Engineering Mutation-Tolerant Genes

by

Prince Kwame Ampofo

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2019 by the
Graduate Supervisory Committee:

Xiaojun Tian, Chair
Samira Kiani
Yang Kuang

ARIZONA STATE UNIVERSITY

May 2019

ABSTRACT

Ideas from coding theory are employed to theoretically demonstrate the engineering of mutation-tolerant genes, genes that can sustain up to some arbitrarily chosen number of mutations and still express the originally intended protein. Attention is restricted to tolerating substitution mutations. Future advances in genomic engineering will make possible the ability to synthesize entire genomes from scratch. This presents an opportunity to embed desirable capabilities like mutation-tolerance, which will be useful in preventing cell deaths in organisms intended for research or industrial applications in highly mutagenic environments. In the extreme case, mutation-tolerant genes (mutols) can make organisms resistant to retroviral infections.

An algebraic representation of the nucleotide bases is developed. This algebraic representation makes it possible to convert nucleotide sequences into algebraic sequences, apply mathematical ideas and convert results back into nucleotide terms. Using the algebra developed, a mapping is found from the naturally-occurring codons to an alternative set of codons which makes genes constructed from them mutation-tolerant, provided no more than one substitution mutation occurs per codon. The ideas discussed naturally extend to finding codons that can tolerate t arbitrarily chosen number of mutations per codon. Finally, random substitution events are simulated in both a wild-type green fluorescent protein (GFP) gene and its mutol variant and the amino acid sequence expressed from each post-mutation is compared with the amino acid sequence pre-mutation.

This work assumes the existence of synthetic protein-assembling entities that function like tRNAs but can read k nucleotides at a time, with $k \geq 5$. The realization of this assumption is presented as a challenge to the research community.

DEDICATION

This work is dedicated to my family for their love and care.

ACKNOWLEDGMENTS

I would like to express my gratitude to Dr. Xiaojun Tian, Dr. Samira Kiani and Dr. Yang Kuang for serving on my committee and for their constructive feedback on my work. Special thanks also to the Mastercard Foundation for the scholarship that made my education at Arizona State University, which culminated in this thesis, possible.

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1

ALGEBRAIC REPRESENTATION OF NUCLEOTIDE BASES

1.1 Introduction

In our effort to develop genes that can tolerate mutations, we will be employing some mathematical ideas. To make this possible, we need to represent nucleotide bases, the building blocks from which DNA and RNA are constructed, with suitable mathematical entities. This way, we can easily apply mathematical ideas to algebraic representations of codons or nucleotide sequences and convert results back into nucleotide terms. This chapter is devoted to finding suitable algebraic representations for the bases: adenine (A), guanine (G), cytosine (C), thymine (T), and uracil (U).

In section 1.2, an interesting algebraic structure called a field is introduced. In section 1.3, we construct a field whose elements will be used as algebraic representations of the bases. In section 1.4, we assign elements of the field constructed in section 1.3 to the bases. Finally, in section 1.5, we discuss how computation is done in a field.

Most concepts utilized throughout this work are from the areas of linear algebra and coding theory. The uninitiated will find the following references helpful: Axler (2007), Ling (2004), and Lin and Costello (1983).

1.2 Field

A field F is a set together with two binary operations addition (denoted by $+$) and multiplication (denoted by $*$) defined such that the following axioms are satisfied:

1. Closure. $\forall\ x, y \in F, \quad x + y \in F \quad$ and $\quad x * y \in F$

2. Commutativity. $\forall\ x, y \in F, \quad x + y = y + x \quad$ and $\quad x * y = y * x$

3. Associativity. $\forall\ x, y, z \in F, \quad x + (y + z) = (x + y) + z \quad$ and $\quad x * (y * z) = (x * y) * z$

4. Existence of additive and multiplicative identities. **(a)** There exists an additive identity $0 \in F$ such that $x+0 = x$ for all $x \in F$. **(b)** There exists a multiplicative identity $1 \in F$ such that $x * 1 = x$ for all $x \in F$

5. Existence of additive and multiplicative inverses. **(a)** For every $x \in F$, there exists an additive inverse denoted $-x \in F$ such that $x+(-x) = 0$. **(b)** For every $x \in F$ except 0, there exists an inverse denoted $x^{-1} \in F$ such that $x * x^{-1} = 1$.

6. Distributivity of multiplication over addition.
   For all $x, y, z \in F, \quad x * (y + z) = x * y + x * z$

7. Distinctness of additive and multiplicative identities. $1 \neq 0$

The axioms outlined above demonstrate why a field is ideal for use in representing nucleotide bases algebraically. The algebraic properties a field possesses, like closure, commutativity, associativity, distributivity, etcetra, as outlined in axioms 1 through 6, are essential for ease of algebraic manipulations. By using elements from a field for the representation of nucleotides, we will be able to easily perform algebraic manipulations and apply mathematical ideas to algebraic equivalents of nucleotide sequences, toward the goal of developing mutation-tolerant genes.

In the next section, we will construct a field with 4 elements. Adenine, Guanine, and Cytosine will each be assigned one of these algebraic elements. Thymine and Uracil will both be assigned the same fourth element since Uracil is the RNA "equivalent" of Thymine.

1.3 Constructing Field of order 4

In general, if p is prime and addition and multiplication are done in modulo-p, the set $\{0, 1, 2, \ldots, p-1\}$ satisfies all the field axioms outlined in Section 1.2 above and thus forms a field of p elements. The number of elements in a field is called its order. When the order of a field is finite, we call it a Galois field and denote it by GF(*number of elements*). Also, when the number of elements in a Galois field is prime, we call the field a prime field.

We desire to construct a field of 4 elements: GF(4). However, 4 is not prime and so $\{0, 1, 2, \ldots, 4-1\} = \{0, 1, 2, 3\}$ is not a field. We can however employ a well-known idea from field theory which allows the construction of any field GF(p^q) from GF(p), where p is prime and q is any positive integer. Thus, we can construct our desired field GF(2^2) from GF(2) = $\{0, 1\}$, which is the field formed from $\{0, 1, 2, \ldots, p-1\}$ when $p = 2$. When GF(2) is extended to get GF(2^2), GF(2) is called the base field and GF(2^2) is called an extension field. To utilize this strategy, we first introduce two concepts:

1. **Polynomials over fields.** If we construct an n-degree polynomial $f(\alpha) = f_0 + f_1\,\alpha + f_2\,\alpha^2 + \cdots + f_{n-1}\,\alpha^{n-1} + f_n\,\alpha^n$ such that its coefficients $f_0, f_1, f_2, \ldots, f_{n-1}, f_n$ are all taken from a field called F, we say that $f(\alpha)$ is a "polynomial over field F". As an example, polynomials over GF(2) are polynomials with all coefficients either 0 or 1. In general, there are 2^n polynomials of degree n that can be constructed over GF(2). To see this, notice that for any polynomial of degree n over GF(2), its coefficients $f_0, f_1, f_2, \ldots, f_{n-1}$ can each have two possible values (0 or 1) whereas coefficient f_n must be 1 since if it is 0, the polynomial will no longer be of degree n. Thus, the total number of n-degree polynomials over GF(2) is $\underbrace{2 * 2 * \cdots * 2}_{\text{n times}} * 1 = 2^n$. For example, there are $2^2 = 4$ polynomials

over GF(2) that are of degree 2. They are:

$$\alpha^2, \quad \alpha + \alpha^2, \quad 1 + \alpha^2, \quad \text{and} \quad 1 + \alpha + \alpha^2 \tag{1.1}$$

2. **Primitive polynomials.** A q-degree polynomial $f(\alpha)$ over a prime field GF(p) is called primitive if the smallest positive integer n for which $f(\alpha)$ divides $\alpha^n + 1$ without a remainder is $n = p^q - 1$. Primitive polynomials are important because they are used in constructing extension fields from base fields. It is difficult to test all polynomials over a given field to find ones that are primitive. As such, tables of primitive polynomials exist from which desired primitive polynomials can be obtained. Using one such table, we find that of the four polynomials of degree 2 over GF(2) (listed in 1.1), only one is primitive:

$$p(\alpha) = 1 + \alpha + \alpha^2 \tag{1.2}$$

With these two concepts introduced, we state without proof a theorem with which we can construct GF(2^2) from GF(2).

Theorem 1.1. *If $f(\alpha)$ is a q-degree primitive polynomial over GF(p), the set of all q-degree polynomials over GF(p) taken as modulo $f(\alpha)$ forms GF(p^q).*

Remark. Just as an integer like 9 can be expressed in modulo 3 by dividing 9 by 3 and keeping the remainder, a polynomial $f_1(\alpha)$ can be expressed as modulo of another polynomial $f_2(\alpha)$ by dividing $f_1(\alpha)$ by $f_2(\alpha)$ and keeping the remainder.

Using Theorem 1.1 above, we will divide each of the 4 polynomials in Equation (1.1) by the primitive polynomial in Equation (1.2), keeping the remainder in each case. The set of these four remainders, under modulo 2 addition and multiplication, will satisfy all the field axioms in Section 1.2 and thus form the desired field

of four elements, GF(2^2).

$$\begin{array}{r|l} & 1 \\ \hline \alpha^2+\alpha+1 & \alpha^2 \\ & \alpha^2+\alpha+1 \\ \hline & \boldsymbol{\alpha+1} \end{array} \qquad \begin{array}{r|l} & 1 \\ \hline \alpha^2+\alpha+1 & \alpha^2+\alpha \\ & \alpha^2+\alpha+1 \\ \hline & \boldsymbol{1} \end{array}$$

$$\begin{array}{r|l} & 1 \\ \hline \alpha^2+\alpha+1 & \alpha^2+1 \\ & \alpha^2+\alpha+1 \\ \hline & \boldsymbol{\alpha} \end{array} \qquad \begin{array}{r|l} & 1 \\ \hline \alpha^2+\alpha+1 & \alpha^2+\alpha+1 \\ & \alpha^2+\alpha+1 \\ \hline & \boldsymbol{0} \end{array}$$

The remainders are 0, 1, α, and $\alpha+1$. Using Theorem 1.1, our desired field[1] is:

$$\text{GF}(2^2) = \{\ 0,\ 1,\ \alpha,\ \alpha+1\ \},\ \text{ with} + \text{and} * \text{done in mod 2} \tag{1.3}$$

Compact form of $\boldsymbol{\alpha+1}$. For every extension field, all the non-zero elements of the set are expressed in terms of one special element called the **primitive element**. By noticing that 1 can be written as α^0, it is easy to spot that α is the primitive element in the field we just constructed (Equation 1.3 above). A well-known result in field theory is that a primitive element is a root of its corresponding primitive polynomial. Thus, $p(\alpha) = 1+\alpha+\alpha^2 = 0$, where p is the primitive polynomial used in constructing GF(2^2). Using this relation, we can find another useful form of the $\alpha+1$ element in GF(2^2):

[1]The phrase "with + and $*$ done in mod 2" in (1.3) is vital because a set by itself is not a field. When we mention the set without the operations, it must be taken as implied.

$$
\begin{aligned}
1 + \alpha + \alpha^2 &= 0 \\
1 + \alpha + (\alpha^2 + \alpha^2) &= 0 + \alpha^2 \qquad \text{(added } \alpha^2 \text{ to both sides)} \\
\alpha^2 + \alpha^2 = (1+1)\alpha^2 &= (0)\alpha^2 = 0 \qquad \text{(} 1 + 1 = 0 \text{ due to modulo 2 addition)} \\
\therefore \quad 1 + \alpha &= \alpha^2 \qquad (1.4)
\end{aligned}
$$

According to (1.4) above, the element $\alpha + 1$ in GF(2^2) is equal to α^2. We can thus write GF(2^2) = $\{0,\ 1,\ \alpha,\ \alpha+1\}$ as $\{0,\ 1,\ \alpha,\ \alpha^2\}$. Because α^2 is more compact, we will use it more often. Being in exponent form, it will also be used primarily in multiplication operations, whereas the $\alpha + 1$ form will be used mainly in addition operations.

1.4 Assignment of Field elements to nucleotide Bases

Our goal at the beginning of this chapter was to find suitable algebraic entities to represent the nucleotide bases. We constructed a field of four elements, GF(2^2), in the preceding section. Here, we assign these elements to the bases.

As outlined in Table 1.1 below, we will represent Adenine with 0, Guanine with α, and Cytosine with α^2. Because the RNA "equivalent" of Thymine is Uracil, they are both assigned 1. These assignments are not arbitrary. They have been done such that adding 1 to the algebraic representation of any base gives the algebraic representation of its Watson-Crick base pair. Examples[2]:

1) 0 (Adenine) + 1 = 1 (Thymine)

2) 1 (Thymine) + 1 = 0 (Adenine)

3) α (Guanine) + 1 = α^2 (Cytosine)

4) α^2 (Cytosine) + 1 = $(\alpha + 1) + 1 = \alpha$ (Guanine)

[2]All operations are done in modulo 2. Also, note that $\alpha^2 = \alpha + 1$ (Equation 1.4)

Base	Algebraic Representation
Adenine (A)	0
Guanine (G)	α
Cytosine (C)	α^2
Thymine (T) & Uracil (U)	1

Table 1.1: Algebraic representation of Nucleotide Bases. Note that $\alpha^2 = \alpha + 1$

We will refer to the set of elements used to represent the bases as $\boldsymbol{B}$ (as in **B**ases), where the subscripts used are simply to indicate which base the element represents and may be left out for the sake of compactness:

$$\boldsymbol{B} = \{0_A,\ \ 1_{T \text{ or } U},\ \ \alpha_G,\ \ \alpha^2{}_C\} \tag{1.5}$$

As stated earlier, notice that 1 is used to represent both Thymine (T) and Uracil (U) since each can be thought of as the DNA or RNA "equivalent" of the other. As such, when 1 appears in the algebraic representation of a nucleotide sequence, it should be seen as representing Thymine or Uracil based on the context. Obviously, if 1 appears in a DNA sequence, it represents Thymine. In an RNA sequence, it represents Uracil.

1.5 Doing Algebra with the Base Representations

Our goal for this chapter was to find suitable algebraic representations for the nucleotide bases so that we can write any nucleotide sequence as a sequence of algebraic entities, apply mathematical ideas to it, and map results back into nucleotide terms, with the aim of developing mutation-tolerant genes. Having found such suitable algebraic entities (elements of set B), we now discuss how addition, multiplication, subtraction, and division can be done with them.

GF(2) $= \{0, 1\}$, the base field from which our extension field of 4 elements, $\mathbf{B} = \{0,\ 1,\ \alpha,\ \alpha^2\}$, was constructed, is only a field under mod 2 addition and multiplication. By implication, **B** is also a field under mod 2 addition and multiplication. As such all operations will be done in mod 2. Note that because we used elements from a field to represent the bases, all the field axioms outlined in Section 1.2 hold. As such, for all operations, the commutative, associative, and distributive properties can be used. It's also vital to remember that $\alpha^2 = \alpha + 1$ (Equation 1.4).

- **Addition.** Adding any elements of **B** is just like normal addition, keeping in mind that in mod 2, $\quad 0 + 0 = 0, \quad 0 + 1 = 1, \quad 1 + 0 = 1 \quad$ and $\quad 1 + 1 = 0$. An addition table is shown in Table 1.2a. Adding any element to itself gives 0. Also, as indicated earlier, adding α^2 (which equals $\alpha + 1$) to any element gives its Watson-Crick base pair.

- **Multiplication.** Multiplication can also be done by keeping in mind that in mod 2, $\quad 0 * 0 = 0, \quad 0 * 1 = 0, \quad 1 * 0 = 0 \quad$ and $\quad 1 * 1 = 1$. Note that for any element b,

$$b^i * b^k = \underbrace{b * b * \cdots * b}_{\text{i times}} \quad * \quad \underbrace{b * b * \cdots * b}_{\text{k times}} = b^{i+k}$$

  One relation that will be useful in simplifying multiplication operations is $\alpha^3 = 1$. To see this, notice that $\alpha^3 = \alpha^2 * \alpha$, and from the multiplication table (Table 1.2b), $\alpha^2 * \alpha = 1$. For example, $\alpha * \alpha^2 * \alpha^2 = \alpha^5$ can be simplified as $\alpha^5 = \alpha^3 * \alpha^2 = 1 * \alpha^2 = \alpha^2$.

Remark. Though only the two operations discussed above are explicitly defined on a field, we can still do subtraction and division because subtraction and division can be written in terms of addition and multiplication, respectively.

$+$	0	1	α	α^2
0	0	1	α	α^2
1	1	0	α^2	α
α	α	α^2	0	1
α^2	α^2	α	1	0

(a) Addition Table

$*$	0	1	α	α^2
0	0	0	0	0
1	0	1	α	α^2
α	0	α	α^2	1
α^2	0	α^2	1	α

(b) Multiplication Table

Table 1.2: Arithmetic table for GF(2^2). Note that $+$ and $*$ are done mod 2.

- **Subtraction.** Note that $a - b = a + (-b)$, where $-b$ is the additive inverse of b. Thus, to subtract b from a, we first find the additive inverse of b (which is denoted by $-b$) and add it to a. Note that an additive inverse of an element b is the element which when added to b, gives 0. Looking at the addition table in Table 1.2a, we notice that adding any element to itself gives 0. This means that for set **B**, the additive inverse of any element is itself. This has two interesting implications: 1) Subtracting an element from another is the same as adding them. For example, $a - b = a + (-b) = a + b$ since $-b$, which denotes the inverse of b, equals b itself. 2) Because of implication 1, the addition table shown in Table 1.2a also acts as a subtraction table for the elements in B.

- **Division.** Note that $\frac{a}{b} = a * b^{-1}$, where b^{-1} is the multiplicative inverse of b. Thus, to divide a by b, we first find the multiplicative inverse of b then multiply it by a. Remember that the multiplicative inverse of an element b is an element which when multiplied by b, gives 1. For example, to find $\frac{1}{\alpha^2}$, we first find the multiplicative inverse of α^2 by looking at the multiplicative table (Table 1.2b) to find the element which when multiplied by α^2, gives 1. Since $\alpha^2 * \alpha = 1$, α is the multiplicative inverse of α^2. Thus, $\frac{1}{\alpha^2} = 1 * (\alpha^2)^{-1} = 1 * \alpha = \alpha$.

1.6 Algebraic representation of Codons

Having found suitable algebraic representation for the nucleotide bases, we can now write each of the 64 DNA sense ($5' \rightarrow 3'$) codons in terms of the algebraic representation of its nucleotide bases.

$$\left.\begin{array}{l}(\alpha\ \alpha^2\ 1)_{\ GCT}\\(\alpha\ \alpha^2\ 0)_{\ GCA}\\(\alpha\ \alpha^2\ \alpha^2)_{\ GCC}\\(\alpha\ \alpha^2\ \alpha)_{\ GCG}\end{array}\right\}\text{Ala}\qquad \left.\begin{array}{l}(0\ 0\ 1)_{\ AAT}\\(0\ 0\ \alpha^2)_{\ AAC}\end{array}\right\}\text{Asn}\qquad \left.\begin{array}{l}(\alpha\ 0\ 1)_{\ GAT}\\(\alpha\ 0\ \alpha^2)_{\ GAC}\end{array}\right\}\text{Asp}\qquad \left.\begin{array}{l}(1\ \alpha\ 1)_{\ TGT}\\(1\ \alpha\ \alpha^2)_{\ TGC}\end{array}\right\}\text{Cys}$$

$$\left.\begin{array}{l}(\alpha\ 0\ 0)_{\ GAA}\\(\alpha\ 0\ \alpha)_{\ GAG}\end{array}\right\}\text{Glu}\qquad \left.\begin{array}{l}(\alpha^2\ 0\ 0)_{\ CAA}\\(\alpha^2\ 0\ \alpha)_{\ CAG}\end{array}\right\}\text{Gln}\qquad \left.\begin{array}{l}(\alpha\ \alpha\ 1)_{\ GGT}\\(\alpha\ \alpha\ 0)_{\ GGA}\\(\alpha\ \alpha\ \alpha^2)_{\ GGC}\\(\alpha\ \alpha\ \alpha)_{\ GGG}\end{array}\right\}\text{Gly}\qquad \left.\begin{array}{l}(\alpha^2\ 0\ 1)_{\ CAT}\\(\alpha^2\ 0\ \alpha^2)_{\ CAC}\end{array}\right\}\text{His}$$

$$\left.\begin{array}{l}(0\ 1\ 1)_{\ AAT}\\(0\ 1\ \alpha^2)_{\ ATC}\\(0\ 1\ 0)_{\ ATA}\end{array}\right\}\text{Ile}\qquad \left.\begin{array}{l}(1\ 1\ 0)_{\ TTA}\\(1\ 1\ \alpha)_{\ TTG}\\(\alpha^2\ 1\ 1)_{\ CTT}\\(\alpha^2\ 1\ 0)_{\ CTA}\\(\alpha^2\ 1\ \alpha^2)_{\ CTC}\\(\alpha^2\ 1\ \alpha)_{\ CTG}\end{array}\right\}\text{Leu}\qquad \left.\begin{array}{l}(0\ 0\ 0)_{AAA}\\(0\ 0\ \alpha)_{\ AAG}\end{array}\right\}\text{Lys}\qquad (0\,1\,\alpha)_{\ ATG}\ \text{Met}$$

$$\left.\begin{array}{l}(\alpha^2\ \alpha^2\ 1)_{\ CCT}\\(\alpha^2\ \alpha^2\ 0)_{\ CCA}\\(\alpha^2\ \alpha^2\ \alpha^2)_{\ CCC}\\(\alpha^2\ \alpha^2\ \alpha)_{\ CCG}\end{array}\right\}\text{Pro}\qquad \left.\begin{array}{l}(1\ 1\ 1)_{\ TTT}\\(1\ 1\ \alpha^2)_{\ TTC}\end{array}\right\}\text{Phe}\qquad \left.\begin{array}{l}(0\ \alpha^2\ 1)_{\ ACT}\\(0\ \alpha^2\ \alpha^2)_{\ ACC}\\(0\ \alpha^2\ 0)_{\ ACA}\\(0\ \alpha^2\ \alpha)_{\ ACG}\end{array}\right\}\text{Thr}\qquad \left.\begin{array}{l}(1\ 0\ 1)_{\ TAT}\\(1\ 0\ \alpha^2)_{\ TAC}\end{array}\right\}\text{Tyr}$$

$$\left.\begin{array}{l}(\alpha\ 1\ 1)_{\ GTT}\\(\alpha\ 1\ 0)_{\ GTA}\\(\alpha\ 1\ \alpha^2)_{\ GTC}\\(\alpha\ 1\ \alpha)_{\ GTG}\end{array}\right\}\text{Val}\qquad \left.\begin{array}{l}(1\ \alpha^2\ 1)_{\ TCT}\\(1\ \alpha^2\ 0)_{\ TCA}\\(1\ \alpha^2\ \alpha^2)_{\ TCC}\\(1\ \alpha^2\ \alpha)_{\ TCG}\\(0\ \alpha\ 1)_{\ AGT}\\(0\ \alpha\ \alpha^2)_{\ AGC}\end{array}\right\}\text{Ser}\qquad (1\,\alpha\,\alpha)_{\ TGG}\quad \text{Trp}\qquad \left.\begin{array}{l}(\alpha^2\ \alpha\ 1)_{\ CGT}\\(\alpha^2\ \alpha\ \alpha^2)_{\ CGC}\\(\alpha^2\ \alpha\ 0)_{\ CGA}\\(\alpha^2\ \alpha\ \alpha)_{\ CGG}\\(0\ \alpha\ 0)_{\ AGA}\\(0\ \alpha\ \alpha)_{\ AGG}\end{array}\right\}\text{Arg}$$

$$\left.\begin{array}{l}(1\ 0\ 0)_{\ TAA}\\(1\ 0\ \alpha)_{\ TAG}\\(1\ \alpha\ 0)_{\ TGA}\end{array}\right\}\text{Stop}$$

Chapter 2

CODONS AS VECTORS

2.1 Introduction

In the previous chapter, a suitable algebraic representation was found for the nucleotide bases: $\mathbf{B} = \{0_A,\ 1_{T \text{ or } U},\ \alpha_G,\ \alpha^2{}_C\}$. In section 1.6, each DNA codon was then written in terms of the algebraic representation of its nucleotides. For example, codon ATG in terms of the algebraic representation of its nucleotide bases is 01α since A, T, and G are represented by $0, 1,$ and α, respectively.

$$ATG \xrightarrow{\text{algebraic form}} 0\ 1\ \alpha$$

To distinguish between codons and their algebraic forms, we will call the algebraic forms **p-codons**. Thus, the p-codon of ATG is 01α. In section 2.2, we consider the set of all 64 p-codons corresponding to the 64 DNA codons and show that this set forms a vector space where the p-codons are vectors and $0,\ 1,\ \alpha,$ and α^2 are scalars. From section 2.3 through 2.5, relevant vector space concepts are discussed in the context of this codon vector space.

2.2 Codon Vector Space

Consider set C below whose elements are the 64 p-codons. We wish to show that this set is a vector space. This is important because by confirming that C is indeed a vector space, we can make use of all theory that has been developed about vector spaces, toward our goal of developing mutation-tolerant genes. Let's first define a vector space.

$$\boldsymbol{C} = \left\{ \begin{array}{cccccccc}
(0\ 0\ 0), & (0\ 0\ 1), & (0\ 0\ \alpha), & (0\ 0\ \alpha^2), & (0\ 1\ 0), & (0\ 1\ 1), & (0\ 1\ \alpha), & (0\ 1\ \alpha^2), \\
(0\ \alpha\ 0), & (0\ \alpha\ 1), & (0\ \alpha\ \alpha), & (0\ \alpha\ \alpha^2), & (0\ \alpha^2\ 0), & (0\ \alpha^2\ 1), & (0\ \alpha^2\ \alpha), & (0\ \alpha^2\ \alpha^2), \\
(1\ 0\ 0), & (1\ 0\ 1), & (1\ 0\ \alpha), & (1\ 0\ \alpha^2), & (1\ 1\ 0), & (1\ 1\ 1), & (1\ 1\ \alpha), & (1\ 1\ \alpha^2), \\
(1\ \alpha\ 0), & (1\ \alpha\ 1), & (1\ \alpha\ \alpha), & (1\ \alpha\ \alpha^2), & (1\ \alpha^2\ 0), & (1\ \alpha^2\ 1), & (1\ \alpha^2\ \alpha), & (1\ \alpha^2\ \alpha^2), \\
(\alpha\ 0\ 0), & (\alpha\ 0\ 1), & (\alpha\ 0\ \alpha), & (\alpha\ 0\ \alpha^2), & (\alpha\ 1\ 0), & (\alpha\ 1\ 1), & (\alpha\ 1\ \alpha), & (\alpha\ 1\ \alpha^2), \\
(\alpha\ \alpha\ 0), & (\alpha\ \alpha\ 1), & (\alpha\ \alpha\ \alpha), & (\alpha\ \alpha\ \alpha^2), & (\alpha\ \alpha^2\ 0), & (\alpha\ \alpha^2\ 1), & (\alpha\ \alpha^2\ \alpha), & (\alpha\ \alpha^2\ \alpha^2), \\
(\alpha^2\ 0\ 0), & (\alpha^2\ 0\ 1), & (\alpha^2\ 0\ \alpha), & (\alpha^2\ 0\ \alpha^2), & (\alpha^2\ 1\ 0), & (\alpha^2\ 1\ 1), & (\alpha^2\ 1\ \alpha), & (\alpha^2\ 1\ \alpha^2), \\
(\alpha^2\ \alpha\ 0), & (\alpha^2\ \alpha\ 1), & (\alpha^2\ \alpha\ \alpha), & (\alpha^2\ \alpha\ \alpha^2), & (\alpha^2\ \alpha^2\ 0), & (\alpha^2\ \alpha^2\ 1), & (\alpha^2\ \alpha^2\ \alpha), & (\alpha^2\ \alpha^2\ \alpha^2)
\end{array} \right\}$$

Definition 1. (Vector Space) A vector space V is a set of objects (called vectors) that can be added $(+)$ together and multiplied $(\cdot)$ by scalars such that the following axioms are satisfied for any choice of vectors $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{w}$ and scalars a and b :

1. Closure. $\boldsymbol{u} + \boldsymbol{w} \in V$ and $a \cdot \boldsymbol{v} \in V$
2. Commutativity. $\boldsymbol{u} + \boldsymbol{w} = \boldsymbol{w} + \boldsymbol{u}$
3. Additive identity. There exists a zero vector, denoted $\mathbf{0}$, in C such that $\boldsymbol{u} + \mathbf{0} = \boldsymbol{u}$
4. Multiplicative identity. $1 \cdot \boldsymbol{v} = \boldsymbol{v}$
5. Additive inverses. For any vector $\boldsymbol{u} \in V$, there exists $-\boldsymbol{u} \in V$ such that $\boldsymbol{u} + (-\boldsymbol{u}) = \mathbf{0}$
6. Associativity. $\boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{w}) = (\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{w}$ and $a \cdot (b \cdot \boldsymbol{v}) = (a \cdot b) \cdot \boldsymbol{v}$
7. Distributivity. $a \cdot (\boldsymbol{u} + \boldsymbol{v}) = (a \cdot \boldsymbol{u}) + (a \cdot \boldsymbol{v})$ and $(a + b) \cdot \boldsymbol{u} = (a \cdot \boldsymbol{u}) + (b \cdot \boldsymbol{u})$

What the definition says is that if we form a set and define a way of adding the elements of the set and also define multiplication of the elements by some scalars such that the addition and multiplication operations obey all the 7 axioms listed, then we have a vector space. Thus, to demonstrate that the set of p-codons $\mathbf{C}$ (with $\mathbf{B} = \{0,\ 1,\ \alpha,\ \alpha^2\}$ as scalars) is a vector space, we need to do two things

as per the definition: 1) define two operations: the addition of two p-codons and multiplication of a p-codon by a scalar and 2) show that the 7 axioms are satisfied under those operations. We do these in sections 2.2.1 through 2.2.3 below. Addition of p-codons and multiplication of p-codons by scalars are defined in terms of tuples of length n so as to generalize to our later needs. In the context of C, we simply set $n = 3$.

2.2.1 Addition of p-codons.

We define the addition of p-codons $\boldsymbol{a} = (a_1,\ a_2, ...,\ a_n)$ and $\boldsymbol{b} = (b_1,\ b_2, ...,\ b_n)$ as $\boldsymbol{a} + \boldsymbol{b} = (a_1 + b_1,\ a_2 + b_2, ...,\ a_n + b_n)$ where the + in $a_i + b_i$ for $i = 1, 2, ..., n$ is modulo 2 addition as given in addition table 1.2a.

2.2.2 Multiplication of p-codons by scalars.

We define the multiplication of a p-codon $\boldsymbol{a} = (a_1,\ a_2, ...,\ a_n)$ by a scalar λ in $\mathbf{B}$ as $\lambda \cdot \boldsymbol{a} = (\lambda a_1,\ \lambda a_2, ...,\ \lambda a_n)$. We will sometimes write $\lambda \cdot \boldsymbol{a}$ simply as $\lambda \boldsymbol{a}$.

2.2.3 Proof that axioms 1 through 7 are satisfied.

Having defined addition of p-codons and scalar multiplication, we are now in a position to show that set $\mathbf{C}$ satisfies axioms 1 through 7 and is thus a vector space. For the proofs below, it is important to keep two things in mind. First, the set of scalars $\mathbf{B} = \{0,\ 1,\ \alpha,\ \alpha^2\}$ is a field and so addition or multiplication of any members of the set always results in another member of the set (see Table 1.2). Second, $\mathbf{C}$ is the set of all 3-tuples whose 3 elements are chosen from $\mathbf{B}$ and so to show that some arbitrary 3-tuple $(\lambda_1\ \ \lambda_2\ \ \lambda_3)$ is a member of $\mathbf{C}$, we only need to show that $\lambda_1, \lambda_2,$ and λ_3 are elements of $\mathbf{B}$.

1. Closure.

(a) For any p-codons $\boldsymbol{u}$ and $\boldsymbol{w}$ in $\mathbf{C}$, $\boldsymbol{u}+\boldsymbol{w}$ is another p-codon in $\mathbf{C}$

Let $\boldsymbol{u} = (u_1\ u_2\ u_3)$ and $\boldsymbol{w} = (w_1\ w_2\ w_3)$. Then $\boldsymbol{u}+\boldsymbol{w} = (u_1+w_1 \quad u_2+w_2 \quad u_3+w_3)$. Since the sum of any elements in $\mathbf{B}$ yields another element in $\mathbf{B}$, $u_1+w_1,\ u_2+w_2,$ and u_3+w_3 are in $\mathbf{B}$. We conclude that $\boldsymbol{u}+\boldsymbol{w}$ is in $\mathbf{C}$, as desired.

(b) For any p-codon $\boldsymbol{v}$ in C and scalar a in $\mathbf{B}$, $a \cdot \boldsymbol{v}$ is a p-codon in C

Let $\boldsymbol{v} = (v_1\ v_2\ v_3)$. Then $a \cdot \boldsymbol{v} = (av_1\ \ av_2\ \ av_3)$. Since the product of any scalars in $\mathbf{B}$ yields another in $\mathbf{B}$, $av_1,\ av_2,$ and av_3 are in $\mathbf{B}$. We conclude that $a \cdot \boldsymbol{v}$ is in C, as desired.

2. Commutativity. For any p-codons $\boldsymbol{u}$ and $\boldsymbol{w}$ in C, $\boldsymbol{u}+\boldsymbol{w} = \boldsymbol{w}+\boldsymbol{u}$

Let $\boldsymbol{u} = (u_1\ u_2\ u_3)$ and $\boldsymbol{w} = (w_1\ w_2\ w_3)$. Then $\boldsymbol{u}+\boldsymbol{w} = (u_1+w_1 \quad u_2+w_2 \quad u_3+w_3)$ and $\boldsymbol{w}+\boldsymbol{u} = (w_1+u_1 \quad w_2+u_2 \quad w_3+u_3)$. Addition of scalars from B is commutative and so $u_i+w_i = w_i+u_i$ for $i = 1,\ 2,\ 3$. Thus, $\boldsymbol{u}+\boldsymbol{w} = \boldsymbol{w}+\boldsymbol{u}$.

3. Additive identity. There exists a zero p-codon, denoted $\mathbf{0}$, in C such that for any p-codon $\boldsymbol{u}$ in C, $\boldsymbol{u}+\mathbf{0} = \boldsymbol{u}$

Note that 0 is a scalar in B. When $a = 0$ in $1(b)$ above, we find that there exists a vector $\mathbf{0} = (0,\ 0,\ 0)$ in C. Let $\boldsymbol{u} = (u_1\ u_2\ u_3)$. Then $\boldsymbol{u}+\mathbf{0} = (u_1\ u_2\ u_3) + (0\ 0\ 0) = (u_1+0 \quad u_2+0 \quad u_3+0) = (u_1\ u_2\ u_3) = \boldsymbol{u}$, as desired.

4. Multiplicative Identity. For any p-codon $\boldsymbol{v}$ in C, $1 \cdot \boldsymbol{v} = \boldsymbol{v}$

Note that 1 is the multiplicative identity element in B. Let $\boldsymbol{v} = (v_1\ v_2\ v_3)$. Then $1 \cdot \boldsymbol{v} = (1 \cdot v_1 \quad 1 \cdot v_2 \quad 1 \cdot v_3) = (v_1\ v_2\ v_3) = \boldsymbol{v}$, as desired.

5. Additive inverses. For any p-codon $\boldsymbol{u} \in C$, there exists an additive inverse $-\boldsymbol{u} \in C$ such that $\boldsymbol{u}+(-\boldsymbol{u}) = \mathbf{0}$. Notice that each element in B has an additive inverse which is itself because addition is done in modulo 2 : $0+0 = 0$,

$1+1=0, \quad \alpha+\alpha=0, \quad$ and $\quad \alpha^2+\alpha^2=0$. Because addition of p-codons is defined componentwise, for any p-codon $\boldsymbol{u}=(u_1\ u_2\ u_3), \quad \boldsymbol{u}+\boldsymbol{u}=\mathbf{0}$ and so $\boldsymbol{u}$ is its own additive inverse. Thus, $-\boldsymbol{u}=-1\cdot\boldsymbol{u}=1\cdot\boldsymbol{u}=\boldsymbol{u}$. We conclude that each p-codon in C has an additive inverse, as required.

6. Associativity. For any $\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\in C$ and $a,b\in B$,

   (a) $\underline{\boldsymbol{u}+(\boldsymbol{v}+\boldsymbol{w})=(\boldsymbol{u}+\boldsymbol{v})+\boldsymbol{w}}$

$$\begin{aligned}\boldsymbol{u}+(\boldsymbol{v}+\boldsymbol{w}) &= (u_1\ u_2\ u_3)\ +\ (v_1+w_1,\ v_2+w_2,\ v_3+w_3)\\ &= (u_1+v_1+w_1,\ u_2+v_2+w_2,\ u_3+v_3+w_3)\\ &= (u_1+v_1,\ u_2+v_2,\ u_3+v_3)\ +\ (w_1\ w_2\ w_3)\\ &= (\boldsymbol{u}+\boldsymbol{v})+\boldsymbol{w}\end{aligned}$$

   (b) $\underline{a\cdot(b\cdot\boldsymbol{v})=(a\cdot b)\cdot\boldsymbol{v}}$

$$\begin{aligned}a\cdot(b\cdot\boldsymbol{v}) &= a\cdot(b\cdot v_1,\ b\cdot v_2,\ b\cdot v_3)\\ &= (a\cdot b\cdot v_1,\ a\cdot b\cdot v_2,\ a\cdot b\cdot v_3)\\ &= (a\cdot b)\cdot(v_1\ v_2\ v_3)\\ &= (a\cdot b)\cdot\boldsymbol{v}\end{aligned}$$

7. Distributivity. For any $\boldsymbol{u},\boldsymbol{v}\in C$ and $a,b\in B$,

(a) $\underline{a \cdot (\boldsymbol{u} + \boldsymbol{v}) = (a \cdot \boldsymbol{u}) + (a \cdot \boldsymbol{v})}$

$$
\begin{aligned}
a \cdot (\boldsymbol{u} + \boldsymbol{v}) &= a \cdot (u_1 + v_1,\ u_2 + v_2,\ u_3 + v_3) \\
&= (a \cdot (u_1 + v_1),\ a \cdot (u_2 + v_2),\ a \cdot (u_3 + v_3)) \\
&= (a \cdot u_1 + a \cdot v_1,\ a \cdot u_2 + a \cdot v_2,\ a \cdot u_3 + a \cdot v_3) \qquad \text{(step 3)} \\
&= (a \cdot u_1,\ a \cdot u_2,\ a \cdot u_3) \ + \ (a \cdot v_1,\ a \cdot v_2,\ a \cdot v_3) \\
&= a \cdot (u_1\ \ u_2\ \ u_3) \ + \ a \cdot (v_1\ \ v_2\ \ v_3) \\
&= (a \cdot \boldsymbol{u}) \ + (a \cdot \boldsymbol{v})
\end{aligned}
$$

(b) $\underline{(a + b) \cdot \boldsymbol{u} = (a \cdot \boldsymbol{u}) \ + \ (b \cdot \boldsymbol{u})}$

$$
\begin{aligned}
(a + b) \cdot \boldsymbol{u} &= (a + b) \cdot (u_1\ \ u_2\ \ u_3) \\
&= ((a + b) \cdot u_1,\ \ (a + b) \cdot u_2,\ \ (a + b) \cdot u_3) \\
&= (a \cdot u_1 + b \cdot u_1,\ \ a \cdot u_2 + b \cdot u_2,\ \ a \cdot u_3 + b \cdot u_3) \qquad \text{(step 3)} \\
&= (a \cdot u_1,\ \ a \cdot u_2,\ \ a \cdot u_3) \ + \ (b \cdot u_1,\ \ b \cdot u_2,\ \ b \cdot u_3) \\
&= a \cdot (u_1\ \ u_2\ \ u_3) \ + \ b \cdot (u_1\ \ u_2\ \ u_3) \\
&= (a \cdot \boldsymbol{u}) \ + \ (b \cdot \boldsymbol{u})
\end{aligned}
$$

Step 3 in (a) and (b) above both follow from their respective step 2 due to field axiom 6 in section 1.2. All other steps follow from the definition of p-codon addition and multiplication of p-codons by scalars given earlier in this section.

Having shown that C, the set of all 64 p-codons is a vector space, we can refer to the p-codons as vectors. We next discuss vector space concepts that are essential to our work. In doing this, we adopt some conventions. We will write $\mathbf{B}^3$ to mean the set of all 3-tuples whose elements are chosen from the set of scalars B. Since the vector space C fits this description, we will henceforth refer to it as $\mathbf{B}^3$. We also note

that the arguments given to show that C satisfies the vector space axioms extend naturally to $\mathbf{B}^n$, where n is a positive integer. Thus, $\mathbf{B}^n$ is a vector space under the same addition and scalar multiplication defined in sections 2.2.1 and 2.2.2 and is defined as:

$$\mathbf{B}^n = \{(u_1, u_2, ..., u_n) : u_i \in B \text{ for } i = 1, 2, ..., n\} \tag{2.1}$$

The vector space $\mathbf{B}^n$, where $n > 3$ will be needed in the next chapter and so the discussions below are done in the general context of $\mathbf{B}^n$. Obviously, when $n = 3$ we obtain the vector space C.

2.3 Subspaces of B^n

A subspace of $\mathbf{B}^n$ is defined as any nonempty subset of $\mathbf{B}^n$ which is itself a vector space under the same vector addition and scalar multiplication defined on $\mathbf{B}^n$ (given in sections 2.2.1 and 2.2.2). Let C be a subset of $\mathbf{B}^n$. To verify that C is a subspace, we only need to check that the closure axiom is satisfied on C. Thus, we only need to verify (2.2) below:

$$\text{For any } \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in C \text{ and } \lambda \in B, \quad (a)\ \boldsymbol{u} + \boldsymbol{v} \in C \qquad (b)\ \lambda \cdot \boldsymbol{w} \in C \tag{2.2}$$

To see why the closure axiom is sufficient to verify if C is a subspace, notice that because C is a subset of a vector space, axioms[1] $2, 4, 6$, and 7 are automatically satisfied. Also, because every vector in C is its own additive inverse due to modulo 2 addition, axiom 5 is satisfied on C, leaving the closure (axiom 1) and additive identity (axiom 3) axioms to be checked. The additive identity axiom requires that there be an identity vector which when added to any vector, leaves the vector unchanged. Because we have defined addition as componentwise, we only need show that $\mathbf{0} = (0, 0, ..., 0)$ is in C to satisfy the additive identity axiom. When $\lambda = 0$ in (2.2) above, the closure

[1] Given in the vector space definition (Definition 1)

axiom verifies the membership of $\mathbf{0}$ in C. Thus, any nonempty subset of $\mathbf{B}^n$ that satisfies the closure axiom satisfies all the vector space axioms and is a subspace. Examples: $\{(0,0,0),\ (0,0,1),\ (0,0,\alpha),\ (0,0,\alpha^2)\}$ and $\{(0,0,0),\ (0,1,\alpha),\ (0,\alpha,\alpha^2),\ (0,\alpha^2,1)\}$ are two subspaces of $\mathbf{B}^3$ since the closure axiom (stated in (2.2)) is satisfied on them.

2.4 Linear Combination, Span, and Linear Independence

2.4.1 Linear Combination

Let $\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_k}$ be vectors in $\mathbf{B}^n$. A linear combination of $\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_k}$ is another vector in $\mathbf{B}^n$ of the form $\lambda_1 \boldsymbol{v_1} + \lambda_2 \boldsymbol{v_2} + \cdots + \lambda_k \boldsymbol{v_k}$, where $\lambda_1, \lambda_2, \ldots, \lambda_k$ are scalars chosen from B. For example, one linear combination of the vectors $(0,1,\alpha)$, $(0,0,1)$, and $(0,1,\alpha^2)$ in $\mathbf{B}^3$ is $0\cdot(0,1,\alpha) + \alpha\cdot(0,0,1) + \alpha^2\cdot(0,1,\alpha^2) = (0,\alpha^2,0)$, which is another vector in $\mathbf{B}^3$. Another linear combination is obtained by choosing different scalars: $1\cdot(0,1,\alpha) + 0\cdot(0,0,1) + 0\cdot(0,1,\alpha^2) = (0,1,\alpha)$.

2.4.2 Span

Let $U = \{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_k}\}$ be a nonempty subset of $\mathbf{B}^n$. The span of U, denoted span(U), is the set of all linear combinations of the vectors in U.

$$\text{span}(U) = \{\lambda_1 \boldsymbol{u_1} + \lambda_2 \boldsymbol{u_2} + \cdots + \lambda_k \boldsymbol{u_k} \ :\ \lambda_i \in B\} \tag{2.3}$$

For example, let $U = \{(1,0,1)\}$ be a subset of $\mathbf{B}^3$. Then span$(U) = \{0\cdot(1,0,1),\ 1\cdot(1,0,1),\ \alpha\cdot(1,0,1),\ \alpha^2\cdot(1,0,1)\} = \{(0,0,0),\ (1,0,1),\ (\alpha,0,\alpha),\ (\alpha^2,0,\alpha^2)\}$. We note that span$(U) = \{(0,0,0),\ (1,0,1),\ (\alpha,0,\alpha),\ (\alpha^2,0,\alpha^2)\}$ is a subspace of $\mathbf{B}^3$. This claim is verified by observing that the addition of any two vectors in span(U) yields a vector in span(U) and multiplying any vector in span(U) by a scalar in B yields a vector in span(U). Span(U) is not unique in being a subspace. In general, the span

of any nonempty subset of $\mathbf{B}^n$ is a subspace. This claim is proved below.

Proof: If U is a nonempty subset of $\mathbf{B}^n$, span(U) is a subspace.

Let $U = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k\}$ be a nonempty subset of $\mathbf{B}^n$. Then span(U) is given in (2.3) and vectors in span(U) are of the form $\lambda_1\boldsymbol{u}_1 + \lambda_2\boldsymbol{u}_2 + \cdots + \lambda_k\boldsymbol{u}_k$ where $\lambda_i \in B$. Let $\boldsymbol{x} = \eta_1\boldsymbol{u}_1 + \eta_2\boldsymbol{u}_2 + \cdots + \eta_k\boldsymbol{u}_k$ and $\boldsymbol{y} = \gamma_1\boldsymbol{u}_1 + \gamma_2\boldsymbol{u}_2 + \cdots + \gamma_k\boldsymbol{u}_k$ be any two vectors in span(U).

Then $\boldsymbol{x}+\boldsymbol{y} = (\eta_1+\gamma_1)\boldsymbol{u}_1+(\eta_2+\gamma_2)\boldsymbol{u}_2+\cdots+(\eta_k+\gamma_k)\boldsymbol{u}_k$. Because $\eta_i+\gamma_i \in B$, $\boldsymbol{x}+\boldsymbol{y} \in$ span(U). Let λ be any scalar in B. Then $\lambda\boldsymbol{x} = (\lambda\eta_1)\boldsymbol{u}_1+(\lambda\eta_2)\boldsymbol{u}_2+\cdots+(\lambda\eta_k)\boldsymbol{u}_k$. Because $\lambda\eta_i \in B$, $\lambda \cdot \boldsymbol{x} \in$ span(U). It has been shown that the closure axiom is satisfied on span(U). As discussed in section 2.3, this is sufficient to conclude that span(U) is a subspace. ■

Important terminology: Let U be a nonempty subset of $\mathbf{B}^n$ and C a subspace of $\mathbf{B}^n$. If span$(U) = C$, we say that U *spans* C and call U a *generating* or *spanning set* of C. This terminology is appropriate because we can have only U and still obtain all vectors in the subspace C simply by forming linear combinations of the vectors in U. Using this terminology in the context of the example given earlier, the subset $U = \{(1,0,1)\}$ of $\mathbf{B}^3$ spans or generates the subspace $C = \{(0,0,0),\ (1,0,1),\ (\alpha,0,\alpha),\ (\alpha^2,0,\alpha^2)\}$ since span$(U) = C$.

2.4.3 Linear Independence

A set of vectors from $\mathbf{B}^n$ $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k\}$ is linearly independent if the only choice of scalars $\lambda_1, \lambda_2, \ldots, \lambda_k$ from B that makes $\lambda_1\boldsymbol{u}_1 + \lambda_2\boldsymbol{u}_2 + \cdots + \lambda_k\boldsymbol{u}_k$ equal $\mathbf{0}$ is $\lambda_1 = \lambda_2 = \cdots = \lambda_k = 0$. For example, $\{(1,0,0),\ (0,0,\alpha)\}$ is a linearly independent set of vectors from $\mathbf{B}^3$ because λ_1 and λ_2 must both be 0 in order for $\lambda_1 \cdot (1,0,0) + \lambda_2 \cdot (0,0,\alpha) = (0,0,0) = \mathbf{0}$ to be true. In contrast, $\{(\alpha,0,1),\ (1,0,\alpha^2)\}$ is not linearly

independent since we can choose values for λ_1 and λ_2 other than $\lambda_1 = \lambda_2 = 0$ such that $\lambda_1 \cdot (1,0,0) \ + \ \lambda_2 \cdot (0,0,\alpha) = \mathbf{0}$. For example, $\alpha^2 \cdot (1,0,0) \ + \ 1 \cdot (0,0,\alpha) = \mathbf{0}$. Such a set that is not linearly independent is called linearly dependent.

Consider the subset $U = \{(1,0,1)\}$ of $\mathbf{B}^3$. As discussed in section 2.4.2, U spans the subspace $C = \{(0,0,0),\ (1,0,1),\ (\alpha,0,\alpha),\ (\alpha^2,0,\alpha^2)\}$. Thus, span($U$)= C. Note that U is linearly independent since the only λ that makes $\lambda \cdot (1,0,1) = \mathbf{0}$ is $\lambda = 0$. Now consider the set $V = \{(1,0,1),\ (\alpha,0,\alpha),\ (\alpha^2,0,\alpha^2)\}$ which was formed by adding two vectors, $(\alpha,0,\alpha)$ and $(\alpha^2,0,\alpha^2)$, to U. Note that V is linearly dependent since $1 \cdot (1,0,1) \ + \ 1 \cdot (\alpha,0,\alpha) \ + \ 1 \cdot (\alpha^2,0,\alpha^2) = \mathbf{0}$. It can be checked that span(V)= C. Thus, despite having two extra vectors, V still has the same span as U. This is explained by the fact that V is linearly dependent. In any linearly dependent set, there is at least one vector which is already in the span of the other vectors in the set such that its removal does not change the span of the set[2].

This illustrates an important fact which we will not formally prove: given a linearly independent set and a linearly dependent set that both span a subspace of $\mathbf{B}^n$, the linearly independent set will have fewer number of vectors.

2.5 Bases and Dimension of B^n

2.5.1 Bases

A basis of the vector space $\mathbf{B}^n$ is defined as a linearly independent set that spans $\mathbf{B}^n$. Let W be such a basis of $\mathbf{B}^n$. Then by definition, W spans $\mathbf{B}^n$ and so every vector in $\mathbf{B}^n$ can be obtained from a linear combination of the vectors in W. Thus, though $\mathbf{B}^n$ has 4^n vectors, we need only keep track of the $|W|$ vectors in the basis W. Also, the linear independence requirement in the definition implies that $|W|$ is

[2]This does not hold when the linearly dependent set in question is $\{(0,0,0)\}$. We could however define span of the empty set to be $\{(0,0,0)\}$ for this case.

the smallest number of vectors needed to span $\mathbf{B}^n$.

Several sets of vectors that are linearly independent and span $\mathbf{B}^n$ exist. Thus, there are several possible sets to use as basis of $\mathbf{B}^n$, but we will use the simplest:

$$W = \{\underbrace{(1,0,\ldots,0)}_{1\text{st}},\ \underbrace{(0,1,0,\ldots,0)}_{2\text{nd}},\ldots,\ \underbrace{(0,\ldots,0,1)}_{n\text{th}}\} \tag{2.4}$$

$|W| = n$ and each vector in W is an n-tuple. The first vector has 1 in its first position and 0 in the other positions, the second vector has 1 in its second position and 0 in the other positions, and so forth. Finally, the nth vector has 1 in its nth position and 0 in other positions. Next, we prove that W is indeed linearly independent and spans $\mathbf{B}^n$.

$$\lambda_1 \cdot (1,0,\ldots,0) + \lambda_2 \cdot (0,1,0,\ldots,0) + \cdots + \lambda_n \cdot (0,\ldots,0,1) \ = \ (\lambda_1, \lambda_2, \ldots, \lambda_n) \tag{2.5}$$

The linear combination of the basis vectors is given in (2.5), where $\lambda_i \in B$. Because the only scalars that make $(\lambda_1, \lambda_2, \ldots, \lambda_n) = (0,0,\ldots,0)$ true is $\lambda_1 = \cdots = \lambda_n = 0$, we conclude that (2.4) is linearly independent. With the linear combination of the vectors in W given in (2.5) as $(\lambda_1, \lambda_2, \ldots, \lambda_n)$, it is clear that by choosing $\lambda_1 = v_1,\ \lambda_2 = v_2, \ldots, \lambda_n = v_n$, any vector $(v_1, v_2, \ldots, v_n)$ in $\mathbf{B}^n$ can be written as a linear combination of the vectors in W. We conclude that W spans $\mathbf{B}^n$. As such, we have verified that (2.4) is indeed a basis of $\mathbf{B}^n$.

A basis is by definition required to be linearly independent. This linear independence requirement leads to the fact that there is a unique way of expressing any vector $\boldsymbol{v}$ in $\mathbf{B}^n$ as a linear combination of the basis vectors. For example, in $\mathbf{B}^3$, (2.4) reduces to the basis $W = \{(1,0,0),\ (0,1,0),\ (0,0,1)\}$ and the only way of expressing the vector $(1, \alpha, \alpha^2)$ in $\mathbf{B}^3$ as a linear combination of the basis is $1 \cdot (1,0,0) \ + \ \alpha \cdot (0,1,0) \ + \ \alpha^2 \cdot (0,0,1)$. Though we prove this fact in the context of our chosen basis (given in (2.4)), it holds for every other basis of $\mathbf{B}^n$.

Let $\boldsymbol{v} \in \mathbf{B}^n$. Assume that $\boldsymbol{v}$ can be written as two different linear combinations of the basis vectors. Thus $\boldsymbol{v} = \lambda_1(1,0,\ldots,0) + \cdots + \lambda_n(0,\ldots,0,1)$ and $\boldsymbol{v} = \eta_1 \cdot (1,0,\ldots,0) + \cdots + \eta_n \cdot (0,\ldots,0,1)$. Subtracting these two expressions results in

$$\mathbf{0} = (\lambda_1 - \eta_1) \cdot (1,0,\ldots,0) + \cdots + (\lambda_n - \eta_n) \cdot (0,\ldots,0,1) \tag{2.6}$$

Recalling that $\{(1,0,\ldots,0),\ (0,1,0,\ldots,0),\ldots,\ (0,\ldots,0,1)\}$ is a linearly independent set, (2.6) holds only if $(\lambda_1 - \eta_1) = (\lambda_2 - \eta_2) = \ldots = (\lambda_n - \eta_n) = 0$. This implies that $\lambda_1 = \eta_1,\ \lambda_2 = \eta_2, \ldots, \lambda_n = \eta_n$. We thus conclude that the two supposedly different linear combinations are actually the same and so the expression of $\boldsymbol{v}$ as a linear combination of the basis vectors is unique.

2.5.2 Dimension

Dimension of a vector space is defined as the number of vectors in its basis. $W = \{(1,0,\ldots,0),\ (0,1,0,\ldots,0),\ldots,\ (0,\ldots,0,1)\}$ is a basis of $\mathbf{B}^n$. $|W| = n$ and so the dimension of $\mathbf{B}^n$ is n. As mentioned earlier, several other basis of $\mathbf{B}^n$ exist. It can however be shown that each such basis has n vectors[3]. As such, the dimension of $\mathbf{B}^n$ is independent of which basis of $\mathbf{B}^n$ is being considered. We will write $dim(V)$ to mean the dimension of some vector space V.

2.6 Orthogonal Complements

Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_n)$ and $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ be vectors in $\mathbf{B}^n$. The inner product of $\boldsymbol{u}$ and $\boldsymbol{w}$, denoted $\boldsymbol{u} \cdot \boldsymbol{w}$, is defined as $\boldsymbol{u} \cdot \boldsymbol{w} = u_1w_1 + u_1w_1 + \cdots + u_nw_n$. When $\boldsymbol{u} \cdot \boldsymbol{w} = 0$ we say $\boldsymbol{u}$ and $\boldsymbol{v}$ are orthogonal to each other.

Let A be a nonempty subset of $\mathbf{B}^n$. The orthogonal complement of A, denoted $A^{\perp}$,

[3]Proof of this statement is not given as it's not directly relevant to our work.

is the set of all vectors in $\mathbf{B}^n$ that are orthogonal to every vector in A. Mathematically,

$$A^{\perp} = \{\ \boldsymbol{x} \in \mathbf{B}^n \ \ : \ \ \boldsymbol{x} \cdot \boldsymbol{a} = 0 \ \text{ for all } \boldsymbol{a} \in A\ \} \tag{2.7}$$

The orthogonal complement of any nonempty subset of $\mathbf{B}^n$ is a subspace. We offer a proof below. As discussed in section 2.3, the closure axiom alone is sufficient to prove that a subset of a vector space is a subspace.

Proof: Given any nonempty subset $A \subseteq \mathbf{B}^n$, $A^{\perp}$ is a subspace.

Let $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in A^{\perp}$, $\boldsymbol{a} \in A$, and $\lambda \in \mathbf{B}$. Then $(\boldsymbol{u} + \boldsymbol{v}) \cdot \boldsymbol{a} = \boldsymbol{u} \cdot \boldsymbol{a} + \boldsymbol{v} \cdot \boldsymbol{a} = 0$, which implies that $(\boldsymbol{u} + \boldsymbol{v}) \in A^{\perp}$. Also, $(\lambda \boldsymbol{w}) \cdot \boldsymbol{a} = \lambda(\boldsymbol{w} \cdot \boldsymbol{a}) = 0$, which implies $(\lambda \boldsymbol{w}) \in A^{\perp}$. Because $(\boldsymbol{u} + \boldsymbol{v}) \in A^{\perp}$ and $(\lambda \boldsymbol{w}) \in A^{\perp}$, the closure axiom is satisfied and we conclude that $A^{\perp}$ is a subspace. ■

The fact proved above will be useful to our work for the case where A is not only a subset but also a subspace. As such, we take A as a subspace in the discussions that follow. Being subspaces, A and $A^{\perp}$ each have a set of basis vectors and hence a dimension. A well-known linear algebra result is that the dimensions of a subspace plus the dimension of its orthogonal complement equals the dimension of the vector space they are subsets of. In our case, since $dim(\mathbf{B}^n) = n$ and $A, A^{\perp} \subseteq \mathbf{B}^n$,

$$dim(A) + dim(A^{\perp}) = n \tag{2.8}$$

Let G and H be matrices whose rows are the basis vectors of A and $A^{\perp}$, respectively. Such matrices are called generator matrices. If $dim(A) = k$, G is a $k \times n$ matrix and H is an $(n - k) \times n$ matrix. Note that G and H both have n columns because vectors in $\mathbf{B}^n$ are of length n. Because every vector in A is orthogonal to every other vector in $A^{\perp}$, the following equations hold and will be used in chapter 3.

$$\boldsymbol{a} H^T = \mathbf{0} \ \text{ for any } \ \boldsymbol{a} \in A \tag{2.9}$$

$$GH^T = \mathbf{0} \tag{2.10}$$

Chapter 3

TOLERATING SUBSTITUTION MUTATIONS

3.1 Introduction

In the previous chapter, we noted that the algebraic representations of naturally-occurring codons form a vector space, which we denoted as $\mathbf{B}^3$. One vector in $\mathbf{B}^3$ is $(0\ 1\ \alpha)$, which is the algebraic representation of ATG, which in a DNA sense strand, represents Methionine. Consider a substitution mutation event which changes the p-codon $(0\ 1\ \alpha)$ to $(0\ \alpha^2\ \alpha)$, which is the algebraic representation of ACG and will ultimately be translated as Threonine, instead of the originally intended Methionine.

Due to the degeneracy of the genetic code, some substitution mutations result in another codon that still codes for the same amino acid. Take Cysteine as an example. In a DNA sense strand, Cysteine is coded for by both TGT (p-codon $(1\ \alpha\ 1)$) and TGC (p-codon $(1\ \alpha\ \alpha^2)$). If the last nucleotide is mutated from C to T or T to C, there will be no subsequent change in the resulting amino acid after translation. Any other substitution mutation will result in a codon which codes for an unintended amino acid.

Our goal in this chapter is to find a set of 64 'unnatural' codons that are each able to tolerate an arbitrarily chosen number of substitution mutations to any nucleotide. That is, codons that can sustain some chosen number of mutation errors and still guarantee that upon translation, the amino acid sequence will not differ from that originally intended. To do this, we will employ ideas from coding theory. These ideas sum up to building some sort of redundancy into the naturally occuring codons. As such, the codons we develop will necessarily each have more than 3 nucleotides and

will be chosen from the vector space $\mathbf{B}^n$, where $n > 3$. Exactly which n we will use will be noted later. We introduce a few concepts that will be needed.

3.2 Hamming Distance and Weight

3.2.1 Hamming Distance

The Hamming distance (Hamming, 1950) between two vectors $\boldsymbol{u} = (u_1, u_2, ..., u_n)$ and $\boldsymbol{v} = (v_1, v_2, ..., v_n)$ in $\mathbf{B}^n$ is the number of places where $u_i \neq v_i$. For example, the Hamming distance between $\boldsymbol{u} = (0\ 1\ \alpha)$ and $\boldsymbol{v} = (1\ 0\ \alpha)$, is 2 since they differ in two places. We will denote the Hamming distance between $\boldsymbol{u}$ and $\boldsymbol{v}$ by $d(\boldsymbol{u}, \boldsymbol{v})$. Note that $d(\boldsymbol{u}, \boldsymbol{v})$ can only take the values $0, 1, 2, \ldots, n$. Also note that $d(\boldsymbol{u}, \boldsymbol{v}) = d(\boldsymbol{v}, \boldsymbol{u})$. Another important property of the Hamming distance is that it satisfies the triangle inequality[1]. Thus, given vectors $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$, $\ d(\boldsymbol{u}, \boldsymbol{w}) \leq d(\boldsymbol{u}, \boldsymbol{v}) + d(\boldsymbol{v}, \boldsymbol{w})$.

3.2.2 Hamming Weight

Let $\boldsymbol{u} = (u_1, u_2, ..., u_n)$ be some vector in $\mathbf{B}^n$. Its Hamming weight, which we will denote as $wt(\boldsymbol{u})$, is the number of places where $u_i \neq 0$. In other words, $wt(\boldsymbol{u}) = d(\boldsymbol{u}, 0)$. For example, the weight of $\boldsymbol{u} = (0\ 1\ \alpha^2)$ is $wt(\boldsymbol{u}) = 2$.

3.2.3 Important Relationship between Hamming Distance and Weight

Consider vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ in $\mathbf{B}^n$ and let $\boldsymbol{w} = \boldsymbol{u} + \boldsymbol{v}$. Then, for $i = 1, 2, \ldots, n$, $\boldsymbol{w_i} = 0$ if and only if $\boldsymbol{u_i} = \boldsymbol{v_i}$ since we use base 2 addition. In other words, the coordinates of $\boldsymbol{w}$ will be non-zero only at coordinates where $\boldsymbol{u}$ and $\boldsymbol{v}$ differ. As such, we can find the Hamming distance between two vectors by first adding them and then

[1] A proof of this will not be directly relevant to our work. As such, it is not given.

taking the Hamming weight of the result. Mathematically, we write

$$d(\boldsymbol{u}, \boldsymbol{v}) = wt(\boldsymbol{u} + \boldsymbol{v}) \tag{3.1}$$

The above relationship becomes very useful when we desire to find the Hamming distance between all unique pairs of vectors in a vector space (or subspace) and report the minimum, called the minimum Hamming distance and denoted d_{min}. Take $\mathbf{B}^n$ as the vector space of interest. Since $\mathbf{B}^n$ has 4^n vectors, if we naively tried to find $d_{min}(\mathbf{B}^n)$ without the relationship given in the previous paragraph, we will have to find the Hamming distance between $(4^n * (4^n - 1))/2$ pairs of vectors and report the minimum as $d_{min}(\mathbf{B}^n)$. By making use of the relationship $d(\boldsymbol{u}, \boldsymbol{v}) = wt(\boldsymbol{u} + \boldsymbol{v})$, we need only find the weight of each of the $4^n - 1$ non-zero vectors and report the minimum as $d_{min}(\mathbf{B}^n)$. Thus, the claim is that given some vector space V,

$$d_{min}(V) = wt_{min}(V) \tag{3.2}$$

where $wt_{min}(V)$ is the minimum of the weights of the non-zero vectors in V.

To see why (3.2) is true, first note that if $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ are different vectors in the vector space V, then their sum, $\boldsymbol{w} = \boldsymbol{v_1} + \boldsymbol{v_2}$, is a non-zero vector also in V. Thus, if we put all the vectors in V into all possible pairs of two, say $(\boldsymbol{v_i}, \boldsymbol{v_j})$, where $i \neq j$, then the sum of each pair will yield some $\boldsymbol{w}$ such that the set of all the $\boldsymbol{w}$s will be equal to the non-zero vectors in V. Combining this with (3.1), it follows that if we want to find $d_{min}(V)$, instead of finding the Hamming distance between all possible pairs of unique vectors and checking the minimum, we can simply check the weight of each non-zero vector in V and record the minimum as our $d_{min}(V)$ because each non-zero vector in V is the result of summing some two different vectors in V.

3.3 Tolerating t Substitution Mutations per Codon

Our goal has been to find some 64 'unnatural' codons to used in place of the naturally-occurring codons in the construction of genes such that the genes are capable of tolerating an arbitrarily chosen number of substitution mutations per codon. We will put the p-codons of these 64 unnatural codons in a set $M = \{\boldsymbol{m_1}, \boldsymbol{m_2}, \ldots, \boldsymbol{m_{64}}\}$. To make use of equation (3.2), we will require that M be a subspace of $\mathbf{B}^n$, where $n > 3$. This requirement also simplifies our work greatly because as a subspace, M is completely defined by its basis. So, we need only find the basis vectors of M. All the other vectors can be obtained via linear combinations of the basis vectors.

3.3.1 Substitution Mutation as Vector Addition

Consider the p-codon $\boldsymbol{u} = (0\ 1\ \alpha) \in \mathbf{B}^3$ and assume that some substitution mutation modified $\boldsymbol{u}$ into $\widetilde{\boldsymbol{u}} = (1\ 0\ \alpha)$. Since $\widetilde{\boldsymbol{u}} = \boldsymbol{u} + (1\ 1\ 0)$, we can refer to $(1\ 1\ 0)$ as the mutation vector. In general, for any $\boldsymbol{u} \in \mathbf{B}^n$, if a mutation event results in $\boldsymbol{u}$ becoming $\widetilde{\boldsymbol{u}}$ such that $\widetilde{\boldsymbol{u}} = \boldsymbol{u} + \boldsymbol{w}$, where $\boldsymbol{w} \in \mathbf{B}^3$, we shall call $\boldsymbol{w}$ the mutation vector and write $\widetilde{m}(\boldsymbol{u}, \boldsymbol{w})$ to mean the vector $\widetilde{\boldsymbol{u}}$ that results from mutating $\boldsymbol{u}$ with the mutation vector $\boldsymbol{w}$. Thus,

$$\widetilde{m}(\boldsymbol{u}, \boldsymbol{w}) = \widetilde{\boldsymbol{u}} = \boldsymbol{u} + \boldsymbol{w} \tag{3.3}$$

We now present a theorem that is essential to finding M. It gives the restriction in terms of Hamming distance that must be placed on the vectors in M to ensure that they have properties (also given in the theorem) necessary to tolerate mutations.

Theorem 3.1. *Let $M = \{\boldsymbol{m_1}, \boldsymbol{m_2}, \ldots, \boldsymbol{m_{64}}\}$ be a subspace of $\boldsymbol{B}^n$, where $n > 3$. If $d_{min}(M) \geq 2t+1$, then if t or less substitution mutations occur in any codon $\boldsymbol{m_i} \in M$ so that it becomes $\widetilde{\boldsymbol{m_i}}$, then*

1. $d(\boldsymbol{m_i}, \widetilde{\boldsymbol{m_i}}) \leq t$

2. $\widetilde{\boldsymbol{m_i}} \notin M$

3. $d(\widetilde{\boldsymbol{m_i}}, \boldsymbol{m_j}) > t$ *for any* $\boldsymbol{m_j} \in M$, *where* $\boldsymbol{m_j} \neq \boldsymbol{m_i}$

Proof

1. Since t or less substitution mutations occurred in $\boldsymbol{m_i}$, it follows that $\widetilde{\boldsymbol{m_i}}$ and $\boldsymbol{m_i}$ differ in at most t places and hence $d(\boldsymbol{m_i}, \widetilde{\boldsymbol{m_i}}) \leq t$.

2. It takes a mutation vector of weight $\geq 2t+1$ to turn any codon in M into a different codon in M since $d_{min}(M) \geq 2t+1$. Since $\leq t$ substitution mutations occurred in $\boldsymbol{m_i}$, $\widetilde{\boldsymbol{m_i}}$ cannot possibly be in M.

3. We prove this by contradiction so assume that $d(\widetilde{\boldsymbol{m_i}}, \boldsymbol{m_j}) \leq t$. We know from 1) that $d(\boldsymbol{m_i}, \widetilde{\boldsymbol{m_i}}) \leq t$. Then, $d(\boldsymbol{m_i}, \widetilde{\boldsymbol{m_i}}) + d(\widetilde{\boldsymbol{m_i}}, \boldsymbol{m_j}) \leq 2t$. Using the triangle inequality, we have that $d(\boldsymbol{m_i}, \boldsymbol{m_j}) \leq d(m_i, \widetilde{\boldsymbol{m_i}}) + d(\widetilde{\boldsymbol{m_i}}, \boldsymbol{m_j}) \leq 2t$. This is a contradiction, since we know that $d(\boldsymbol{m_i}, \boldsymbol{m_j}) \geq 2t+1$ because $d_{min}(M) \geq 2t+1$. We conclude that $d(\widetilde{\boldsymbol{m_i}}, \boldsymbol{m_j}) > t$, as desired. ■

Using theorem 3.1, our task of developing codons capable of tolerating t mutations is simplified and can be stated as: find some set of vectors $M = \{\boldsymbol{m_1}, \boldsymbol{m_2}, \ldots, \boldsymbol{m_{64}}\}$ such that $d_{min}(M) \geq 2t+1$. Upon finding such an M, we are guaranteed that any mutation event with a mutation vector of weight t or less that occurs in any codon in M will result in a new codon that is none of the vectors in M but still closest in Hamming distance to the original codon and can thus be decoded as the original codon. We next use the concept of orthogonal complements previously discussed in section 2.6.

3.3.2 Finding Generator Matrices for $M^{\perp}$ and M

Let $H = [h_1, h_2, \ldots, h_n]$, where h_i is the ith column of H, be the generator matrix for $M^{\perp}$. Then from equation (2.9) we know that $\boldsymbol{a}H^T = \mathbf{0}$ for $\boldsymbol{a} \in M$. Thus, if $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, $a_1h_1 + a_2h_2 + \ldots a_nh_n = \mathbf{0}$. Let $k = 2t + 1$. According to equation (3.2), $d_{min}(M) = k$ implies that $wt_{min}(M) = k$, which has two implications that combine with the fact that $a_1h_1 + a_2h_2 + \ldots a_nh_n = \mathbf{0}$ to tell us how to construct H. These are discussed below:

1. $wt_{min}(M) = k$ implies that at least one vector in M has weight k.

   Let $m = (m_1, m_2, \ldots, m_k) \in M$ be such a vector with weight k and let $m_{i1}, m_{i2}, \ldots, m_{ik}$ be its k non-zero coordinates. Then $m_1h_1 + m_2h_2 + \ldots + m_kh_k = m_{i1}h_{i1} + m_{i2}h_{i2} + \ldots + m_{ik}h_{ik} = \mathbf{0}$. This implies that the generator matrix H has some k columns that are linearly dependent.

2. $wt_{min}(M) = k$ implies that there is no vector in M with weight $k - 1$.

   This means there is no $m \in M$ with the $k - 1$ non-zero coordinates m_{i1}, m_{i2} $, \ldots, m_{i(k-1)}$ such that $m_{i1}h_{i1} + m_{i2}h_{i2} + \ldots + m_{i(k-1)}h_{i(k-1)} = \mathbf{0}$. This implies that any $k - 1$ columns of H are linearly independent.

The two discussions above tell us how the requirement of $d_{min}(M) = k$ on M translates to restrictions on the generator matrix H of its orthogonal complement $M^{\perp}$. These restrictions tell us how to construct H: construct H such that 1) some k columns are linearly dependent and 2) any $k - 1$ columns are linearly independent, where $k = 2t + 1$ and t is the arbitrary number of substitution mutations we want to tolerate per codon. Once H is constructed, we can then find the generator matrix of M, call it G, from the equation $GH^T = \mathbf{0}$, which was given in section 2.6. Note that G completely tells us the vectors in $M = \{\boldsymbol{m_1}, \boldsymbol{m_2}, \ldots, \boldsymbol{m_{64}}\}$ since its rows are a basis of M. The approach is demonstrated in the next section.

3.4 Example: Tolerating 1 Substitution Mutation per Codon

Here, we use the ideas discussed in the previous sections, particularly section 3.3.2, to find an example set of 64 alternative codons $M = \{\boldsymbol{m_1}, \boldsymbol{m_2}, \ldots, \boldsymbol{m_{64}}\}$, where $M \subseteq \mathbf{B}^n$ and $n > 3$, such that a gene constructed from these codons is guaranteed to tolerate any 1 substitution mutation per codon[2].

We know that if $d_{min}(M) = wt_{min}(M) \geq 2t + 1$, then M will have the structure necessary for tolerating any mutation with mutation vector of weight less than or equal to t. For our example, $t = 1$ and so we will form an M with $d_{min}(M) = wt_{min}(M) = 3$. Using ideas discussed in the prior section, this requires that we find G, the generator matrix of M, from $GH^T = \mathbf{0}$ where H, the generator matrix of $M^{\perp}$, is a matrix with two restrictions: 1) some 3 columns are linearly dependent and 2) any 2 columns of H are linearly independent.

As noted in section 2.6, if G has size $k \times n$, H will have size $(n - k) \times n$. To find H and then G, we need to know what n and k are. M is a subspace of 64 vectors, one to 'replace' each naturally-occurring codon. Given that we have 4 scalars, $0, 1, \alpha,$ and α^2, M must have 3 basis vectors so that all possible linear combinations of the 3 basis vectors yields 64 vectors. By definition, the rows of G are the basis vectors of M. Thus, G has $k = 3$ rows. We will choose the subspace M from the vector space $\mathbf{B}^5$ and so $n = 5$. This choice will be explained in the next section. Given $n = 5$ and $k = 3$, G and H have sizes 3×5 and 2×5, respectively. Let's construct H.

Consider the set of vectors A=$\{(1, \alpha^2),\ (1, \alpha),\ (\alpha, \alpha),\ (1, 0),\ (0, 1)\}$. Since $1 \cdot (\alpha, \alpha) + \alpha \cdot (1, 0) + \alpha \cdot (0, 1) = \mathbf{0}$, (α, α), $(1, 0)$ and $(0, 1)$ are linearly dependent. Also, one can verify that any two vectors from A are linearly independent. The vectors in

[2]Thus, any codon can tolerate a mutation with mutation vector of weight 1.

A thus satisfy the two requirements on the the column vectors of H. As such, we can use these vectors as the column vectors of H, as shown below.

$$H = \begin{bmatrix} 1 & 1 & \alpha & 1 & 0 \\ \alpha^2 & \alpha & \alpha & 0 & 1 \end{bmatrix} \tag{3.4}$$

Note that H is not unique. Any matrix of size 2×5 with column vectors that satisfy the two requirements mentioned earlier can be used. Also note that the submatrix formed from the last two columns is the 2×2 identity matrix $I_2 = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. This choice was made so we could make use of the fact that if $H = [P \mid I_{n-k}]$, where I_{n-k} is the $n-k \times n-k$ identity matrix, then $G = [I_k \mid P^T]$ satisfies $GH^T = \mathbf{0}$. Using this fact and H given in equation (3.4), we have G as:

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & \alpha^2 \\ 0 & 1 & 0 & 1 & \alpha \\ 0 & 0 & 1 & \alpha & \alpha \end{bmatrix} \tag{3.5}$$

The rows of G form a basis of the set $M = \{\boldsymbol{m_1}, \boldsymbol{m_2}, \ldots, \boldsymbol{m_{64}}\}$. Thus, every $\boldsymbol{m_i} \in M$ can be obtained from a linear combination of the form $\boldsymbol{m_i} = \lambda_1 \cdot (1, 0, 0, 1, \alpha^2) \ + \ \lambda_2 \cdot (0, 1, 0, 1, \alpha) \ + \ \lambda_3 \cdot (0, 0, 1, \alpha, \alpha)$, where λ_i are scalars. This linear combination can be written as the matrix multiplication given below:

$$\boldsymbol{m_i} = (\lambda_1, \lambda_2, \lambda_3) \cdot G \tag{3.6}$$

Thus, G maps each vector $(\lambda_1, \lambda_2, \lambda_3) \in \boldsymbol{B}^3$ uniquely to a vector $(\omega_1, \omega_2, \omega_3, \omega_4, \omega_5) \in M$. In essence, G tells us which unnatural codon of length 5 to use in place of each naturally-occuring codon of length 3 such that genes constructed from the unnatural

codons can tolerate up to one substitution mutation per codon. For example, the p-codon of ATG is $(0\ 1\ \alpha)$. $(0\ 1\ \alpha)\cdot G = (0\ 1\ \alpha\ \alpha\ 1)$, which is the p-codon for ATGGT. We will thus use the synthetic codon ATGGT in place of ATG. Table 3.1 shows results from evaluating equation (3.6) for all p-codons of the naturally occuring codons. It shows how all 64 naturally-occuring codons should be mapped to the mutation-tolerant (mutol) codons of length 5 in set M such that genes constructed from the mutol codons will be tolerant of up to one substitution mutation per codon. Since we represented both thymine and uracil with scalar 1 as outlined in table 1.1, T in the codons column of table 3.1 can simply be replaced with U to get the corresponding RNA codons.

Three observations can be made about the 5-letter codons in table 3.1 we claim will confer mutation-tolerance upon genes constructed from them: 1) each of the 5-letter unnatural codons can have any 1 of its nucleotides replaced with any other nucleotide without turning into any of the other 63 codons. 2) upon substituting any 1 nucleotide in any of the 5-letter codons, the resulting 5-letter codon is still most similar to the originally intended codon in terms of Hamming Distance. 3) any 1 nucleotide substitution in any of the 5-letter codons results in a 5-letter codon that is not in Table 3.1. These observations are desirable and are simply consequences of theorem 3.1 for the case $t = 1$. Observation 3 however poses a challenge, which is addressed in the next section.

Table 3.1: Mapping to Alternative Codons

p-codons		DNA Codons	
$(0\ 0\ 0)$	$(0\ 0\ 0\ 0\ 0)$	AAA	AAAAA
$(0\ 0\ 1)$	$(0\ 0\ 1\ a\ a)$	AAT	AATGG
$(0\ 0\ a)$	$(0\ 0\ a\ a^2\ a^2)$	AAG	AAGCC
$(0\ 0\ a^2)$	$(0\ 0\ a^2\ 1\ 1)$	AAC	AACTT
$(0\ 1\ 0)$	$(0\ 1\ 0\ 1\ a)$	ATA	ATATG
$(0\ 1\ 1)$	$(0\ 1\ 1\ a^2\ 0)$	ATT	ATTCA
$(0\ 1\ a)$	$(0\ 1\ a\ a\ 1)$	ATG	ATGGT
$(0\ 1\ a^2)$	$(0\ 1\ a^2\ 0\ a^2)$	ATC	ATCAC
$(0\ a\ 0)$	$(0\ a\ 0\ a\ a^2)$	AGA	AGAGC
$(0\ a\ 1)$	$(0\ a\ 1\ 0\ 1)$	AGT	AGTAT
$(0\ a\ a)$	$(0\ a\ a\ 1\ 0)$	AGG	AGGTA
$(0\ a\ a^2)$	$(0\ a\ a^2\ a^2\ a)$	AGC	AGCCG
$(0\ a^2\ 0)$	$(0\ a^2\ 0\ a^2\ 1)$	ACA	ACACT
$(0\ a^2\ 1)$	$(0\ a^2\ 1\ 1\ a^2)$	ACT	ACTTC
$(0\ a^2\ a)$	$(0\ a^2\ a\ 0\ a)$	ACG	ACGAG
$(0\ a^2\ a^2)$	$(0\ a^2\ a^2\ a\ 0)$	ACC	ACCGA
$(1\ 0\ 0)$	$(1\ 0\ 0\ 1\ a^2)$	TAA	TAATC
$(1\ 0\ 1)$	$(1\ 0\ 1\ a^2\ 1)$	TAT	TATCT
$(1\ 0\ a)$	$(1\ 0\ a\ a\ 0)$	TAG	TAGGA
$(1\ 0\ a^2)$	$(1\ 0\ a^2\ 0\ a)$	TAC	TACAG
$(1\ 1\ 0)$	$(1\ 1\ 0\ 0\ 1)$	TTA	TTAAT
$(1\ 1\ 1)$	$(1\ 1\ 1\ a\ a^2)$	TTT	TTTGC

Table 3.1: Mapping to Alternative Codons

p-codons		DNA Codons	
$(1\ 1\ a)$	$(1\ 1\ a\ a^2\ a)$	TTG	TTGCG
$(1\ 1\ a^2)$	$(1\ 1\ a^2\ 1\ 0)$	TTC	TTCTA
$(1\ a\ 0)$	$(1\ a\ 0\ a^2\ 0)$	TGA	TGACA
$(1\ a\ 1)$	$(1\ a\ 1\ 1\ a)$	TGT	TGTTG
$(1\ a\ a)$	$(1\ a\ a\ 0\ a^2)$	TGG	TGGAC
$(1\ a\ a^2)$	$(1\ a\ a^2\ a\ 1)$	TGC	TGCGT
$(1\ a^2\ 0)$	$(1\ a^2\ 0\ a\ a)$	TCA	TCAGG
$(1\ a^2\ 1)$	$(1\ a^2\ 1\ 0\ 0)$	TCT	TCTAA
$(1\ a^2\ a)$	$(1\ a^2\ a\ 1\ 1)$	TCG	TCGTT
$(1\ a^2\ a^2)$	$(1\ a^2\ a^2\ a^2\ a^2)$	TCC	TCCCC
$(a\ 0\ 0)$	$(a\ 0\ 0\ a\ 1)$	GAA	GAAGT
$(a\ 0\ 1)$	$(a\ 0\ 1\ 0\ a^2)$	GAT	GATAC
$(a\ 0\ a)$	$(a\ 0\ a\ 1\ a)$	GAG	GAGTG
$(a\ 0\ a^2)$	$(a\ 0\ a^2\ a^2\ 0)$	GAC	GACCA
$(a\ 1\ 0)$	$(a\ 1\ 0\ a^2\ a^2)$	GTA	GTACC
$(a\ 1\ 1)$	$(a\ 1\ 1\ 1\ 1)$	GTT	GTTTT
$(a\ 1\ a)$	$(a\ 1\ a\ 0\ 0)$	GTG	GTGAA
$(a\ 1\ a^2)$	$(a\ 1\ a^2\ a\ a)$	GTC	GTCGG
$(a\ a\ 0)$	$(a\ a\ 0\ 0\ a)$	GGA	GGAAG
$(a\ a\ 1)$	$(a\ a\ 1\ a\ 0)$	GGT	GGTGA
$(a\ a\ a)$	$(a\ a\ a\ a^2\ 1)$	GGG	GGGCT
$(a\ a\ a^2)$	$(a\ a\ a^2\ 1\ a^2)$	GGC	GGCTC
$(a\ a^2\ 0)$	$(a\ a^2\ 0\ 1\ 0)$	GCA	GCATA

Table 3.1: Mapping to Alternative Codons

p-codons		DNA Codons	
$(a\ a^2\ 1)$	$(a\ a^2\ 1\ a^2\ a)$	GCT	GCTCG
$(a\ a^2\ a)$	$(a\ a^2\ a\ a\ a^2)$	GCG	GCGGC
$(a\ a^2\ a^2)$	$(a\ a^2\ a^2\ 0\ 1)$	GCC	GCCAT
$(a^2\ 0\ 0)$	$(a^2\ 0\ 0\ a^2\ a)$	CAA	CAACG
$(a^2\ 0\ 1)$	$(a^2\ 0\ 1\ 1\ 0)$	CAT	CATTA
$(a^2\ 0\ a)$	$(a^2\ 0\ a\ 0\ 1)$	CAG	CAGAT
$(a^2\ 0\ a^2)$	$(a^2\ 0\ a^2\ a\ a^2)$	CAC	CACGC
$(a^2\ 1\ 0)$	$(a^2\ 1\ 0\ a\ 0)$	CTA	CTAGA
$(a^2\ 1\ 1)$	$(a^2\ 1\ 1\ 0\ a)$	CTT	CTTAG
$(a^2\ 1\ a)$	$(a^2\ 1\ a\ 1\ a^2)$	CTG	CTGTC
$(a^2\ 1\ a^2)$	$(a^2\ 1\ a^2\ a^2\ 1)$	CTC	CTCCT
$(a^2\ a\ 0)$	$(a^2\ a\ 0\ 1\ 1)$	CGA	CGATT
$(a^2\ a\ 1)$	$(a^2\ a\ 1\ a^2\ a^2)$	CGT	CGTCC
$(a^2\ a\ a)$	$(a^2\ a\ a\ a\ a)$	CGG	CGGGG
$(a^2\ a\ a^2)$	$(a^2\ a\ a^2\ 0\ 0)$	CGC	CGCAA
$(a^2\ a^2\ 0)$	$(a^2\ a^2\ 0\ 0\ a^2)$	CCA	CCAAC
$(a^2\ a^2\ 1)$	$(a^2\ a^2\ 1\ a\ 1)$	CCT	CCTGT
$(a^2\ a^2\ a)$	$(a^2\ a^2\ a\ a^2\ 0)$	CCG	CCGCA
$(a^2\ a^2\ a^2)$	$(a^2\ a^2\ a^2\ 1\ a)$	CCC	CCCTG

3.4.1 Codon Degeneracy

We first demonstrate observation 3 made in the previous section. Let $\boldsymbol{x}$ = ATGGT, which is the 5-letter mutation-tolerant variant of the naturally-occuring codon ATG. Assume a mutation event that substitutes the third nucleotide of $\boldsymbol{x}$, guanine, with adenine. The resulting codon will be $\widetilde{\boldsymbol{x}}$=ATAGT. We can confirm that ATAGT is indeed not one of the 64 5-letter codons given in table 3.1. This is also true of any 5-letter codon which results from a 1 nucleotide substitution mutation of any of the other 63 codons.

This poses the following challenge. Assume we have a gene, call it K, constructed using the mutation-tolerant 5-letter codons given in table 3.1 and artificial tRNAs that recognize the 64 corresponding 5-letter RNA codons. Further assume that some 1 nucleotide substitution has occurred in a codon $\boldsymbol{x}$ of K to become $\widetilde{\boldsymbol{x}}$. We know that $\widetilde{\boldsymbol{x}}$ is not in table 3.1. Thus, the translation of gene K cannot progress beyond the RNA equivalent of codon $\widetilde{\boldsymbol{x}}$ since its pattern is not one of the 64 patterns recognized by the the artificial tRNAs. In fact, there will be $4^5 - 64$ RNA codons not recognized by the artificial tRNAs.

The solution is obvious. We ought to have artificial tRNAs that recognize RNA codons beyond those for the 64 5-letter codons given in 3.1. For example, instead of having an artificial tRNA decode only AUGGU as methionine (the amino acid coded for by the naturally-occurring AUG) we will have several artificial tRNAs that decode several 5-letter RNA codons, including the 'original' AUGGU, as methionine. Our approach to doing that is as follows.

As discussed in section, nucleotide substitutions can be described as the addition of a mutation vector to a p-codon. As such, being tolerant of single nucleotide substitutions means that the 5-letter codons in table 3.1 tolerate any mutation event whose

mutation vector is one of the rows [3] in Z below.

$$Z = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \alpha & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 \\ 0 & 0 & \alpha & 0 & 0 \\ 0 & 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & 0 & \alpha \\ \alpha^2 & 0 & 0 & 0 & 0 \\ 0 & \alpha^2 & 0 & 0 & 0 \\ 0 & 0 & \alpha^2 & 0 & 0 \\ 0 & 0 & 0 & \alpha^2 & 0 \\ 0 & 0 & 0 & 0 & \alpha^2 \end{bmatrix} \tag{3.7}$$

Using this fact, given any 5-letter codon $\boldsymbol{x}$ from table 3.1, our approach to deciding which 5-letter codons to decode in the same manner as $\boldsymbol{x}$ is as follows: add the p-codon of $\boldsymbol{x}$ to each row of Z, then have artificial tRNAs that decode all resulting codons as you would $\boldsymbol{x}$. Combining this approach with equation (3.6), we give equation (3.8) which takes as input the p-codon $(\lambda_1, \lambda_2, \lambda_3)$ of a naturally-occurring 3-letter codon and yields K, a 16×1 matrix whose rows are p-codons of 16 $5-$letter codons where the first codon is the mutation-tolerant codon to be used in place of the naturally-occurring 3-letter codon in the construction of a gene and all 16 5-letter codons are

[3]The zero mutation vector leads to a silent mutation but is added to facilitate use of Z in a latter equation.

to be decoded by artificial tRNAs during translation as the 3-letter codon would be in the natural genetic code.

$$\begin{aligned} K &= \varphi_{16}[\ (\lambda_1\lambda_2\lambda_3)\cdot G\ ] + Z \\ \text{where } \varphi_n[i] &= [\underbrace{i,\ i, \ldots,\ i}_{\text{n times}}]^T \end{aligned} \tag{3.8}$$

Take codon ATG as an example. In a DNA sense strand, it codes for methionine. Its p-codon is (01α). Using equation (3.8), $K = [\ (01\alpha\alpha1),\ (11\alpha\alpha1),\ (00\alpha\alpha1),$ $(01\alpha^2\alpha1), (01\alpha\alpha^21),\ (01\alpha\alpha0),\ (\alpha1\alpha\alpha1),\ (0\alpha^2\alpha\alpha1),\ (010\alpha1),\ (01\alpha01),\ (01\alpha\alpha\alpha^2),$ $(\alpha^21\alpha\alpha1),\ (0\alpha\alpha\alpha1),\ (011\alpha1),\ (01\alpha11),\ (01\alpha\alpha\alpha)\ ]^T$. The first p-codon in K is $(01\alpha\alpha1)$. In nucleotide terms, it is ATGGT. Thus, in 'converting' a naturally-occurring gene to its mutol variant, codon ATGGT should be used in place of ATG. Also, we ought to have artificial tRNAs that recognize and decode all 16 5-letter codons corresponding to the p-codons in K (given below) as ATG would naturally be decoded — methionine.

$\text{ATGGT}_{(01\alpha\alpha1)}$	$\text{ATGCT}_{(01\alpha\alpha^21)}$	$\text{ATAGT}_{(010\alpha1)}$	$\text{AGGGT}_{(0\alpha\alpha\alpha1)}$
$\text{TTGGT}_{(11\alpha\alpha1)}$	$\text{ATGGA}_{(01\alpha\alpha0)}$	$\text{ATGAT}_{(01\alpha01)}$	$\text{ATTGT}_{(011\alpha1)}$
$\text{AAGGT}_{(00\alpha\alpha1)}$	$\text{GTGGT}_{(\alpha1\alpha\alpha1)}$	$\text{ATGGC}_{(01\alpha\alpha\alpha^2)}$	$\text{ATGTT}_{(01\alpha11)}$
$\text{ATCGT}_{(01\alpha^2\alpha1)}$	$\text{ACGGT}_{(0\alpha^2\alpha\alpha1)}$	$\text{CTGGT}_{(\alpha^21\alpha\alpha1)}$	$\text{ATGGG}_{(01\alpha\alpha\alpha)}$

Just as we did above for codon ATG, equation (3.8) has been evaluated for all 64 3-letter codons and the results are given in Appendix A in table A.1.

3.4.2 Demonstration of Mutation-Tolerance Capability

Here, we simulate random substitution mutations in both a wild type GFP gene (NCBI Reference Sequence: NC_011521.1) and its mutol variant to demonstrate the superior mutation-tolerance of the mutol variant. By mutol variant[4], we mean the DNA sequence obtained by replacing the naturally occurring 3-letter codons with the 5-letter codons we constructed. In Table 3.3 below, the top sequence in each row is the wild type GFP DNA sequence. The middle sequence on each row is the DNA sequence of the mutol variant with each 5-letter codon right below its corresponding 3-letter codon. As previously mentioned, the underlying assumption is that there exists some synthetic tRNAs capable of 'reading' these 5-letter codons. The bottom sequence in each row also indicates the amino acid coded for by the codons. The first and last codon in each row are numbered for ease of reference.

We randomly selected 20 same-numbered codons from the wild type GFP DNA and its mutol variant. For each of these codons, we randomly substituted one of its nucleotides. We then noted the amino acids coded for by the post-mutation codons in both the wild type and the mutol variant. The results for each of 20 mutated codons in both sequences are given in table 3.4. The columns named "Mutation" have a syntax of the form X $\rightarrow$ Y, which simply means the original codon was X and became Y after the mutation. The same syntax is used in the columns named "Translation", where the amino acid on the left of the arrow is the originally intended amino acid and the amino acid on the right is the amino acid coded for by the mutated codon. From the table, we see that only codon number 25 and codon number 167 in the wild type sequence still coded for the original amino acid post-mutation. In contrast, all 20 codons that were mutated in the mutol variant still coded for the originally intended amino acid post-mutation.

[4]Mutol variant as in **mu**tation-**tol**erant variant

Table 3.3: GFP DNA Sequence and Its Mutol Variant

5′ ATG [1]	AGT	AAA	GGA	GAA	GAA	CTT [7]
5′ ATGCT	AGTAT	AAAAA	GGAAG	GAAGT	GAAGT	CTTAG
M	S	K	G	E	E	L
TTC [8]	ACT	GGA	GTT	GTC	CCA	ATT [14]
TTCTA	ACTTC	GGAAG	GTTTT	GTCGG	CCAAC	ATTCA
F	T	G	V	V	P	I
CTT [15]	GTT	GAA	TTA	GAT	GGT	GAT [21]
CTTAG	GTTTT	GAAGT	TTAAT	GATAC	GGTGA	GATAC
L	V	E	L	D	G	D
GTT [22]	AAT	GGG	CAC	AAA	TTT	TCT [28]
GTTTT	AATGG	GGGCT	CACGC	AAAAA	TTTGC	TCTAA
V	N	G	H	K	F	S
GTC [29]	AGT	GGA	GAG	GGT	GAA	GGT [35]
GTCGG	AGTAT	GGAAG	GAGTG	GGTGA	GAAGT	GGTGA
V	S	G	E	G	E	G
GAT [36]	GCA	ACA	TAC	GGA	AAA	CTT [42]
GATAC	GCATA	ACACT	TACAG	GGAAG	AAAAA	CTTAG
D	A	T	Y	G	K	L
ACC [43]	CTT	AAA	TTT	ATT	TGC	ACT [49]
ACCGA	CTTAG	AAAAA	TTTGC	ATTCA	TGCGT	ACTTC
T	L	K	F	I	C	T
ACT [50]	GGA	AAA	CTA	CCT	GTT	CCA [56]
ACTTC	GGAAG	AAAAA	CTAGA	CCTGT	GTTTT	CCAAC
T	G	K	L	P	V	P
TGG [57]	CCA	ACA	CTT	GTC	ACT	ACT [63]
TGGAC	CCAAC	ACACT	CTTAG	GTCGG	ACTTC	ACTTC
W	P	T	L	V	T	T
TTC [64]	GGT	TAT	GGT	GTT	CAA	TGC [70]
TTCTA	GGTGA	TATCT	GGTGA	GTTTT	CAACG	TGCGT
F	G	Y	G	V	Q	C
TTT [71]	GCG	AGA	TAC	CCA	GAT	CAT [77]
TTTGC	GCGGC	AGAGC	TACAG	CCAAC	GATAC	CATTA
F	A	R	Y	P	D	H
ATG [78]	AAA	CAG	CAT	GAC	TTT	TTC [84]
ATGGT	AAAAA	CAGAT	CATTA	GACCA	TTTGC	TTCTA
M	K	Q	H	D	F	F

Table 3.3: GFP DNA Sequence and Its Mutol Variant

AAG [85] AAGCC K	AGT AGTAT S	GCC GCCAT A	ATG ATGGT M	CCT CCTGT P	GAA GAAGT E	GGT [91] GGTGA G
TAT [92] TATCT Y	GTA GTACC V	CAG CAGAT Q	GAA GAAGT E	AGA AGAGC R	ACT ACTTC T	ATA [98] ATATG I
TTT [99] TTTGC F	TTC TTCTA F	AAA AAAAA K	GAT GATAC D	GAC GACCA D	GGG GGGCT G	AAC [105] AACTT N
TAC [106] TACAG Y	AAG AAGCC K	ACA ACACT T	CGT CGTCC R	GCT GCTCG A	GAA GAAGT E	GTC [112] GTCGG V
AAG [113] AAGCC K	TTT TTTGC F	GAA GAAGT E	GGT GGTGA G	GAT GATAC D	ACC ACCGA T	CTT [119] CTTAG L
GTT [120] GTTTT V	AAT AATGG N	AGA AGAGC R	ATC ATCAC I	GAG GAGTG E	TTA TTAAT L	AAA [126] AAAAA K
GGT [127] GGTGA G	ATT ATTCA I	GAT GATAC D	TTT TTTGC F	AAA AAAAA K	GAA GAAGT E	GAT [133] GATAC D
GGA [134] GGAAG G	AAC AACTT N	ATT ATTCA I	CTT CTTAG L	GGA GGAAG G	CAC CACGC H	AAA [140] AAAAA K
TTG [141] TTGCG L	GAA GAAGT E	TAC TACAG Y	AAC AACTT N	TAT TATCT Y	AAC AACTT N	TCA [147] TCAGG S
CAC [148] CACGC H	AAT AATGG N	GTA GTACC V	TAC TACAG Y	ATC ATCAC I	ATG ATGGT M	GCA [154] GCATA A
GAC [155] GACCA D	AAA AAAAA K	CAA CAACG Q	AAG AAGCC K	AAT AATGG N	GGA GGAAG G	ATC [161] ATCAC I
AAA [162] AAAAA K	GTT GTTTT V	AAC AACTT N	TTC TTCTA F	AAA AAAAA K	ATT ATTCA I	AGA [168] AGAGC R

Table 3.3: GFP DNA Sequence and Its Mutol Variant

CAC [169] CACGC H	AAC AACTT N	ATT ATTCA I	GAA GAAGT E	GAT GATAC D	GGA GGAAG G	AGC [175] AGCCG S
GTT [176] GTTTT V	CAA CAACG Q	CTA CTAGA L	GCA GCATA A	GAC GACCA D	CAT CATTA H	TAT [182] TATCT Y
CAA [183] CAACG Q	CAA CAACG Q	AAT AATGG N	ACT ACTTC T	CCA CCAAC P	ATT ATTCA I	GGC [189] GGCTC G
GAT [190] GATAC D	GGC GGCTC G	CCT CCTGT P	GTC GTCGG V	CTT CTTAG L	TTA TTAAT L	CCA [196] CCAAC P
GAC [197] GACCA D	AAC AACTT N	CAT CATTA H	TAC TACAG Y	CTG CTGTC L	TCC TCCCC S	ACA [203] ACACT T
CAA [204] CAACG Q	TCT TCTAA S	GCC GCCAT A	CTT CTTAG L	TCG TCGTT S	AAA AAAAA K	GAT [210] GATAC D
CCC [211] CCCTG P	AAC AACTT N	GAA GAAGT E	AAG AAGCC K	AGA AGAGC R	GAC GACCA D	CAC [217] CACGC H
ATG [218] ATGGT M	GTC GTCGG V	CTT CTTAG L	CTT CTTAG L	GAG GAGTG E	TTT TTTGC F	GTA [224] GTACC V
ACA [225] ACACT T	GCT GCTCG A	GCT GCTCG A	GGG GGGCT G	ATT ATTCA I	ACA ACACT T	CAT [231] CATTA H
GGC [232] GGCTC G	ATG ATGGT M	GAT GATAC D	GAA GAAGT E	CTA CTAGA L	TAC TACAG Y	AAA [238] AAAAA K
TAA [239] 3′ TAATC 3′ Stop						

	Wildtype GFP		**Mutol Variant**	
Codon No.	Mutation	Translation	Mutation	Translation
10	GGA → GCA	G → A	GGAAG → GCAAG	G → G
13	CCA → GCA	P → A	CCAAC → CCAAG	P → P
15	CTT → GTT	L → V	CTTAG → CCTAG	L → L
18	TTA → GTA	L → V	TTAAT → TTAGT	L → L
19	GAT → CAT	D → H	GATAC → AATAC	D → D
23	AAT → AAA	N → K	AATGG → AATGT	N → N
25	CAC → CAT	H → H	CACGC → CACAC	H → H
39	TAC → TTC	Y → F	TACAG → TCCAG	Y → Y
55	GTT → ATT	V → I	GTTTT → GCTTT	V → V
83	TTT → TAT	F → Y	TTTGC → TTGGC	F → F
94	CAG → CAC	Q → H	CAGAT → CAGCT	Q → Q
102	GAT → GTT	D → V	GATAC → GCTAC	D → D
108	ACA → ATA	T → I	ACACT → ACAGT	T → T
112	GTC → GCC	V → A	GTCGG → TTCGG	V → V
130	TTT → ATT	F → I	TTTGC → TATGC	F → F
138	GGA → TGA	G → Stop	GGAAG → GGGAG	G → G
150	GTA → GGA	V → G	GTACC → GTACA	V → V
167	ATT → ATA	I → I	ATTCA → AGTCA	I → I
211	CCC → ACC	P → T	CCCTG → CTCTG	P → P
233	ATG → ATA	M → I	ATGGT → ATTGT	M → M

Table 3.4: Mutation Simulation Results

REFERENCES

Axler, S., *Linear Algebra Done Right* (Springer, 2007).

Hamming, R. W., "Error detecting and error correcting codes", Bell System Technical Journal **29**, 2, 147160 (1950).

Lin, S. and D. J. Costello, *Error control coding: fundamentals and applications* (Prentice-Hall, 1983).

Ling, S., *Coding theory : a first course* (Cambridge University Press, Cambridge, UK New York, 2004).

APPENDIX A

3-LETTER CODONS TO 5-LETTER CODONS

Table A.1: Degeneracy Mapping

AAA (000)	$\mathrm{AAAAA}_{(00000)}$	$\mathrm{AAATA}_{(00010)}$	$\mathrm{AAGAA}_{(00a00)}$	$\mathrm{ACAAA}_{(0a^20000)}$
	$\mathrm{TAAAA}_{(10000)}$	$\mathrm{AAAAT}_{(00001)}$	$\mathrm{AAAGA}_{(000a0)}$	$\mathrm{AACAA}_{(00a^200)}$
	$\mathrm{ATAAA}_{(01000)}$	$\mathrm{GAAAA}_{(a0000)}$	$\mathrm{AAAAG}_{(0000a)}$	$\mathrm{AAACA}_{(000a^20)}$
	$\mathrm{AATAA}_{(00100)}$	$\mathrm{AGAAA}_{(0a000)}$	$\mathrm{CAAAA}_{(a^20000)}$	$\mathrm{AAAAC}_{(0000a^2)}$
AAT (001)	$\mathrm{AATGG}_{(001aa)}$	$\mathrm{AATCG}_{(001a^2a)}$	$\mathrm{AACGG}_{(00a^2aa)}$	$\mathrm{ACTGG}_{(0a^21aa)}$
	$\mathrm{TATGG}_{(101aa)}$	$\mathrm{AATGC}_{(001aa^2)}$	$\mathrm{AATAG}_{(0010a)}$	$\mathrm{AAGGG}_{(00aaa)}$
	$\mathrm{ATTGG}_{(011aa)}$	$\mathrm{GATGG}_{(a01aa)}$	$\mathrm{AATGA}_{(001a0)}$	$\mathrm{AATTG}_{(0011a)}$
	$\mathrm{AAAGG}_{(000aa)}$	$\mathrm{AGTGG}_{(0a1aa)}$	$\mathrm{CATGG}_{(a^201aa)}$	$\mathrm{AATGT}_{(001a1)}$
AAG (00a)	$\mathrm{AAGCC}_{(00aa^2a^2)}$	$\mathrm{AAGGC}_{(00aaa^2)}$	$\mathrm{AAACC}_{(000a^2a^2)}$	$\mathrm{ACGCC}_{(0a^2aa^2a^2)}$
	$\mathrm{TAGCC}_{(10aa^2a^2)}$	$\mathrm{AAGCG}_{(00aa^2a)}$	$\mathrm{AAGTC}_{(00a1a^2)}$	$\mathrm{AATCC}_{(001a^2a^2)}$
	$\mathrm{ATGCC}_{(01aa^2a^2)}$	$\mathrm{GAGCC}_{(a0aa^2a^2)}$	$\mathrm{AAGCT}_{(00aa^21)}$	$\mathrm{AAGAC}_{(00a0a^2)}$
	$\mathrm{AACCC}_{(00a^2a^2a^2)}$	$\mathrm{AGGCC}_{(0aaa^2a^2)}$	$\mathrm{CAGCC}_{(a^20aa^2a^2)}$	$\mathrm{AAGCA}_{(00aa^20)}$
AAC ($00a^2$)	$\mathrm{AACTT}_{(00a^211)}$	$\mathrm{AACAT}_{(00a^201)}$	$\mathrm{AATTT}_{(00111)}$	$\mathrm{ACCTT}_{(0a^2a^211)}$
	$\mathrm{TACTT}_{(10a^211)}$	$\mathrm{AACTA}_{(00a^210)}$	$\mathrm{AACCT}_{(00a^2a^21)}$	$\mathrm{AAATT}_{(00011)}$
	$\mathrm{ATCTT}_{(01a^211)}$	$\mathrm{GACTT}_{(a0a^211)}$	$\mathrm{AACTC}_{(00a^21a^2)}$	$\mathrm{AACGT}_{(00a^2a1)}$
	$\mathrm{AAGTT}_{(00a11)}$	$\mathrm{AGCTT}_{(0aa^211)}$	$\mathrm{CACTT}_{(a^20a^211)}$	$\mathrm{AACTG}_{(00a^21a)}$
ATA (010)	$\mathrm{ATATG}_{(0101a)}$	$\mathrm{ATAAG}_{(0100a)}$	$\mathrm{ATGTG}_{(01a1a)}$	$\mathrm{AGATG}_{(0a01a)}$
	$\mathrm{TTATG}_{(1101a)}$	$\mathrm{ATATC}_{(0101a^2)}$	$\mathrm{ATACG}_{(010a^2a)}$	$\mathrm{ATCTG}_{(01a^21a)}$
	$\mathrm{AAATG}_{(0001a)}$	$\mathrm{GTATG}_{(a101a)}$	$\mathrm{ATATA}_{(01010)}$	$\mathrm{ATAGG}_{(010aa)}$
	$\mathrm{ATTTG}_{(0111a)}$	$\mathrm{ACATG}_{(0a^201a)}$	$\mathrm{CTATG}_{(a^2101a)}$	$\mathrm{ATATT}_{(01011)}$
ATT (011)	$\mathrm{ATTCA}_{(011a^20)}$	$\mathrm{ATTGA}_{(011a0)}$	$\mathrm{ATCCA}_{(01a^2a^20)}$	$\mathrm{AGTCA}_{(0a1a^20)}$
	$\mathrm{TTTCA}_{(111a^20)}$	$\mathrm{ATTCT}_{(011a^21)}$	$\mathrm{ATTTA}_{(01110)}$	$\mathrm{ATGCA}_{(01aa^20)}$
	$\mathrm{AATCA}_{(001a^20)}$	$\mathrm{GTTCA}_{(a11a^20)}$	$\mathrm{ATTCG}_{(011a^2a)}$	$\mathrm{ATTAA}_{(01100)}$
	$\mathrm{ATACA}_{(010a^20)}$	$\mathrm{ACTCA}_{(0a^21a^20)}$	$\mathrm{CTTCA}_{(a^211a^20)}$	$\mathrm{ATTCC}_{(011a^2a^2)}$
ATG (01a)	$\mathrm{ATGGT}_{(01aa1)}$	$\mathrm{ATGCT}_{(01aa^21)}$	$\mathrm{ATAGT}_{(010a1)}$	$\mathrm{AGGGT}_{(0aaa1)}$
	$\mathrm{TTGGT}_{(11aa1)}$	$\mathrm{ATGGA}_{(01aa0)}$	$\mathrm{ATGAT}_{(01a01)}$	$\mathrm{ATTGT}_{(011a1)}$
	$\mathrm{AAGGT}_{(00aa1)}$	$\mathrm{GTGGT}_{(a1aa1)}$	$\mathrm{ATGGC}_{(01aaa^2)}$	$\mathrm{ATGTT}_{(01a11)}$
	$\mathrm{ATCGT}_{(01a^2a1)}$	$\mathrm{ACGGT}_{(0a^2aa1)}$	$\mathrm{CTGGT}_{(a^21aa1)}$	$\mathrm{ATGGG}_{(01aaa)}$

Table A.1: Degeneracy Mapping

ATC $(01a^2)$	$\mathrm{ATCAC}_{(01a^20a^2)}$	$\mathrm{ATCTC}_{(01a^21a^2)}$	$\mathrm{ATTAC}_{(0110a^2)}$	$\mathrm{AGCAC}_{(0aa^20a^2)}$
	$\mathrm{TTCAC}_{(11a^20a^2)}$	$\mathrm{ATCAG}_{(01a^20a)}$	$\mathrm{ATCGC}_{(01a^2aa^2)}$	$\mathrm{ATAAC}_{(0100a^2)}$
	$\mathrm{AACAC}_{(00a^20a^2)}$	$\mathrm{GTCAC}_{(a1a^20a^2)}$	$\mathrm{ATCAT}_{(01a^201)}$	$\mathrm{ATCCC}_{(01a^2a^2a^2)}$
	$\mathrm{ATGAC}_{(01a0a^2)}$	$\mathrm{ACCAC}_{(0a^2a^20a^2)}$	$\mathrm{CTCAC}_{(a^21a^20a^2)}$	$\mathrm{ATCAA}_{(01a^200)}$
AGA $(0a0)$	$\mathrm{AGAGC}_{(0a0aa^2)}$	$\mathrm{AGACC}_{(0a0a^2a^2)}$	$\mathrm{AGGGC}_{(0aaaa^2)}$	$\mathrm{ATAGC}_{(010aa^2)}$
	$\mathrm{TGAGC}_{(1a0aa^2)}$	$\mathrm{AGAGG}_{(0a0aa)}$	$\mathrm{AGAAC}_{(0a00a^2)}$	$\mathrm{AGCGC}_{(0aa^2aa^2)}$
	$\mathrm{ACAGC}_{(0a^20aa^2)}$	$\mathrm{GGAGC}_{(aa0aa^2)}$	$\mathrm{AGAGT}_{(0a0a1)}$	$\mathrm{AGATC}_{(0a01a^2)}$
	$\mathrm{AGTGC}_{(0a1aa^2)}$	$\mathrm{AAAGC}_{(000aa^2)}$	$\mathrm{CGAGC}_{(a^2a0aa^2)}$	$\mathrm{AGAGA}_{(0a0a0)}$
AGT $(0a1)$	$\mathrm{AGTAT}_{(0a101)}$	$\mathrm{AGTTT}_{(0a111)}$	$\mathrm{AGCAT}_{(0aa^201)}$	$\mathrm{ATTAT}_{(01101)}$
	$\mathrm{TGTAT}_{(1a101)}$	$\mathrm{AGTAA}_{(0a100)}$	$\mathrm{AGTGT}_{(0a1a1)}$	$\mathrm{AGGAT}_{(0aa01)}$
	$\mathrm{ACTAT}_{(0a^2101)}$	$\mathrm{GGTAT}_{(aa101)}$	$\mathrm{AGTAC}_{(0a10a^2)}$	$\mathrm{AGTCT}_{(0a1a^21)}$
	$\mathrm{AGAAT}_{(0a001)}$	$\mathrm{AATAT}_{(00101)}$	$\mathrm{CGTAT}_{(a^2a101)}$	$\mathrm{AGTAG}_{(0a10a)}$
AGG $(0aa)$	$\mathrm{AGGTA}_{(0aa10)}$	$\mathrm{AGGAA}_{(0aa00)}$	$\mathrm{AGATA}_{(0a010)}$	$\mathrm{ATGTA}_{(01a10)}$
	$\mathrm{TGGTA}_{(1aa10)}$	$\mathrm{AGGTT}_{(0aa11)}$	$\mathrm{AGGCA}_{(0aaa^20)}$	$\mathrm{AGTTA}_{(0a110)}$
	$\mathrm{ACGTA}_{(0a^2a10)}$	$\mathrm{GGGTA}_{(aaa10)}$	$\mathrm{AGGTG}_{(0aa1a)}$	$\mathrm{AGGGA}_{(0aaa0)}$
	$\mathrm{AGCTA}_{(0aa^210)}$	$\mathrm{AAGTA}_{(00a10)}$	$\mathrm{CGGTA}_{(a^2aa10)}$	$\mathrm{AGGTC}_{(0aa1a^2)}$
AGC $(0aa^2)$	$\mathrm{AGCCG}_{(0aa^2a^2a)}$	$\mathrm{AGCGG}_{(0aa^2aa)}$	$\mathrm{AGTCG}_{(0a1a^2a)}$	$\mathrm{ATCCG}_{(01a^2a^2a)}$
	$\mathrm{TGCCG}_{(1aa^2a^2a)}$	$\mathrm{AGCCC}_{(0aa^2a^2a^2)}$	$\mathrm{AGCTG}_{(0aa^21a)}$	$\mathrm{AGACG}_{(0a0a^2a)}$
	$\mathrm{ACCCG}_{(0a^2a^2a^2a)}$	$\mathrm{GGCCG}_{(aaa^2a^2a)}$	$\mathrm{AGCCA}_{(0aa^2a^20)}$	$\mathrm{AGCAG}_{(0aa^20a)}$
	$\mathrm{AGGCG}_{(0aaa^2a)}$	$\mathrm{AACCG}_{(00a^2a^2a)}$	$\mathrm{CGCCG}_{(a^2aa^2a^2a)}$	$\mathrm{AGCCT}_{(0aa^2a^21)}$
ACA $(0a^20)$	$\mathrm{ACACT}_{(0a^20a^21)}$	$\mathrm{ACAGT}_{(0a^20a1)}$	$\mathrm{ACGCT}_{(0a^2aa^21)}$	$\mathrm{AAACT}_{(000a^21)}$
	$\mathrm{TCACT}_{(1a^20a^21)}$	$\mathrm{ACACA}_{(0a^20a^20)}$	$\mathrm{ACATT}_{(0a^2011)}$	$\mathrm{ACCCT}_{(0a^2a^2a^21)}$
	$\mathrm{AGACT}_{(0a0a^21)}$	$\mathrm{GCACT}_{(aa^20a^21)}$	$\mathrm{ACACC}_{(0a^20a^2a^2)}$	$\mathrm{ACAAT}_{(0a^2001)}$
	$\mathrm{ACTCT}_{(0a^21a^21)}$	$\mathrm{ATACT}_{(010a^21)}$	$\mathrm{CCACT}_{(a^2a^20a^21)}$	$\mathrm{ACACG}_{(0a^20a^2a)}$
ACT $(0a^21)$	$\mathrm{ACTTC}_{(0a^211a^2)}$	$\mathrm{ACTAC}_{(0a^210a^2)}$	$\mathrm{ACCTC}_{(0a^2a^21a^2)}$	$\mathrm{AATTC}_{(0011a^2)}$
	$\mathrm{TCTTC}_{(1a^211a^2)}$	$\mathrm{ACTTG}_{(0a^211a)}$	$\mathrm{ACTCC}_{(0a^21a^2a^2)}$	$\mathrm{ACGTC}_{(0a^2a1a^2)}$
	$\mathrm{AGTTC}_{(0a11a^2)}$	$\mathrm{GCTTC}_{(aa^211a^2)}$	$\mathrm{ACTTT}_{(0a^2111)}$	$\mathrm{ACTGC}_{(0a^21aa^2)}$
	$\mathrm{ACATC}_{(0a^201a^2)}$	$\mathrm{ATTTC}_{(0111a^2)}$	$\mathrm{CCTTC}_{(a^2a^211a^2)}$	$\mathrm{ACTTA}_{(0a^2110)}$

Table A.1: Degeneracy Mapping

ACG $(0a^2a)$	$\mathrm{ACGAG}_{(0a^2a0a)}$	$\mathrm{ACGTG}_{(0a^2a1a)}$	$\mathrm{ACAAG}_{(0a^200a)}$	$\mathrm{AAGAG}_{(00a0a)}$
	$\mathrm{TCGAG}_{(1a^2a0a)}$	$\mathrm{ACGAC}_{(0a^2a0a^2)}$	$\mathrm{ACGGG}_{(0a^2aaa)}$	$\mathrm{ACTAG}_{(0a^210a)}$
	$\mathrm{AGGAG}_{(0aa0a)}$	$\mathrm{GCGAG}_{(aa^2a0a)}$	$\mathrm{ACGAA}_{(0a^2a00)}$	$\mathrm{ACGCG}_{(0a^2aa^2a)}$
	$\mathrm{ACCAG}_{(0a^2a^20a)}$	$\mathrm{ATGAG}_{(01a0a)}$	$\mathrm{CCGAG}_{(a^2a^2a0a)}$	$\mathrm{ACGAT}_{(0a^2a01)}$
ACC $(0a^2a^2)$	$\mathrm{ACCGA}_{(0a^2a^2a0)}$	$\mathrm{ACCCA}_{(0a^2a^2a^20)}$	$\mathrm{ACTGA}_{(0a^21a0)}$	$\mathrm{AACGA}_{(00a^2a0)}$
	$\mathrm{TCCGA}_{(1a^2a^2a0)}$	$\mathrm{ACCGT}_{(0a^2a^2a1)}$	$\mathrm{ACCAA}_{(0a^2a^200)}$	$\mathrm{ACAGA}_{(0a^20a0)}$
	$\mathrm{AGCGA}_{(0aa^2a0)}$	$\mathrm{GCCGA}_{(aa^2a^2a0)}$	$\mathrm{ACCGG}_{(0a^2a^2aa)}$	$\mathrm{ACCTA}_{(0a^2a^210)}$
	$\mathrm{ACGGA}_{(0a^2aa0)}$	$\mathrm{ATCGA}_{(01a^2a0)}$	$\mathrm{CCCGA}_{(a^2a^2a^2a0)}$	$\mathrm{ACCGC}_{(0a^2a^2aa^2)}$
TAA (100)	$\mathrm{TAATC}_{(1001a^2)}$	$\mathrm{TAAAC}_{(1000a^2)}$	$\mathrm{TAGTC}_{(10a1a^2)}$	$\mathrm{TCATC}_{(1a^201a^2)}$
	$\mathrm{AAATC}_{(0001a^2)}$	$\mathrm{TAATG}_{(1001a)}$	$\mathrm{TAACC}_{(100a^2a^2)}$	$\mathrm{TACTC}_{(10a^21a^2)}$
	$\mathrm{TTATC}_{(1101a^2)}$	$\mathrm{CAATC}_{(a^2001a^2)}$	$\mathrm{TAATT}_{(10011)}$	$\mathrm{TAAGC}_{(100aa^2)}$
	$\mathrm{TATTC}_{(1011a^2)}$	$\mathrm{TGATC}_{(1a01a^2)}$	$\mathrm{GAATC}_{(a001a^2)}$	$\mathrm{TAATA}_{(10010)}$
TAT (101)	$\mathrm{TATCT}_{(101a^21)}$	$\mathrm{TATGT}_{(101a1)}$	$\mathrm{TACCT}_{(10a^2a^21)}$	$\mathrm{TCTCT}_{(1a^21a^21)}$
	$\mathrm{AATCT}_{(001a^21)}$	$\mathrm{TATCA}_{(101a^20)}$	$\mathrm{TATTT}_{(10111)}$	$\mathrm{TAGCT}_{(10aa^21)}$
	$\mathrm{TTTCT}_{(111a^21)}$	$\mathrm{CATCT}_{(a^201a^21)}$	$\mathrm{TATCC}_{(101a^2a^2)}$	$\mathrm{TATAT}_{(10101)}$
	$\mathrm{TAACT}_{(100a^21)}$	$\mathrm{TGTCT}_{(1a1a^21)}$	$\mathrm{GATCT}_{(a01a^21)}$	$\mathrm{TATCG}_{(101a^2a)}$
TAG $(10a)$	$\mathrm{TAGGA}_{(10aa0)}$	$\mathrm{TAGCA}_{(10aa^20)}$	$\mathrm{TAAGA}_{(100a0)}$	$\mathrm{TCGGA}_{(1a^2aa0)}$
	$\mathrm{AAGGA}_{(00aa0)}$	$\mathrm{TAGGT}_{(10aa1)}$	$\mathrm{TAGAA}_{(10a00)}$	$\mathrm{TATGA}_{(101a0)}$
	$\mathrm{TTGGA}_{(11aa0)}$	$\mathrm{CAGGA}_{(a^20aa0)}$	$\mathrm{TAGGG}_{(10aaa)}$	$\mathrm{TAGTA}_{(10a10)}$
	$\mathrm{TACGA}_{(10a^2a0)}$	$\mathrm{TGGGA}_{(1aaa0)}$	$\mathrm{GAGGA}_{(a0aa0)}$	$\mathrm{TAGGC}_{(10aaa^2)}$
TAC $(10a^2)$	$\mathrm{TACAG}_{(10a^20a)}$	$\mathrm{TACTG}_{(10a^21a)}$	$\mathrm{TATAG}_{(1010a)}$	$\mathrm{TCCAG}_{(1a^2a^20a)}$
	$\mathrm{AACAG}_{(00a^20a)}$	$\mathrm{TACAC}_{(10a^20a^2)}$	$\mathrm{TACGG}_{(10a^2aa)}$	$\mathrm{TAAAG}_{(1000a)}$
	$\mathrm{TTCAG}_{(11a^20a)}$	$\mathrm{CACAG}_{(a^20a^20a)}$	$\mathrm{TACAA}_{(10a^200)}$	$\mathrm{TACCG}_{(10a^2a^2a)}$
	$\mathrm{TAGAG}_{(10a0a)}$	$\mathrm{TGCAG}_{(1aa^20a)}$	$\mathrm{GACAG}_{(a0a^20a)}$	$\mathrm{TACAT}_{(10a^201)}$
TTA (110)	$\mathrm{TTAAT}_{(11001)}$	$\mathrm{TTATT}_{(11011)}$	$\mathrm{TTGAT}_{(11a01)}$	$\mathrm{TGAAT}_{(1a001)}$
	$\mathrm{ATAAT}_{(01001)}$	$\mathrm{TTAAA}_{(11000)}$	$\mathrm{TTAGT}_{(110a1)}$	$\mathrm{TTCAT}_{(11a^201)}$
	$\mathrm{TAAAT}_{(10001)}$	$\mathrm{CTAAT}_{(a^21001)}$	$\mathrm{TTAAC}_{(1100a^2)}$	$\mathrm{TTACT}_{(110a^21)}$
	$\mathrm{TTTAT}_{(11101)}$	$\mathrm{TCAAT}_{(1a^2001)}$	$\mathrm{GTAAT}_{(a1001)}$	$\mathrm{TTAAG}_{(1100a)}$

Table A.1: Degeneracy Mapping

TTT (111)	$\mathrm{TTTGC}_{(111aa^2)}$	$\mathrm{TTTCC}_{(111a^2a^2)}$	$\mathrm{TTCGC}_{(11a^2aa^2)}$	$\mathrm{TGTGC}_{(1a1aa^2)}$
	$\mathrm{ATTGC}_{(011aa^2)}$	$\mathrm{TTTGG}_{(111aa)}$	$\mathrm{TTTAC}_{(1110a^2)}$	$\mathrm{TTGGC}_{(11aaa^2)}$
	$\mathrm{TATGC}_{(101aa^2)}$	$\mathrm{CTTGC}_{(a^211aa^2)}$	$\mathrm{TTTGT}_{(111a1)}$	$\mathrm{TTTTC}_{(1111a^2)}$
	$\mathrm{TTAGC}_{(110aa^2)}$	$\mathrm{TCTGC}_{(1a^21aa^2)}$	$\mathrm{GTTGC}_{(a11aa^2)}$	$\mathrm{TTTGA}_{(111a0)}$
TTG $(11a)$	$\mathrm{TTGCG}_{(11aa^2a)}$	$\mathrm{TTGGG}_{(11aaa)}$	$\mathrm{TTACG}_{(110a^2a)}$	$\mathrm{TGGCG}_{(1aaa^2a)}$
	$\mathrm{ATGCG}_{(01aa^2a)}$	$\mathrm{TTGCC}_{(11aa^2a^2)}$	$\mathrm{TTGTG}_{(11a1a)}$	$\mathrm{TTTCG}_{(111a^2a)}$
	$\mathrm{TAGCG}_{(10aa^2a)}$	$\mathrm{CTGCG}_{(a^21aa^2a)}$	$\mathrm{TTGCA}_{(11aa^20)}$	$\mathrm{TTGAG}_{(11a0a)}$
	$\mathrm{TTCCG}_{(11a^2a^2a)}$	$\mathrm{TCGCG}_{(1a^2aa^2a)}$	$\mathrm{GTGCG}_{(a1aa^2a)}$	$\mathrm{TTGCT}_{(11aa^21)}$
TTC $(11a^2)$	$\mathrm{TTCTA}_{(11a^210)}$	$\mathrm{TTCAA}_{(11a^200)}$	$\mathrm{TTTTA}_{(11110)}$	$\mathrm{TGCTA}_{(1aa^210)}$
	$\mathrm{ATCTA}_{(01a^210)}$	$\mathrm{TTCTT}_{(11a^211)}$	$\mathrm{TTCCA}_{(11a^2a^20)}$	$\mathrm{TTATA}_{(11010)}$
	$\mathrm{TACTA}_{(10a^210)}$	$\mathrm{CTCTA}_{(a^21a^210)}$	$\mathrm{TTCTG}_{(11a^21a)}$	$\mathrm{TTCGA}_{(11a^2a0)}$
	$\mathrm{TTGTA}_{(11a10)}$	$\mathrm{TCCTA}_{(1a^2a^210)}$	$\mathrm{GTCTA}_{(a1a^210)}$	$\mathrm{TTCTC}_{(11a^21a^2)}$
TGA $(1a0)$	$\mathrm{TGACA}_{(1a0a^20)}$	$\mathrm{TGAGA}_{(1a0a0)}$	$\mathrm{TGGCA}_{(1aaa^20)}$	$\mathrm{TTACA}_{(110a^20)}$
	$\mathrm{AGACA}_{(0a0a^20)}$	$\mathrm{TGACT}_{(1a0a^21)}$	$\mathrm{TGATA}_{(1a010)}$	$\mathrm{TGCCA}_{(1aa^2a^20)}$
	$\mathrm{TCACA}_{(1a^20a^20)}$	$\mathrm{CGACA}_{(a^2a0a^20)}$	$\mathrm{TGACG}_{(1a0a^2a)}$	$\mathrm{TGAAA}_{(1a000)}$
	$\mathrm{TGTCA}_{(1a1a^20)}$	$\mathrm{TAACA}_{(100a^20)}$	$\mathrm{GGACA}_{(aa0a^20)}$	$\mathrm{TGACC}_{(1a0a^2a^2)}$
TGT $(1a1)$	$\mathrm{TGTTG}_{(1a11a)}$	$\mathrm{TGTAG}_{(1a10a)}$	$\mathrm{TGCTG}_{(1aa^21a)}$	$\mathrm{TTTTG}_{(1111a)}$
	$\mathrm{AGTTG}_{(0a11a)}$	$\mathrm{TGTTC}_{(1a11a^2)}$	$\mathrm{TGTCG}_{(1a1a^2a)}$	$\mathrm{TGGTG}_{(1aa1a)}$
	$\mathrm{TCTTG}_{(1a^211a)}$	$\mathrm{CGTTG}_{(a^2a11a)}$	$\mathrm{TGTTA}_{(1a110)}$	$\mathrm{TGTGG}_{(1a1aa)}$
	$\mathrm{TGATG}_{(1a01a)}$	$\mathrm{TATTG}_{(1011a)}$	$\mathrm{GGTTG}_{(aa11a)}$	$\mathrm{TGTTT}_{(1a111)}$
TGG $(1aa)$	$\mathrm{TGGAC}_{(1aa0a^2)}$	$\mathrm{TGGTC}_{(1aa1a^2)}$	$\mathrm{TGAAC}_{(1a00a^2)}$	$\mathrm{TTGAC}_{(11a0a^2)}$
	$\mathrm{AGGAC}_{(0aa0a^2)}$	$\mathrm{TGGAG}_{(1aa0a)}$	$\mathrm{TGGGC}_{(1aaaa^2)}$	$\mathrm{TGTAC}_{(1a10a^2)}$
	$\mathrm{TCGAC}_{(1a^2a0a^2)}$	$\mathrm{CGGAC}_{(a^2aa0a^2)}$	$\mathrm{TGGAT}_{(1aa01)}$	$\mathrm{TGGCC}_{(1aaa^2a^2)}$
	$\mathrm{TGCAC}_{(1aa^20a^2)}$	$\mathrm{TAGAC}_{(10a0a^2)}$	$\mathrm{GGGAC}_{(aaa0a^2)}$	$\mathrm{TGGAA}_{(1aa00)}$
TGC $(1aa^2)$	$\mathrm{TGCGT}_{(1aa^2a1)}$	$\mathrm{TGCCT}_{(1aa^2a^21)}$	$\mathrm{TGTGT}_{(1a1a1)}$	$\mathrm{TTCGT}_{(11a^2a1)}$
	$\mathrm{AGCGT}_{(0aa^2a1)}$	$\mathrm{TGCGA}_{(1aa^2a0)}$	$\mathrm{TGCAT}_{(1aa^201)}$	$\mathrm{TGAGT}_{(1a0a1)}$
	$\mathrm{TCCGT}_{(1a^2a^2a1)}$	$\mathrm{CGCGT}_{(a^2aa^2a1)}$	$\mathrm{TGCGC}_{(1aa^2aa^2)}$	$\mathrm{TGCTT}_{(1aa^211)}$
	$\mathrm{TGGGT}_{(1aaa1)}$	$\mathrm{TACGT}_{(10a^2a1)}$	$\mathrm{GGCGT}_{(aaa^2a1)}$	$\mathrm{TGCGG}_{(1aa^2aa)}$

Table A.1: Degeneracy Mapping

TCA $(1a^20)$	$\mathrm{TCAGG}_{(1a^20aa)}$	$\mathrm{TCACG}_{(1a^20a^2a)}$	$\mathrm{TCGGG}_{(1a^2aaa)}$	$\mathrm{TAAGG}_{(100aa)}$
	$\mathrm{ACAGG}_{(0a^20aa)}$	$\mathrm{TCAGC}_{(1a^20aa^2)}$	$\mathrm{TCAAG}_{(1a^200a)}$	$\mathrm{TCCGG}_{(1a^2a^2aa)}$
	$\mathrm{TGAGG}_{(1a0aa)}$	$\mathrm{CCAGG}_{(a^2a^20aa)}$	$\mathrm{TCAGA}_{(1a^20a0)}$	$\mathrm{TCATG}_{(1a^201a)}$
	$\mathrm{TCTGG}_{(1a^21aa)}$	$\mathrm{TTAGG}_{(110aa)}$	$\mathrm{GCAGG}_{(aa^20aa)}$	$\mathrm{TCAGT}_{(1a^20a1)}$
TCT $(1a^21)$	$\mathrm{TCTAA}_{(1a^2100)}$	$\mathrm{TCTTA}_{(1a^2110)}$	$\mathrm{TCCAA}_{(1a^2a^200)}$	$\mathrm{TATAA}_{(10100)}$
	$\mathrm{ACTAA}_{(0a^2100)}$	$\mathrm{TCTAT}_{(1a^2101)}$	$\mathrm{TCTGA}_{(1a^21a0)}$	$\mathrm{TCGAA}_{(1a^2a00)}$
	$\mathrm{TGTAA}_{(1a100)}$	$\mathrm{CCTAA}_{(a^2a^2100)}$	$\mathrm{TCTAG}_{(1a^210a)}$	$\mathrm{TCTCA}_{(1a^21a^20)}$
	$\mathrm{TCAAA}_{(1a^2000)}$	$\mathrm{TTTAA}_{(11100)}$	$\mathrm{GCTAA}_{(aa^2100)}$	$\mathrm{TCTAC}_{(1a^210a^2)}$
TCG $(1a^2a)$	$\mathrm{TCGTT}_{(1a^2a11)}$	$\mathrm{TCGAT}_{(1a^2a01)}$	$\mathrm{TCATT}_{(1a^2011)}$	$\mathrm{TAGTT}_{(10a11)}$
	$\mathrm{ACGTT}_{(0a^2a11)}$	$\mathrm{TCGTA}_{(1a^2a10)}$	$\mathrm{TCGCT}_{(1a^2aa^21)}$	$\mathrm{TCTTT}_{(1a^2111)}$
	$\mathrm{TGGTT}_{(1aa11)}$	$\mathrm{CCGTT}_{(a^2a^2a11)}$	$\mathrm{TCGTC}_{(1a^2a1a^2)}$	$\mathrm{TCGGT}_{(1a^2aa1)}$
	$\mathrm{TCCTT}_{(1a^2a^211)}$	$\mathrm{TTGTT}_{(11a11)}$	$\mathrm{GCGTT}_{(aa^2a11)}$	$\mathrm{TCGTG}_{(1a^2a1a)}$
TCC $(1a^2a^2)$	$\mathrm{TCCCC}_{(1a^2a^2a^2a^2)}$	$\mathrm{TCCGC}_{(1a^2a^2aa^2)}$	$\mathrm{TCTCC}_{(1a^21a^2a^2)}$	$\mathrm{TACCC}_{(10a^2a^2a^2)}$
	$\mathrm{ACCCC}_{(0a^2a^2a^2a^2)}$	$\mathrm{TCCCG}_{(1a^2a^2a^2a)}$	$\mathrm{TCCTC}_{(1a^2a^21a^2)}$	$\mathrm{TCACC}_{(1a^20a^2a^2)}$
	$\mathrm{TGCCC}_{(1aa^2a^2a^2)}$	$\mathrm{CCCCC}_{(a^2a^2a^2a^2a^2)}$	$\mathrm{TCCCT}_{(1a^2a^2a^21)}$	$\mathrm{TCCAC}_{(1a^2a^20a^2)}$
	$\mathrm{TCGCC}_{(1a^2aa^2a^2)}$	$\mathrm{TTCCC}_{(11a^2a^2a^2)}$	$\mathrm{GCCCC}_{(aa^2a^2a^2a^2)}$	$\mathrm{TCCCA}_{(1a^2a^2a^20)}$
GAA $(a00)$	$\mathrm{GAAGT}_{(a00a1)}$	$\mathrm{GAACT}_{(a00a^21)}$	$\mathrm{GAGGT}_{(a0aa1)}$	$\mathrm{GCAGT}_{(aa^20a1)}$
	$\mathrm{CAAGT}_{(a^200a1)}$	$\mathrm{GAAGA}_{(a00a0)}$	$\mathrm{GAAAT}_{(a0001)}$	$\mathrm{GACGT}_{(a0a^2a1)}$
	$\mathrm{GTAGT}_{(a10a1)}$	$\mathrm{AAAGT}_{(000a1)}$	$\mathrm{GAAGC}_{(a00aa^2)}$	$\mathrm{GAATT}_{(a0011)}$
	$\mathrm{GATGT}_{(a01a1)}$	$\mathrm{GGAGT}_{(aa0a1)}$	$\mathrm{TAAGT}_{(100a1)}$	$\mathrm{GAAGG}_{(a00aa)}$
GAT $(a01)$	$\mathrm{GATAC}_{(a010a^2)}$	$\mathrm{GATTC}_{(a011a^2)}$	$\mathrm{GACAC}_{(a0a^20a^2)}$	$\mathrm{GCTAC}_{(aa^210a^2)}$
	$\mathrm{CATAC}_{(a^2010a^2)}$	$\mathrm{GATAG}_{(a010a)}$	$\mathrm{GATGC}_{(a01aa^2)}$	$\mathrm{GAGAC}_{(a0a0a^2)}$
	$\mathrm{GTTAC}_{(a110a^2)}$	$\mathrm{AATAC}_{(0010a^2)}$	$\mathrm{GATAT}_{(a0101)}$	$\mathrm{GATCC}_{(a01a^2a^2)}$
	$\mathrm{GAAAC}_{(a000a^2)}$	$\mathrm{GGTAC}_{(aa10a^2)}$	$\mathrm{TATAC}_{(1010a^2)}$	$\mathrm{GATAA}_{(a0100)}$
GAG $(a0a)$	$\mathrm{GAGTG}_{(a0a1a)}$	$\mathrm{GAGAG}_{(a0a0a)}$	$\mathrm{GAATG}_{(a001a)}$	$\mathrm{GCGTG}_{(aa^2a1a)}$
	$\mathrm{CAGTG}_{(a^20a1a)}$	$\mathrm{GAGTC}_{(a0a1a^2)}$	$\mathrm{GAGCG}_{(a0aa^2a)}$	$\mathrm{GATTG}_{(a011a)}$
	$\mathrm{GTGTG}_{(a1a1a)}$	$\mathrm{AAGTG}_{(00a1a)}$	$\mathrm{GAGTA}_{(a0a10)}$	$\mathrm{GAGGG}_{(a0aaa)}$
	$\mathrm{GACTG}_{(a0a^21a)}$	$\mathrm{GGGTG}_{(aaa1a)}$	$\mathrm{TAGTG}_{(10a1a)}$	$\mathrm{GAGTT}_{(a0a11)}$

Table A.1: Degeneracy Mapping

GAC $(a0a^2)$	$\mathrm{GACCA}_{(a0a^2a^20)}$	$\mathrm{GACGA}_{(a0a^2a0)}$	$\mathrm{GATCA}_{(a01a^20)}$	$\mathrm{GCCCA}_{(aa^2a^2a^20)}$
	$\mathrm{CACCA}_{(a^20a^2a^20)}$	$\mathrm{GACCT}_{(a0a^2a^21)}$	$\mathrm{GACTA}_{(a0a^210)}$	$\mathrm{GAACA}_{(a00a^20)}$
	$\mathrm{GTCCA}_{(a1a^2a^20)}$	$\mathrm{AACCA}_{(00a^2a^20)}$	$\mathrm{GACCG}_{(a0a^2a^2a)}$	$\mathrm{GACAA}_{(a0a^200)}$
	$\mathrm{GAGCA}_{(a0aa^20)}$	$\mathrm{GGCCA}_{(aaa^2a^20)}$	$\mathrm{TACCA}_{(10a^2a^20)}$	$\mathrm{GACCC}_{(a0a^2a^2a^2)}$
GTA $(a10)$	$\mathrm{GTACC}_{(a10a^2a^2)}$	$\mathrm{GTAGC}_{(a10aa^2)}$	$\mathrm{GTGCC}_{(a1aa^2a^2)}$	$\mathrm{GGACC}_{(aa0a^2a^2)}$
	$\mathrm{CTACC}_{(a^210a^2a^2)}$	$\mathrm{GTACG}_{(a10a^2a)}$	$\mathrm{GTATC}_{(a101a^2)}$	$\mathrm{GTCCC}_{(a1a^2a^2a^2)}$
	$\mathrm{GAACC}_{(a00a^2a^2)}$	$\mathrm{ATACC}_{(010a^2a^2)}$	$\mathrm{GTACT}_{(a10a^21)}$	$\mathrm{GTAAC}_{(a100a^2)}$
	$\mathrm{GTTCC}_{(a11a^2a^2)}$	$\mathrm{GCACC}_{(aa^20a^2a^2)}$	$\mathrm{TTACC}_{(110a^2a^2)}$	$\mathrm{GTACA}_{(a10a^20)}$
GTT $(a11)$	$\mathrm{GTTTT}_{(a1111)}$	$\mathrm{GTTAT}_{(a1101)}$	$\mathrm{GTCTT}_{(a1a^211)}$	$\mathrm{GGTTT}_{(aa111)}$
	$\mathrm{CTTTT}_{(a^21111)}$	$\mathrm{GTTTA}_{(a1110)}$	$\mathrm{GTTCT}_{(a11a^21)}$	$\mathrm{GTGTT}_{(a1a11)}$
	$\mathrm{GATTT}_{(a0111)}$	$\mathrm{ATTTT}_{(01111)}$	$\mathrm{GTTTC}_{(a111a^2)}$	$\mathrm{GTTGT}_{(a11a1)}$
	$\mathrm{GTATT}_{(a1011)}$	$\mathrm{GCTTT}_{(aa^2111)}$	$\mathrm{TTTTT}_{(11111)}$	$\mathrm{GTTTG}_{(a111a)}$
GTG $(a1a)$	$\mathrm{GTGAA}_{(a1a00)}$	$\mathrm{GTGTA}_{(a1a10)}$	$\mathrm{GTAAA}_{(a1000)}$	$\mathrm{GGGAA}_{(aaa00)}$
	$\mathrm{CTGAA}_{(a^21a00)}$	$\mathrm{GTGAT}_{(a1a01)}$	$\mathrm{GTGGA}_{(a1aa0)}$	$\mathrm{GTTAA}_{(a1100)}$
	$\mathrm{GAGAA}_{(a0a00)}$	$\mathrm{ATGAA}_{(01a00)}$	$\mathrm{GTGAG}_{(a1a0a)}$	$\mathrm{GTGCA}_{(a1aa^20)}$
	$\mathrm{GTCAA}_{(a1a^200)}$	$\mathrm{GCGAA}_{(aa^2a00)}$	$\mathrm{TTGAA}_{(11a00)}$	$\mathrm{GTGAC}_{(a1a0a^2)}$
GTC $(a1a^2)$	$\mathrm{GTCGG}_{(a1a^2aa)}$	$\mathrm{GTCCG}_{(a1a^2a^2a)}$	$\mathrm{GTTGG}_{(a11aa)}$	$\mathrm{GGCGG}_{(aaa^2aa)}$
	$\mathrm{CTCGG}_{(a^21a^2aa)}$	$\mathrm{GTCGC}_{(a1a^2aa^2)}$	$\mathrm{GTCAG}_{(a1a^20a)}$	$\mathrm{GTAGG}_{(a10aa)}$
	$\mathrm{GACGG}_{(a0a^2aa)}$	$\mathrm{ATCGG}_{(01a^2aa)}$	$\mathrm{GTCGA}_{(a1a^2a0)}$	$\mathrm{GTCTG}_{(a1a^21a)}$
	$\mathrm{GTGGG}_{(a1aaa)}$	$\mathrm{GCCGG}_{(aa^2a^2aa)}$	$\mathrm{TTCGG}_{(11a^2aa)}$	$\mathrm{GTCGT}_{(a1a^2a1)}$
GGA $(aa0)$	$\mathrm{GGAAG}_{(aa00a)}$	$\mathrm{GGATG}_{(aa01a)}$	$\mathrm{GGGAG}_{(aaa0a)}$	$\mathrm{GTAAG}_{(a100a)}$
	$\mathrm{CGAAG}_{(a^2a00a)}$	$\mathrm{GGAAC}_{(aa00a^2)}$	$\mathrm{GGAGG}_{(aa0aa)}$	$\mathrm{GGCAG}_{(aaa^20a)}$
	$\mathrm{GCAAG}_{(aa^200a)}$	$\mathrm{AGAAG}_{(0a00a)}$	$\mathrm{GGAAA}_{(aa000)}$	$\mathrm{GGACG}_{(aa0a^2a)}$
	$\mathrm{GGTAG}_{(aa10a)}$	$\mathrm{GAAAG}_{(a000a)}$	$\mathrm{TGAAG}_{(1a00a)}$	$\mathrm{GGAAT}_{(aa001)}$
GGT $(aa1)$	$\mathrm{GGTGA}_{(aa1a0)}$	$\mathrm{GGTCA}_{(aa1a^20)}$	$\mathrm{GGCGA}_{(aaa^2a0)}$	$\mathrm{GTTGA}_{(a11a0)}$
	$\mathrm{CGTGA}_{(a^2a1a0)}$	$\mathrm{GGTGT}_{(aa1a1)}$	$\mathrm{GGTAA}_{(aa100)}$	$\mathrm{GGGGA}_{(aaaa0)}$
	$\mathrm{GCTGA}_{(aa^21a0)}$	$\mathrm{AGTGA}_{(0a1a0)}$	$\mathrm{GGTGG}_{(aa1aa)}$	$\mathrm{GGTTA}_{(aa110)}$
	$\mathrm{GGAGA}_{(aa0a0)}$	$\mathrm{GATGA}_{(a01a0)}$	$\mathrm{TGTGA}_{(1a1a0)}$	$\mathrm{GGTGC}_{(aa1aa^2)}$

Table A.1: Degeneracy Mapping

GGG (aaa)	$\mathrm{GGGCT}_{(aaaa^2 1)}$	$\mathrm{GGGGT}_{(aaaa1)}$	$\mathrm{GGACT}_{(aa0a^2 1)}$	$\mathrm{GTGCT}_{(a1aa^2 1)}$
	$\mathrm{CGGCT}_{(a^2aaa^2 1)}$	$\mathrm{GGGCA}_{(aaaa^2 0)}$	$\mathrm{GGGTT}_{(aaa11)}$	$\mathrm{GGTCT}_{(aa1a^2 1)}$
	$\mathrm{GCGCT}_{(aa^2aa^2 1)}$	$\mathrm{AGGCT}_{(0aaa^2 1)}$	$\mathrm{GGGCC}_{(aaaa^2a^2)}$	$\mathrm{GGGAT}_{(aaa01)}$
	$\mathrm{GGCCT}_{(aaa^2a^2 1)}$	$\mathrm{GAGCT}_{(a0aa^2 1)}$	$\mathrm{TGGCT}_{(1aaa^2 1)}$	$\mathrm{GGGCG}_{(aaaa^2a)}$
GGC (aaa^2)	$\mathrm{GGCTC}_{(aaa^2 1a^2)}$	$\mathrm{GGCAC}_{(aaa^2 0a^2)}$	$\mathrm{GGTTC}_{(aa11a^2)}$	$\mathrm{GTCTC}_{(a1a^2 1a^2)}$
	$\mathrm{CGCTC}_{(a^2aa^2 1a^2)}$	$\mathrm{GGCTG}_{(aaa^2 1a)}$	$\mathrm{GGCCC}_{(aaa^2a^2a^2)}$	$\mathrm{GGATC}_{(aa01a^2)}$
	$\mathrm{GCCTC}_{(aa^2a^2 1a^2)}$	$\mathrm{AGCTC}_{(0aa^2 1a^2)}$	$\mathrm{GGCTT}_{(aaa^2 11)}$	$\mathrm{GGCGC}_{(aaa^2aa^2)}$
	$\mathrm{GGGTC}_{(aaa1a^2)}$	$\mathrm{GACTC}_{(a0a^2 1a^2)}$	$\mathrm{TGCTC}_{(1aa^2 1a^2)}$	$\mathrm{GGCTA}_{(aaa^2 10)}$
GCA (aa^20)	$\mathrm{GCATA}_{(aa^2010)}$	$\mathrm{GCAAA}_{(aa^2000)}$	$\mathrm{GCGTA}_{(aa^2a10)}$	$\mathrm{GAATA}_{(a0010)}$
	$\mathrm{CCATA}_{(a^2a^2010)}$	$\mathrm{GCATT}_{(aa^2011)}$	$\mathrm{GCACA}_{(aa^20a^20)}$	$\mathrm{GCCTA}_{(aa^2a^210)}$
	$\mathrm{GGATA}_{(aa010)}$	$\mathrm{ACATA}_{(0a^2010)}$	$\mathrm{GCATG}_{(aa^201a)}$	$\mathrm{GCAGA}_{(aa^20a0)}$
	$\mathrm{GCTTA}_{(aa^2110)}$	$\mathrm{GTATA}_{(a1010)}$	$\mathrm{TCATA}_{(1a^2010)}$	$\mathrm{GCATC}_{(aa^201a^2)}$
GCT (aa^21)	$\mathrm{GCTCG}_{(aa^21a^2a)}$	$\mathrm{GCTGG}_{(aa^21aa)}$	$\mathrm{GCCCG}_{(aa^2a^2a^2a)}$	$\mathrm{GATCG}_{(a01a^2a)}$
	$\mathrm{CCTCG}_{(a^2a^21a^2a)}$	$\mathrm{GCTCC}_{(aa^21a^2a^2)}$	$\mathrm{GCTTG}_{(aa^211a)}$	$\mathrm{GCGCG}_{(aa^2aa^2a)}$
	$\mathrm{GGTCG}_{(aa1a^2a)}$	$\mathrm{ACTCG}_{(0a^21a^2a)}$	$\mathrm{GCTCA}_{(aa^21a^20)}$	$\mathrm{GCTAG}_{(aa^210a)}$
	$\mathrm{GCACG}_{(aa^20a^2a)}$	$\mathrm{GTTCG}_{(a11a^2a)}$	$\mathrm{TCTCG}_{(1a^21a^2a)}$	$\mathrm{GCTCT}_{(aa^21a^21)}$
GCG (aa^2a)	$\mathrm{GCGGC}_{(aa^2aaa^2)}$	$\mathrm{GCGCC}_{(aa^2aa^2a^2)}$	$\mathrm{GCAGC}_{(aa^20aa^2)}$	$\mathrm{GAGGC}_{(a0aaa^2)}$
	$\mathrm{CCGGC}_{(a^2a^2aaa^2)}$	$\mathrm{GCGGG}_{(aa^2aaa)}$	$\mathrm{GCGAC}_{(aa^2a0a^2)}$	$\mathrm{GCTGC}_{(aa^21aa^2)}$
	$\mathrm{GGGGC}_{(aaaaa^2)}$	$\mathrm{ACGGC}_{(0a^2aaa^2)}$	$\mathrm{GCGGT}_{(aa^2aa1)}$	$\mathrm{GCGTC}_{(aa^2a1a^2)}$
	$\mathrm{GCCGC}_{(aa^2a^2aa^2)}$	$\mathrm{GTGGC}_{(a1aaa^2)}$	$\mathrm{TCGGC}_{(1a^2aaa^2)}$	$\mathrm{GCGGA}_{(aa^2aa0)}$
GCC (aa^2a^2)	$\mathrm{GCCAT}_{(aa^2a^201)}$	$\mathrm{GCCTT}_{(aa^2a^211)}$	$\mathrm{GCTAT}_{(aa^2101)}$	$\mathrm{GACAT}_{(a0a^201)}$
	$\mathrm{CCCAT}_{(a^2a^2a^201)}$	$\mathrm{GCCAA}_{(aa^2a^200)}$	$\mathrm{GCCGT}_{(aa^2a^2a1)}$	$\mathrm{GCAAT}_{(aa^2001)}$
	$\mathrm{GGCAT}_{(aaa^201)}$	$\mathrm{ACCAT}_{(0a^2a^201)}$	$\mathrm{GCCAC}_{(aa^2a^20a^2)}$	$\mathrm{GCCCT}_{(aa^2a^2a^21)}$
	$\mathrm{GCGAT}_{(aa^2a01)}$	$\mathrm{GTCAT}_{(a1a^201)}$	$\mathrm{TCCAT}_{(1a^2a^201)}$	$\mathrm{GCCAG}_{(aa^2a^20a)}$
CAA (a^200)	$\mathrm{CAACG}_{(a^200a^2a)}$	$\mathrm{CAAGG}_{(a^200aa)}$	$\mathrm{CAGCG}_{(a^20aa^2a)}$	$\mathrm{CCACG}_{(a^2a^20a^2a)}$
	$\mathrm{GAACG}_{(a00a^2a)}$	$\mathrm{CAACC}_{(a^200a^2a^2)}$	$\mathrm{CAATG}_{(a^2001a)}$	$\mathrm{CACCG}_{(a^20a^2a^2a)}$
	$\mathrm{CTACG}_{(a^210a^2a)}$	$\mathrm{TAACG}_{(100a^2a)}$	$\mathrm{CAACA}_{(a^200a^20)}$	$\mathrm{CAAAG}_{(a^2000a)}$
	$\mathrm{CATCG}_{(a^201a^2a)}$	$\mathrm{CGACG}_{(a^2a0a^2a)}$	$\mathrm{AAACG}_{(000a^2a)}$	$\mathrm{CAACT}_{(a^200a^21)}$

Table A.1: Degeneracy Mapping

CAT ($a^2 01$)	$\mathrm{CATTA}_{(a^2 0110)}$	$\mathrm{CATAA}_{(a^2 0100)}$	$\mathrm{CACTA}_{(a^2 0 a^2 10)}$	$\mathrm{CCTTA}_{(a^2 a^2 110)}$
	$\mathrm{GATTA}_{(a0110)}$	$\mathrm{CATTT}_{(a^2 0111)}$	$\mathrm{CATCA}_{(a^2 01 a^2 0)}$	$\mathrm{CAGTA}_{(a^2 0 a10)}$
	$\mathrm{CTTTA}_{(a^2 1110)}$	$\mathrm{TATTA}_{(10110)}$	$\mathrm{CATTG}_{(a^2 011a)}$	$\mathrm{CATGA}_{(a^2 01a0)}$
	$\mathrm{CAATA}_{(a^2 0010)}$	$\mathrm{CGTTA}_{(a^2 a110)}$	$\mathrm{AATTA}_{(00110)}$	$\mathrm{CATTC}_{(a^2 011 a^2)}$
CAG ($a^2 0a$)	$\mathrm{CAGAT}_{(a^2 0a01)}$	$\mathrm{CAGTT}_{(a^2 0a11)}$	$\mathrm{CAAAT}_{(a^2 0001)}$	$\mathrm{CCGAT}_{(a^2 a^2 a01)}$
	$\mathrm{GAGAT}_{(a0a01)}$	$\mathrm{CAGAA}_{(a^2 0a00)}$	$\mathrm{CAGGT}_{(a^2 0aa1)}$	$\mathrm{CATAT}_{(a^2 0101)}$
	$\mathrm{CTGAT}_{(a^2 1a01)}$	$\mathrm{TAGAT}_{(10a01)}$	$\mathrm{CAGAC}_{(a^2 0a0 a^2)}$	$\mathrm{CAGCT}_{(a^2 0aa^2 1)}$
	$\mathrm{CACAT}_{(a^2 0 a^2 01)}$	$\mathrm{CGGAT}_{(a^2 aa01)}$	$\mathrm{AAGAT}_{(00a01)}$	$\mathrm{CAGAG}_{(a^2 0a0a)}$
CAC ($a^2 0a^2$)	$\mathrm{CACGC}_{(a^2 0 a^2 aa^2)}$	$\mathrm{CACCC}_{(a^2 0 a^2 a^2 a^2)}$	$\mathrm{CATGC}_{(a^2 01aa^2)}$	$\mathrm{CCCGC}_{(a^2 a^2 a^2 aa^2)}$
	$\mathrm{GACGC}_{(a0 a^2 aa^2)}$	$\mathrm{CACGG}_{(a^2 0 a^2 aa)}$	$\mathrm{CACAC}_{(a^2 0 a^2 0 a^2)}$	$\mathrm{CAAGC}_{(a^2 00aa^2)}$
	$\mathrm{CTCGC}_{(a^2 1 a^2 aa^2)}$	$\mathrm{TACGC}_{(10 a^2 aa^2)}$	$\mathrm{CACGT}_{(a^2 0 a^2 a1)}$	$\mathrm{CACTC}_{(a^2 0 a^2 1 a^2)}$
	$\mathrm{CAGGC}_{(a^2 0aaa^2)}$	$\mathrm{CGCGC}_{(a^2 aa^2 aa^2)}$	$\mathrm{AACGC}_{(00 a^2 aa^2)}$	$\mathrm{CACGA}_{(a^2 0 a^2 a0)}$
CTA ($a^2 10$)	$\mathrm{CTAGA}_{(a^2 10a0)}$	$\mathrm{CTACA}_{(a^2 10 a^2 0)}$	$\mathrm{CTGGA}_{(a^2 1aa0)}$	$\mathrm{CGAGA}_{(a^2 a0a0)}$
	$\mathrm{GTAGA}_{(a10a0)}$	$\mathrm{CTAGT}_{(a^2 10a1)}$	$\mathrm{CTAAA}_{(a^2 1000)}$	$\mathrm{CTCGA}_{(a^2 1 a^2 a0)}$
	$\mathrm{CAAGA}_{(a^2 00a0)}$	$\mathrm{TTAGA}_{(110a0)}$	$\mathrm{CTAGG}_{(a^2 10aa)}$	$\mathrm{CTATA}_{(a^2 1010)}$
	$\mathrm{CTTGA}_{(a^2 11a0)}$	$\mathrm{CCAGA}_{(a^2 a^2 0a0)}$	$\mathrm{ATAGA}_{(010a0)}$	$\mathrm{CTAGC}_{(a^2 10aa^2)}$
CTT ($a^2 11$)	$\mathrm{CTTAG}_{(a^2 110a)}$	$\mathrm{CTTTG}_{(a^2 111a)}$	$\mathrm{CTCAG}_{(a^2 1 a^2 0a)}$	$\mathrm{CGTAG}_{(a^2 a10a)}$
	$\mathrm{GTTAG}_{(a110a)}$	$\mathrm{CTTAC}_{(a^2 110 a^2)}$	$\mathrm{CTTGG}_{(a^2 11aa)}$	$\mathrm{CTGAG}_{(a^2 1a0a)}$
	$\mathrm{CATAG}_{(a^2 010a)}$	$\mathrm{TTTAG}_{(1110a)}$	$\mathrm{CTTAA}_{(a^2 1100)}$	$\mathrm{CTTCG}_{(a^2 11 a^2 a)}$
	$\mathrm{CTAAG}_{(a^2 100a)}$	$\mathrm{CCTAG}_{(a^2 a^2 10a)}$	$\mathrm{ATTAG}_{(0110a)}$	$\mathrm{CTTAT}_{(a^2 1101)}$
CTG ($a^2 1a$)	$\mathrm{CTGTC}_{(a^2 1 a1 a^2)}$	$\mathrm{CTGAC}_{(a^2 1a0 a^2)}$	$\mathrm{CTATC}_{(a^2 101 a^2)}$	$\mathrm{CGGTC}_{(a^2 aa1 a^2)}$
	$\mathrm{GTGTC}_{(a1a1 a^2)}$	$\mathrm{CTGTG}_{(a^2 1a1a)}$	$\mathrm{CTGCC}_{(a^2 1aa^2 a^2)}$	$\mathrm{CTTTC}_{(a^2 111 a^2)}$
	$\mathrm{CAGTC}_{(a^2 0a1 a^2)}$	$\mathrm{TTGTC}_{(11a1 a^2)}$	$\mathrm{CTGTT}_{(a^2 1a11)}$	$\mathrm{CTGGC}_{(a^2 1aaa^2)}$
	$\mathrm{CTCTC}_{(a^2 1 a^2 1 a^2)}$	$\mathrm{CCGTC}_{(a^2 a^2 a1 a^2)}$	$\mathrm{ATGTC}_{(01a1 a^2)}$	$\mathrm{CTGTA}_{(a^2 1a10)}$
CTC ($a^2 1a^2$)	$\mathrm{CTCCT}_{(a^2 1 a^2 a^2 1)}$	$\mathrm{CTCGT}_{(a^2 1 a^2 a1)}$	$\mathrm{CTTCT}_{(a^2 11 a^2 1)}$	$\mathrm{CGCCT}_{(a^2 aa^2 a^2 1)}$
	$\mathrm{GTCCT}_{(a1 a^2 a^2 1)}$	$\mathrm{CTCCA}_{(a^2 1 a^2 a^2 0)}$	$\mathrm{CTCTT}_{(a^2 1 a^2 11)}$	$\mathrm{CTACT}_{(a^2 10 a^2 1)}$
	$\mathrm{CACCT}_{(a^2 0 a^2 a^2 1)}$	$\mathrm{TTCCT}_{(11 a^2 a^2 1)}$	$\mathrm{CTCCC}_{(a^2 1 a^2 a^2 a^2)}$	$\mathrm{CTCAT}_{(a^2 1 a^2 01)}$
	$\mathrm{CTGCT}_{(a^2 1aa^2 1)}$	$\mathrm{CCCCT}_{(a^2 a^2 a^2 a^2 1)}$	$\mathrm{ATCCT}_{(01 a^2 a^2 1)}$	$\mathrm{CTCCG}_{(a^2 1 a^2 a^2 a)}$

Table A.1: Degeneracy Mapping

CGA (a^2a0)	$\mathrm{CGATT}_{(a^2a011)}$	$\mathrm{CGAAT}_{(a^2a001)}$	$\mathrm{CGGTT}_{(a^2aa11)}$	$\mathrm{CTATT}_{(a^21011)}$
	$\mathrm{GGATT}_{(aa011)}$	$\mathrm{CGATA}_{(a^2a010)}$	$\mathrm{CGACT}_{(a^2a0a^21)}$	$\mathrm{CGCTT}_{(a^2aa^211)}$
	$\mathrm{CCATT}_{(a^2a^2011)}$	$\mathrm{TGATT}_{(1a011)}$	$\mathrm{CGATC}_{(a^2a01a^2)}$	$\mathrm{CGAGT}_{(a^2a0a1)}$
	$\mathrm{CGTTT}_{(a^2a111)}$	$\mathrm{CAATT}_{(a^20011)}$	$\mathrm{AGATT}_{(0a011)}$	$\mathrm{CGATG}_{(a^2a01a)}$
CGT (a^2a1)	$\mathrm{CGTCC}_{(a^2a1a^2a^2)}$	$\mathrm{CGTGC}_{(a^2a1aa^2)}$	$\mathrm{CGCCC}_{(a^2aa^2a^2a^2)}$	$\mathrm{CTTCC}_{(a^211a^2a^2)}$
	$\mathrm{GGTCC}_{(aa1a^2a^2)}$	$\mathrm{CGTCG}_{(a^2a1a^2a)}$	$\mathrm{CGTTC}_{(a^2a11a^2)}$	$\mathrm{CGGCC}_{(a^2aaa^2a^2)}$
	$\mathrm{CCTCC}_{(a^2a^21a^2a^2)}$	$\mathrm{TGTCC}_{(1a1a^2a^2)}$	$\mathrm{CGTCT}_{(a^2a1a^21)}$	$\mathrm{CGTAC}_{(a^2a10a^2)}$
	$\mathrm{CGACC}_{(a^2a0a^2a^2)}$	$\mathrm{CATCC}_{(a^201a^2a^2)}$	$\mathrm{AGTCC}_{(0a1a^2a^2)}$	$\mathrm{CGTCA}_{(a^2a1a^20)}$
CGG (a^2aa)	$\mathrm{CGGGG}_{(a^2aaaa)}$	$\mathrm{CGGCG}_{(a^2aaa^2a)}$	$\mathrm{CGAGG}_{(a^2a0aa)}$	$\mathrm{CTGGG}_{(a^21aaa)}$
	$\mathrm{GGGGG}_{(aaaaa)}$	$\mathrm{CGGGC}_{(a^2aaaa^2)}$	$\mathrm{CGGAG}_{(a^2aa0a)}$	$\mathrm{CGTGG}_{(a^2a1aa)}$
	$\mathrm{CCGGG}_{(a^2a^2aaa)}$	$\mathrm{TGGGG}_{(1aaaa)}$	$\mathrm{CGGGA}_{(a^2aaa0)}$	$\mathrm{CGGTG}_{(a^2aa1a)}$
	$\mathrm{CGCGG}_{(a^2aa^2aa)}$	$\mathrm{CAGGG}_{(a^20aaa)}$	$\mathrm{AGGGG}_{(0aaaa)}$	$\mathrm{CGGGT}_{(a^2aaa1)}$
CGC (a^2aa^2)	$\mathrm{CGCAA}_{(a^2aa^200)}$	$\mathrm{CGCTA}_{(a^2aa^210)}$	$\mathrm{CGTAA}_{(a^2a100)}$	$\mathrm{CTCAA}_{(a^21a^200)}$
	$\mathrm{GGCAA}_{(aaa^200)}$	$\mathrm{CGCAT}_{(a^2aa^201)}$	$\mathrm{CGCGA}_{(a^2aa^2a0)}$	$\mathrm{CGAAA}_{(a^2a000)}$
	$\mathrm{CCCAA}_{(a^2a^2a^200)}$	$\mathrm{TGCAA}_{(1aa^200)}$	$\mathrm{CGCAG}_{(a^2aa^20a)}$	$\mathrm{CGCCA}_{(a^2aa^2a^20)}$
	$\mathrm{CGGAA}_{(a^2aa00)}$	$\mathrm{CACAA}_{(a^20a^200)}$	$\mathrm{AGCAA}_{(0aa^200)}$	$\mathrm{CGCAC}_{(a^2aa^20a^2)}$
CCA (a^2a^20)	$\mathrm{CCAAC}_{(a^2a^200a^2)}$	$\mathrm{CCATC}_{(a^2a^201a^2)}$	$\mathrm{CCGAC}_{(a^2a^2a0a^2)}$	$\mathrm{CAAAC}_{(a^2000a^2)}$
	$\mathrm{GCAAC}_{(aa^200a^2)}$	$\mathrm{CCAAG}_{(a^2a^200a)}$	$\mathrm{CCAGC}_{(a^2a^20aa^2)}$	$\mathrm{CCCAC}_{(a^2a^2a^20a^2)}$
	$\mathrm{CGAAC}_{(a^2a00a^2)}$	$\mathrm{TCAAC}_{(1a^200a^2)}$	$\mathrm{CCAAT}_{(a^2a^2001)}$	$\mathrm{CCACC}_{(a^2a^20a^2a^2)}$
	$\mathrm{CCTAC}_{(a^2a^210a^2)}$	$\mathrm{CTAAC}_{(a^2100a^2)}$	$\mathrm{ACAAC}_{(0a^200a^2)}$	$\mathrm{CCAAA}_{(a^2a^2000)}$
CCT (a^2a^21)	$\mathrm{CCTGT}_{(a^2a^21a1)}$	$\mathrm{CCTCT}_{(a^2a^21a^21)}$	$\mathrm{CCCGT}_{(a^2a^2a^2a1)}$	$\mathrm{CATGT}_{(a^201a1)}$
	$\mathrm{GCTGT}_{(aa^21a1)}$	$\mathrm{CCTGA}_{(a^2a^21a0)}$	$\mathrm{CCTAT}_{(a^2a^2101)}$	$\mathrm{CCGGT}_{(a^2a^2aa1)}$
	$\mathrm{CGTGT}_{(a^2a1a1)}$	$\mathrm{TCTGT}_{(1a^21a1)}$	$\mathrm{CCTGC}_{(a^2a^21aa^2)}$	$\mathrm{CCTTT}_{(a^2a^2111)}$
	$\mathrm{CCAGT}_{(a^2a^20a1)}$	$\mathrm{CTTGT}_{(a^211a1)}$	$\mathrm{ACTGT}_{(0a^21a1)}$	$\mathrm{CCTGG}_{(a^2a^21aa)}$
CCG (a^2a^2a)	$\mathrm{CCGCA}_{(a^2a^2aa^20)}$	$\mathrm{CCGGA}_{(a^2a^2aa0)}$	$\mathrm{CCACA}_{(a^2a^20a^20)}$	$\mathrm{CAGCA}_{(a^20aa^20)}$
	$\mathrm{GCGCA}_{(aa^2aa^20)}$	$\mathrm{CCGCT}_{(a^2a^2aa^21)}$	$\mathrm{CCGTA}_{(a^2a^2a10)}$	$\mathrm{CCTCA}_{(a^2a^21a^20)}$
	$\mathrm{CGGCA}_{(a^2aaa^20)}$	$\mathrm{TCGCA}_{(1a^2aa^20)}$	$\mathrm{CCGCG}_{(a^2a^2aa^2a)}$	$\mathrm{CCGAA}_{(a^2a^2a00)}$
	$\mathrm{CCCCA}_{(a^2a^2a^2a^20)}$	$\mathrm{CTGCA}_{(a^21aa^20)}$	$\mathrm{ACGCA}_{(0a^2aa^20)}$	$\mathrm{CCGCC}_{(a^2a^2aa^2a^2)}$

Table A.1: Degeneracy Mapping

CCC $(a^2a^2a^2)$	$\mathrm{CCCTG}_{(a^2a^2a^21a)}$	$\mathrm{CCCAG}_{(a^2a^2a^20a)}$	$\mathrm{CCTTG}_{(a^2a^211a)}$	$\mathrm{CACTG}_{(a^20a^21a)}$
	$\mathrm{GCCTG}_{(aa^2a^21a)}$	$\mathrm{CCCTC}_{(a^2a^2a^21a^2)}$	$\mathrm{CCCCG}_{(a^2a^2a^2a^2a)}$	$\mathrm{CCATG}_{(a^2a^201a)}$
	$\mathrm{CGCTG}_{(a^2aa^21a)}$	$\mathrm{TCCTG}_{(1a^2a^21a)}$	$\mathrm{CCCTA}_{(a^2a^2a^210)}$	$\mathrm{CCCGG}_{(a^2a^2a^2aa)}$
	$\mathrm{CCGTG}_{(a^2a^2a1a)}$	$\mathrm{CTCTG}_{(a^21a^21a)}$	$\mathrm{ACCTG}_{(0a^2a^21a)}$	$\mathrm{CCCTT}_{(a^2a^2a^211)}$