Analysis of the genetic code
Characterisation of robustness properties in the framework of graph theory

Analyse des genetishen codes
Characterisierung von robustheit Eigenschaften im Rahmen von Graphen Theorie

by George Dadunashvili

Master Thesis by:
George Dadunashvili
Department of Physics
Ludwig-Maximilians-Universität München

Supervisors:
Prof. Dr. Erwin Frey
M.Sc. Florian Gartner

Munich,
September 8, 2017

ABSTRACT

We know that predicting the spatial structure of proteins is difficult. It is not trivial to predict if a given functional protein will remain functional after a given number of mutations in the DNA that codes for it. Even for a given biological system which performs a very specialized task it is in general not straightforward to determine how robust the proteins are.

This thesis is mainly concerned with the role of the universal genetic code in biological translation and transcription machinery. The goal is to understand its impact on the robustness of proteins. Is the genetic code a relevant factor in understanding the protein properties?

The answer is obviously yes. The above question is also very easy to stumble upon, since the universal genetic code is one out of a huge number of possible codes and yet universal among almost every living organism. This has consequentially led to many investigations of its "specialness". But since the genetic code is a rule of transcription from RNA to proteins its effects are observed inside biological systems. If any special feature in the ensemble of proteins is observed, it is difficult to distinguish which led to this special property, the genetic code or the evolutionary pressure inside this biological system. And probably both factors are very important.

The novelty of this thesis lies in the new description of the genetic code as a graph between amino acids. This allows us to describe the code free from any biological context, and especially to introduce a notion of robustness. Using the weighted adjacency matrix of the genetic code, we define a space of possible codes and a "measure" of closeness between codes. Then we analyze the ensemble of codes "close" to the genetic code and "far away" from it and find very compelling evidence that the universal genetic code is truly very special among the analyzed codes, with respect to the robustness property.

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION FOR INVESTIGATING AMINO ACID SEQUENCES

In biology it is a known fact that the information about living organisms is stored in their genomes which consists of long strands of DNA. Sequences of four nucleic acids adenine (A), cytosine (C), guanine (G) and thymine(T), which are constituent parts of DNA encode the information about the organism. An elaborate molecular machinery inside the cell transcribes the relevant parts of the DNA to messenger RNA. This finally gets translated into proteins, which themselves are made up from sequences of 20 amino acids(see A.2). Proteins are macromolecules which are responsible for most of the functions inside the cell, for example catalyzing chemical reactions. There are many different proteins very diverse in their function[13, Ch.1.4]. Hence understanding protein properties is of central interest in many biological applications.
Given a random amino acid sequence it is generally not possible to determine the 3 dimensional spacial structure of this protein (also referred to as the tertiary structure). In order to make predictions it is necessary to conduct expensive simulations. This is known as the protein folding problem (which is considered as NP-hard [15], though research is still ongoing [8]). Since biological properties or functionality of a protein is heavily influenced by its spatial conformation, the understanding of the protein functionality is contingent on our understanding of its tertiary structure. One might ask a simpler version of the question; Given a sequence of amino acids is it probable that they will fold into a functional protein? In context of some biological process this is the same question as deciding how probable the sequence is to be generated. It stands to reason that if the generation of a sequence is probable then it is plausible to assume that this sequence will fold into a functional protein. Hence finding a model which assigns a probability distribution to any protein sequence within a biological context, can greatly simplify the protein folding problem since it can reduce the number of candidates, which need to be checked for the functionality of the tertiary structure. This approach however, is dependent on the properties of a certain biological system and might not help with our understanding of some general features of proteins.

We can ask a even simpler question. Given an amino acid sequence of a functional protein, how much of it can be changed until the protein looses functionality? This brings us to the notion of robustness. Loosely said a robust protein is one which can withstand minor structural changes without loosing its functionality. In a more mathematical sense it can be understood as a measure, which relates closeness of amino acid sequences[1] to closeness of (or relation in) functionality of proteins.
Since the DNA contains the information about the proteins, random changes occur on the level of the DNA and not on the level of amino acid sequences. And since the mapping from DNA to proteins is degenerate (several DNA sequences can code exactly the same amino acid sequence), there are Mutations on the DNA level which do not change the protein at all, they are referred to as silent mutations (more detail on the universal genetic code will be provided in section 1.3). Furthermore the amino acids are partitioned into chemically related groups, so if one amino acid gets replaced by a chemically related one it might not have an effect on the function of the protein[2][1, pp.248-250].
Taking the "degenerate mapping" and the chemical relation between amino acids together we see that a certain "strength" of mutations in the DNA code might be needed in order to achieve a functional change in the amino acid sequence.

1.2 SPECIFYING THE GOAL

The broad question of the protein robustness can be broken down into more specific parts:

Q1: What is the strength of fluctuations needed, on the DNA level to affect functional change on the protein level?

Q2: How strongly is the robustness determined by the context of a biological system of which the protein is part of?

Q3: How strongly is the functionality determined by the chemical properties (similarities) of the amino acids, in the protein?

1 e.g. Hamming distance between two amino acid sequences
2 Protein functionality is also a vague term, but it can be formalized in terms of fitness. One could for example measure how much the efficiency of performing a given task changes when we change the structure of the protein[16, sec:Genome Fitness]

Q4: How is the robustness of proteins influenced by the genetic code[3]?

Q4.1: How are the statistical properties[4] of amino acid sequences affected by changes in genetic code?

All of this questions are interesting and there is a rich literature discussing robustness of amino acid sequences[2], their functionality and diversity in biological context [4, 10, 11, 12] and the role of the genetic code in optimizing certain measures for robustness[6]. But all of these questions are rich in complexity and as a quick survey of the literature will show, the difficulties arise by large part, because it is not obvious how to quantify heuristics like optimal, robust functional etc. And for each well constructed measure there are many others which would be similarly well suited[16]. This problem of course is not exclusive to this topic of research and is plaguing the entire field of biophysics.
Since the scope of a master thesis is limited, I chose to concentrate my efforts on investigating the the question Q4.1 which circumvents the "messiness" of quantifying robustness. The main idea was to generate a random DNA sequence ensemble and a set of genetic codes and see if the structure of correlations in the resulting amino acid sequences would change from one genetic code to another. Unsurprisingly they did and the hope was that it would be possible to make inferences about the genetic code structure from the observed changes. This investigation was done for a toy model (binary DNA), and it became clear that the dependence on the random DNA ensemble was to great to make inferences about the genetic code. A detailed discussion of this is provided in the appendix A.3. After this I moved away from the toy model and started modeling the 4 letter DNA. Instead of correlations I used Shannon entropy associated to the DNA and protein ensemble (see 2.2) to characterize them and managed to observe the dependence between the change in Shannon entropy after translation and the genetic codes, used for translation.
Probably the clearest statement of the initial goal of this thesis is the following: understanding the structure of the state space of genetic codes, and analyzing the position of the universal code, in it. Within the following chapters I will define the state space of genetic codes and provide a systematic way of describing codes and their "closeness" to the universal code. Using this, I will present my observations on how the Shanon entropy changes on average after the translation of DNA into proteins, depending on the "distance" of the used genetic code from the universal code. During the initial work on the thesis, it became apparent that one could also take a more direct approach towards analyzing genetic codes and attribute a measure of robustness directly to them. In chapter 3 I will

3 The definition of the genetic code is presented in the following section.

4 What is exactly meant by statistical structure will be specified later (see ch.2), but on could imagine for example generating an ensemble of pseudo DNA sequences calculating the typical correlation length and then observe how it changes with different translations.

show how one can accomplish this and how the robustness of the universal code looks compared to others.

1.3 UNIVERSAL GENETIC CODE

As already stated the genetic information about the protein is encoded in the DNA or rather the coding region of the DNA [13, Ch 1.3]. Which is transcribed into RNA and then finally translated to proteins (central dogma of biology). The exact biological details are for the following analysis irrelevant, and from now on I will refer to the whole process as the translation of the DNA into proteins (omitting the RNA step). During this translation 3 consecutive nucleotides of the DNA constitute one "DNA word", called codon which is assigned to one amino acid.
The genetic code can be seen as a dictionary which maps codons onto amino acids. Each codon consists of three consecutive DNA nucleotides, which can be either one of the following four: thymine (T), cytosine (C), adenine (A) and guanine (G), hence there are 64 possible codons. The total number of amino acids used in most organisms is 20 [13, Ch 1.3]. This means that we have to map 64 codons to 20 amino acids, plus an additional stop command for the translation machinery. The canonical genetic code (from now on referred to as the universal code) is a specific choice of this map. Hence a particular assignment of codons to the amino acids they encode. This can be represented as a list of pairs of all 64 codons with their amino acids[5], or a list of 20 amino acids with their corresponding codons (see tab.A.2 and A.1). Most popular pictorial representation of the code (which can be seen in the majority of the literature) can be seen in the figure 1.1, the nucleotide letters should be read from the center outward to construct a codon, with the corresponding amino acid name written in the outer most layer. This picture reveals the fact that most codons which code for the same amino acid only differ in the last letter.

One can however imagine that there is a more complex structure of relations between amino acids hiding inside the code, which simply can not be adequately depicted in this two dimensional representation. Even worse this picture can be misleading. If one looks at amino acids tyrosine (codons TAT and TAC) and asparagine (codons AAT and AAC), they appear very different in their codon content since they are on the opposed sides of the wheel, but the codons only differ in the first nucleotide, hence only one mutation in the first nucleotide could transform tyrosine into asparagine, and in this sense this amino acids could be considered next neighbors. One could of course try to redraw this picture in

5 This is useful, for an efficient implementation of the code in a computer program, as a vector of 64 elements. Each element can be a number between 0 and 20 representing an amino acid while the index of the element represents the codon encoding it.

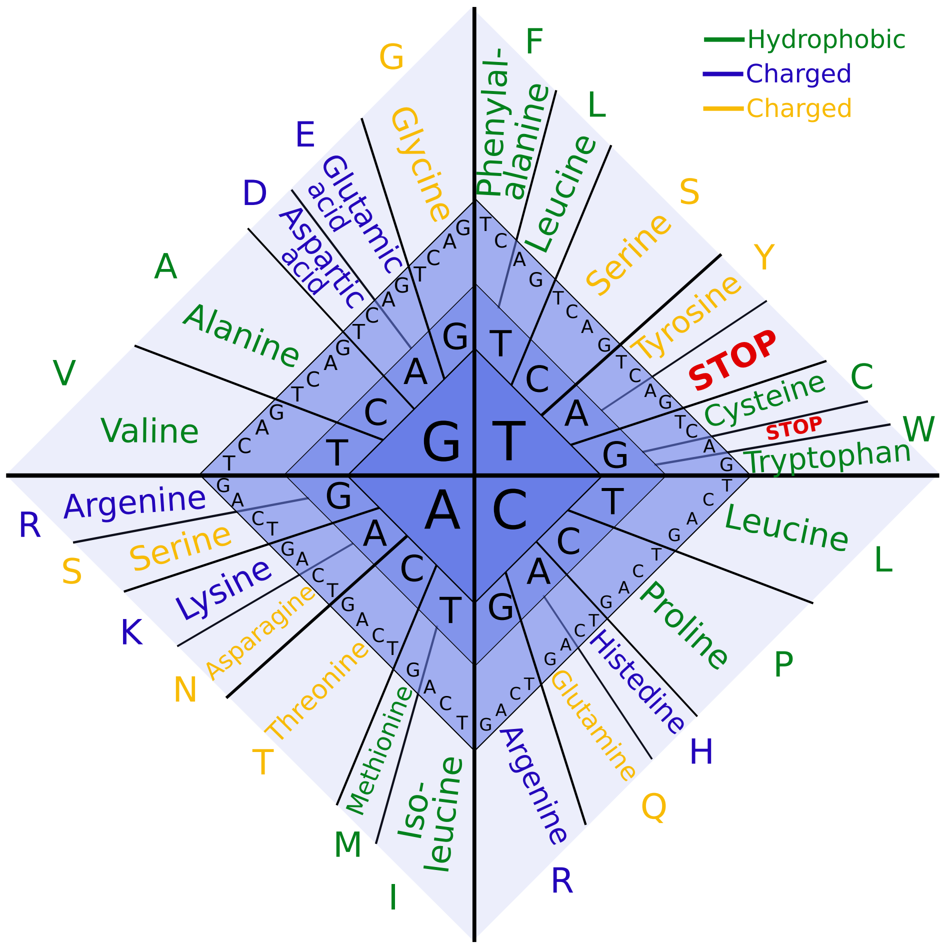


Figure 1.1.: A pictorial representation of the genetic code. Each black letter stands for a DNA nucleotide. **Read from center outward**, the 3 consecutive letters form a codon which encode the amino acid named in the outer most layer.

3 dimensions, but unfortunately one will never arrive at a figure with desirable properties, since there is no regular polyhedron with 64 vertices in 3 dimensions.
The mathematical representation which I chose to describe the genetic code is a graph with 21 nodes (one for each amino acid and stop command). One can construct this graph by first constructing a graph of 64 nodes one fore each codon, and then connect each codon to every other which differs only by one nucleotide replacement (see fig.1.3). Each codon has 3 nucleotides, so one has a possibility to make a single nucleotide replacement at each position. By making all possible replacement one can derive all next neighbors. Each codon has 9. For an example see tab.1.1

nucleotide	next neighbors		
	ATT	TAT	TTA
TTT	CTT	TCT	TTC
	GTT	TGT	TTG

Table 1.1.: This table shows the **example of the codon TTT and its 9 next neighbors**. The connection of TTT to its next neighbours is highlighted in the fig.1.3. As one can see the codons can change into each other, by replacing one nucleotide at a given position (single nucleotide replacement).

One can construct the amino acid graph by merging the nodes of the codon graph, without deleting the edges. One must clump all codons into the nodes of amino acids they encode. Then two amino acid nodes are connected with an edge if the codons of this two amino acids can transform into each other via a single nucleotide substitution. Most amino acids have several connections to each other since they have more then one codon. One than draws one edge (instead of many) between two connected amino acids an weights the edge by the number of possible connections (see fig.1.2). Such graphs are represented by their adjacency matrices[5, Ch. 2] (see tab.A.3 and fig.A.1). This picture clearly shows relations between amino acids with similar codons. If we go back to our previous example of tyrosine (Y) and asparagine (N) we will see that they are connected with a thin line (fig.1.2) and in the adjacency matrix (tab.A.3) we can see that the corresponding entries Y, N and N, Y are equal to 2. This is because there exist two nucleotide replacements for TAC (a codon of tyrosine) which mutates it to either one of the codons of asparagine. On the other hand for each of the codons of asparagine exists one nucleotide replacement which mutates them to TAC. So one can say that there are two nucleotide replacements which mutate tyrosine and asparagine into each other.
The advantage of the graph representation is not only the fact that it is more informative to look at than the conventional table of codon and amino acid pairs. Main advantage comes from the fact that the graphs can be represented as adjacency matrices, which provide an

unique description for each possible genetic code. And we have the tools of graph theory at our disposal to analyze its properties (more on this in later sections 2.1.1 and 3.1).

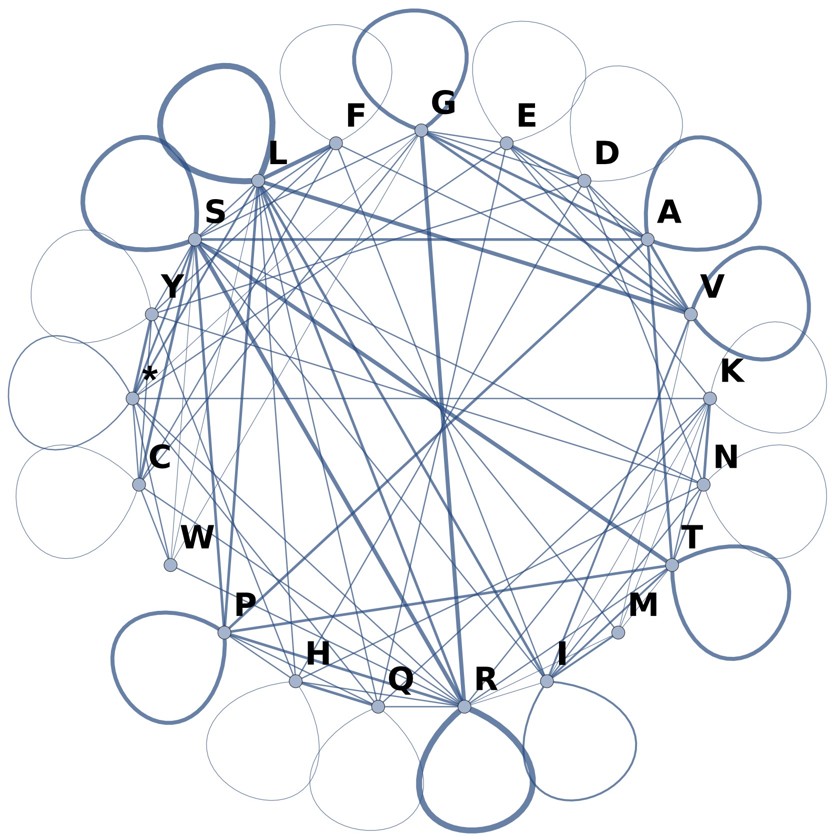


Figure 1.2.: A representation of the genetic code as a graph. Each node represents an amino acid. **Two amino acids are connected by an edge if any of their codons can be mutated into the codons of the other amino acid, by single nucleotide replacement**. Each line is weighted by the number of possible mutations between two amino acids.

1.3.1 *Complications*

Since we want to understand how the statistical properties of any given amino acid sequence ensemble change with different genetic codes, we need to know what the number of possible genetic codes is. For this end I represent the code as an assignment of 64 codons to 20 amino acids and 1 stop command. This is equivalent to distributing 64 particles to 21 boxes with the constraint that each box has to have at least 1 particle in it. Each such configuration can be represented as a set of heights

$$\left\{ h_i \in [1,44] \mid i \in [1,21], \sum_{i=1}^{21} h_i = 64 \right\}. \tag{1.1}$$

We disregard all properties of amino acids besides their codon content, hence the names of amino acids do not contain any information, and our results should be invariant under renaming of amino acids. This means that the ordering of the boxes is irrelevant and we can always order the boxes in weakly decreasing order of their particle content. This makes it easier to calculate the degeneracy of each configuration. The total number of codes (C_{tot}) can be calculated as

$$C_{tot} = \sum_{\{h_i\}_i} \frac{64!}{\prod_{i=1}^{21} \left[h_i! \left(\sum_{j>1}^{21} \delta_{h_i,h_j} + 1 \right) \right]}. \tag{1.2}$$

The derivation of eqn. 1.2 is presented in the appendix A.2.
In order to calculate this sum we need to know each possible hight configuration $\{h_i\}_i$. Since each box will contain at least 1 particle the problem reduces to filling 21 boxes with 43 particles, in a way that the boxes are filled in weakly decreasing order. This corresponds to drawing all Young tableau's with maximal height of 43 and maximal width of 21. Which itself corresponds to the integer partition of 43 into at most 21 integers ([7, p.1]). There are 59755 such partitions and each one of them can be found with Mathematica[9](see A.2). In the end we arrive at

$$C_{tot} \approx 2.95573 \times 10^{64}. \tag{1.3}$$

With my current algorithm the generation of each code takes roughly $10\mu s$. Even if I could speed it up to $1ns$ and I had roughly 10000 cores at my disposal the waiting time to generate all codes would still be 9.37255×10^{43} years. This number is so large that calling it astronomic is not doing it justice.
In order to cope with this problem I first used a simplified model, with binary sequences as a reduced model of DNA and smaller codons with only two letters (a detailed description can be found in chapter A.3). The number of possible genetic codes in this model was reduced to 13. My goal was to get an intuition for the system and see if one could already learn something from the minimal model. Unfortunately it turned out that the

simplifications which I made, were to strong and no useful connections to the real system[6] could be made. In retrospect this is not very surprising, since the goal was to find out more about the statistical properties of the ensemble of genetic codes, and to hope that a study of en ensemble of size 13 could be instructive for the study of an ensemble of size 10^{64} is a little bit naive.
This forced me to start working with the full model, which turned out to be very difficult, exactly because of the large numbers of codes. Since exhaustive sampling was out of question and I could not even hope to generate any sample of meaningful size, I had to develop an algorithm for generating genetic codes, related to the universal code. The main problem here was to define a metric for relatedness of codes and justify its significance (all of which will be discussed in Chapter 2).

1.3.2 *Thesis structure and personal reflections*

During the Christmas break of 2015-2016 I was doing an on-line course in introductory biology when I first heard of the genetic code. The visualization used in that course was similar to the figure 1.1. I was not satisfied by the fact that the neighboring Amino acids in the picture were not necessarily more closely related to each other then the amino acids which were not neighbors. So I used MATHEMATICA to create a graph which would connect amino acids, depending on their codon content. It looked much like the figure 1.2. Then I made a plot of related amino acids (fig. 1.3) which has a very regular structure (this was not surprising since each codon has the same number of next neighbors which are always same distance away i.e. only differ in one nucleotide). So initially my choice of genetic code as a topic of research, was motivated by the "weird" structure of the code graph. At that point I had also read a paper *Maximum entropy models for antibody diversity* by Mora et al. [11] This was an analysis of experimental findings about Zebra-fish antibody diversity[17]. One of the findings was that the used maximum entropy model implied a critical strength of interactions within amino acids of analyzed protein sequences[11, section: Zipfs' Law and Fig.3]. The authors interpretation was that, the set of proteins expressed in the system was restricted to a much smaller state space (than the space of all possible sequences). Motivated by this findings and my curiosity about the structure of the universal code, I decided to tackle a very ambitious problem. Namely, finding out if the universal genetic code imposed a special kind of restrictions on the state space of proteins.[7]

6 With real system I mean DNA consisting of 4 nucleotides and 3 letter long codons.
7 I was well aware that each protein expressed in a living system is a result of a long process of evolutionary selection, which itself restricts the state of possible functional proteins we observe.

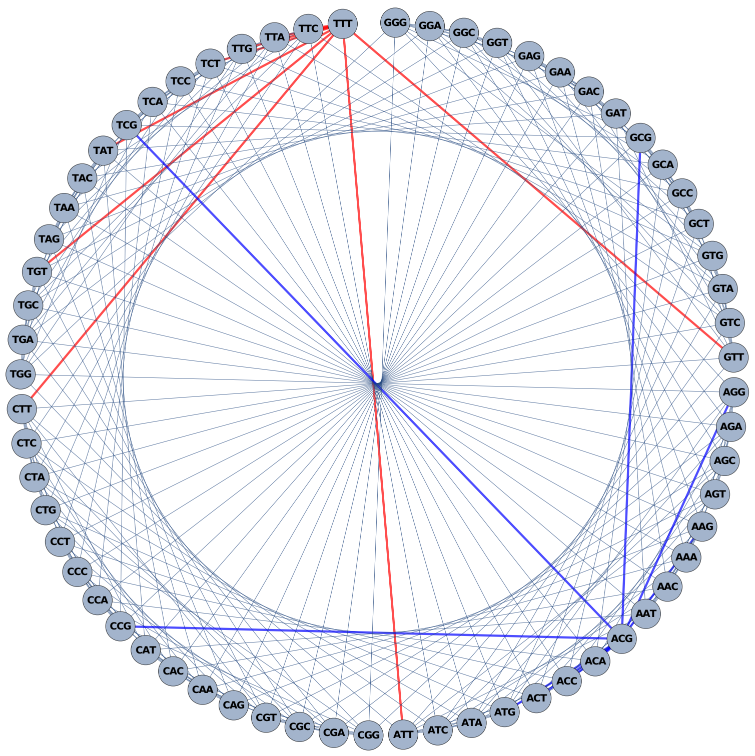


Figure 1.3.: Graph of codons. Showing them as **next neighbors if one codon can be mutated into another by single nucleotide replacement**. Codons *TTT* and *AGC* were chosen at random, and their connections to their next neighbors are displayed bold and in color.

In hindsight it seems very obvious that this question was to broad in its formulation and touched a far to great domain of research, to be feasible within the time constraints of a master thesis. But my limited understanding of biology, statistical physics and computational techniques involved in specifying and solving the problem, left me completely oblivious to this complexity. The result of this circumstances was that the research proceeded in the manner of trial and error (with a big emphasis on error). It was only later when I understood the need to constraint the boundaries of the problem and restricted the goals of the research (as described in section 1.2), that finally some advancements could be made. Even though the final findings of the thesis do not fully meet the initial ambitious goal, I hope that they will be instructive, and provide a better understanding of the properties of the genetic code.

But I wanted to understand if the state space was restricted even before that, simply by the choice of the genetic code.

2 MARKOV CHAIN MODEL OF DNA AND PROTEIN SEQUENCES

The central goal as stated in the previous chapter, is to investigate the structure of the genetic code. This will be attempted in two different ways. One way is to choose a representation of the genetic code, and directly analyze its structure. This kind of analysis will be completely separate from biological systems and can be useful to understand general features of the code. Second way of investigating the structure of the code is to find a theoretical description of DNA and amino acid sequences and observe the change in the structure of amino acid sequences when one changes the genetic code. Then the statistical structure of the DNA is the free parameter of the model, genetic code is the control parameter and statistical structure of amino acid sequences is the observable. This is a more indirect approach but one can see its physical relevance immediately, since one does not observe some abstract property of the genetic code but its influence on the amino acid sequences. I followed both approaches during my thesis in parallel and in following chapters I will present the results of both analyses.
I will first describe the algorithms which generated the genetic codes and the indicators I used to distinguish between codes which where "close" or "far away" from the universal code.

2.1 STATE SPACE OF GENETIC CODES

The number of possible genetic codes with 21 amino acids is very large (see eqn.1.3), in order to restrict the state space I only used genetic codes with the same redundancy as the universal code. Meaning that each amino acid has the same number of codons, (but not necessarily the same ones) as the amino acids of the universal code.

number of amino acids	redundancy
2	1
9	2
2	3
5	4
3	6

Table 2.1.: Redundancy Table. First column of the table lists the number of amino acids with a given redundancy (number of codons) and the second column lists that redundancy. This is the table for the universal genetic code. **All artificial codes generated for this master thesis had the same redundancy as specified here**.

In order to generate alternative genetic codes, two algorithms were used.

1. First algorithm simply assigned codons to amino acids until no codons were left. Since the assigned codons were sampled from an uniform distribution, from now on I will call this genetic codes uniform for simplicity.

2. Second algorithm started from the universal code and permuted codons associated to amino acids (with number of permutations p being the free parameter).

2.1.1 *Uniform genetic codes*

The genetic code can be defined as a set of 64 ordered tuples[1] of codons ($\{c\}$) and their corresponding amino acids ($\{\sigma\}$)

$$\mathcal{G} = \{(c_\alpha, \sigma_\beta) | c_\alpha \neq c_{\alpha'} \forall \alpha \neq \alpha';\ \alpha, \alpha' \in [0, 63];\ \beta \in [0, 20]\}. \tag{2.1}$$

In my algorithms this definition was realized by a vector of length 64 which held numbers between 0 and 20. Such that the index of the vector represents one of the 64 codons and

1 the tuples must be ordered since the mapping from codon to amino acid must be unique but the opposite must not be true

the value at that index represents one of the 21 amino acids.[2] The steps of the generating algorithm are the following:

1. Take first of the 21 amino acids and assign it to a position in the vector.

   The position is chosen randomly from the uniform distribution over the positions (i.e. each free position in the vector has the same probability to be chosen).

2. repeat the first step $r-1$-times, where r is the redundancy of the amino acid.

3. Repeat previous steps for all remaining amino acids

This scheme will generate different codes if one changes the order of the amino acid names, but since we do not distinguish between them (besides their codon content), this codes are equivalent for our purposes. In order to avoid the generation of such equivalent codes I used the same ordering of amino acid names as in the universal genetic code. Hence phenylalanin (F) is the first amino acid leucine (L) is the second one and so on.

F	L	S	Y	*	C	W	P	H	Q	R	I	M	T	N	K	V	A	D	E	G

Table 2.2.: **Canonical order of amino acids**, i.e. same order as in the universal genetic code.

The uniform genetic codes generated in this manner superficially appear very different compared to the universal genetic code (see table2.3 for few examples). Since the state space is very large it is not surprising that such codes will exist and that they will have wildly different properties as the universal code. So this alone is not enough to assert if the universal code is special or not. What we need is an algorithm to generate genetic codes similar to the universal code and, we also need a way to somehow quantify differences between this ensembles.

2 From now on, for simplicity I will use the term 21 amino acids meaning 20 amino acids plus the stop command

FFLLSSSSYY**CC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
FL*SY*CWPHQRIIM*IWHHTSTWPPNKVTANDDCCSKCNIRYFELMRQSTTWSIASGIRVTNH
FLSYSCWPLH*QCLRHIIMRMQITNKVACDELPYPTTDGEPHIKLPCDMLP*CTGFWFM*ADCF
FLS*YCWPFHCQL*CYRCIMTTTPINYMKVLSTVYAADTRNQSEDQLYKRQN*HLMYMRTLSEG
FLSYCWPHQRIMPIHTFWHIIRFFLH*RTYNRYLTYCRKRNV*TKMSIYAQIVVDAKEGYVLD*

Table 2.3.: Several examples of **genetic codes generated by the uniform probability algorithm** (uniform codes, in black), compared to the universal genetic code (first string, in blue). Codes are represented as string of characters, were each character represents an amino acid and $*$ represents the Stop codon. Position of each character associates it with the one of the codons which are numbered between 0 and 63.

2.1.2 *Genetic codes derived from the universal code*

One easy way to generate codes which are intuitively close to the universal code, is through the permutations of vector elements. The algorithm is the following:

1. Choose two positions in the vector of the universal code at random.

   The elements of the vector at chosen positions must be distinct.

   Choose the vector indices to permute by sampling integers $i = 0 \ldots, 20$ from the uniform distribution.

2. Exchange the elements of the vector between the two chosen positions.

3. Repeat previous steps until the desired number of permutations p is reached. p is a free parameter of the algorithm.

4. The resulting vector is the new genetic code.

Unsurprisingly the genetic codes from this ensemble, for $p = 1$ look very similar to the universal code (see tab.2.4), and we expect them to have similar properties as the universal code. Further more when we increase the number of permutations we expect

FFLLSSSSYY**CC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
FFLLSSS*YY*SCC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
FFLLSSSSYY**CC*WLLLLPPPGHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGP

Table 2.4.: Two examples of **genetic codes derived from the universal code by the permutation scheme with** $p = 1$ (derived codes, in black), compared to the universal genetic code (first string, in blue). The permuted characters are highlighted in red. Codes are represented as string of characters, were each character represents an amino acid and $*$ represents the Stop codon. Position of each character associates it with the one of the codons which are numbered between 0 and 63..

this properties to deviate more from the universal code. One can also see that for $p \gg 1$ the characters in the resulting codes will be mixed so often that the information about the initial state will be lost completely. Since the positions for letter exchanges are also chosen from the uniform distribution one can expect that for $p \to \infty$ the first and second algorithm should generate code ensembles with same properties.
In order to illustrate this intuition little bit more we can use the weighted adjacency matrices associated to each code. I will elaborate on how to associate such a matrix to each code later in the chapter 3. All we need to know now is that for each code we can find such a matrix and if we can distinguish two codes then we also can distinguish their matrices. The eigenvalue density of matrices belonging to codes from the same ensemble (hence, codes with same permutation number) is the eigenvalue density distribution of this ensemble. These distributions can be used to see how related the ensembles are.

In fig.2.1 one can immediately see that the eigenvalue density of weighted adjacency matrices, of genetic codes with only one permutation are sharply peaked around the eigenvalues of the WAM associated to the universal code. This means that the WAM's from $p = 1$ ensemble mostly have eigenvalues very close to those of the universal genetic code. the eigenvalue density of WAM's of the ensemble with $p = 3$, still has some peaks around some of the eigenvalues of the universal code, but the weight of intermediate values is also much higher than in the $p = 1$ case. If one increases the number of permutations drastically, this peaks near the eigenvalues of the universal code disappear and the shape of the curve becomes more lake the shape of the eigenvalue density distribution of the WAM's associated to the uniform genetic codes (see fig.2.2).

We can conclude that, **the second of the previously presented two algorithms is able to generate genetic codes which are similar to the universal code. Furthermore we can change the relatedness of this codes to the universal code in a controlled manner by increasing the permutation number**. Thus although we do not have a generic measure of distance in the state space of genetic codes, we can systematically generate genetic codes

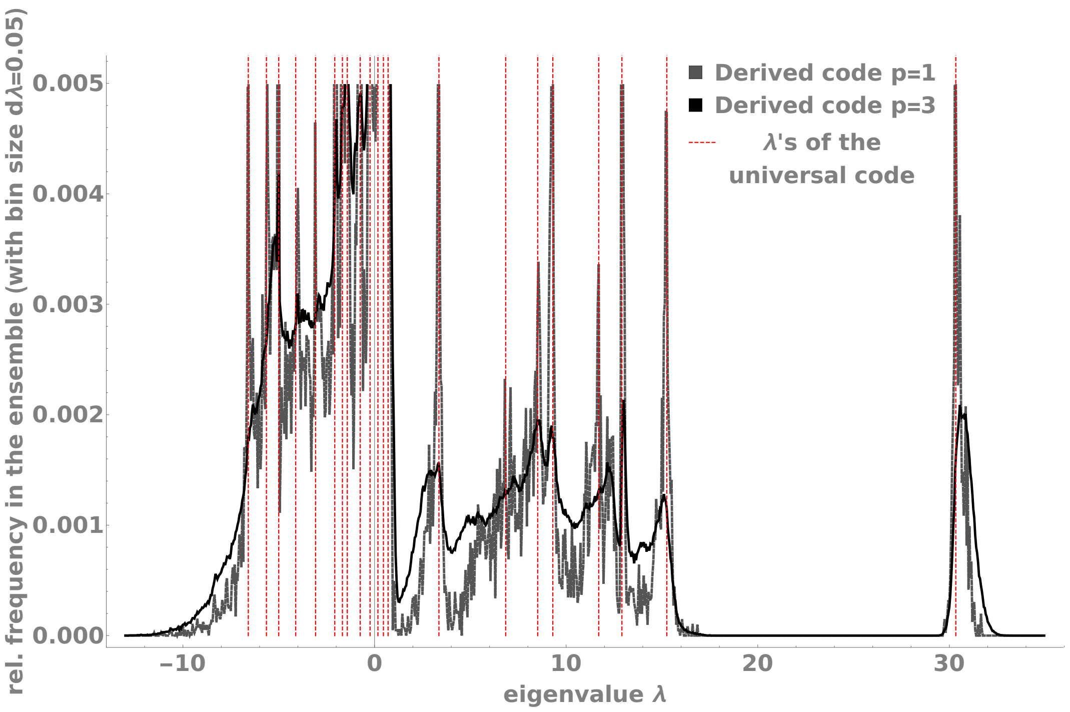


Figure 2.1.: Eigenvalue densities of weighted adjacency matrix (WAM) ensembles. Red dashed, vertical lines are the eigenvalues of the WAM associated to the universal genetic code, gray and black lines are the eigenvalue densities of the WAM's associated to the ensemble of codes generated with 1 and 3 permutations respectively. **The eigenvalue densities of WAM's associated with genetic codes with $p = 1$ peak around the eigenvalues of the universal code. This behavior is less prominent for genetic codes with $p = 3$.** The lines were generated by binning the eigenvalues from the ensemble, with bin size $d\lambda = 0.05$ and normalizing them by the number of bins. Hence the lines essentially represent a normalized histogram of the eigenvalues.

that are increasingly far removed from the universal code by increasing the number of permutations p. These "islands" in the state space of genetic codes serve as a tool that I will use in this and the following chapter to analyze the properties pf the universal code.

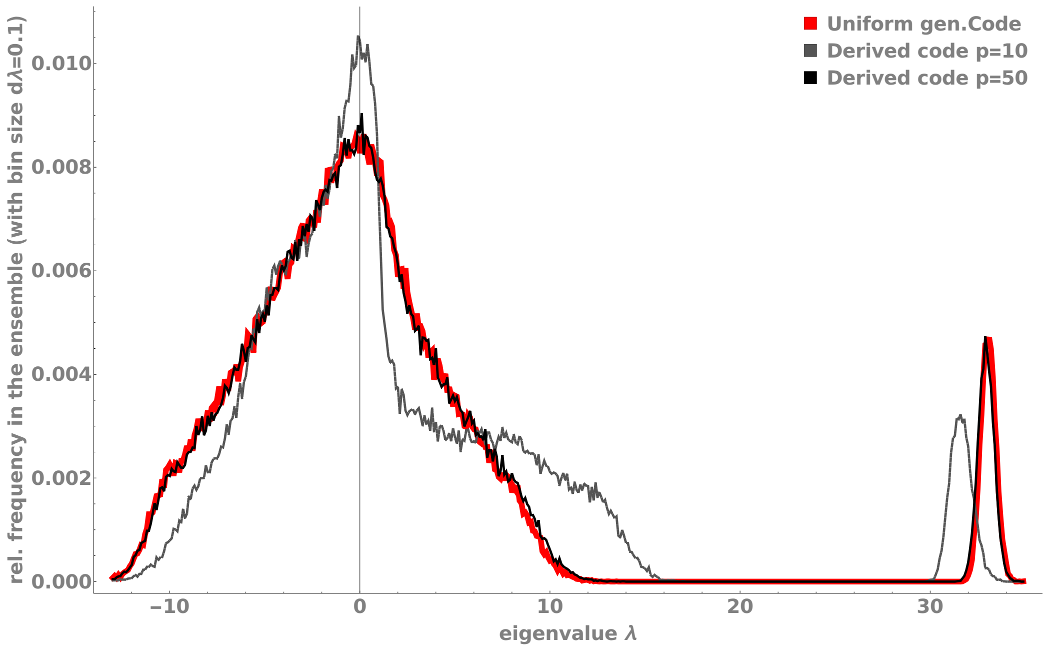


Figure 2.2.: Eigenvalue densities of weighted adjacency matrix (WAM) ensembles. Red thick line represents the eigenvalues of weighted adjacency matrix ensemble associated to uniform codes. gray and black lines are the eigenvalue densities of the WAM's associated to the ensemble of codes generated with 10 and 50 permutations respectively. **The latter two approach the eigenvalue distribution of the WAM's of the uniform codes.** The lines were generated by binning the eigenvalues from the ensemble, with bin size $d\lambda = 0.1$ and normalizing them by the number of bins. Hence the lines essentially represent a normalized histogram of the eigenvalues.

2.2 TRANSITION PROBABILITY MATRIX

In order to quantify what statistical structure of DNA and amino acids are, we need a model which provides us with a probability distribution for a given sequence. I treat DNA

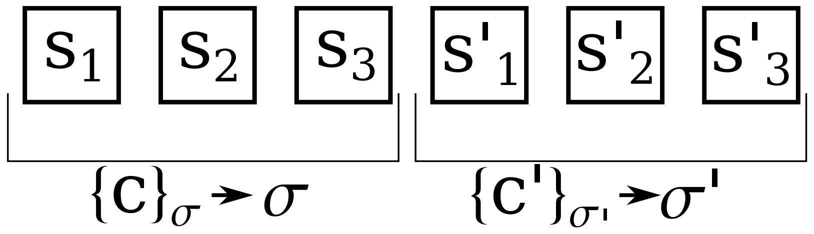


Figure 2.3.: DNA segment of two 3 letter groups, s_1, s_2, s_3 and s'_1, s'_2, s'_3 each group is translated to a codon c and c' respectively. This codons are part of a codon set of some amino acid σ (and σ'). The association of codon sets to amino acids is determined by the genetic code $\mathcal{G}$.

sequences $s = (s_0\ s_1\ \cdots\ s_L)$ as Markov chains of letters $s_i \in T, C, A, G$ in equilibrium, hence the probability transition matrix $P_{s,s'}$[3] fully describes the statistics of a sequence ensemble. The reasoning behind this choice of description is twofold. Firstly, description as Markov chains correspond to the next neighbor interaction model. Which is the lowest level of interacting models, it presents a suitable starting point for a description and can be gradually expanded in the strength of interactions if needed. Secondly one can associate to each real DNA sequence a Markovian Transition matrix, by simply counting the occurrence of the transitions in the sequences and then normalizing the "counted matrix" by its row sum. This provides us with an easy tool to numerically compare the predictions of the here presented analysis with the real biological systems. We see the transition probability matrix $P_{s,s'} = p(s'|s)$ of the DNA as the input parameter of our model. The quantity we are interested in is $\Pi_{\sigma,\sigma'} = \pi(\sigma'|\sigma)$ the transition probability matrix of the amino acids. in order to calculate $\mathbf{\Pi}$ we first write down the the codon transition probability $p(c'|c)$ probability. Since the DNA is modeled as a Markov chain in equilibrium, the probability of having any sequence $\boldsymbol{s}$ is given by

$$p(\vec{s}) = p(s_0, s_1, \ldots, s_L) = p^{st}(s_0) p(s_1|s_0) \cdots p(s_L|s_{L-1}), \tag{2.2}$$

where the stationary distribution is also determined by the Transition matrix $\boldsymbol{P}$ since it has to satisfy the left eigenvector equation (for the eigenvalue 1)

$$\boldsymbol{p}^{st} \cdot (\boldsymbol{P} - \mathbb{1}) = \mathbf{0}. \tag{2.3}$$

Hence the codon transition probability reads:

$$p(c'|c) = p(s'_1, s'_2, s'_3 | s_1, s_2, s_3) = p(s'_1, s'_2, s'_3 | s_3) = p(s'_1|s_3) p(s'_2|s'_1) p(s'_3|s'_2) \tag{2.4}$$

3 This matrix describes the probability that a letter s will be followed by the letter s'

with s_1, s_2, s_3 and s'_1, s'_2, s'_3 being the first second and third letters of c and c' respectively. Codons coding for an amino acid (at a given position), are mutually exclusive events[4] hence,

$$\pi(\sigma|\mathcal{G}) = p(\sum_{\{c\}_\sigma} c|\mathcal{G}) = \sum_{\{c\}_\sigma} p(c|\mathcal{G}), \tag{2.5}$$

where $\{c\}_\sigma$ represents the codon set of σ, determined by the genetic code $\mathcal{G}$(see eqn. 2.1).

$$\pi(\sigma'|\sigma) = \frac{\pi(\sigma,\sigma')}{\pi(\sigma)} \tag{2.6}$$

$$= \frac{p(\sum_{\{c\}_\sigma} c, \sum_{\{c'\}_{\sigma'}} c')}{p(\sum_{\{c\}_\sigma} c)} \tag{2.7}$$

$$\underbrace{=}_{\text{using 2.5}} \frac{\sum_{\{c\}_\sigma,\{c'\}_{\sigma'}} p(c,c')}{\sum_{\{c\}_\sigma} p(c)} \tag{2.8}$$

$$\underbrace{=}_{\text{using 2.4}} \frac{\sum_{\{c\}_\sigma,\{c'\}_{\sigma'}} p^{st}(s_1)p(s_2|s_1)p(s_3|s_2)p(s'_1|s_3)p(s'_2|s'_1)p(s'_3|s'_2)}{\sum_{\{c\}_\sigma} p^{st}(s_1)p(s_2|s_1)p(s_3|s_2)} \tag{2.9}$$

Before deriving this formula all analysis was done for the toy model (with binary DNA and codon length 2), since this restricted the number of possible genetic codes to 13, but unfortunately the toy model was too far from reality to make general inferences about the dependence of the probability structure of protein sequences on the genetic code (for a detailed discussion of the toy model see appendix A.3).
Having an equation for the amino acid transition matrix (for any given DNA transition matrix), enables us to observe the changes in sequence probability structure without the need to generate random DNA sequences and translate them into amino acids. This greatly reduces the time needed to compare the effects of different translations.
The quantity we want to compare between different genetic codes here is related to the Shannon entropy of the transition probability matrix [14].
Shannon entropy is a measure of information in the ensemble generated by the transition matrix and is defined in the following way

$$S[\boldsymbol{P}] = -\sum_{i,j=1}^{q} P_{ij}\log_2 P_{ij} \tag{2.10}$$

where q is the size of the square transition matrix ($q = 21$ for the amino acid transition matrix and $q = 64$ for the codon transition matrix). With shrinking entropy $S[\boldsymbol{P}]$ the statistical structure of the ensemble becomes simpler. $S[\boldsymbol{P}] = 0$ (is minimal) for $\boldsymbol{P} = \boldsymbol{I}$

4 If an amino acid at a position i is coded by a given codon then it obviously can not be coded by another codon at the same position simultaneously.

(where $\boldsymbol{I}$ represents the $q \times q$ identity matrix) and is maximized for $\boldsymbol{P} = \frac{1}{q}\mathbf{1}$, where $\mathbf{1}$ is a matrix of all ones (not the identity matrix). Hence the entropy is confined in the region

$$S[\boldsymbol{P}] \in [0, q \log_2 q]. \tag{2.11}$$

Since the entropy gives a measure of how diverse the generated ensemble can be we might ask;

- How does the average diversity of the ensemble depend on the genetic code?
- How does the change from the entropy of the codon transition matrix to the entropy of the amino acid transition matrix depend on the genetic code?

These questions do not directly relate to the robustness but, their answers can indicate if the universal genetic code can constraint protein sequences to a smaller subset than other possible codes, by reducing the entropy of the ensemble after translation. The value of interest is the relative entropy change before and after translation. Any given DNA transition matrix $\boldsymbol{P}$ will generate a specific codon transition matrix $\Theta_{c,c'}[\boldsymbol{P}] = p(c'|c)$, and then for each genetic code a different amino acid transition matrix $\Pi_{\sigma,\sigma'}[\boldsymbol{P}, \mathcal{G}] = \pi(\sigma'|\sigma)$. So the relative entropy change can be defined as:

$$S_{rel}[\boldsymbol{P}, \mathcal{G}] = \frac{S[\Pi[\boldsymbol{P}, \mathcal{G}]]}{S[\Theta[\boldsymbol{P}]]} \tag{2.12}$$

Since $\boldsymbol{P}$ is a 4×4 matrix with 12 free parameters it is in general not practical to make an exhaustively parameter sweep (with high resolution). I used a simplified subset of DNA transition matrices with $n_{par} = 4$ free parameters:

$$\boldsymbol{P} = \begin{pmatrix} p_{00} & \frac{1-p_{00}}{3} & \frac{1-p_{00}}{3} & \frac{1-p_{00}}{3} \\ \frac{1-p_{11}}{3} & p_{11} & \frac{1-p_{11}}{3} & \frac{1-p_{11}}{3} \\ \frac{1-p_{22}}{3} & \frac{1-p_{22}}{3} & p_{22} & \frac{1-p_{22}}{3} \\ \frac{1-p_{33}}{3} & \frac{1-p_{33}}{3} & \frac{1-p_{33}}{3} & p_{33} \end{pmatrix}, \tag{2.13}$$

and varied each parameter p_{ii} between $min = 0.3$ and $max = 0.7$ in steps of $\Delta = 0.2$, which gave a total of 81 transition matrices. Each of which was translated into amino acid transition matrices for different genetic codes from different ensembles(for more detail see tab.2.5).

	ensemble type	ensemble size
derived	$p = 1$	1927
	$p = 2$	9055
	$p = 4$	10000
	$uniform$	10000

Table 2.5.: Table of genetic code ensembles, which were used to generate the data for the fig.2.4. The size of the ensemble derived by 1 permutation is smaller because the generating algorithm is much more likely to generate same codes (randomly apply the same permutation several times in the row to the universal code) and since I only save unique codes the number of uniquely generated codes after 10000 attempts was only 1927 (same reasoning applies to the ensemble with $p = 2$).

The resulting relative entropies were then averaged over the input parameter space (hence $\boldsymbol{P}$), for each code type ensemble individually. The histogram of average relative entropies can be seen in the figure 2.4. This figure implies that when translating from the codon to the amino acid transition matrix, the average entropy decrease is strongest for codes which differ from the genetic code by one permutation. And the average relative entropy values increase with increasing permutation number. The figure 2.4 also shows that the tails of the distributions for higher permutation values are heavier (the distributions are broader), meaning that not only the average is higher but also the probability that the relative entropy change will be higher than that of the universal genetic code.
This results look promising, and imply that the universal genetic code is optimized for the reduction of the entropy of DNA ensemble after translation to proteins. This claim can be made since the reduction of the entropy on average is strongest for the universal genetic code and codes related to it by one permutation. There is a problem however. The form of the transition probability matrix is a special case of all possible transition matrices, and at this point it is not clear if this restriction is biologically justified. However the restriction is absolutely necessary since the number of transition matrices grows with the number of free parameters to the power of the number of parameter values

$$(n_{par})^{\frac{max-min}{\Delta}}. \tag{2.14}$$

The best course of action would be to use a diverse set of DNA sequences to determine a biologically relevant ensemble of DNA transition matrices and then repeat the relative entropy analysis for this dataset. Because of this problem I will refrain from quantitative analysis at this point.

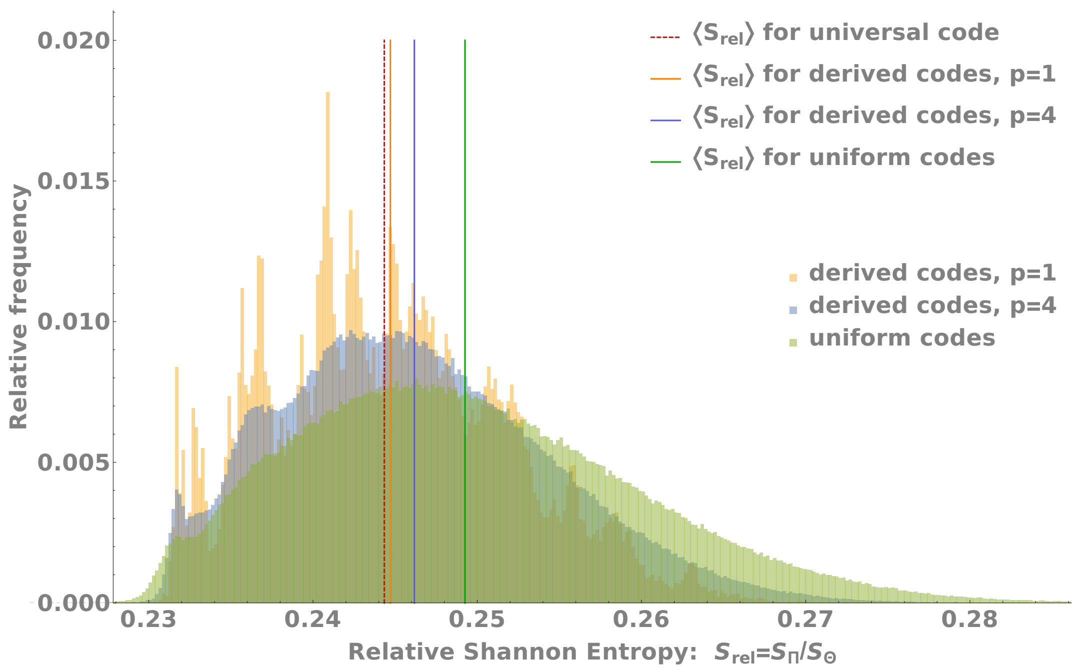


Figure 2.4.: Histogram of the relative Shannon entropy, for different ensembles of genetic codes after the averaging was performed over the parameter space(different DNA transition matrices), separately for each of the ensembles of the different genetic code types (listed in the table2.5). The vertical lines display the average values of resulting distributions (hence averaging $S_{rel}[\boldsymbol{P},\mathcal{G}]$ first over $\boldsymbol{P}$ and then over $\mathcal{G}$ for each of the genetic code ensembles).

In order to avoid this problem of too many free parameters, I will use a different approach in the next chapter. An approach which looks at the structure of the genetic code directly instead of looking at the effects of the genetic code on the translation.

3 GRAPH REPRESENTATION OF THE GENETIC CODE

3.1 WEIGHTED ADJACENCY MATRIX AND THE LOOPINESS INDEX

The genetic code can be represented in many ways, as one can see in tables A.1, A.2 and in figure 1.1. Each of this representations may be beneficial for solving some particular problem. But essentially they all convey the same amount of information they define a map from the set of codons to the set of amino acids, which can be represented as a set of ordered tuples (see eqn. 2.1). Since we are interested in quantifying the robustness of the genetic code it is interesting to know how easily one amino acid can change into another. Mutations on the DNA level can occur in different ways. During the duplication of DNA, errors can lead to replacement of nucleotides. Random deletion and insertion events can lead to frame shifts, and several such events can be combined[12]. I decided to model a single nucleotide replacement as the basic source of error. (assuming that it is the most common one during DNA replication, especially during the early period of evolution when the genetic code was still changing). In this setup the question about protein robustness becomes equivalent to asking how easily an amino acid can mutate into another, or how many silent mutations[1] are allowed in a given genetic code. To answer this questions the representation of the genetic code as a graph of amino acids like in fig.1.2 seems very natural.

In the graph from fig.1.2 each node represents an amino acid. Two amino acids are connected if a codon of one amino acid can be mutated into the codon of another one by a single nucleotide replacement. Since most of the amino acids are coded by more then one codon, there are several ways to mutate one amino acid into another. The edges then are weighted by the number of such possible mutations. In this picture the silent mutations are expressed as self loops in the graph, and they are also wighted by the number of possible silent mutations for each amino acid. If there is no way of changing one amino acid into another by single nucleotide replacement then the weight is 0 (hence there is no edge and similarly there is no self loop if an amino acid has only one codon or its codons are not

1 silent mutations are those which change a codon into another one coding for the same amino acid.

related by one mutation). Each graph is then best described by its weighted adjacency matrix (WAM) $\boldsymbol{W}$, with

$$W_{ij} = \text{Weight of the edge betwenn i and j} \tag{3.1}$$

Hence the genetic codes are in the end represented by a symmetric 21×21 matrix (see tab.A.3 and fig.A.1 for the WAM of the universal code). This allows us to use tools from linear algebra to analyze the structure of the genetic code. Since the main focus of the analysis is the robustness we are interested in characterizing the loop structure of the codes (hence the diagonal elements of the matrix). For this end I introduce a new quantity called average loopiness index of the graph, defined as:

$$\phi = \frac{1}{q} \sum_{i=1}^{q} \phi_i \tag{3.2}$$

where ϕ_i is the the vertex loopiness defined as the ratio of the loop weight to the total weight of all edges connected to the amino acid i, hence

$$\phi = \frac{1}{q} \sum_{i=1}^{q} \frac{W_{ii}}{\sum_{j=1}^{q} W_{ij}}. \tag{3.3}$$

Note that both indices are normalized, $\phi, \phi_i \in [0,1]$ with $\phi = 0$ if $\mathrm{Tr}[\boldsymbol{W}] = 0$ and $\phi = 1$ if $\boldsymbol{W}$ is diagonal.[2] One can interpret ϕ as a quantity which measures the average relative abundance of the silent mutations. Hence codes with larger ϕ will be more likely to translate two different DNA codes to the same protein.
We can determine the average loopiness for any genetic code directly. I used the algorithms described in te section 2.1 to generate ensembles of genetic codes with increasing relative closeness of the ensembles to the universal code. Then measured the average loopiness for each code and plotted the histograms for each ensemble. The results can be seen in fig.3.1 for a small number of different ensembles. In fig.3.2 one can see the distributions for 99 different ensembles of codes with number of permutations increasing from 1 to 99.

3.1.1 *Discussion*

In fig.3.2 the distribution of ϕ shifts to smaller values with increasing number of permutations towards the ensemble of uniform codes (this behavior is continued for higher values of permutations, see fig.3.2). Meanwhile the average loopiness of the universal

2 It is easy to see that $\phi = 1$ is not possible for the graphs of the genetic code since it would mean that each amino acid is immutable.

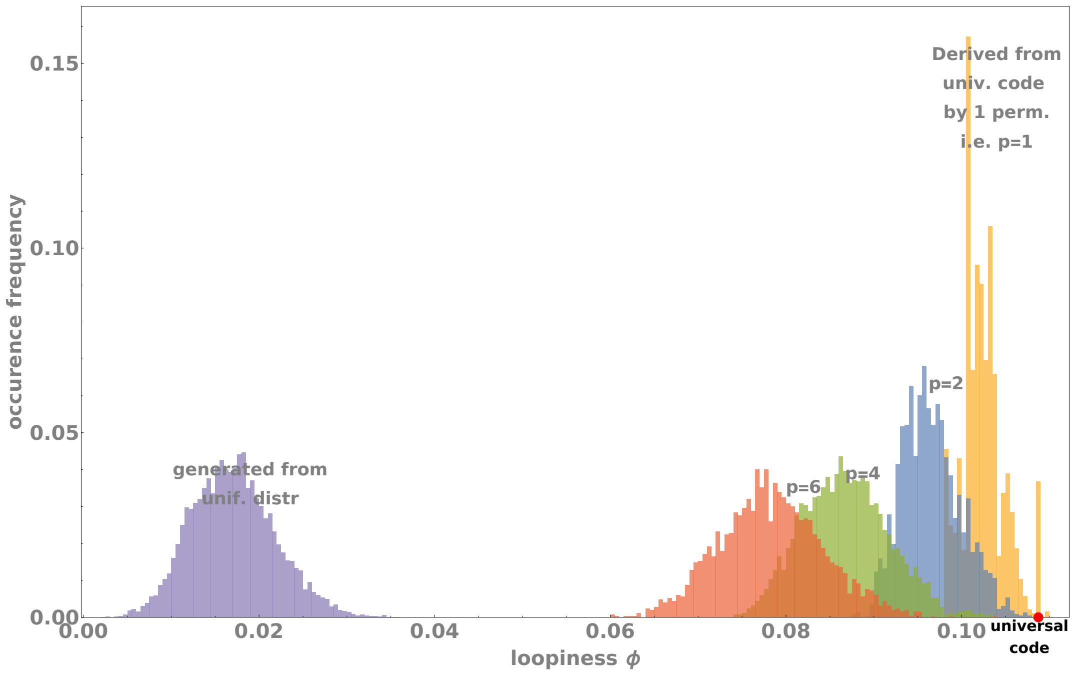


Figure 3.1.: Each distribution (of a different color), on the right side of the plot, represents an ensemble of average loopiness of genetic codes derived with different number of permutations from the universal code (the ensemble sizes are 1927, 90543, 99998 and 100000 for $p = 1, 2, 4, 6$ respectively. For the generating algorithm see subsection 2.1.2). The left most distribution was generated by the first algorithm (uniform codes, ensemble size is 10000. For the generating algorithm see subsection 2.1.1). **The average loopiness of the universal code ϕ_{uc} is displayed in the lower right corner of the plot by the red dot**.

code ϕ_{uc} has one of the largest values and can be seen in the right most corner of the plot. This is quite compelling evidence that the genetic code is optimized for robustness w.r.t. single nucleotide mutations in the DNA. In order to clearly see the shift of ϕ distributions towards smaller values, I calculated the expected value of the average loopiness $\langle\phi\rangle_p$. The average was taken over all ϕ 's in an ensemble of genetic codes for a given p (the ensemble data is the same as in the fig.3.2). Then I plotted $\langle\phi\rangle_p$ against the permutation number of the ensemble (see fig.3.3). We can see that the average value decreases fast in the beginning and then approaches a plateau. If we take the average of last 10 values from the plot, we can get an approximation for the values of $\langle\phi\rangle_p$ on the plateau. Comparing this value $\langle\phi\rangle_p^{plateu}$ with ϕ_{uc} shows us that the average loopiness is decreased roughly sixfold, when

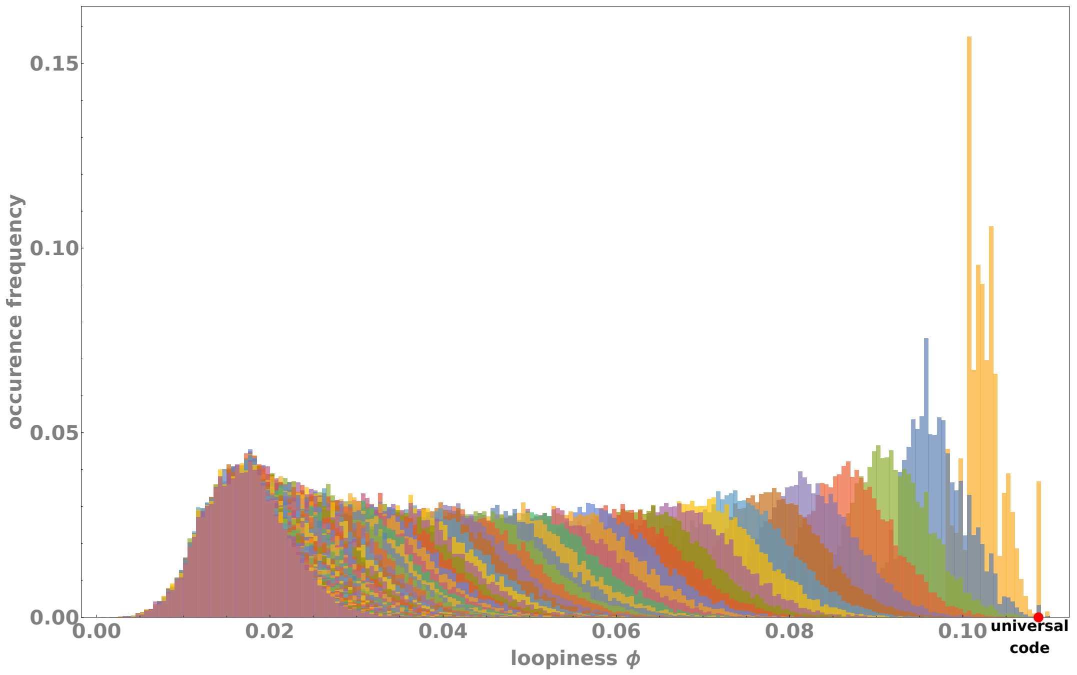


Figure 3.2.: Progression of ϕ distributions with increasing number of permutations in genetic codes, from right to left($p = 1, \ldots, 99$). All ensembles in this graph are generated by the second algorithm (see subsection2.1.2), through permutations of the universal code (The ensemble sizes are 1927, 19243 and 19980 for $p = 1, 2, 3$ respectively and 20000 for $p = 4, \ldots, 99$). **The average loopiness of the universal code ϕ_{uc} is displayed in the lower right corner of the plot by the red dot**.

we deviate fare from the universal genetic code through permutations of the codon content of amino acids.

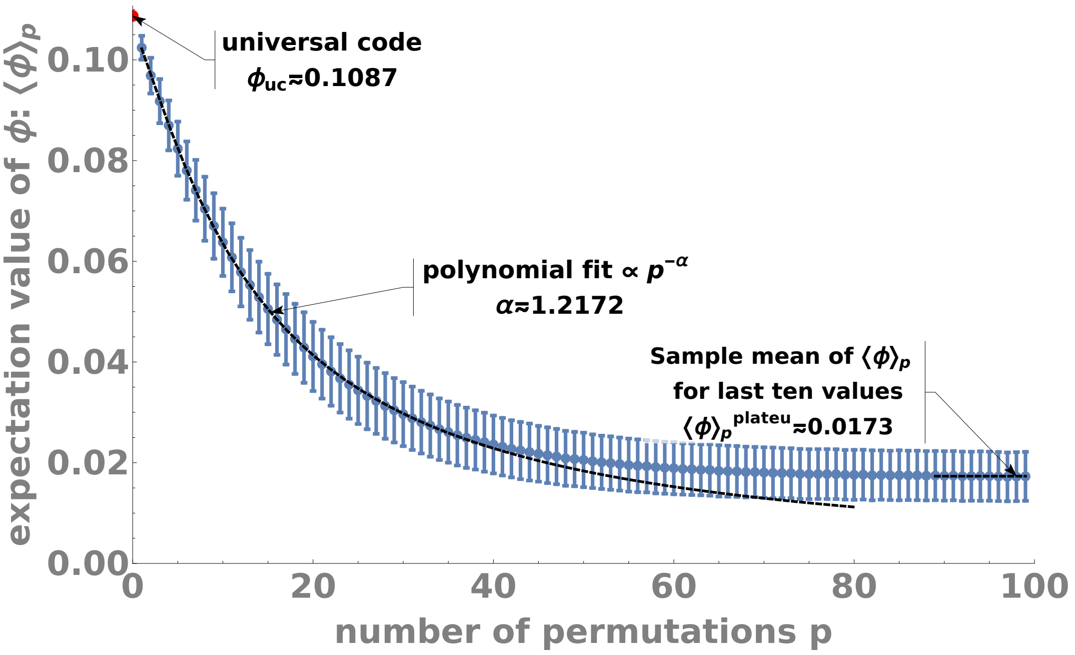


Figure 3.3.: Blue dots are the expected values of ϕ (ensemble average for each group of genetic codes, with different p). The error bars are the ensemble standard deviation. **The average loopiness of the genetic code has the largest value compared to the expected values of the average loopiness of any other ensemble**. I fitted the points between $p = 1$ and $p = 40$ with a polynomial function $\frac{c_1}{c_2 p^\alpha + 1}$, to get a sense of the speed of decrease. For small $c_2 p$ the exponent is roughly 1.2.

4 CONCLUSIONS AND OUTLOOK

The investigations of the structure of the universal code proofed quite difficult. In order to make any progress a variety of new approaches needed to be implemented. First step was to find an useful description of the code which was done by associating a weighted adjacency matrix to it. Since this matrices are high dimensional constructs it was initially not obvious how to explore the state space of possible alternative genetic codes in a controlled manner. This problem was eventually solved by introducing the permutation scheme described in the subsection 2.1.2, which turned out to be the most useful tool in later analysis. The problem with high number of free parameters was encountered again in the Markov matrix model of DNA and protein sequences. Here the problem could not be fully solved and I needed to resort to averaging out the initial parameters. The analysis however still revealed that the universal code and codes differing from it by one permutation, reduce the entropy of a DNA ensemble after translation to proteins, on average more then genetic codes further removed from the universal code (see fig.2.4). The lack of constraints on the parameter space of the model (Markov matrix of DNA $\boldsymbol{P}$) made it impossible to perform an in depth quantitative analysis, which would help to make more tangible and biologically relevant statements. However I developed an algorithm to assign a Markov matrix to any ensemble of DNA sequences which can be used to repeat the analysis for real biological systems and see if the results of chapter 2 can be replicated. In the end the approach to analyze the weighted adjacency matrices directly turned out to be most successful. The newly introduced measure for robustness, the average loopiness index ϕ was evaluated for a large number of genetic codes. The universal code proofed to be consistently better over the whole analyzed state space. In fact in fig.4.1 one can see the absolute and relative numbers of analyzed genetic codes which had a higher or equal value of ϕ. This number is highest for codes which were derived from the genetic code, though their percentage is only 3.8%, it decreases continuously for larger values of permutations and vanishes continuously for $p > 4$. For $p \leq 4$ the average loopiness index of the universal code is larger than the average loopiness indices of 99.96% of observed genetic codes. The largest value being $\phi_{max} \approx 0.1099$ which is only slightly larger (roughly 1.0109 times), than the value for the universal code ($\phi_{uc} \approx 0.1087$). A closer look at fig.3.3 also reveals that the average ϕ for ensembles decreases quickly with increasing permutations,

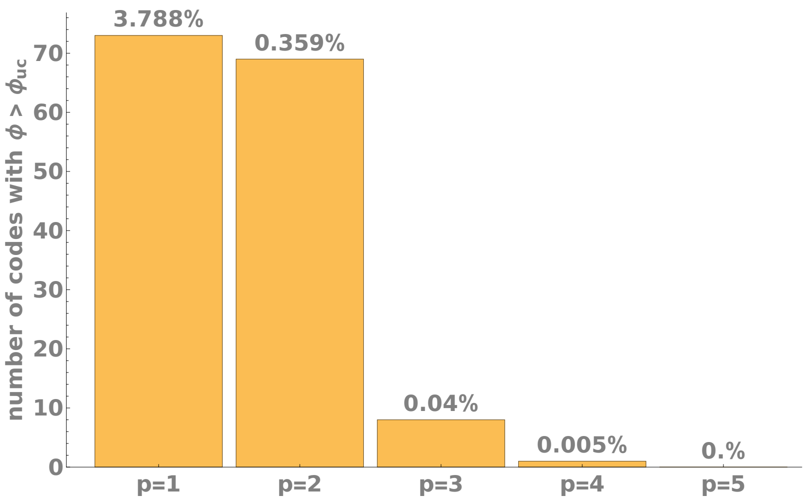


Figure 4.1.: Number of genetic codes which have larger or equal average loopiness then the universal code. The number of such codes is highest for the ensemble with $p = 1$ and decreases continuously. **There are no codes with greater or equal ϕ for ensembles with $p > 4$. For all codes from ensembles with $p \leq 4$ only a tiny amount of** 0.038% **have a larger loopiness then the universal code**.

meaning that the high value of average loopiness expressed by the universal code is rare and diminishes quickly when one deviates from it.
So one could answer the questions Q4 and Q4.1 posed in the beginning (see section 1.2) as follows. Concluding from the reviewed evidence it seems that the universal genetic code maximizes robustness against single nucleotide mutations in the DNA and facilitates largest reduction in the entropy of the DNA ensemble after translation to proteins (on average, compared to other genetic codes).

OUTLOOK

The best course of action to improve the Markov chain model further would be to describe the stop codon as the end state of the Markov chain, which would allow to describe protein sequences of varying length naturally[3, Ch.3]. In order to solve the problem with high parameter space one could apply the analysis of chapter 2 to real DNA ensembles. Most interesting case would be the genomes of the oldest known organisms, since it stands to reason that they are closest to the state of DNA sequences which an early genetic code(still

undergoing evolutionary changes) might have encountered.
In my opinion however a more promising alley of research would be the improvement of graph representation of the genetic code. In my discussion (see ch.3) I only used graphs which consider two codons as next neighbors if they can be mutated into each other by a single nucleotide replacement (see tab.1.1). As discussed, this is not the only kind of mutation which can occur in a DNA sequence. A simple extension would be to look at graphs where codons are related to each other by up to two nucleotide replacements, or consider a single nucleotide deletion which would shift the reading frame of the DNA strand. One could find next neighbors generated by this operation, for each codon and construct graphs based on this notion next neighbors. This would generate very different adjacency matrices and their analysis could reveal new information about the universal code.

A APPENDIX

A.1 GENETIC CODE, TABLES

This part of the appendix contains supplementary graphs and tables for diverse representations of the universal genetic code.

codon	a.a.	codon	a.a	codon	a.a	codon	a.a
AAA	K	CAA	Q	GAA	E	TAA	*
AAC	N	CAC	H	GAC	D	TAC	Y
AAG	K	CAG	Q	GAG	E	TAG	*
AAT	N	CAT	H	GAT	D	TAT	Y
ACA	T	CCA	P	GCA	A	TCA	S
ACC	T	CCC	P	GCC	A	TCC	S
ACG	T	CCG	P	GCG	A	TCG	S
ACT	T	CCT	P	GCT	A	TCT	S
AGA	R	CGA	R	GGA	G	TGA	*
AGC	S	CGC	R	GGC	G	TGC	C
AGG	R	CGG	R	GGG	G	TGG	W
AGT	S	CGT	R	GGT	G	TGT	C
ATA	I	CTA	L	GTA	V	TTA	L
ATC	I	CTC	L	GTC	V	TTC	F
ATG	M	CTG	L	GTG	V	TTG	L
ATT	I	CTT	L	GTT	V	TTT	F

Table A.1.: Universal genetic code represented as a list of pairs, of DNA codons and corresponding amino acids(1 letter notation). The asterisk (*) represents the stop command. Often this code is show as a list of pairs of RNA codons (see [1, Fig. 4.2]) and amino acids, That is to say that the DNA nucleotide thymine (*T*) is replaced by the RNA nucleotide uracil (*U*).

Amino acids 3 letter notation	Amino acids 1 letter notation	codons
Met	M	ATG
Trp	W	TGG
Tyr	Y	TAT, TAC
Phe	F	TTT, TTC
Cys	C	TGT, TGC
Asn	N	AAT, AAC
Asp	D	GAT, GAC
Gln	Q	CAA, CAG
Glu	E	GAA, GAG
His	H	CAT, CAC
Lys	K	AAA, AAG
Ile	I	ATT, ATC, ATA
Gly	G	GGT, GGC, GGA, GGG
Ala	A	GCT, GCC, GCA, GCG
Val	V	GTT, GTC, GTA, GTG
Thr	T	ACT, ACC, ACA, ACG
Pro	P	CCT, CCC, CCA, CCG
Leu	L	CTT, CTC, CTA, CTG, TTA, TTG
Ser	S	TCT, TCC, TCA, TCG, AGT, AGC
Arg	R	CGT, CGC, CGA, CGG, AGA, AGG
Stop	*	TAA, TAG, TGA

Table A.2.: Universal genetic code represented as a list of amino acids with corresponding lists of codons, which encode them in the DNA.

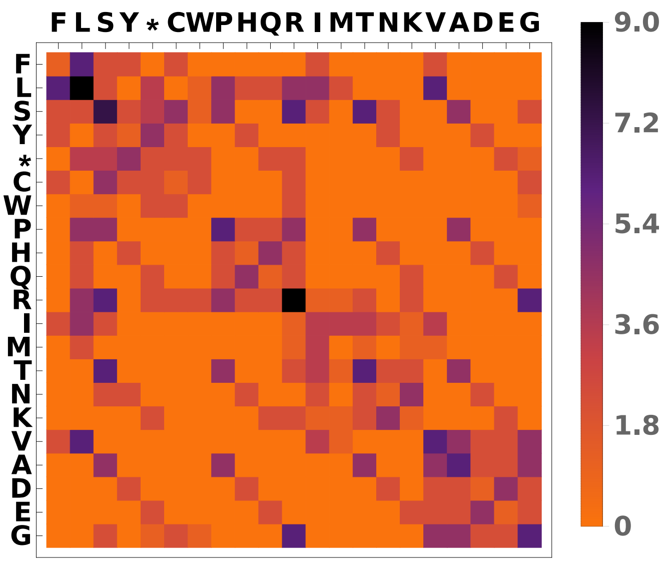


Figure A.1.: Heat-map plot of the weighted adjacency matrix of the universal code.

	F	L	S	Y	*	C	W	P	H	Q	R	I	M	T	N	K	V	A	D	E	G
F	1	6	2	2	0	2	0	0	0	0	0	2	0	0	0	0	2	0	0	0	0
L	6	9	2	0	3	0	1	4	2	2	4	4	2	0	0	0	6	0	0	0	0
S	2	2	7	2	3	4	1	4	0	0	6	2	0	6	2	0	0	4	0	0	2
Y	2	0	2	1	4	2	0	0	2	0	0	0	0	0	2	0	0	0	2	0	0
*	0	3	3	4	2	2	2	0	0	2	2	0	0	0	0	2	0	0	0	2	1
C	2	0	4	2	2	1	2	0	0	0	2	0	0	0	0	0	0	0	0	0	2
W	0	1	1	0	2	2	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1
P	0	4	4	0	0	0	0	6	2	2	4	0	0	4	0	0	0	4	0	0	0
H	0	2	0	2	0	0	0	2	1	4	2	0	0	0	2	0	0	0	2	0	0
Q	0	2	0	0	2	0	0	2	4	1	2	0	0	0	0	2	0	0	0	2	0
R	0	4	6	0	2	2	2	4	2	2	9	1	1	2	0	2	0	0	0	0	6
I	2	4	2	0	0	0	0	0	0	0	1	3	3	3	2	1	3	0	0	0	0
M	0	2	0	0	0	0	0	0	0	0	1	3	0	1	0	1	1	0	0	0	0
T	0	0	6	0	0	0	0	4	0	0	2	3	1	6	2	2	0	4	0	0	0
N	0	0	2	2	0	0	0	0	2	0	0	2	0	2	1	4	0	0	2	0	0
K	0	0	0	0	2	0	0	0	0	2	2	1	1	2	4	1	0	0	0	2	0
V	2	6	0	0	0	0	0	0	0	0	0	3	1	0	0	0	6	4	2	2	4
A	0	0	4	0	0	0	0	4	0	0	0	0	0	4	0	0	4	6	2	2	4
D	0	0	0	2	0	0	0	0	2	0	0	0	0	0	2	0	2	2	1	4	2
E	0	0	0	0	2	0	0	0	0	2	0	0	0	0	0	2	2	2	4	1	2
G	0	0	2	0	1	2	1	0	0	0	6	0	0	0	0	0	4	4	2	2	6

Table A.3.: Weighted adjacency matrix of the graph representation of the universal genetic code.

The number of all genetic codes is related to the number of partitions of 43 into at most 21 integers which is 59755. In order to make the derivation steps of the eqn.1.2 more illustrative, I will start with the example of 8 codons which need to be assigned to 4 amino acids.

First such assignment would be 5 codons to one amino acid and the remaining 3 codons to the one amino acid each, like depicted in fig.A.2a. Determining the last 3 assignments (i.e. n_1) also completely determines the configuration. We also have to remember that the renaming of amino acids does not change the model, hence after we have chosen 3 codons it does not matter to which amino acids they get assigned to (hence the factor 1/3!)

$$C_a = n_1 \times \frac{1}{3!} = 8 \times 7 \times 4 \times \frac{1}{3!} = \binom{8}{1}\binom{7}{1}\binom{6}{1}\frac{1}{3!} = 56 \tag{A.1}$$

To determine the configuration in the second type of assignment seen in fig.A.2b, we need to find out the number of possible assignments for last two codons divided by their degeneracy, for each of those assignments we also have assignments of two codons to one amino acid, hence

$$C_b = n_2 \times n_3 \times \frac{1}{2!} = \binom{8}{1}\binom{7}{1}\binom{6}{2}\frac{1}{2!} = 420 \tag{A.2}$$

Through analogous argumentations for the remaining tableau's we can see that:

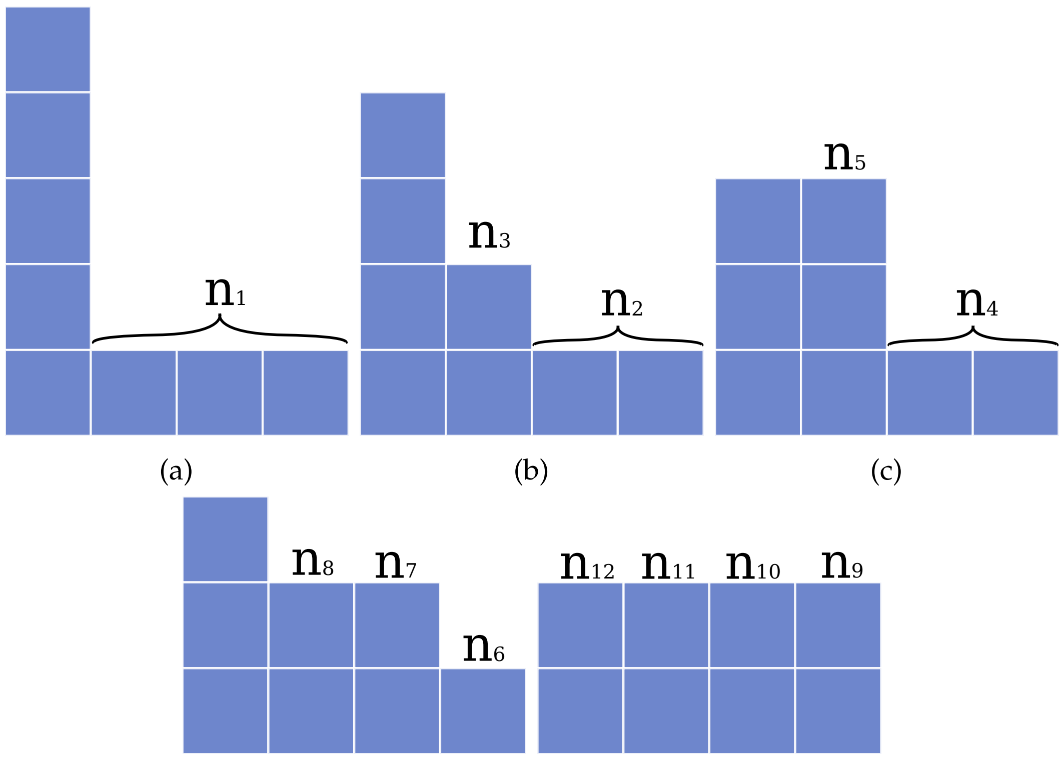
$$C_c = n_4 \times \frac{1}{2!} \times n_5 \times \frac{1}{2!} = \binom{8}{1}\binom{7}{1}\binom{6}{3}\frac{1}{4} = 280 \tag{A.3}$$

$$C_d = n_6 \times n_7 \times n_8 \times \frac{1}{2!} = \binom{8}{1}\binom{7}{2}\binom{5}{2}\frac{1}{2} = 840 \tag{A.4}$$

$$C_e = n_9 \times n_{10} \times n_{11} \times n_{12} \times \frac{1}{4!} = \binom{8}{2}\binom{6}{2}\binom{4}{2}\frac{1}{24} = 105 \tag{A.5}$$

And finally we can sum over all C_i to get the total number of configurations. The calculation for the universal genetic code is analogous. The number of choices in the last position is h_{21} out of 64 codons, h_{20} out of $64 - h_{21}$ for the next and so forth, giving:

$$\begin{aligned} C(\{h_i\}) &= \frac{1}{\mathcal{N}}\binom{64}{h_{21}}\binom{64-h_{21}}{h_{20}}\cdots\binom{64-h_{21}-\ldots-h_3}{h_2} && \text{(A.6)}\\ &= \frac{1}{\mathcal{N}}\frac{64!}{h_{21}!(64-h_{21})!}\frac{(64-h_{21})!}{h_{20}!(64-h_{21}-h_{20})!}\cdots\frac{(64-h_{21}-\ldots-h_3)!}{h_2!(64-h_{21}-\ldots-h_2)!} && \text{(A.7)}\\ &= \frac{1}{\mathcal{N}}\frac{64!}{\prod_{i=1}^{21} h_i!} && \text{(A.8)} \end{aligned}$$



(d)

(e)

Where $\frac{1}{\mathcal{N}}$ is the normalization factor which counts states which are degenerate with respect to amino acid renaming. Since all we are interested in is the codon content of amino acids, we have to divide by the number of possible name permutations (renamings), which do not change the code, hence renaming between amino acids with same codon content (i.e. same height h_i). So we arrive at the correct normalization factor, pairwise comparing all heights:

$$\mathcal{N} = \prod_{i=1}^{21} \left(\sum_{j>1}^{21} \delta_{h_i,h_j} + 1 \right) \tag{A.9}$$

Combining everything we arrive at the eqn.1.2. In order to calculate each of the 59755 different height configurations I used the **IntegerPartitions** function from Mathematica to calculate the partitions of 43 into at most 21 integers. Then added each partition element wise to a vector of ones (of length 21). this provided 59755 vectors of length 21 where each vector corresponded to a height configuration $\{h_i\}_i$ with $i-$th element of the vector corresponding to h_i. Using a Mathematica implementation of the eqn.1.2 I performed the sum over all vectors and arrived at the result from eqn.1.3.

In order to reduce the number of possible genetic codes one can look at simplified toy model, of binary DNA sequences

$$\boldsymbol{s} = \{s_0, \ldots, s_N\}, \tag{A.10}$$

with

$$s_i \in \mathcal{A}_s = \{0, 1\}. \tag{A.11}$$

And codons with two letters ($\ell_c = 2$). Hence we get 4 codons $\{00, 01, 10, 11\}$. This codons can be grouped in 13 different distinct ways to form degenerate codon to amino acid maps (see A.5)
We assume only next neighbor interactions between DNA nucleotides, hence model the binary sequences as Markov Chains which have been known to be useful in modeling regions of DNA[3, ch.3.1]. This means that the sequence is fully characterized by its transition probability matrix, defined as

$$\boldsymbol{P} = \begin{pmatrix} p_{00} & 1 - p_{00} \\ 1 - p_{11} & p_{11} \end{pmatrix}, \tag{A.12}$$

Where $p_{ss'}$ is the probability that s' will be observed in a sequence at position $i+1$ if we observed s at the position i.[1] Hence

$$p_{ss'} = p(s'|s), \tag{A.13}$$

or in braket notation

$$p(s'|s) = \langle s' | \vec{P} | s \rangle . \tag{A.14}$$

The statistics of the sequence ensemble is fully characterized if we can write down the probability distribution $p(\boldsymbol{s})$ for any arbitrary sequence $\boldsymbol{s}$. If we know the starting letters s_0 of the sequence we can use the transition probabilities (and the fact that the system is Markovian) to characterize the full probability of any sequence $\boldsymbol{s}$ of length L.

$$p(\boldsymbol{s}|s_0) = p(s_L|p_{L-1}) \cdots p(s_1|s_0) \tag{A.15}$$

This can be rewritten in terms of the transition probability matrix using A.14,

$$p(\boldsymbol{s}|s_0) = \langle s_0 | \boldsymbol{P} | s_1 \rangle \cdots \langle s_{L-1} | \boldsymbol{P} | s_L \rangle \tag{A.16}$$

1 In this model any given nucleotide only depends on its left neighbor, this is to reflect the $3'$ to $5'$ assembly of DNA. Since DNA strands are assembled from left to right, a probability of nucleotide occurrence can not depend on its right neighbor, since it is not part of the sequence yet. However when we look at the sequences in equilibrium, the interactions appear symmetric.

Multiplying both sides by $p(s_0)$ and using the product rule gives us the following expression for the full probability of a given sequence:

$$p(\boldsymbol{s}) = p(s_0) \langle s_0 | \boldsymbol{P} | s_1 \rangle \cdots \langle s_{L-1} | \boldsymbol{P} | s_L \rangle \tag{A.17}$$

We assume that the system is in stationary state meaning that:

$$p(s_i) = p^{st}(s_i) = \langle p^{st} | s_i \rangle \quad \forall s_i \tag{A.18}$$

where, $|p\rangle^{st}$ is the stationary probability, and $|s\rangle$ is the unit vector (in $|\mathcal{A}_s|$ dimensions) with only one nonzero entry, representing the values of the alphabet, i.e.

$$|s\rangle \in \{ |\hat{e}_\alpha\rangle \, | \alpha \in [1, |\mathcal{A}_s|] \} \tag{A.19}$$

From this condition follows that

$$\langle s_i \rangle = \langle p^{st} | s_i \rangle \quad \forall s_i \tag{A.20}$$

since,

$$\begin{aligned} \langle s_i \rangle &= \sum_{s_0,\dots,s_L} |s_i\rangle \langle p^{st} | s_0 \rangle \langle s_0 | \boldsymbol{P} | s_1 \rangle \cdots |s_i\rangle \langle s_i | \boldsymbol{p} \cdots \langle s_{L-1} | \boldsymbol{p} | s_L \rangle & \text{(A.21)} \\ &= \sum_{s_i, s_L} |s_i\rangle \langle p^{st} | \boldsymbol{P}^i | s_i \rangle \underbrace{\langle s_i | \boldsymbol{P}^{L-i} | s_L \rangle}_{(*)}, & \text{(A.22)} \end{aligned}$$

the expression $(*)$ evaluates to one,

$$\begin{aligned} \sum_{s_L} (*) &= \sum_{s_L} \langle s_i | \boldsymbol{P}^{L-i} | s_L \rangle & \text{(A.23)} \\ &= \langle s_i | \sum_{\alpha\beta} P_{\alpha\beta} | \hat{e}_\alpha \rangle & \text{(A.24)} \\ &= \sum_{\alpha} \langle s_i | \hat{e}_\alpha \rangle = 1 \;\; \forall \, i, & \text{(A.25)} \end{aligned}$$

in eqn. A.24 and I used A.19 and the fact that the row sum of the Markovian Probability matrix is 1. Inserting this back in eqn.A.21, as well as using the fact that the stationary distribution of the Markovian probability matrix is its left eigenvector leads to

$$\langle s_i \rangle = \sum_{s_0} |s_i\rangle \langle p^{st} | s_0 \rangle = \sum_{s_0} |s_i\rangle \langle s_0 | p^{st} \rangle = |s_i\rangle \,. \tag{A.26}$$

Similarly we can derive the expression for the covariance as

$$\langle s_i s_{i+l} \rangle = \sum_{s_i, s_{i+l}} \underbrace{\langle s_i | \, s_{i+l} \rangle}_{\delta_{s_i, s_{i+l}}} p^{st}(s_i) \, \langle s_i | \, \boldsymbol{P}^l \, | s_{i+l} \rangle \tag{A.27}$$

$$= \sum_{s_i, s_{i+l}} p^{st}(s_i) \, \langle s_i | \, \boldsymbol{P}^l \, | s_i \rangle \tag{A.28}$$

$$= \sum_{\alpha}^{|\mathcal{A}_s|} \boldsymbol{p}^{st}{}_{\alpha} (\boldsymbol{P}^l)_{\alpha\alpha}, \tag{A.29}$$

where $\boldsymbol{p}^{st} \equiv \left| p^{st} \right\rangle$. All previously derived equations(besides A.12) are applicable to a model with any size of the alphabet, but here we are interested in binary DNA, i.e.

$$|s = 0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{A.30}$$

$$|s = 1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{A.31}$$

The stationary probability can be derived by solving the equation

$$\left\langle p^{st} \right| \boldsymbol{P} = \left\langle p^{st} \right| . \tag{A.32}$$

For the binary alphabet we get the solution

$$\left| p^{st} \right\rangle = \frac{1}{p_{00} + p_{11} - 2} \begin{pmatrix} p_{11} - 1 \\ p_{00} - 1 \end{pmatrix} . \tag{A.33}$$

In the end we can see that the probability distribution of the DNA sequences is fully parametrized by p_{00} and p_{11}. A quantity which is very easy to analyze for this system is a typical correlation length. Since the correlation length is just one scalar, associated to any DNA ensemble characterized py the probability distribution $p(\boldsymbol{s}, p_{00}, p_{11})$, we can make 3D plots and observe the distribution of the correlation length $\xi(p_{00}, p_{11})$ in the whole parameter space of DNA (see fig.A.4). Then we can apply each of the 13 genetic codes and see how the functional form of the correlation length changes depending on the code fig.A.5. One can see from the figures A.5 that not each genetic code has an unique impact on the correlation landscape and graphs which are related by the exchange symmetry of the letters $0, 1$ have correlation landscapes which are also related by flipping the p_{00} and p_{11} axes (e.g. A.5a and A.5f or A.5b and A.5e).
This model also provides a good opportunity to test the analytic equation for the amino acid level correlations (eqn.2.4), for the full parameter space of the model and an all possible translations. Figure A.3 shows that the scatter plot of elements of the theoretically predicted and measured transition probability matrices. All points lie on the diagonal. Meaning that the theoretically predicted and measured values are the same.

Unfortunately the toy model is too simple to make any quantitative analysis of the impact of genetic code on the statistical structure, hence the dependence of the transition matrix $\boldsymbol{P}$ on the weighted adjacency matrix $\boldsymbol{W}$ of the graphs. But two useful observations can be made,

- statistical properties of genetic codes are related to the structure of the genetic code graphs (e.g. symmetries of the code are reflected in symmetries of the correlation landscape).
- the analytically derived values of the Transition matrices are in excellent agreement with the measured values from the simulated sequences.

A.3.1 *Simulation parameters and figures*

In simulations a parameter sweep between $(p_{00}, p_{11}) = (0.05, 0.05)$ and $(p_{00}, p_{11}) = (0.95, 0.95)$ was made, with a step size of $dp = 0.05$ for each parameter. For each of 400 pairs of points an ensemble of $N = 32768$ binary DNA sequences with length $L = 14$ was generated, and then translated with each of the 13 genetic codes of the model. The probability transition matrices and correlation length were measured for each of these ensemble and are presented in the figures of this section. For the calculation of the correlation length I used the Pearson correlation function defined as:

$$C(s_i, s_j) = \frac{\langle s_i s_j \rangle - \langle s_i \rangle \langle s_j \rangle}{\sqrt{Var(s_i)Var(s_j)}} \tag{A.34}$$

the expressions for covariance and average value of a nucleotide have already been derived for the model (see equations A.20 and A.27), using the definition of the variance $\langle s_i^2 \rangle - \langle s_i \rangle^2$ and using the expression for covariance with $i = j$, we arrive at the formula for the correlation function between two nucleotides which are l sites apart

$$C(l) = \frac{\sum_{\alpha}^{|\mathcal{A}_s|} \boldsymbol{p}^{st}{}_{\alpha} (\boldsymbol{P}^l)_{\alpha\alpha} - (\boldsymbol{p}^{st})^2}{1 - (\boldsymbol{p}^{st})^2}. \tag{A.35}$$

The correlation length is taken to be

$$\xi(\boldsymbol{P}) = \frac{-1}{\log[C(1)]}. \tag{A.36}$$

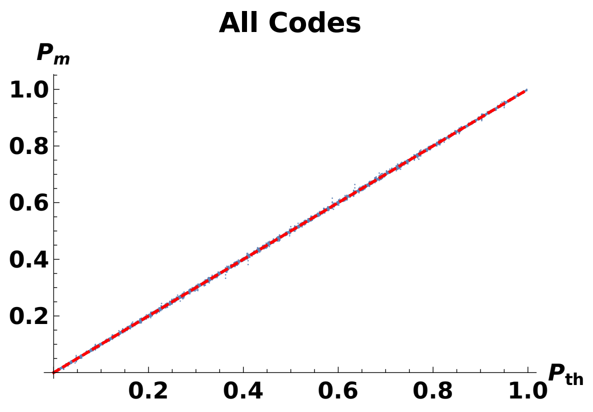


Figure A.3.: Blue dots represent the scatter plot of elements of the theoretically predicted and measured transition probability matrices. Red dashed line is the expected position of the dots in case of a good agreement.

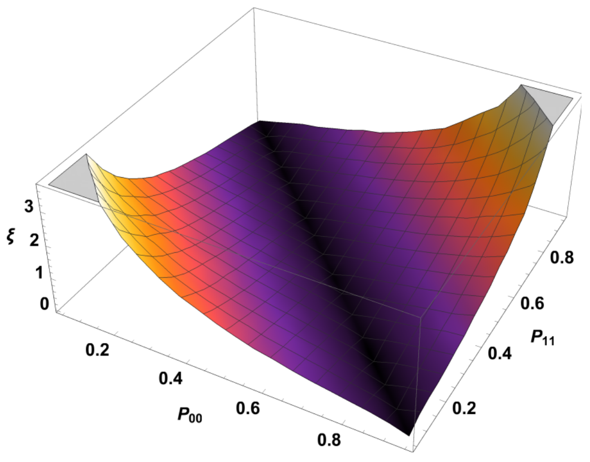
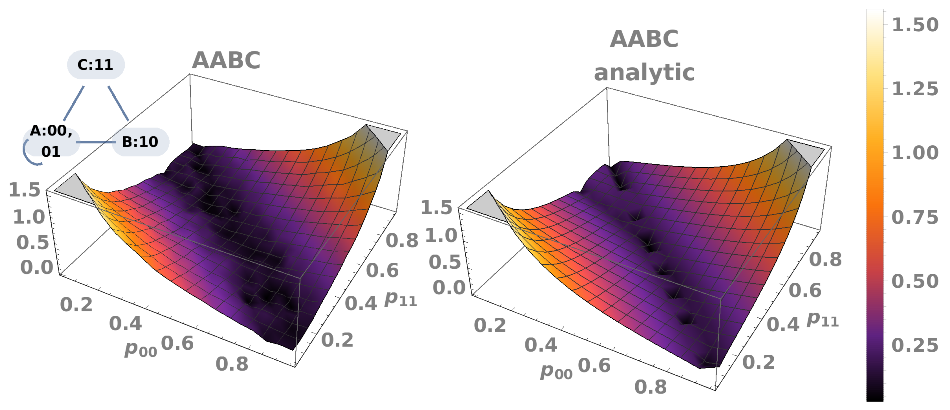
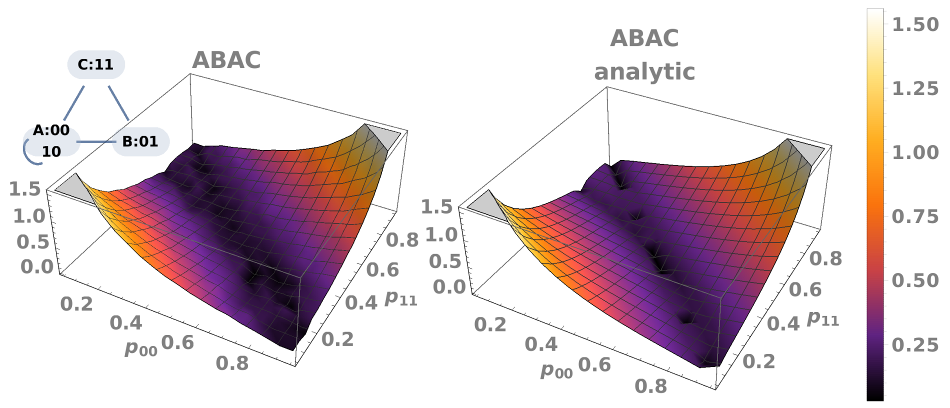


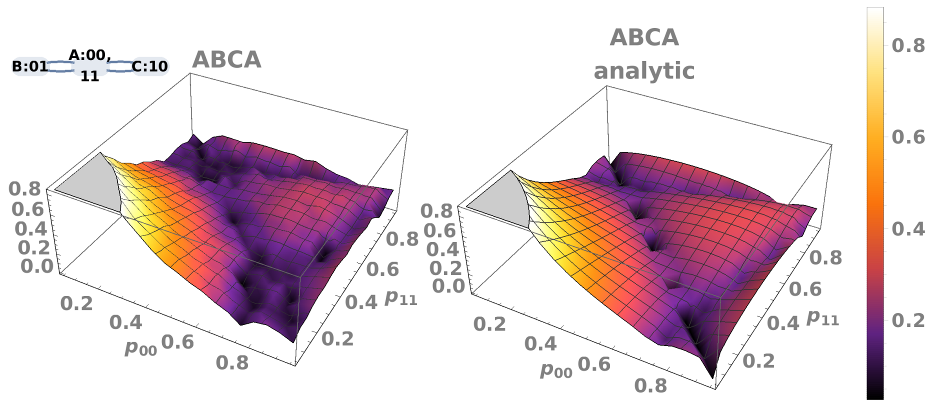
Figure A.4.: Measured Correlation length of binary DNA


C:11
A:00, 01
B:10
AABC
AABC analytic
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
p_{00}
p_{11}
1.50
1.25
1.00
0.75
0.50
0.25

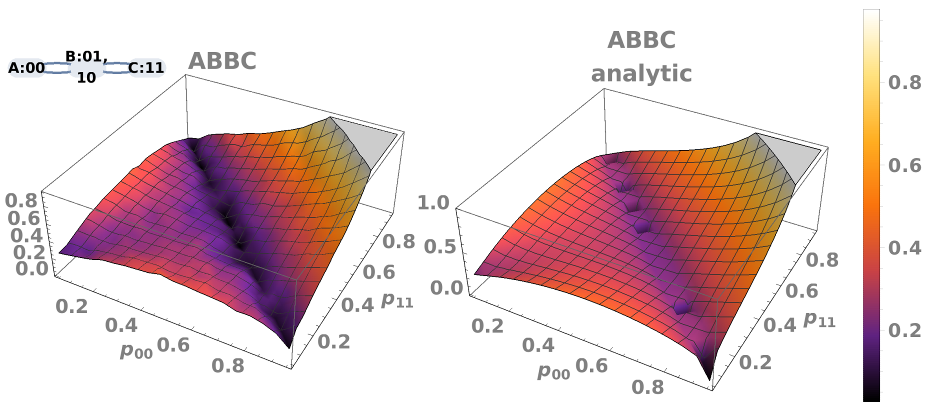
(a)


C:11
A:00 10
B:01
ABAC
ABAC analytic
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
p_{00}
p_{11}
1.50
1.25
1.00
0.75
0.50
0.25

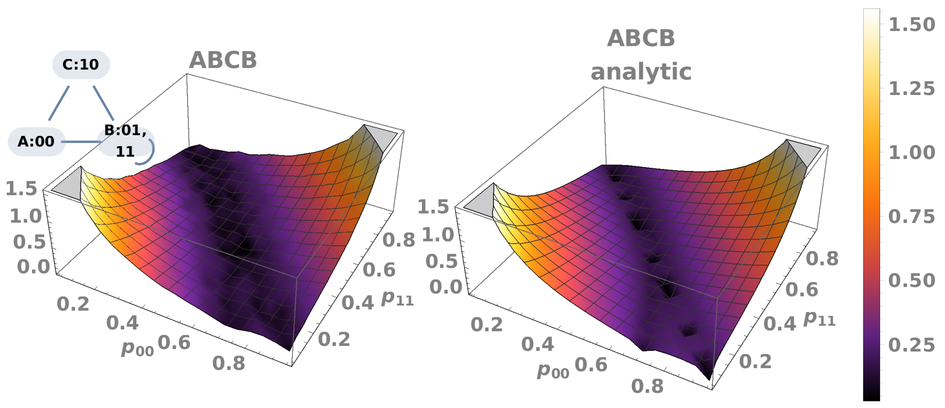
(b)


B:01
A:00, 11
C:10
ABCA
ABCA analytic
0.8
0.6
0.4
0.2
0.0
p_{00}
p_{11}

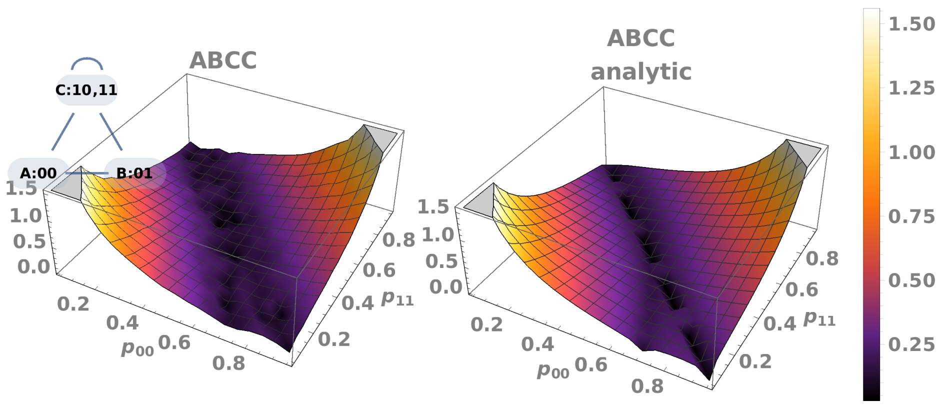
(c)


A:00
B:01, 10
C:11
ABBC
ABBC analytic
p_{00}
p_{11}

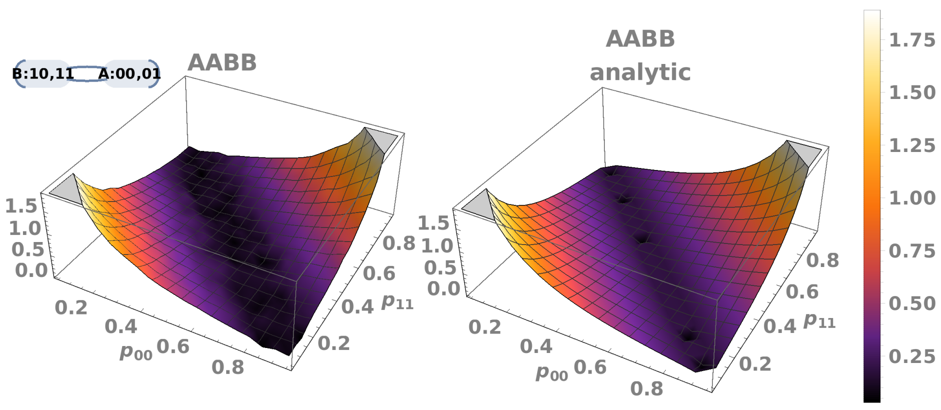
(d)


C:10
A:00
B:01, 11
ABCB
ABCB analytic
p_{00}
p_{11}

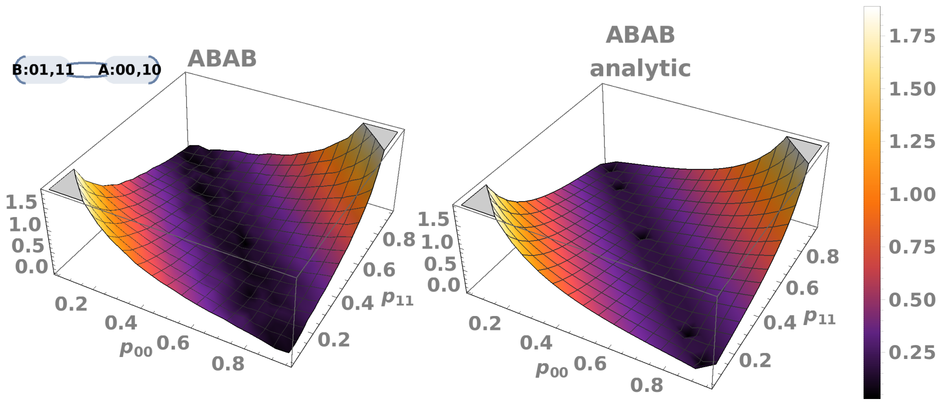
(e)


C:10,11
A:00
B:01
ABCC
ABCC analytic
p_{00}
p_{11}

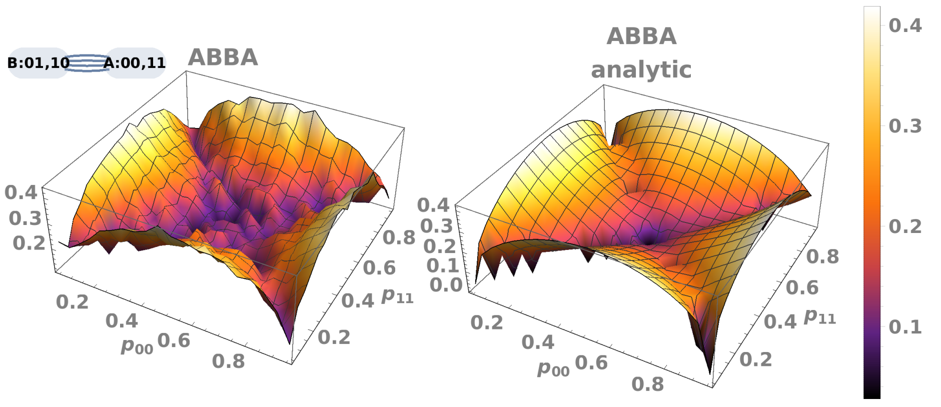
(f)


B:10,11
A:00,01
AABB
AABB analytic
p_{00}
p_{11}

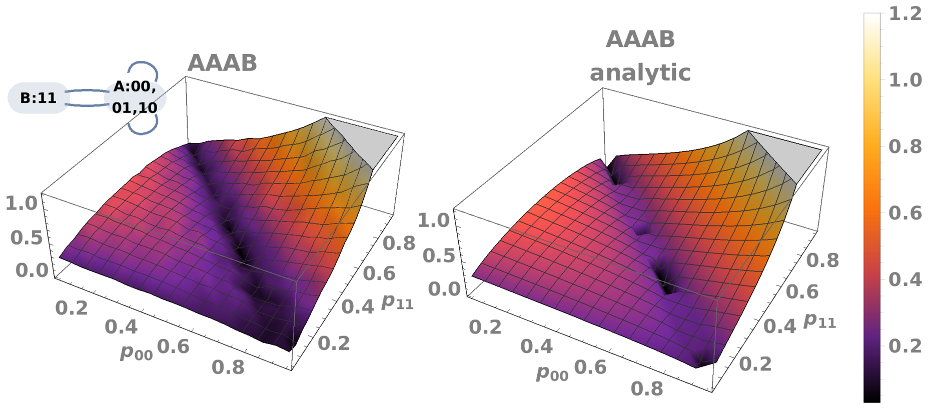
(g)


B:01,11
A:00,10
ABAB
ABAB analytic
p_{00}
p_{11}

(h)

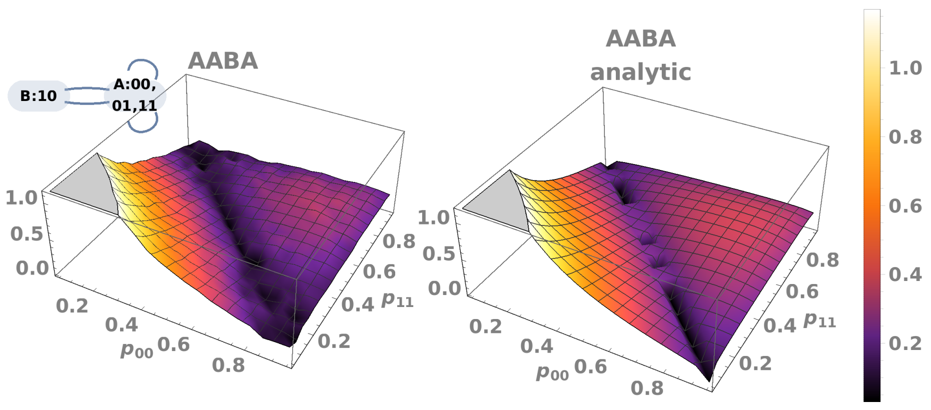

B:01,10
A:00,11
ABBA
ABBA analytic
p_{00}
p_{11}

(i)



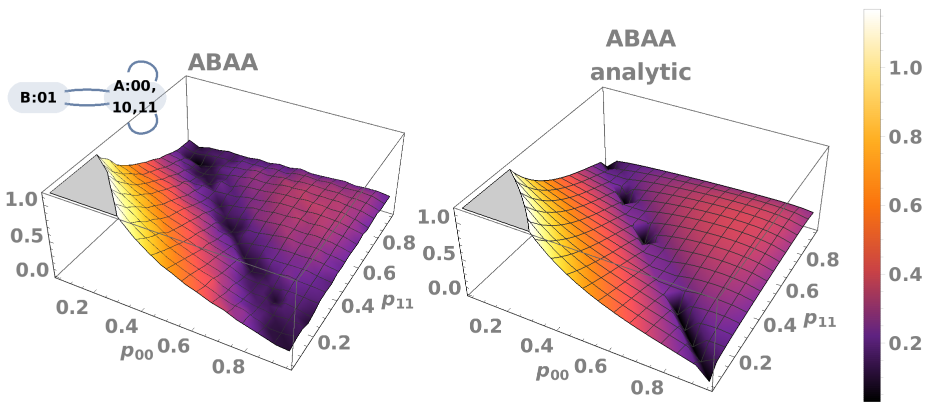
B:11
A:00, 01,10
AAAB
AAAB analytic
1.0
0.5
0.0
0.2
0.4
p_{00}
0.6
0.8
p_{11}
1.2

(j)



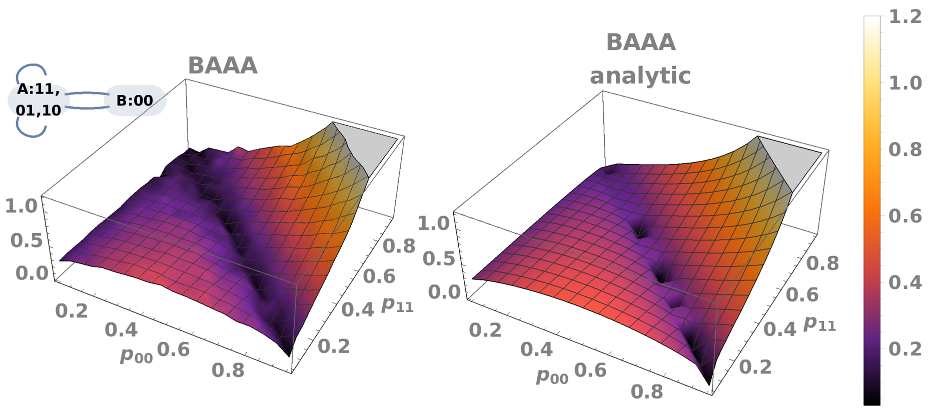
B:10
A:00, 01,11
AABA
AABA analytic
1.0
0.5
0.0
0.2
0.4
p_{00}
0.6
0.8
p_{11}

(k)



B:01
A:00, 10,11
ABAA
ABAA analytic
1.0
0.5
0.0
0.2
0.4
p_{00}
0.6
0.8
p_{11}

(l)



(m)

Figure A.5.: Toy Model. Each pair of graphs shows a **comparison between measured and predicted correlation lengths** (as defined by the eqn.A.36). Measurements were made by generating the binary DNA sequences from a 2×2 transition matrix A.12 and translating them with the corresponding genetic code, as described in the beginning of this section. The theoretical transition matrix was generated by the eqn.2.8for the toy model.

This master thesis is accompanied by a DVD containing the supplementary material. On the top level in the DVD are the following 6 folders:

1. **cppGeneticCodeGenerator**

   This folder contains the c++ code, which among many other things can generate alternative genetic codes. Both algorithms 1 and 2 described in subsections 2.1.1 and 2.1.2 respectively, are implemented here. The unit-tests for the code are in the sub-folder **test_suit**.

2. **genCoMathematica**

   This folder contains the implementation of the MATHEMATICA package *genCo.m* which provides all tools to analyze ensembles of genetic codes. It also contains tools to generate DNA sequences, using the Markov chain model described in the Chapter 2 and translate them to proteins. And measure the Markov matrix for these sequences. An implementation of the equation 2.4 is also provided here. This package was extensively unit-tested and benchmarked all of the tests are contained in the aptly named sub-folder **unitTestAndBenchmarks**.

3. **EntropyHist**

   This folder contains a sub-folder with data (JSON files with genetic codes generated by the c++ program) and a MATHEMATICA notebook. Which contain the analysis of the impact of genetic codes on the Shannon entropy change(see end of ch.2).

4. **WAMstructureAnalysis**

   This folder contains a sub-folder with data (JSON files with genetic codes generated by the c++ program) and two MATHEMATICA notebooks. Which contain the analysis of the genetic code ensembles presented throughout the chapters 2 and 3.

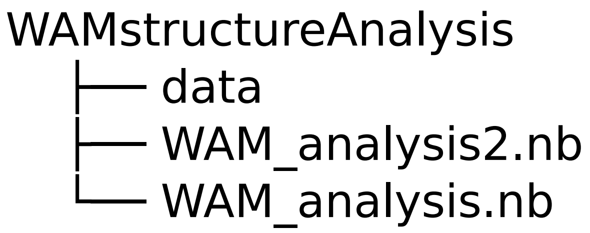
5. **Toy model correlations code**

this folder contains a Mathematica package *ToyModelSequences.m* which is a precursor of *genCo.m* and mainly contains functionality related to the toy model (see sec.A.3). Another script in the folder *ToyModelSequences_script.m* generated all the data used for plots in sec.A.3. The plots themselves were generate with a notebook, which is also in the folder.

6. **figures**

   This is the folder containing all plots used in this thesis. All of the above folders may also contain some additional plots or sub-folders with plots.

All of the code was tested and should be functional if the Folder structure remains intact (Mathematica notebooks rely on the *genCo.m* package file). C++ code was developed in c++14 and and the Gnu compiler (version 7.1.1-4) was used to create the executable file. Mathematica code was written in Mathematica11.1.



```
WAMstructureAnalysis
├── data
├── WAM_analysis2.nb
└── WAM_analysis.nb
```

Figure A.6.: A typical structure of a folder on the supplementary material CD. Folders generally contain a folder called data and Mathematica notebooks, or other source code.

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

[1] William Bialek. *Biophysics: searching for principles*. Princeton University Press, 2012.

[2] Matthew H J Cordes, Alan R. Davidson, and Robert T. Sauer. Sequence space, folding and protein design. *Current Opinion in Structural Biology*, 6(1):3–10, 1996.

[3] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[4] Yuval Elhanati, Anand Murugan, Curtis G Callan, Thierry Mora, and Aleksandra M Walczak. Quantifying selection in immune receptor repertoires. *Proceedings of the National Academy of Sciences of the United States of America*, 111(27):9875–80, 2014.

[5] E. Estrada. *The structure of complex networks: theory and applications*. 2011.

[6] S J Freeland and L D Hurst. Load minimization of the genetic code: history does not explain the pattern. *Proceedings of the Royal Society B: Biological Sciences*, 265(1410):2111–2119, 1998.

[7] William Fulton. *Young tableaux: with applications to representation theory and geometry*, volume 35. Cambridge University Press, 1997.

[8] Christophe Guyeux, Nathalie M-L Côté, Jacques M. Bahi, and Wojciech Bienia. Is protein folding problem really a NP-complete one? First investigations. *Journal of bioinformatics and computational biology*, 12(1):1350017, 2014.

[9] Wolfram Research, Inc. Mathematica, Version 11.1. Champaign, IL, 2017.

[10] Thierry Mora and Aleksandra M Walczak. Quantifying lymphocyte receptor diversity. pages 1–10.

[11] Thierry Mora, Aleksandra M Walczak, William Bialek, and Curtis G Callan. Maximum entropy models for antibody diversity. *Proceedings of the National Academy of Sciences*, 107(12):5405–5410, 2010.

[12] A. Murugan, T. Mora, A. M. Walczak, and C. G. Callan. Statistical inference of the generation probability of T-cell receptors from sequence repertoires. *Proceedings of the National Academy of Sciences*, 109(40):16161–16166, 2012.

[13] Rob Phillips, Jane Kondev, Julie Theriot, and Hernan Garcia. *Physical biology of the cell*. Garland Science, 2012.

[14] Claude E Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[15] Ron Unger and John Moult. Finding the lowest free energy conformation of a protein is an NP-hard problem: Proof and implications. *Bulletin of Mathematical Biology*, 55(6):1183–1198, 1993.

[16] Kalin Vetsigian, Carl Woese, and Nigel Goldenfeld. Collective evolution and the genetic code. *Proceedings of the National Academy of Sciences*, 103(28):10696–10701, 2006.

[17] Joshua A J A Weinstein, Ning N Jiang, Richard A R A White, Daniel S D S Fisher, and Stephen R S R Quake. High-throughput sequencing of the zebrafish antibody repertoire. *Science*, 324(5928):807–810, 2009.

ACKNOWLEDGMENTS

I want to thank Professor Erwin Frey, first for awakening my interest in biological Physics as a lecturer and then for giving me the opportunity to work in his group, by accepting me as his master student. The time I spend in the group was both very enjoyable and educational.
I also want to thank my supervisor Florian Gartner for his support and helpful discussions.
I want to also thank every member of the group. The exchanges of ideas and discussions with them were always very instructive. Special thanks to the diligent attendees of the numerics, and complex networks clouds.
Thanks to my friends Marius Wagner and Timo Haas for checking parts of this thesis for spelling errors. I sure hope they did a good job.
Very special thanks to Karsten Miermans, who always found time for in depth discussions with me about this thesis, and physics in general. His support over the last year helped me to clarify lots of ideas. And his continued lobbying to systematically unit-test my code saved me from an uncountable number of bugs. His help made this thesis indisputably a better one.
Finally I want to thank my parents for their unconditional support, not only during this thesis but the entire duration of my studies. They made lots of demanding times much easier.

SELBSTSTÄNDIGKEITSERKLÄRUNG

Erklärung:

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst zu haben und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt zu haben.

München, September 8, 2017	
Ort, Datum	Unterschrift