HALF-PRODUCT CODES

A Thesis

by

SANTOSH KUMAR EMMADI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2014

Major Subject: Electrical Engineering

ABSTRACT

A class of codes, half-product codes, derived from product codes, is characterized. These codes have the implementation advantages of product codes and possess a special structural property which leads them to have larger (at least $\frac{3}{2}$ times more) minimum distance than product codes. With the same length and rate, they have better scaling in the error floor than product codes. They also have a larger minimum stopping-set size under iterative decoding which provides better scaling. The main results of this thesis are summarized as follows:

1. Encoding and decoding methods of half-product codes are described.
2. The minimum distance of these codes is derived, and proved to be at least $\frac{3}{2}$ times larger than that of the product codes for the same rate and block length.
3. The performance of iterative decoding in the error floor region is analyzed by enumerating the minimum stopping-set patterns for these codes. The results are compared with product codes.

Simulations are also performed to compare the half-product codes with product codes. We conclude that half-product codes scale better in the error floor than product codes in the region where the minimum stopping-sets dominate the error floor, and that they have same threshold as product codes when rate is same and code length is increased to infinity .

DEDICATION

For believing in me, and for all the unyielding love and support,

I dedicate this thesis to you, my brother, *Mahesh Emmadi*.

ACKNOWLEDGEMENTS

I would like to thank my thesis committee members for their time and effort in conducting my defense. Especially, I would like to thank Dr. Krishna Narayanan for his patient, insightful and very fruitful discussions with me regarding this work. A part of this work couldn't have been possible if it were not stimulated by his curiosity.

I would like to thank my fellow group members, and friends, Avinash Vem, Santhosh Vanaparthy, and Swanand Kadhe for their constant support, motivation, and encouragement throughout writing this report. Discussions with them were engrossing and helpful to this work.

I would like to thank my family for their continuous, granted for life, love and support. I would like to thank my extended family, my more-than-friends here at College Station, Sriharsha Madala, Shantanu Deshpande, Abhishek Yada, Amruth Juturu, Vamseedhar Raja, Srikanth PVK, and Sindhu Chittireddy for sharing, caring, and being there for me in every stage of my life here, and for the life ahead. It is because of the interactions with these wonderful people that I have learnt many things in life.

This section is not complete without thanking the one important person, my advisor, Dr. Henry Pfister. Since the beginning of my thesis work with him, he has been the epitome of the supportiveness. He has been patient with me, and had given me all the time and freedom I needed. Discussions with him are always inspiring. I would like to thank him for believing in me through my lows and highs. And, I would like to thank him for taking me under his wing and giving me the opportunity to work with him.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

1. INTRODUCTION

This thesis considers a classical family of codes called product codes (PCs) and develops the theory for related class of codes, called half-product codes (HPCs). In particular, this thesis claims that HPCs are superior to PCs when compared fairly.

1.1 Background

Even though product codes were proposed a long time ago [1], they are still being used in practice because of their efficient implementation. In particular they are used for high data rate applications through low error rate channels such as Optical Transport Networks (OTN), where the data rate can be 100 Gbit/sec. These are also used in the applications where frame lengths are in excess of 10^5 bits and the code rates are greater than 0.9 [2]. These codes can be found used in the CD standard IEC-908, the CD-ROM standard ECMA-130, and in the DVD standard [3].

Product codes are useful for a variety of reasons including their burst-error correcting capabilities [4]. They were first introduced by Elias in [1] around five decades ago. Elias showed that the minimum distance of binary product codes is equal to the product of the minimum distances of their component codes. They have been extensively studied ever since, because of their ease in the implementation. There are many decoders proposed for these codes, but one among them stood out to be efficient in implementation, i.e., the cascade decoding, which was introduced by N. Abramson in [5]. In his words, the cascade decoder is an efficient decoder because of two things, one, they can be easily built, and two, instead of correcting all error patterns with weight less than some fixed value and no error patterns of greater weight, the cascade decoders correct many more error patterns beyond their guaranteed correction capability [5]. There are many versions of row-column decoding that have been proposed and analyzed in the literature. There are also many modifications to product codes that have been proposed for specific applications [2]. In [2], Justesen analyzes the performance of the product codes and related structures with iterative decoding. In the same paper, Justesen suggests an idea of a possible modified construction of the product codes called *half-product codes*. However, the paper doesn't provide many details related to these codes. This thesis explores some important properties and details of half-product codes.

1.2 Our work

In our work on the half-product codes, the encoding and decoding methods of these codes are suggested. Their properties such as d_{min}, minimum stopping set patterns, and enumeration of these patterns are derived. Then, the codes are compared with the parent codes of these, the product codes. The error floor of these codes have been analytically calculated and is compared with the error floor of the product codes. It is concluded that the error floor in the half-product codes scales better than that of the product codes. Simulation results are also presented and the performance is discussed.

1.3 Organization

In this report, chapter 2 gives an introduction, and a summary on the product codes. Chapter 3 presents our work in the half-product codes. It details on encoding and decoding methods. It presents the derivation and proof on d_{min}(lower bound) of the half-product codes. It gives the error floor analysis, and concludes with a comparison to the product codes. Chapter 4 shows the simulations performed in this work, discusses the results obtained, and describes our conclusions.

2. PRODUCT CODES - A SUMMARY

In this chapter, some required description on the product codes is provided. This chapter also serves to build the notation required for the next chapters.

2.1 Preliminaries

Product codes (PCs) are formed by arranging the code symbols in a matrix form with rows forming one component code and columns forming another [5]. Let $C_1(n_1, k_1, d_1)$ and $C_2(n_2, k_2, d_2)$ be two binary linear codes of lengths n_1 and n_2, dimensions k_1 and k_2, and minimum distances d_1 and d_2, respectively. The product code $\mathcal{C} = C_1 \otimes C_2$ consists of all matrices whose rows are in C_1 and whose columns are in C_2. The product code $\mathcal{C}$ is a (n, k, d) code where its length is $n = n_1 n_2$, dimension is $k = k_1 k_2$, and its minimum distance is $d = d_1 d_2$. The structure of $\mathcal{C}$ is illustrated in Figure 2.1.

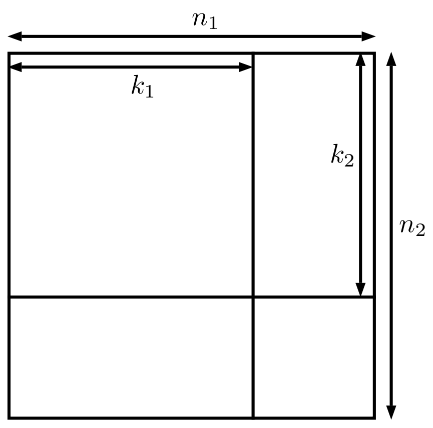

Figure 2.1: Structure of product code.

2.2 Encoding

The top left part of the matrix contains $k_1 k_2$ information symbols. The first k_1 columns are encoded into n_1 columns so that each row is contained in C_1. Now, after the first k_2 rows are encoded, they are encoded into n_2 rows so that each column is contained in C_2. This process of encoding can also be seen in terms of the generator matrix. The generator matrix of the product code is the kronecker or the tensor product of the generator matrices of the component codes [6]. The encoding operation can be concisely given in the following theorem.

Theorem 2.1 (*Encoding*)**.** *If the component codes of the product code $\mathcal{C}$, given by $C_1(n_1, k_1)$ and $C_2(n_2, k_2)$, have generator matrices G_1 and G_2 respectively, then the encoding of the message matrix, $M_{k_1 \times k_2}$, into a codeword $c \in \mathcal{C}$ is given by*

$$c = G_1^T M G_2.$$

The proof for this theorem can be easily seen by the construction process given above.

2.3 Decoding

The decoding of product codes can be done in an iterative fashion. In each step of iteration, some errors are corrected, until the codeword is corrected or can no longer be corrected. The structure of the product code also suggests a decoding process—the process where decoding of the rows and columns occur independently. So, the natural method of decoding is the cascade decoding, introduced by N. Abramson in [5], in which a received codeword is passed through the cascade of row and column decoders. The row decoder decodes all rows at once, and then the column decoder decodes all columns at once. This process is carried out iteratively until the decoding may no longer result in a new word (see figure 2.2).

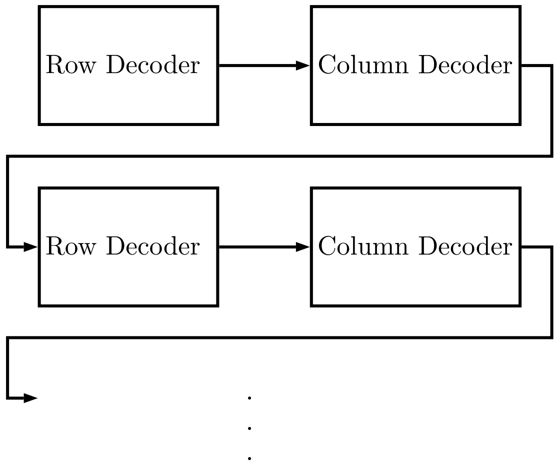

Figure 2.2: Cascade Decoding.

Let us see the decoding capability of the product codes. The structure of the product codes with cascade decoding method makes a subtle difference in the decodability of the codes, to the product codes from the regular linear block codes. The difference is that the relation between the bound for the number of errors decodable and the minimum distance is no longer valid. That is, the number of correctable errors by the cascade decoder for the product codes is not $\frac{d_{min}-1}{2}$. This is because of certain patterns of errors occurring, called *stopping-set patterns.* For example let us consider the following simple stopping-set pattern.

$$\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Figure 2.3: 4 error stopping-set

Figure 2.3 shows a pattern of errors that is uncorrectable when 1-error correcting component codes, for example—$(7,4)$ Hamming codes, are used for the rows and columns. The minimum distance of each component code is 3 and so the minimum distance of the product code formed by these component codes is 9. We see that this pattern can not be reduced in row decoding as the pattern has 2 errors in two rows and the row component code can only correct 1-error, and similarly, it can not be reduced in column decoding. So, this pattern remains uncorrected even though 4 errors is correctable by minimum distance decoding because $4 \leq \frac{9-1}{2}$.

2.4 Decoding Capability of Product Codes

It is clear from above that the relation between the number of correctable errors and the minimum distance of the product code is different from that of a regular linear block code. Now, we will compute the expression for the minimum number of errors that are uncorrectable by cascade decoding of product codes, as is given in [5].

Let the row component code, $C_1(n_1, k_1)$, be a t_1-error correcting, and let the column component code, $C_2(n_2, k_2)$, be a t_2-error correcting. Then, we define the minimum number of errors that can not be corrected by the product code $\mathcal{C}(n_1 n_2, k_1 k_2)$ to be s_{min}^{PC}. Any error pattern with at least (t_2+1) errors in each of at least (t_1+1) different columns will be a stopping-set pattern, and hence, can not be corrected by the cascade decoding procedure. So, the minimum number of errors that can not be corrected equals the number of errors in the minimum stopping-set. Hence,

$$\begin{aligned} s_{min}^{PC} &= (t_1+1)(t_2+1) \\ &= t_1 t_2 + t_1 + t_2 + 1 \end{aligned}$$

But according to the minimum Hamming distance relation, the code can correct any pattern of t errors or less with minimum distance decoding, where

$$\begin{aligned} t &= \frac{d_{min} - 1}{2} \\ &= \frac{(2t_1 + 1)(2t_2 + 1) - 1}{2} \\ &= 2t_1 t_2 + t_1 + t_2 \end{aligned}$$

So, there is a difference of $t - s_{min}^{PC} + 1 = t_1 t_2$ in the number of errors that can not always be corrected due to the structure of the decoder. Because of these error patterns, cascade decoding gives rise to an error floor in the low error rate region in the plot of probability of codeword error against the error rate of the channel.

Now, we will enumerate the minimum stopping-set patterns. To enumerate the number of minimum weight stopping-set error patterns is to enumerate the number of ways of choosing $(t_1 + 1)$ columns in n_1 columns and $(t_2 + 1)$ rows in n_2 rows. Then, the following is imminent.

Enumeration 1. *The total number of error patterns of minimum stopping-sets of size s_{min}^{PC} is given by*

$$N_{PC} = \binom{n_1}{t_1 + 1} \binom{n_2}{t_2 + 1}$$

2.5 Stopping-set Performance of Product Codes

In this section, we will evaluate the error floor performance, i.e. the error performance due to the minimum stopping-sets in product codes. In the previous section, it has already been explained that, in the low error rate regions, as N tends to infinity, the error floor performance is dominated by the stopping-set patterns. Now, we will calculate this error probability, due to these dominating stopping-sets. In the enumeration 1, the total number of possible error patterns of minimum stopping-sets has been evaluated. Using this, the probability of error in the error floor is given by,

$$\begin{aligned}
P(\text{Error}) &\geq P(\text{Error due to min stopping-sets}) \\
&\approx (\text{No. of min stopping-sets}) \times p^{(\text{weight of the min stopping-set})} \\
&= N_{PC} \times p^{\frac{(t_1+1)(t_2+1)}{2}} \\
&= \binom{n_1}{t_1+1}\binom{n_2}{t_2+1} p^{\frac{(t+1)(t+2)}{2}} \qquad \text{(PC Error floor performance)}
\end{aligned}$$

3. HALF-PRODUCT CODES

Half-product codes (HPCs) are introduced in [2], as a variation of product codes, without describing the details of their encoding, decoding, or performance. In this chapter, we will explain the details of encoding and decoding procedures. Also, some interesting properties pertaining to their decoding capability are derived.

Half-product codes are formed by arranging the code elements in a symmetric matrix with rows and columns from the same component codes, and taking only the upper (or lower) half of this matrix. The diagonal elements are all taken to be zeros. The half-product code formed by the component code $C(n, k)$ is given by $\mathcal{C}(N, K)$ with length $N = \frac{n(n-1)}{2}$ and dimension $K = \frac{k(k-1)}{2}$. Even though we only take the half of the encoded matrix for transmission, we will consider its equivalent full form for analysis and other purposes. The codeword, in full symmetric matrix form, is called the *codeword in full form*, and the upper half triangle is called the *codeword in half form*. Following Figure 3.1 gives an example for $(21, 6)$ half-product codeword, formed by $(7, 4)$ Hamming code as component code.

$$\begin{array}{ccccccc}
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
 & 0 & 1 & 1 & 1 & 0 & 0 \\
 & & 0 & 1 & 0 & 0 & 0 \\
 & & & 0 & 1 & 0 & 0 \\
 & & & & 0 & 0 & 0 \\
 & & & & & 0 & 0 \\
 & & & & & & 0
\end{array}
\qquad\qquad
\begin{array}{ccccccc}
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Half form Full form

Figure 3.1: $(21, 6)$ Half-product codeword in half form, and full form

3.1 Encoding

The encoding of the half-product codes is illustrated in the following Figure 3.2. Consider an $k \times k$ matrix with unfilled positions.

$$\begin{array}{ccccc} \square & \square & \square & \dots & \square \\ \square & \square & \square & \dots & \square \\ \square & \square & \square & \dots & \square \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \square & \square & \square & \dots & \square \end{array}$$

Unfilled matrix.

$$\begin{array}{ccccc} 0 & m_1 & m_2 & \dots & m_{k-1} \\ \square & 0 & m_k & \dots & m_{2k-3} \\ \square & \square & 0 & \dots & m_{3k-5} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \square & \square & \square & \dots & 0 \end{array}$$

Upper triangle filled.

$$\begin{array}{ccccc} 0 & m_1 & m_2 & \dots & m_{k-1} \\ m_1 & 0 & m_k & \dots & m_{2k-3} \\ m_2 & m_k & 0 & \dots & m_{3k-5} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{k-1} & m_{2k-3} & m_{3k-5} & \dots & 0 \end{array}$$

Message matrix, $M_{(k \times k)}$ in full form.

Figure 3.2: Message matrix of the half-product codes.

Let us start filling the matrix row by row. The first row is filled with 0 in the diagonal position and $k-1$ information symbols in the next $k-1$ positions. The second row is with 0 in the diagonal position and $k-2$ information symbols in the next $k-2$ positions. Similarly, the third row is started with 0 in the diagonal position and $k-3$ symbols in the following $k-3$ positions. This process continues until the $(k-1)^{th}$ row is filled with 0 in the diagonal position and 1 information symbol in the following position. Now, all the available positions for information symbols are filled. Then, k^{th} row is only filled in the diagonal position with 0. Now, we have upper triangle of size $k \times k$ filled with information symbols. This is the message matrix in the half form. As the half-product codes are symmetric in the full form, to fill the up the message matrix, we take the lower half triangle to be the transpose of the upper half triangle. This way, the lower triangle of size $k \times k$ is also filled.

We are now left with $k \times k$ message matrix, in full form, with diagonal elements equal to 0. Let us denote it by M. The encoding of this, now, is same as the encoding of a message matrix in the product codes, except in this, both row and column component codes are same. Each row of length k is encoded into a codeword of length n, contained in the component code $C(n, k)$ and then, each column of length k is encoded into a codeword of length n, contained in the same component code $C(n, k)$. This forms the full symmetric product code. Taking the upper (or, lower) half triangle gives the half-product code $\mathcal{C}(N, K)$.

Theorem 3.1. *Encoding of the above message matrix, M will give rise to a half-product codeword matrix which is symmetric and has all the the diagonal positions as zeros if the component code is in $GF(2^m)$.*

Proof. Let the generator matrix of systematic code $C(n, k)$ be $G = [I\ P]_{k\times n}$, where I is the Identity matrix of size $k \times k$ and P is the parity part of the generator matrix which generates the parity symbols in the codeword. Let it be given by

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n-k} \\ p_{11} & p_{12} & \cdots & p_{1n-k} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k1} & p_{k2} & \cdots & p_{kn-k} \end{pmatrix}$$

The message matrix in full symmetric form be given by M, and let the corresponding half-product codeword in full form be given by c. Then, the encoding of the half-product code $\mathcal{C}$ in full form is given by the following operation, from the theorem 2.1.

$$\begin{aligned} c &= G^T M G \\ &= [I\ P]^T M [I\ P] \\ &= \begin{pmatrix} M & MP \\ P^T M & P^T M P \end{pmatrix} \end{aligned}$$

The diagonal elements of c are, collectively, the diagonal elements of M and the diagonal elements of $P^T MP$. We already know that the diagonal elements of M are all zeros. So, we only need to find the diagonal elements of $P^T MP$. They are given by,

$$(P^T MP)_{ii} = \sum_{l=1}^{k} \sum_{j=1}^{k} p_{ji} M_{jl} p_{li}, \qquad \text{for } i = 1, 2, \ldots, k$$

But we already know that matrix M is symmetric and has zeros as the diagonal elements, so

$$\begin{aligned} M_{jl} = M_{lj} &= m_r, &\quad \text{if } j \neq l \\ M_{jl} &= 0, &\quad \text{if } j = l \end{aligned}$$

where r is a function of l and j.

Thus,

$$\begin{aligned} (P^T MP)_{ii} &= \sum_{l=1}^{k} \sum_{j>l}^{k} (p_{ji} M_{jl} p_{li} + p_{li} M_{lj} p_{ji}) \\ &= \sum_{l=1}^{k} \sum_{j>l}^{k} m_r (p_{ji} p_{li} + p_{li} p_{ji}), \qquad \text{for } i = 1, 2, \ldots, k \end{aligned}$$

But, if the elements of G, (and hence, the elements of P) belong to $GF(2^m)$, then $p_{ji}p_{li} = \alpha \in GF(2^m)$ and so, $\alpha + \alpha = 0$. Thus,

$$(P^T MP)_{ii} = 0, \qquad \text{for } i = 1, 2, \ldots, k.$$

Therefore, the diagonal elements of $P^T MP$ are zeros. And so, all the diagonal elements of the half-product codeword, c in full form, are all zeros. □

The above theorem guarantees zeros in every diagonal position of the half-product codeword if the symbols are from $GF(2^m)$, and the diagonal positions of the message matrix are all taken to be zeros. The reason why we need the zeros in the diagonal is because the diagonal positions may be weakly protected when compared to the off diagonal positions. We can observe that in the half-product code matrix structure, each code element that is in non diagonal position belongs to two different component codewords, whereas the element in the diagonal position belongs to only one codeword. That's why these weakly protected positions are fixed with zeros in the half-product codes.

3.2 On d_{min} of Half-product Codes

We have seen the encoding of the half-product codes in the previous section. Now, we will see an important property of any linear code, i.e, the minimum distance, d_{min} for the half-product code. In this section, we will derive a lower bound on d_{min} for the half-product codes.

We have already known about the minimum distance of the product codes from the previous chapter. Its expression is given by the product of the minimum distances of the component codes ($d_{min} = d_1 \times d_2$). But, it is not as simple as that for half-product codes. Half-product codes in full form are product codes which are symmetric and have zeros in the diagonal. These two constraints restrict the patterns of the codewords and make them not so easy to comprehend. The same applies to the minimum weight pattern also. From the basic coding theory, it is known that in a linear code, the minimum distance is equal to the weight of the minimum weight codeword. So, the art of finding the minimum distance of a code translates to finding the minimum weight codewords in that code. We use this principle in deriving the lower bound for minimum distance of the half-product code, in the following.

Definition 3.1 (*Support Set*)**.** The support set of a set of codewords X, denoted by $\chi(X)$, is the set of locations $i \in [n] = \{1, 2, \ldots, n\}$ such that $x_i \neq 0$ for some $x \in X$.

Definition 3.2 (*2nd Generalized Hamming weight*)**.** The second generalized hamming weight of the code C, denoted by $d_2(C)$, is the length of the minimum support set of the two non-zero codewords taken at a time.

Lemma 3.2. *For a linear code C with $d_{min} = d$,*

$$d_2(C) \geq \begin{cases} \frac{3d}{2} & \text{if } d \text{ is even} \\ \frac{(3d+1)}{2} & \text{if } d \text{ is odd.} \end{cases}$$

Proof. Let us first consider the case when d is even. Take two nonzero codewords from C into the set X. Each of them has weight of at least d, and the distance between them is at least d. So, the support of X, $\chi(X)$, has at least $d + \frac{d}{2}$ elements. Hence, $d_2(C) = |\chi(X)| \geq \frac{3d}{2}$. The equality happens when X has two codewords of minimum weight, and are separated by the minimum distance in C.

Now, let us consider the case when d is odd. Take two nonzero codewords from C into the set X. One of them has weight of at least d, then the other has the weight of at least $d+1$, because the distance between them is at least d, which is odd. So, the support of X, $\chi(X)$ has at least $d+\frac{(d+1)}{2}$ elements. Hence, $d_2(C)=|\chi(X)|\geq\frac{3d+1}{2}$. The equality happens when X has one codeword of minimum weight, d, in C, and is separated with other codeword of weight $d+1$ by minimum distance, d, or has one codeword of minimum weight, d, in C, and is separated with other codeword of weight d by minimum distance, $d+1$. $\square$

Theorem 3.3. *Let a linear code C with $d_{min}=d$ form a half-product code $\mathcal{C}$. Then, the minimum distance of $\mathcal{C}$ is given by*

$$D_{min}\geq\begin{cases}\frac{3d^2}{4} & \textit{if } d \textit{ is even}\\ \frac{(d+1)(3d-1)}{4} & \textit{if } d \textit{ is odd}\end{cases}$$

Proof. The first bound can be proved like this. In a half-product codeword, take the support set of the component codewords. Let it be S. Then,

$$\begin{aligned}|S| &\geq |\text{Support of two distinct codewords}|\\ &\geq d_2(C) && \text{(by definition)}\\ &\geq \frac{3d}{2} && \text{(by lemma 3.2)}\end{aligned}$$

The rows (columns) corresponding to the elements of S are all non-zeros, because there exists a row codeword that has a non-zero in the position corresponding to each element of S. And, a non-zero codeword has weight of at least d. So, the weight of the half-product codeword is at least $|S|$ times d, and we have

$$|S|\times d\geq\begin{cases}\frac{3d^2}{2} & \text{if } d \text{ is even}\\ \frac{d(3d+1)}{2} & \text{if } d \text{ is odd}\end{cases}$$

So, in the half form,

$$D_{min} \geq \begin{cases} \frac{3d^2}{4} & \text{if } d \text{ is even} \\ \frac{d(3d+1)}{4} & \text{if } d \text{ is odd} \end{cases}$$

The above part derives only the lower bound on d_{min}. This still doesn't prove its tightness, especially in the case when d is odd. In other words, it doesn't give a way to construct the minimum weight half-product codeword. In the following, we are going to construct a minimum weight codeword for the half-product code $\mathcal{C}$ formed by the linear component code C of $d_{min} = d$.

The process of constructing a minimum weight half-product codeword is done by choosing a codeword, from the set of component codewords C, for each row (column) to complete the half-product codeword in the full form. The number, l, of distinct codewords taken from C to form the half-product codeword is varied as $l = 1, 2, 3, \ldots$ For each l, if possible, a minimum weight half-product codeword is constructed and its weight is calculated. It will be proved that the minimum weight is achieved for a minimum half-product codeword constructed using $l = 3$. For $l = 4$ or more, the half-product codeword constructed will have weight more than the minimum weight.

Let us first consider the case where a half-product codeword is formed by single component codeword ($l = 1$). It is clear from the encoding process of the half-product code that every row has zero in the diagonal position. If we are to form a half-product codeword from a single component codeword, let's say c, then it must have zeros in the diagonal positions of the rows it is in. Let it be filled in i^{th} row, then, $c_i = 0$. So, i^{th} column will have zeros in the positions corresponding to the rows with c. That implies, $c_i = 0 \; \forall i$. Hence, the (only) half-product codeword formed by a single component codeword is the all zero codeword. In other words, we can not construct a non-zero half-product codeword with just a single component codeword.

Second, consider the case where a half-product codeword is constructed by two non-zero component codewords ($l = 2$). Let us consider the half-product codeword X formed by two distinct non-zero component codewords c_1 and c_2. Let c_1 and c_2 have support sets S_1 and S_2. Let i^{th} row, denoted by X_i, be filled by c_1, so $X_i = c_1$. Then, $X_{ij} = c_{1j} \neq 0 \; \forall j \in S_1$. Because of the symmetry, we have $X_{ji} = X_{ij} \neq 0 \; \forall j \in S_1$. So, j^{th} row is non-zero for each $j \in S_1$. As the diagonal position $X_{ii} = c_{1i} = 0$, we have $i \notin S_1$ for each row i in which c_1 is filled. Therefore, each row X_j such that $j \in S_1$ can not be filled by c_1. Let us fill those with component codeword c_2. Then, $X_{jk} = c_{2k} \neq 0 \; \forall j \in S_1, \; \forall k \in S_2$. Because of the symmetry, we have $X_{jk} = X_{kj} \neq 0 \; \forall j \in S_1, \; \forall k \in S_2$. So, k^{th} row is non-zero for each $k \in S_2$. As the diagonal position $X_{jj} = c_{2j} = 0$, we have $j \notin S_2$ for each row $j \in S_1$. Therefore, each row X_k such that $k \in S_2$ can not be filled by c_2. We can fill these rows with c_1 if and only if S_1 and S_2 are disjoint. Now, the number of non-zero rows is at least $|S_1|+|S_2|$. To get the minimum codeword, we consider exactly $|S_1| + |S_2|$ rows. In that case, we have $|S_1|$ number of rows filled with c_2 and $|S_2|$ number of rows filled with c_1. The only constraint to construct this half-product codeword is that the component codewords c_1 and c_2 should have disjoint support sets. If such a codeword pair is not in the component code, then we can not form a half-product codeword just by two non-zero codewords. If the component code C has such a pair with weights d_1 ($= |S_1|$) and d_2 ($= |S_2|$), then a half-product codeword is formed by these, and will have the weight which equals $d_1 d_2 + d_2 d_1 = 2 d_1 d_2$, which is at least $2d^2$, as each $d_1, d_2 \geq d$.

Now, consider the case where a half-product codeword is constructed by three non-zero component codewords ($l = 3$). The following construction procedure gives a minimum half-product codeword. Take two non-zero codewords $c_1, c_2 \in C$ such that $|\chi(\{c_1, c_2\})| = d_2(C)$. Consider $c_3 = c_1 + c_2$, is also a codeword $\in C$, as C is a linear code and sum of two codewords is also a codeword. We observe that $\chi(\{c_1, c_2, c_3\}) = \chi(\{c_1, c_2\})$ and denote this by S. This is because of addition of zero positions in c_1 with zero positions in c_2 gives zero positions in c_3, so there are no extra non-zero positions in c_3. Fill the rows (columns) corresponding to the indices from S with these codewords such that the diagonal positions are always zeros. That is, a codeword c_1, c_2, or c_3 is filled in an i^{th} row if the i^{th} position of the respective codeword c_1, c_2, or c_3 is a zero, for all $i \in S$. We observe that this can be uniquely filled, because no zero-position $i \in S$ in a codeword is repeated in other two codewords. This gives a symmetric structure because of the following arguments. Let the half-product codeword formed be X, and its i^{th} row be denoted by X_i. Let $J = S - \chi(c_1)$, which has the elements in S that are zero positions in c_1, i.e. $c_{1j} = 0 \ \forall j \in J$. From the construction process, an i^{th} row is filled with c_1 if $i \in J$, i.e $X_i = c_1 \ \forall i \in J$. Therefore, $X_{ij} = c_{1j} = 0 \ \forall i \in J, \ \forall j \in J$, which implies that $X_{ij} = X_{ji} = 0 \ \forall i \in J, \ \forall j \in J$. Similarly, we can argue that $X_{ij} = X_{ji} = 1 \ \forall i \in \chi(c_1), \ \forall j \in \chi(c_1)$. This proves the symmetry of the rows and the corresponding columns filled with c_1 using the above construction procedure. Similarly, one can prove it for other codewords, c_2 and c_3. This way, all the rows and columns corresponding to S are filled with c_1, c_2, or c_3, and are symmetric. Then the remaining rows, and so columns, are all filled with zeros. This construction gives a half-product codeword which is of minimum weight.

We now refer to the example in Figure 3.3, which gives an illustration of the construction process for a codeword of minimum weight 6. In the figure we consider only the rows and columns corresponding to S that are rearranged with a unique mapping such that the first codeword, c_1, has all non-zeros continuously.

This construction gives a minimum weight codeword, because of the following.

$$\begin{array}{ccccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & & & & & & & & \\
1 & & & & & & & & \\
1 & & & & & & & & \\
1 & & & & & & & & \\
1 & & & & & & & & \\
1 & & & & & & & & \\
0 & & & & & & & & \\
0 & & & & & & & &
\end{array}$$

Starting with c_1

$$\begin{array}{ccccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & & & & & & & \\
1 & 1 & & & & & & & \\
1 & 1 & & & & & & & \\
1 & 0 & & & & & & & \\
1 & 0 & & & & & & & \\
0 & 1 & & & & & & & \\
0 & 1 & & & & & & &
\end{array}$$

Filling with c_2

$$\begin{array}{ccccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & & & & & & \\
1 & 1 & 0 & & & & & & \\
1 & 0 & 1 & & & & & & \\
1 & 0 & 1 & & & & & & \\
0 & 1 & 1 & & & & & & \\
0 & 1 & 1 & & & & & &
\end{array}$$

Filling with c_3

$$\begin{array}{ccccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & & & & \\
1 & 0 & 1 & 1 & 1 & & & & \\
0 & 1 & 1 & 1 & 1 & & & & \\
0 & 1 & 1 & 1 & 1 & & & &
\end{array}$$

Completing with c_3

$$\begin{array}{ccccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & & \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & &
\end{array}$$

Completing with c_2

$$\begin{array}{ccccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0
\end{array}$$

Completing with c_1

Figure 3.3: Construction of the minimum half-product codeword from component code of $d_{min} = 6$.

First, for every element in S, there is a non-zero row contributing to the weight of the codeword. So, the weight is minimum only when the cardinality of S, the support set $\chi(\{c_1, c_2, c_3\})$, is minimum. That's why choosing c_1, c_2 such that $|\chi(\{c_1, c_2\})| = d_2(C)$ guarantees it to be minimum. If we choose any more distinct codewords for construction of the half-product codeword, their support set increases more than $d_2(C)$ causing the number of non-zero rows, and hence, the overall weight to increase. So, we can't get any lesser weight codeword if we use 4 or more distinct codewords.

Let us, now, calculate the weight of this codeword thus constructed using c_1, c_2, and c_3. The codeword has $|S| = d_2(C)$ number of rows, which, from lemma 3.2, is minimum when the code C has two codewords of minimum weight d and separated by minimum distance d, in the case when d is even; or the code C has one codeword of minimum weight d that is separated from one codeword of weight $d+1$ by the minimum distance d, in the case when d is odd. So,

- in the case when d is even: If c_1, c_2 are $d-$weight, $d-$distance separated, then c_3 is also $d-$weight and $d-$distance separated from each of $\{c_1, c_2\}$. So, $d_2(C) = \frac{3d}{2}$ and weight of the codeword in full form is $d_2(C) \times d = \frac{3d^2}{2}$.

- in the case when d is odd: If c_1 is $d-$weight and c_2 is $(d+1)-$weight, and are separated by distance of d, then c_3 is $d-$weight and is separated by distances of $(d+1)$ and d respectively from c_1 and c_2. So, $d_2(C) = \frac{3d+1}{2}$ and weight of the codeword in full form is $\frac{(d+1)}{2} \times d + \frac{d-1}{2} \times (d+1) = \frac{(d+1)(3d-1)}{2}$.

Observing the half-product codes above, the minimum weight comes from the codeword constructed by three non-zero component codewords. Therefore, in the half form

$$D_{min} \geq \begin{cases} \frac{3d^2}{4} & \text{if } d \text{ is even} \\ \frac{(d+1)(3d-1)}{4} & \text{if } d \text{ is odd} \end{cases}$$

□

3.3 Decoding

The decoding of half-product codes is performed in an iterative fashion, similar to product codes. But, for half-product codes, cascade decoding happens at every row and column continuously in every iterative step. So, a row is passed to the row decoding, and immediately the corrected codeword is updated along the symmetric column (which is equivalent to passing the symmetric column through the column decoding independently). An iteration is over when all the rows are decoded. The corrected codeword matrix is again passed to the cascade decoder in the next iteration. This continues until the decoding leads to no new codeword matrix at the end of an iteration.

As this process is iterative and involves cascade decoding, a stopping-set to be formed while decoding and may affect the performance. To give an example of a stopping-set for half-product codes, let us consider the following pattern. The error

$$\begin{array}{ccccccc}
0 & 0 & 1 & 1 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & 0 & 1 & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 & 0 \\
 & & & & 0 & 0 & 0 \\
 & & & & & 0 & 0 \\
 & & & & & & 0
\end{array}
\qquad\qquad
\begin{array}{ccccccc}
0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

In half form In full form

Figure 3.4: Minimum stopping-set pattern for $(21, 6)$ half-product code formed by $(7, 4)$ Hamming code.

pattern, given in the Figure 3.4 in both half and full forms, is a minimum stopping-set pattern for $(21, 6)$ half-product code formed by $(7, 4)$ Hamming code as component code. When we look at the full form of the error pattern, we can see that it can not be corrected when it is passed through the row decoder, as the non-zero rows have 2 errors in them, which is more than error correctability of the component code (1-error correcting). We can also see that it can not be corrected when it is passed through the column decoder for the same argument. This shows that the half-product codes have stopping-sets using the above decoding procedure.

3.4 Enumeration of the Minimum Stopping-sets

In this section, we will calculate the number of possible minimum stopping-set error patterns that can happen when a half-product code is transmitted through a channel with certain non-zero error probability. First, we will look at the error pattern that leads to a minimum stopping-set in the half-product codes, before enumerating the number of such patterns possible.

Consider a $t-$error correcting code $C(n,k)$ as the component code. Let it form the half-product code $\mathcal{C}(N = \frac{n(n-1)}{2}, K = \frac{k(k-1)}{2})$. As the component code is $t-$error correcting, the maximum number of errors the decoder can correct is t along either row or column, in the full form of the pattern. Hence, the stopping-set error pattern should have at least $t+1$ errors along every non-zero row or column. So, the minimum stopping pattern has exactly $(t+1)$ errors along each non-zero row and column. The following pattern is exactly that. This is the pattern formed by taking only the non-

$$\begin{array}{cccccc}
0 & 1_1 & 1_2 & 1_3 & \dots & 1_{t+1} \\
 & 0 & 1_2 & 1_3 & \dots & 1_{t+1} \\
 & & 0 & 1_3 & \dots & 1_{t+1} \\
 & & & \ddots & \ddots & \vdots \\
 & & & & 0 & 1_{t+1} \\
 & & & & & 0
\end{array}
\qquad\qquad
\begin{array}{cccccc}
0 & 1_1 & 1_2 & 1_3 & \dots & 1_{t+1} \\
1_1 & 0 & 1_2 & 1_3 & \dots & 1_{t+1} \\
1_1 & 1_2 & 0 & 1_3 & \dots & 1_{t+1} \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
1_1 & 1_2 & 1_3 & \dots & 0 & 1_{t+1} \\
1_1 & 1_2 & 1_3 & \dots & 1_{t+1} & 0
\end{array}$$

In half form — In full form

Figure 3.5: Minimum stopping-set pattern for half-product code formed by $t-$error correcting component code. The subscript represents the position of the error in that corresponding row.

zero rows and rearranging to form a compact error pattern with zeros as diagonal elements. The pattern symbolizes all the equivalent patterns of minimum stopping-sets with exactly $(t+1)$ errors in the non-zero rows (columns). By looking at the pattern, it can be observed that the total number of (non-zero) rows is $(t+2)$, and so the total number errors count to $2s_{min}^{HPC} = (t+1)(t+2)$ in the full form, where s_{min}^{HPC} is the number of errors in a minimum uncorrectable pattern in the half form. To enumerate the number of different stopping-set patterns is to enumerate the number of ways of choosing $(t+2)$ rows out of n rows. We note that choosing the rows itself decides the placements of errors in the rows, because of the symmetry of the half-product code in the full form. The following is, therefore apparent.

Enumeration 2. *The total number of error patterns of minimum stopping-sets of size s_{min}^{HPC} is given by*

$$N_{HPC} = \binom{n}{t+2}$$

3.5 Stopping-set Performance of Half-product Codes

In this section, we will evaluate the error performance due to the minimum stopping-sets. In the low error rate region, as N tends to infinity, the error performance of the decoder depends only on the minimum stopping-set patterns, as the decoder can not correct these error patterns anymore. We need to calculate the performance degradation due to these stopping-sets in order to ensure that the required performance is met. From the enumeration 2, we calculated the number of possible minimum stopping-set error patterns, to be N_{HPC}. So, the probability of error in the error floor is given by the following.

$$\begin{aligned} P(\text{Error}) &\geq P(\text{Error due to min stopping-sets}) \\ &\approx (\text{No. of min stopping-sets}) \times p^{(\text{weight of the min stopping-set})} \\ &= N_{HPC} \times p^{\frac{(t+1)(t+2)}{2}} \\ &= \binom{n}{t+2} p^{\frac{(t+1)(t+2)}{2}} \qquad \text{(HPC Error floor performance)} \end{aligned}$$

3.6 Comparison to Product Codes

We now have the expressions for error performance due to the minimum stopping-sets for both product and half-product codes. Thus, we can compare both of them to see which one is better in the low error rate region as N tends to infinity. Let us set up the parameters for an approximately fair comparison.

Consider a half-product code $\mathcal{C}_1(N = \frac{n(n-1)}{2}, K = \frac{k(k-1)}{2})$, and a product code $\mathcal{C}_2(N = \frac{n^2}{2}, K = \frac{k^2}{2})$. Let $\mathcal{C}_1$ be formed by the component code $C_1(n, k)$, a $t-$error correcting code, and let $\mathcal{C}_2$ be formed by the component code $C_2(\frac{n}{\sqrt{2}}, \frac{k}{\sqrt{2}})$, a $\frac{t}{\sqrt{2}}-$correcting code, for both rows and columns. The factor of $\sqrt{2}$ is to make the lengths of the both codes to be approximately same as $n \to \infty$. Let us consider the following analysis in the region as $n \to \infty$ keeping the ratio $\frac{t}{n} = f$, a constant. This constant ratio will ensure the rate to be constant as $n \to \infty$, see table 3.1.

Codes	C_1	C_2	$\mathcal{C}_1$	$\mathcal{C}_2$
Rates	$1-2f$	$1-2f$	$(1-2f)^2 - o(\frac{1}{n})$	$(1-2f)^2$
Rates as $n \to \infty$	$1-2f$	$1-2f$	$(1-2f)^2$	$(1-2f)^2$

Table 3.1: Comparison of the rates

We observe in Table 3.1 that the rates of half-product code $\mathcal{C}_1$, and product code $\mathcal{C}_2$ are same, as $n \to \infty$, so that the comparison in table 3.2 is almost fair. Table 3.2 also gives comparison for d_{min}, size of the stopping-set s_{min}, number of minimum stopping-sets, and the bound on the error performance due to the minimum stopping-sets. We can observe the following results from the table.

1. d_{min} for half-product codes is larger than that for product codes.

2. The size of the minimum stopping-set for half-product codes is $O(t) = O(n)$ more than the minimum stopping-set for product codes, as $s_{min}^{HPC} - s_{min}^{PC} = \frac{3-2\sqrt{2}}{2}t \approx 0.08t$.

	Codes	$\mathcal{C}_1$	$\mathcal{C}_2$
1.	d_{min}	$\geq \frac{3d^2}{4}$	$\frac{d^2}{2}$
2.	s_{min}	$\frac{(t+1)(t+2)}{2} = \frac{t^2+3t+2}{2}$	$(\frac{t}{\sqrt{2}}+1)^2 = \frac{t^2+2\sqrt{2}t+2}{2}$
3.	No. of ss	$\begin{aligned} N_{HPC} &= \binom{n}{t+2} \\ &= O\left(\left(\frac{n}{t+2}\right)^{t+2}\right) \\ &= O\left(\left(\frac{1}{f+\frac{2}{n}}\right)^{fn+2}\right) \\ &= O\left(F^n\right), \end{aligned}$ where $F = \left(\frac{1}{f}\right)^f > 1$	$\begin{aligned} N_{PC} &= \binom{\frac{n}{\sqrt{2}}}{\frac{t}{\sqrt{2}}+1}^2 \\ &= O\left(\left(\frac{n}{t+\sqrt{2}}\right)^{2\left(\frac{t}{\sqrt{2}}+1\right)}\right) \\ &= O\left(\left(\frac{1}{f+\frac{\sqrt{2}}{n}}\right)^{\sqrt{2}fn+2}\right) \\ &= O\left(F^{\sqrt{2}n}\right), \end{aligned}$ where $F = \left(\frac{1}{f}\right)^f > 1$
4.	PoE	$\approx O\left(F^n\right) p^{\frac{t^2+3t+2}{2}}$	$\approx O\left(F^{\sqrt{2}n}\right) p^{\frac{t^2+2\sqrt{2}t+2}{2}}$

Table 3.2: Comparison of the sizes of the minimum stopping-sets

3. The number of minimum stopping-sets for half-product code increase exponentially with n in the exponent, which is $\sqrt{2}$ times lower than the exponent with which the number of minimum stopping-sets for product code increase.

4. Points 2 and 3 imply that the error rate due to the minimum stopping-sets of half-product codes should be much lower than the error rate due to the minimum stopping-sets of product codes.

4. SIMULATION RESULTS

In this chapter, the simulation results are presented and conclusions are drawn based on these.

4.1 Comparative Simulations between HPC and PC

We have considered two different MDS codes $C_1(n_1 = n, k_1 = k, t_1 = t)$ and $C_2(n_2 = \frac{n}{\sqrt{2}}, k_2 = \frac{k}{\sqrt{2}}, t_2 = \frac{t}{\sqrt{2}})$ (where the third component in the parameter triplet represents the error correction capability of the code), as component codes for the half-product code (HPC) $\mathcal{C}_1(N = \frac{n(n-1)}{2}, K = \frac{k(k-1)}{2})$ and product code (PC) $\mathcal{C}_2(N = \frac{n^2}{2}, K = \frac{k^2}{2})$. The channel is considered to be a channel with symbol error probability of p for both the codes. For both codes, all-zero codeword is transmitted through the channel. Thus, transmitted codeword has errors uniformly from the channel with symbol error probability of p. The corresponding received codewords are decoded using the respective cascade decoders for HPC and PC.

Monte Carlo simulations have been performed to calculate the probability of block or codeword error for varying values of p from 0.03 to 0.04, in each case of $n = 2^8 - 1, 2^9 - 1$, and $2^{10} - 1$ with rate of the HPC and PC being constant, Rate = 0.95. These probability of error values are plotted in log scale against the values of p in the figure 4.1.

4.2 Observations

From the plot in the figure 4.1, the following observations can be made.

1. For small n ($n_1 = 255$), equivalently small t ($t_1 = 6$), we see that the half-product codes perform better than the product codes.

2. For large n ($n_1 = 511,\ 1023$), equivalently large t ($t_1 = 13,\ 26$), we see that

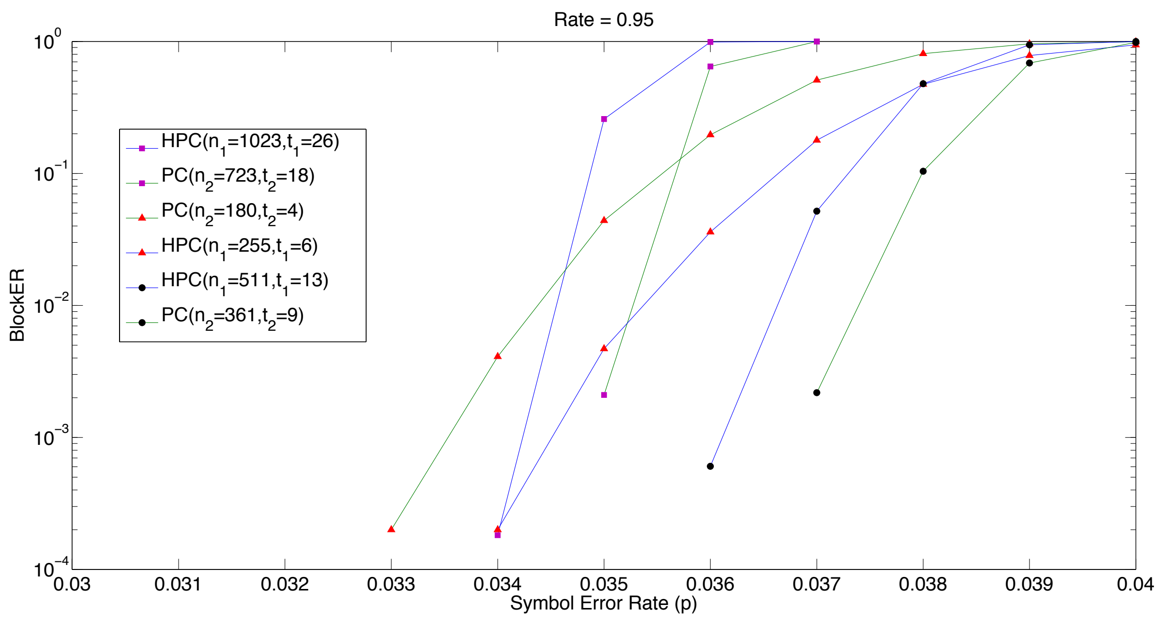

Figure 4.1: Comparative plot of HPC and PC for increasing values of n_1 and n_2 with fixed Rate $= 0.95$

the product codes perform better than the half-product codes.

3. One can also observe that the waterfall behavior of the curve becomes steeper, and both the curves coming closer, as n gets larger.

4.3 Discussion

Based on the observations in section 4.2 of the plot in the Figure 4.1, we can say the following.

In the observation 1, we see that half-product codes perform better than product codes. That is because, when n is smaller, t is smaller and so, the stopping-set sizes are smaller in both the codes. The small sizes of the stopping-set patterns make the performance due to these patterns dominate the error floor. But, in Section 3.6, it is analytically evaluated that the scaling of the error floor performances is better in half-product codes than in product codes. So, the observation 1 is only the experimental evidence supporting this comparative evaluation.

In the observation 2, we see that product codes perform better than half-product codes. We observe this, contrary to the intuition portrayed in the previous paragraph, because in this regime of large n, and so, large t, the sizes of the stopping-sets is so high that the error performance due to the minimum stopping-sets is no longer dominating, and is in fact negligible compared to all the other stopping-sets. This reversal of error performances from the half-product codes and the product codes is because of the combined effect of all these stopping-sets. Let us now see an example of this by taking the immediate next stopping-set in both codes and calculating the error performance due to this. It can be observed that the next stopping-set will have the size of $\frac{(t+1)(t+3)}{2}$ in half-product codes, which is, now, $O(n)$ less than the size of the next stopping-sets, $(\frac{t}{\sqrt{2}}+1)(\frac{t}{\sqrt{2}}+2)$, in product codes. This is opposite to what we have observed in case of the minimum stopping-sets. Although, the number of such stopping-sets, if computed for both codes, will be such that the ratio of the number in product code to that in half-product code is approximately in $O(n^{gn})$, where $g < 1$ is constant for fixed rate. When we compute the error rate due to this next stopping-set in both codes, we see that the error rate in half-product codes is lower than that of product codes for up to certain n and then the error rate reverses after that point. This is similarly carried to the next few dominating stopping-sets. After taking the combined error rate due to these, the error rate will be seen higher in half-product codes than in product codes.

From the observation 3, it is clear that the thresholds of the half-product codes and the product codes tend to be same as n tends to infinity. Analysis of the thresholds is left to the future work.

4.4 Conclusions

We have seen the observations and discussed about them in the previous sections. From this discussion we can conclude that,

1. Half-product codes scale better in the error floor than that of the product codes. So, if the application is in low t region, where the stopping-sets dominate the error performance, then the half-product code is probably better than the product code.

2. Half-product codes have the same threshold as the product codes, as the length of the component codes tends to infinity, so half-product codes are not worse than product codes in theory.

REFERENCES

[1] P. Elias, "Error-free coding," *Information Theory, Transactions of the IRE Professional Group on*, vol. 4, pp. 29–37, September 1954.

[2] J. Justesen, "Performance of product codes and related structures with iterated decoding," *Communications, IEEE Transactions on*, vol. 59, pp. 407–415, February 2011.

[3] E. Rosnes, "Stopping set analysis of iterative row-column decoding of product codes," *Information Theory, IEEE Transactions on*, vol. 54, pp. 1551–1560, April 2008.

[4] H. Burton and J. Weldon, E., "Cyclic product codes," *Information Theory, IEEE Transactions on*, vol. 11, pp. 433–439, Jul 1965.

[5] N. Abramson, "Cascade decoding of cyclic product codes," *Communication Technology, IEEE Transactions on*, vol. 16, pp. 398–402, June 1968.

[6] D. Slepian, "Some further theory of group codes," *Bell System Technical Journal, The*, vol. 39, pp. 1219–1252, Sept 1960.