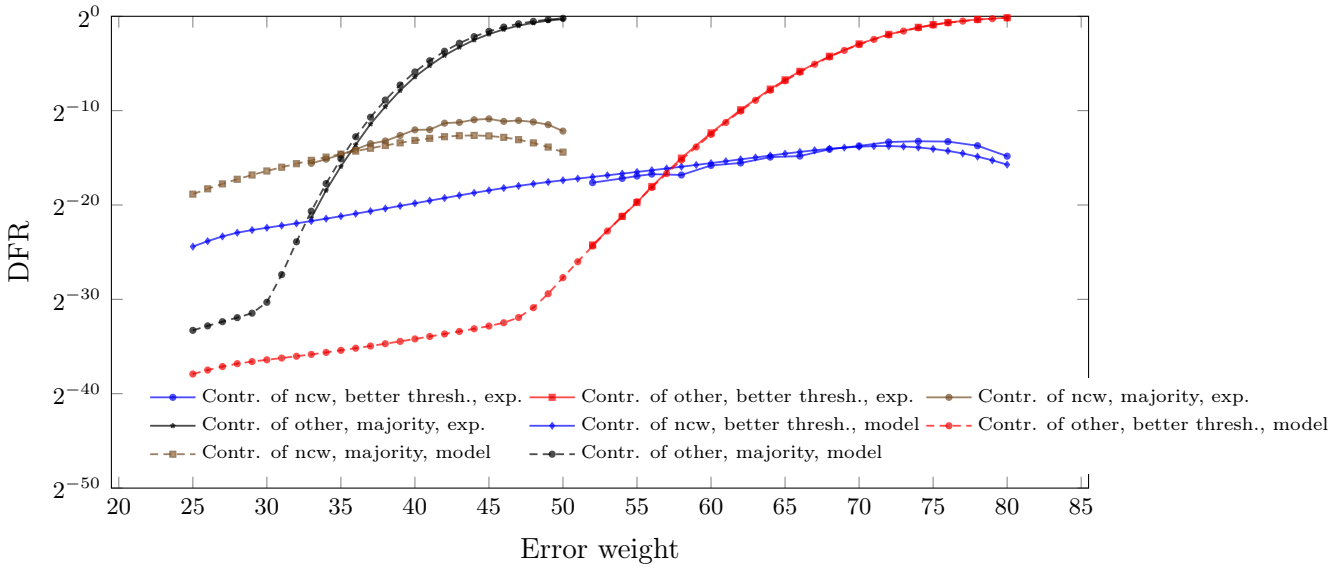


FIGURE 12. DFR vs. error weight ($r = 1723$, $d = 17$), experiments vs. model.

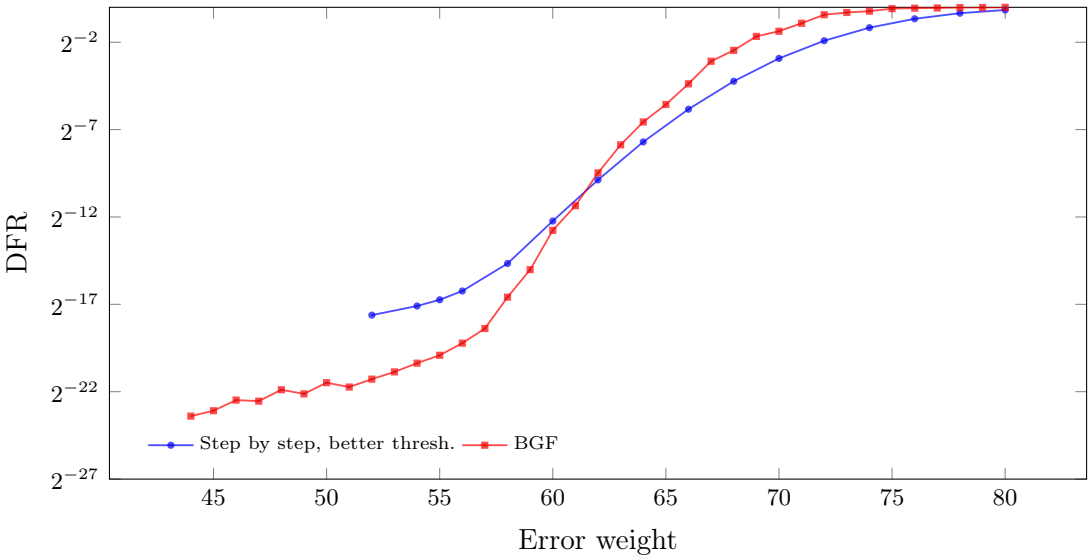


FIGURE 13. DFR vs. error weight ($r = 1723$, $d = 17$), BGF vs. step by step.

Finally after all this experimental evidence, we have run the model of this section on a key chosen at random for the step by step decoder by choosing several different thresholds for various block sizes for $d = 71$ and $t = 134$. The results are given in Figure 14.

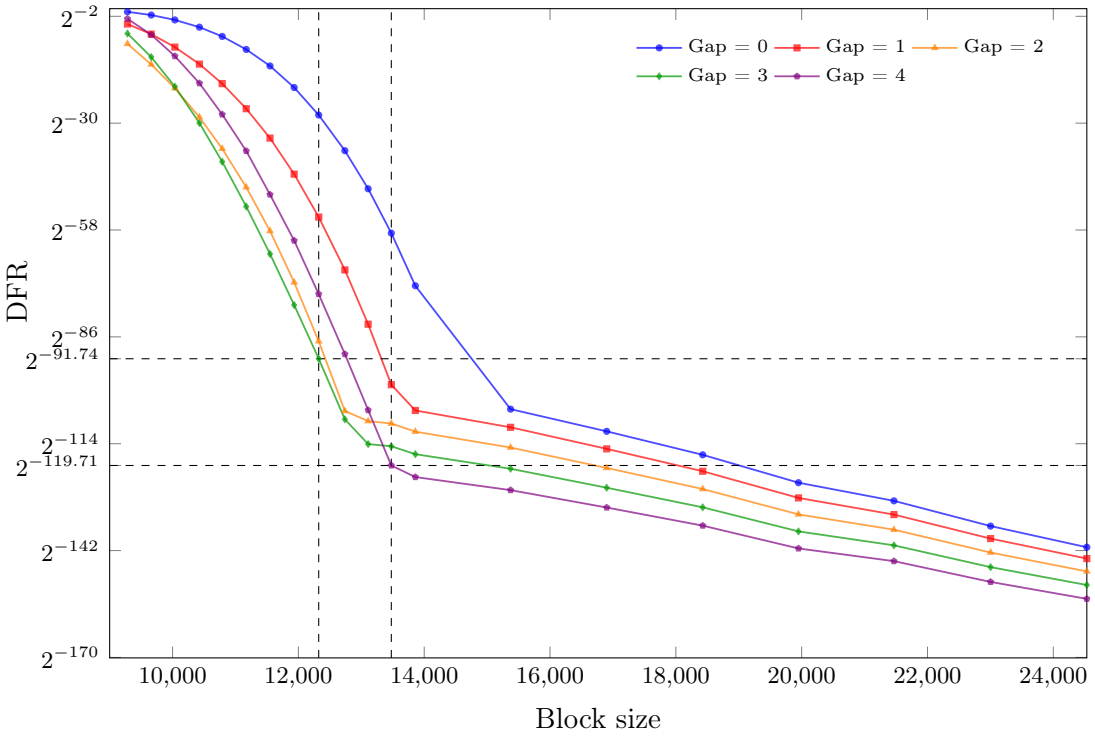


FIGURE 14. DFR vs. block size ($d = 71$, $t = 134$).

We have checked the influence of the gap parameter δ used for decoding. Here the threshold T for flipping a bit depends only on the syndrome weight s, the block length r, the error weight t and the gap parameter δ as follows

$$T = \min\left(d, \max\left(A_{r,t}(s) + \delta, \frac{d+1}{2}\right)\right), \tag{23}$$

where $A_{r,t}(s)$ is an affine function $\alpha \cdot s + \beta$ of s depending on r and t which is computed with the same method as for BIKE. We have obtained for $t = 134$ and various values of r the coefficients given in Subsection C.1 Table 1.

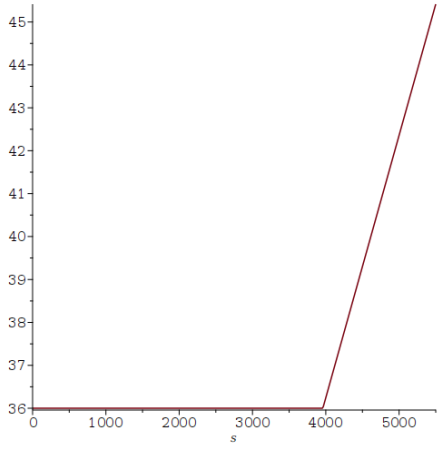
We observe several different interesting points:

- Using the same gap parameter $\delta = 3$ as the one used in the latest BIKE decoder seems to be optimal for the actual BIKE 1 parameter which is $r = 12323$. It results in a DFR of about $2^{-91.7}$ which is unfortunately above $2^{-\lambda}$.
- Moreover, even if we increase the block length r we do not improve by much the DFR, because the error floor kicks in right after this value of r. We attain a DFR of $2^{-114.1}$ with this gap for $r = 13109$ but need then to go to $r = 18427$ to go below $2^{-\lambda}$. It is $2^{-130.6}$ in this case.

- Using a gap δ equal to 4 allows to an error floor which appears a little bit later and allows to obtain a DFR which is about $2^{-119.6}$ for $r = 13477$. This is less than 10 percent more than the actual BIKE parameter.

A closer inspection of the actual choice for the affine function $A_{r,t}$ shows that for a large set of values of s (roughly between 0 and more than 70 per cent of the typical initial syndrome value) the threshold takes the value $\frac{d+1}{2}$. A typical example is given in Figure 15 where $d = 71$ and $(d+1)/2 = 36$.

FIGURE 15. An example for the BIKE 1 block length $r = 12323$ and the affine function given in Subsection C.1 Table 1.



This seems to be a bad choice. Indeed, the majority threshold rule is the one which maximizes the probability of flipping a suspicious bit and thus provoking the avalanche effect leading to the convergence to a near-codeword and thus to the error floor phenomenon. It is tempting to conjecture that using instead of (23) the following modified threshold function might be better

$$T = \min\left(d, \max\left(A_{r,t}(s) + \delta, T_{\min}\right)\right), \tag{24}$$

with a value $T_{\min}$ which is above $\frac{d+1}{2}$. In the case of the BIKE 1 parameter, d is equal to 71 and $\frac{d+1}{2} = 36$. Choosing larger values for $T_{\min}$ has a significant effect on the error floor and does not deteriorate the waterfall behavior too much. This is illustrated by the following table which shows the improvement we may expect. We have studied this point in detail for a gap $\delta = 3$ and for $r = 13109$. Moreover to see the effect on the DFR we have decomposed the DFR as

$$\mathrm{DFR} = \mathrm{DFR_e} + \mathrm{DFR_w}.$$

$\mathrm{DFR_e}$ is the contribution of the DFR coming solely from the near-convergence to a near codeword. Experimental evidence shows that this term is very good for predicting the error floor. $\mathrm{DFR_w}$ is defined as $\mathrm{DFR} - \mathrm{DFR_e}$ and is also very good for predicting the waterfall behavior. For $\delta = 3$ we have the following table

$T_{\min}$	$\mathrm{DFR_e}$	$\mathrm{DFR_w}$
36	-113.046457928823	-123.378559395570
37	-118,243581226345	-123,378559395558
38	-123,405489448735	-123,378559389356
39	-128,809000162163	-123,378537439281
40	-134,809143919309	-123,310820300205

These results were obtained for the following choice of the affine function $A_{r,t}(s) = 0.0061804625681607 \cdot s + 12.8998430883248$. We see that we gain roughly 5 or 6 bits on $\mathrm{DFR_e}$ by increasing $T_{\min}$ by 1. It seems that by choosing in a better way the threshold function should allow to reach a DFR which is below $2^{-\lambda}$. This is indeed easy to achieve for $r = 13477$ with the following threshold function

- $s \in]0, 3800]$, $T(s) = 38$
- $s \in]3800, 4300]$, $T(s) = 39$
- $s \in]4300, 4550]$, $T(s) = 40$
- $s > 4550$, $T(s) = 0.005917328871862742 \cdot s + 13.57245637$.

We obtain in this case

$$\mathrm{DFR_e} \approx 2^{-129.565}, \ \ \mathrm{DFR_w} \approx 2^{-137.316}, \ \ \mathrm{DFR} \approx 2^{-129.55}.$$

7. Concluding Remarks

A new, accurate Markov model. This work highlights the central role that the near codewords put forward in [Vas21b] play in the error floor regime. We enrich the Markov model of [SV19b, Vas21b] with the size of the intersection of the error with a near codeword and obtaining a model which is key dependent, we obtain an accurate prediction of the DFR in the error floor regime, matching experimental data for various decoders and keys.
Attaining a DFR which is below $2^{-\lambda}$. If we run this Markov model on the BIKE level 1 parameter with the former threshold function, we fall short of reaching a DFR which is below $2^{-\lambda}$. We get a DFR of about 2^{-91} in this case. Moreover this problem cannot simply be addressed by taking slightly larger parameters as one could expect since the error floor kicks in just after the block size $r = 12323$ chosen for BIKE 1. However, this error floor seems to be due to the fact that the threshold function is equal to its minimum possible value $\frac{d+1}{2}$ for a very large range of values of the syndrome weight s. This can be addressed adequately by increasing this minimum threshold value from 36 to 38 in the case of BIKE 1. In this case, a simple threshold function leads to a DFR for a step by step decoder which is below $2^{-129.5}$ for a block size $r = 13477$. This is slightly less than a 10% overhead over the current BIKE 1 parameter.
Analyzing a parallel decoder. There are reasons to believe [Vas21b] that the parallel decoders used in the actual BIKE implementation produce an even lower DFR. Whereas up to now, only the beginning of the waterfall region could be explored for cryptographic parameters, this work opens a new road for exploring the ability of various decoders to attain even more efficiently the needed $2^{-\lambda}$ DFR and for understanding the effect of various weak keys on the DFR. Concerning the first point, it would be interesting to use the slightly modified

Markov chain approach followed in [ABP24a] (for the full version see [ABP24b]) to analyze parallel decoders for instance.

References

[ABB+21] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Santosh Ghosh, Shay Gueron, Tim Güneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Jan Richter-Brockmann, Nicolas Sendrier, Jean-Pierre Tillich, Valentin Vasseur, and Gilles Zémor. BIKE: Bit flipping key encapsulation - spec v4.2. `https://bikesuite.org/files/v4.2/BIKE_Spec.2021.07.26.1.pdf`, 2021.

[ABH+22] Sarah Arpin, Tyler Raven Billingsley, Daniel Rayor Hast, Jun Bo Lau, Ray A. Perlner, and Angela Robinson. A study of error floor behavior in QC-MDPC codes. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*, volume 13512 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2022.

[ABH+24] Sarah Arpin, Tyler Raven Billingsley, Daniel Rayor Hast, Jun Bo Lau, Ray Perlner, and Angela Robinson. A graph-theoretic approach to analyzing decoding failures of BIKE. Cryptology ePrint Archive, Paper 2024/1736, 2024.

[ABP24a] Alessandro Annechini, Alessandro Barenghi, and Gerardo Pelosi. Bit-flipping decoder failure rate estimation for $(v, w)-$regular codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 3374–3379. IEEE, 2024.

[ABP24b] Alessandro Annechini, Alessandro Barenghi, and Gerardo Pelosi. Bit-flipping decoder failure rate estimation for $(v, w)-$regular codes. *CoRR*, abs/2401.16919, 2024.

[BBC+21] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. Performance bounds for QC-MDPC codes decoders. In Antonia Wachter-Zeh, Hannes Bartz, and Gianluigi Liva, editors, *Code-Based Cryptography - 9th International Workshop, CBCrypto 2021, Munich, Germany, June 21-22, 2021 Revised Selected Papers*, volume 13150 of *Lecture Notes in Computer Science*, pages 95–122. Springer, 2021.

[Cha17] Julia Chaulet. *Étude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques*. PhD thesis, University Pierre et Marie Curie, March 2017.

[DGK20] Nir Drucker, Shay Gueron, and Dusan Kostic. QC-MDPC decoders with several shades of gray. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography*, pages 35–50. Springer, Cham, 2020.

[Gal63] Robert G. Gallager. *Low Density Parity Check Codes*. M.I.T. Press, Cambridge, Massachusetts, 1963.

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on MDPC with CCA security using decoding errors. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 789–815, 2016.

[HB18] Yoones Hashemi and Amir H. Banihashemi. Characterization of elementary trapping sets in irregular LDPC codes and the corresponding efficient exhaustive search algorithms. *IEEE Trans. Inf. Theory*, 64(5):3411–3430, 2018.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *Theory of Cryptography Conference*, pages 341–371. Springer, 2017.

[NSP+23] Mohammad Reza Nosouhi, Syed W. Shah, Lei Pan, Yevhen Zolotavkin, Ashish Nanda, Praveen Gauravaram, and Robin Doss. Weak-key analysis for BIKE post-quantum key encapsulation mechanism. *IEEE Trans. Inf. Forensics Secur.*, 18:2160–2174, 2023.

[Ric03] Tom Richardson. Error floors of LDPC codes. In *Proc. of the 41th Annual Allerton Conf. on Communication, Control, and Computing*, volume 41, pages 1426–1435, 2003.

[Sen24] Nicolas Sendrier. BIKE decoding failure – weak keys & error floors. Slides of the panel discussion of the fifth PQC standardization conference, April 2024.

[SV19a] Nicolas Sendrier and Valentin Vasseur. About low DFR for QC-MDPC decoding. Cryptology ePrint Archive, Paper 2019/1434, 2019. `https://eprint.iacr.org/2019/1434`.

[SV19b] Nicolas Sendrier and Valentin Vasseur. On the decoding failure rate of QC-MDPC bit-flipping decoders. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography 2019*, volume 11505 of *LNCS*, pages 404–416, Chongquing, China, May 2019. Springer.

[Tan81] Robert Michael Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, 27(5):533–547, 1981.

[Vas21a] Valentin Vasseur. *Post-quantum cryptography: a study of the decoding of QC-MDPC codes*. PhD thesis, Université de Paris, Mar 2021.

[Vas21b] Valentin Vasseur. *Post-quantum cryptography: a study of the decoding of QC-MDPC codes*. Theses, Université de Paris, March 2021.

[Vas24] Valentin Vasseur. qcmdpc_markov Github codebase, 2024. Accessed: 2024-08-26.

[VCN14] Bane Vasić, Shashi Kiran Chilappagari, and Dung Viet Nguyen. Chapter 6 - failures and error floors of iterative decoders. In David Declerq, Marc Fossorier, and Ezio Biglieri, editors, *Academic Press Library in Mobile and Wireless Communications*, pages 299–341. Academic Press, Oxford, 2014.

[WWW23] Tianrui Wang, Anyu Wang, and Xiaoyun Wang. Exploring decryption failures of BIKE: New class of weak keys and key recovery attacks. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, volume 14083 of *LNCS*, pages 70–100. Springer, 2023.

Appendix A. Computing the stationary probability distribution

The aim of this section is to show how to compute efficiently the stationary probability distribution $\mathbf{P}_\infty(\texttt{blocked}|s,t,u)$ which is the probability that, after an infinite number of iterations, we end up in the blocked state (where decoding fails) given that we started with a syndrome of weight s, an error of weight t and maximal intersection with a near-codeword of size u. There is a first step to simplify the Markov chain and to remove loops. It consists of two steps.

I. For each state (s,t,u) we compute the probability that we attain the `blocked` state after performing n loops around the state. This emulates the fact that we verified that all n bits would not get flipped because their counters is below the threshold chosen for flipping the bit. The probability $\mathbf{P}_\infty(\texttt{blocked})$ to attain the `blocked` state is given by

(25)
$$\mathbf{P}_\infty(\texttt{blocked}) = \left[\sum_{\sigma<T}\mathbf{P}_{\text{good}}(\sigma)\right]^{n_{\text{good}}}\cdot\left[\sum_{\sigma<T}\mathbf{P}_{\text{norm}}(\sigma)\right]^{n_{\text{norm}}}\cdot\left[\sum_{\sigma<T}\mathbf{P}_{\text{sus}}(\sigma)\right]^{n_{\text{sus}}}\cdot\left[\sum_{\sigma<T}\mathbf{P}_{\text{bad}}(\sigma)\right]^{n_{\text{bad}}},$$

where :
- T is the threshold which is chosen to flip a bit,
- $\mathbf{P}_{\text{good}}(\sigma)$, $\mathbf{P}_{\text{norm}}(\sigma)$, $\mathbf{P}_{\text{sus}}(\sigma)$ and $\mathbf{P}_{\text{bad}}(\sigma)$ are respectively the probability that a good/normal/suspicious/bad bit has a counter equal to σ,
- $n_{\text{good}} \stackrel{\text{def}}{=} n-d-t-u$, $n_{\text{norm}} \stackrel{\text{def}}{=} t-u$, $n_{\text{sus}} \stackrel{\text{def}}{=} d-u$ and $n_{\text{bad}} \stackrel{\text{def}}{=} u$ are respectively the number of good bits, the number of normal bits in error, the normal of suspicious bits (which are therefore not in error) and the number of bad bits.

II. We remove the loops around each state (s,t,u) by first computing the probability that there is no flip in one iteration of the sequential decoder, that we compute as

$$\mathbf{P}(\texttt{noflip}) = \sum_{\sigma<T} \mathbf{P}_{\text{good}}(\sigma)\frac{n_{\text{good}}}{n} + \mathbf{P}_{\text{norm}}(\sigma)\frac{n_{\text{norm}}}{n} + \mathbf{P}_{\text{sus}}(\sigma)\frac{n_{\text{sus}}}{n} + \mathbf{P}_{\text{bad}}(\sigma)\frac{n_{\text{bad}}}{n}. \tag{26}$$

Then according to the kind of transition from the state (s,t,u) to a state (s',t',u') we are considering, say the counter of the bit we are considering is equal to σ, which entails that $s' = s + d - 2\sigma$

- Case of a good bit : here necessarily $t' = t+1$, $u' = u$. We set
$$\mathbf{P}(s',t',u'|s,t,u) = \mathbf{P}_{\text{good}}(\sigma)\frac{n_{\text{good}}}{n}\frac{1}{1-\mathbf{P}(\texttt{noflip})}.$$
- Case of a normal bit in error : $t' = t-1$, $u' = u$. We set
$$\mathbf{P}(s',t',u'|s,t,u) = \mathbf{P}_{\text{norm}}(\sigma)\frac{n_{\text{norm}}}{n}\frac{1}{1-\mathbf{P}(\texttt{noflip})}.$$
- Case of a suspicious bit : $t' = t+1$, $u' = u+1$. We set
$$\mathbf{P}(s',t',u'|s,t,u) = \mathbf{P}_{\text{sus}}(\sigma)\frac{n_{\text{sus}}}{n}\frac{1}{1-\mathbf{P}(\texttt{noflip})}.$$
- Case of a bad bit : $t' = t-1$, $u' = u-1$. We set
$$\mathbf{P}(s',t',u'|s,t,u) = \mathbf{P}_{\text{bad}}(\sigma)\frac{n_{\text{bad}}}{n}\frac{1}{1-\mathbf{P}(\texttt{noflip})}.$$

We also restrict the range of values that s' can take (and renormalize accordingly all the transition probabilities). In the case of a perfect key we set for instance $\mathbf{P}(\sigma) = 0$ in the case where

$$\begin{aligned} s' &< 0 \\ s' &< d(2u'-t') - u'(u'-1) \\ s' &> dt' - u'(u'-1). \end{aligned}$$

Once the Markov chain has been simplified by using these rules, we now observe that on one hand

$$\mathbf{P}_\infty(\texttt{blocked}|s,t,u) = \mathbf{P}(\texttt{blocked}|s,t,u) + \sum_{s',t',u'} \mathbf{P}_\infty(\texttt{blocked}|s',t',u')\cdot\mathbf{P}(s',t',u'|s,t,u) \tag{27}$$

and on the other hand, the only transition probabilities $\mathbf{P}(s',t',u'|s,t,u)$ which are non zero are now such that $s' < s$. This suggests to compute the probabilities $\mathbf{P}_\infty(\texttt{blocked}|s,t,u)$ by starting from $s=0$ and increasing s by 1 each time and the recursion formula (27) to compute the remaining ones. For $s=0$ we use the following rule

$$\begin{aligned} \mathbf{P}_\infty(\texttt{blocked}|0,t,u) &= 0 \text{ if } t < 2d \\ \mathbf{P}_\infty(\texttt{blocked}|0,t,u) &= 1 \text{ otherwise.} \end{aligned}$$

The first rule accounts for the fact that there should be no codeword of weight less than $w = 2d$. We also set

$$\mathbf{P}_\infty(\texttt{blocked}|s,t,u) = 0 \text{ if } s < t(d-t+1) \text{ and } t \leqslant d \tag{28}$$

which basically accounts for the fact that we make the assumption that there is no near-codeword of size $t < d$ which has a smaller syndrome than a subset of size t of a near codeword of $\mathcal{N}$. This leads finally to the following algorithm

Algorithm 1 Algorithm for computing the DFR

for all t, u **do** ▷ initialization
  if $t < 2d$ **then**
    $\mathbf{P}_\infty(\texttt{blocked}|0,t,u) \leftarrow 0$
  else
    $\mathbf{P}_\infty(\texttt{blocked}|0,t,u) \leftarrow 1$
for all s,t,u s.t. $s < t(d-t+1)$ and $t \leqslant d$ **do**
  $\mathbf{P}_\infty(\texttt{blocked}|0,t,u) \leftarrow 0$
for $s = 1$ to $s_{\max}$ **do** ▷ main loop
  for all t, u **do**
    $\mathbf{P}_\infty(\texttt{blocked}|s,t,u) \leftarrow \sum_{s'<s,t',u'} \mathbf{P}_\infty(\texttt{blocked}|s',t',u') \cdot \mathbf{P}(s',t',u'|s,t,u) + \mathbf{P}(\texttt{blocked}|s,t,u)$
$\texttt{DFR} \leftarrow \sum_{s,t,u} \mathbf{P}_\infty(\texttt{blocked}|s,t,u)\,\mathbf{P}(s,t,u)$.

Appendix B. Computing the initial state distribution

Let random variables:

- S denote the syndrome weight
- T denote the total error weight
- U denote the maximal intersection size between the error vector and any near codeword
- B denote the index of the block (0 or 1) containing the near codeword with maximal intersection

For a QC-MDPC code of row weight d where the degree distributions n_Δ are known for both blocks.

Additionally define auxiliary random variables:

- T_0 and T_1 denote the error weights in blocks 0 and 1 respectively, satisfying $T_0 + T_1 = T$
- U_0 and U_1 denote the maximum intersection sizes between the error vector and any near codeword in blocks 0 and 1 respectively

In this section, we compute $\mathbf{P}(S = s, U = u, B = b|T = t)$. We will use the formula

$$\mathbf{P}(S = s, U = u, B = b|T = t) = \mathbf{P}(U = u, B = b|T = t)\,\mathbf{P}(S = s|U = u, B = b, T = t).$$

B.1. **Distribution $\mathbf{P}(U = u, B = b|T = t)$.** For a realization $T_0 = t_0$, $T_1 = t_1$ summing to t, where t_i is the weight in block i, the probability of this partition occurring is given by the hypergeometric distribution

$$\frac{\binom{r}{t_0}\binom{r}{t_1}}{\binom{n}{t}}.$$

For each t_i, the distribution of intersections with any particular near codeword in block i follows a hypergeometric distribution with probability mass function

$$f_i(k) = \frac{\binom{d}{k}\binom{r-d}{t_i-k}}{\binom{r}{t_i}}$$

for intersection size k. Let $F_i(k)$ denote the corresponding CDF:

$$F_i(k) = \sum_{j=0}^{k} f_i(j)$$

Since we take the maximum of r independent such random variables, the CDF of the maximum F_i is:

$$F_i^r(k) = (F_i(k))^r$$

And the PMF of the maximum is:

$$\mathbf{P}(U_i = k|T_i = t_i) = F_i^r(k) - F_i^r(k-1)$$

Then there are three cases to consider:

- Block 0 contains a unique maximum intersection of size u:

$$\mathbf{P}(U_0 = u|T_0 = t_0)\,\mathbf{P}(U_1 < u|T_1 = t_1)\,\mathbf{P}(T_0 = t_0, T_1 = t_1)$$

- Block 1 contains a unique maximum intersection of size u:

$$\mathbf{P}(U_1 = u|T_1 = t_1)\,\mathbf{P}(U_0 < u|T_0 = t_0)\,\mathbf{P}(T_0 = t_0, T_1 = t_1)$$

- Both blocks have maximum intersection of size u:

$$\mathbf{P}(U_0 = u|T_0 = t_0)\,\mathbf{P}(U_1 = u|T_1 = t_1)\,\mathbf{P}(T_0 = t_0, T_1 = t_1)$$

Cases 1 and 2 determine $b = 0/1$ respectively, and case 3 split evenly between them.

B.2. **Distribution $\mathbf{P}(S = s|U = u, B = b, T = t)$.** To compute this probability, we need to track both:

(1) The number of check nodes involved in an odd number of errors
(2) The total number of errors connected to each check node

The total number of edges connected to errors in the Tanner graph must equal dt.

We partition the check nodes based on their degree Δ in $\mathscr{G}$, with counts given by n_Δ. For each degree Δ, we use $\rho_{\Delta,\ell}$ for the distribution of the errors in a parity check equation. Here, $\rho_{\Delta,\ell}$ gives the probability that a check node of degree Δ in $\mathscr{G}$ involves exactly ℓ errors, where these errors are distributed between: the u bits common to both the error vector and nearest near codeword and the remaining $t - u$ error bits not in the near codeword.

Let R_i be the random variable for the number of errors connected to check node i:

$$\mathbf{P}(R_i = \ell | \deg(i) = \Delta) = \rho_{\Delta,\ell}$$

where $\deg(i)$ is the node's degree in $\mathscr{G}$.

Our goal is to compute, for all values of s:

$$\mathbf{P}\Big(\sum_i R_i = dt, \text{exactly } s \text{ values of } R_i \text{ are odd and } r-s \text{ are even} \Big| T = t, U = u\Big)$$

given the known distribution of degrees. Then we can easily deduce

$$\mathbf{P}\Big(\text{exactly } s \text{ values of } R_i \text{ are odd and } r-s \text{ are even} \Big| \sum_i R_i = dt, T = t, U = u\Big)$$

Which is what we will set for $\mathbf{P}(S = s | U = u, B = b, T = t)$.

We decompose s into components by degree, where s_Δ represents the number of degree-Δ parity check equations in $\mathscr{G}$ with an odd number of errors, for $\Delta = 0, 1, \ldots, \Delta_{\max}$. This gives us $s = s_0 + s_1 + \cdots + s_{\Delta_{\max}}$. The number of possible configurations with this decomposition is given by the multinomial coefficient:

$$\binom{r}{s_0, n_0 - s_0, s_1, n_1 - s_1, \ldots, s_{\Delta_{\max}}, n_{\Delta_{\max}} - s_{\Delta_{\max}}}$$

For each individual degree Δ, we compute:

$$\mathbf{P}\Big(\sum_{i \text{ has degree } \Delta} R_i = \ell, \text{exactly } s_\Delta \text{ values of } R_i \text{ are odd and } n_\Delta - s_\Delta \text{ are even} \Big| T = t, U = u\Big)$$

using convolutions. We then convolve these distributions and multiply by the multinomial coefficient. Finally, we retain only the values where the sum equals $d \cdot t$ and normalize the distribution to obtain our desired probability.

The actual computational procedure consists of several steps. First, for each degree Δ, we create vectors mapping ℓ to $\rho_{\Delta,\ell}$. These vectors are then split into their odd and even components without normalization. Next, for each s_Δ in $\{0, \ldots, n_\Delta\}$, we compute s_Δ convolutions of the odd vector and $n_\Delta - s_\Delta$ convolutions of the even vector, dividing the results by $s_\Delta!$ and $(n_\Delta - s_\Delta)!$ respectively. We then perform a series of convolutions, first within each fixed Δ group we obtain for each possible value s_Δ the distribution of the sum of R_i. We then convolve across different Δ groups thus obtaining the sum of s_Δ together with the corresponding distributions of the sum of R_i. The final result is multiplied by $r!$ to account for the multinomial coefficient. Finally, we retain only the values where the sum equals $d \cdot t$ and normalize the distribution to obtain our desired probability.

Appendix C. Additional Information on the Experiments

C.1. **Affine part of the threshold function for the curves of Figure 14.**

TABLE 1. Table giving the affine part $A_{r,t}$ of the threshold function corresponding to Figure 14. The threshold $T_r(s)$ is given for a block length r and a syndrome weight s by the formula $T_r(s) = \min\left(d, \max\left(A_{r,t}(s), \frac{d+1}{2}\right)\right)$.

r	α	β
9283	0.007835394510038963	13.297180113728302
9661	0.007393105300920274	13.567391558872282
10037	0.007061848501098616	13.594659657463222
10427	0.006798500986468654	13.456553521076303
10789	0.006606617361836905	13.228860262727327
11171	0.006444309408601018	12.921143566520255
11549	0.0063141209528076135	12.572979649160397
11933	0.006205362314938619	12.192085979992001
12323	0.0061135837435673315	11.789323238365313
12739	0.00603182965107241	11.351398154970589
13109	0.005970219758260547	10.96007467501192
13477	0.005917328871862742	10.572456366568613
13859	0.0058697311301313835	10.173985156386603
15373	0.005733664525418103	8.655332501782194
16901	0.00565184187736275	7.233670336326141
18427	0.005604464463216208	5.915303733756538
19949	0.005580554453148446	4.683511294611625
21467	0.0055736761566337325	3.521321141011274
23003	0.005579882422098863	2.399420709618189
24533	0.005596416334069782	1.32559949672806