Research Experience for Undergraduates
Research Reports
Indiana University, Bloomington
Summer 2010

November 26, 2012

Contents

Preface

During the summer of 2012 nine students participated in the Research Experiences for Undergraduates program in Mathematics at Indiana University. The program ran for eight weeks, from June 4 through July 27, 2012. Six faculty served as research advisers. Three faculty members oversaw a pair of related projects; all other faculty advised one student each.

The program opened with an introductory pizza party. On the following morning, students began meeting with their faculty mentors; these meetings continued regularly throughout the first few weeks. During week one, there were short presentations by faculty mentors briefly introducing the problem to be investigated. Several other IU faculty gave talks on their favorite topics during the first half of the program. Students also received an orientation to the mathematics library. Week two featured a tour of the IU Art Museum led by undergraduate docent, math, and English student and REU alumnus John Brown, and an introduction to professional ethics for researchers in mathematics. In week three, students gave short, informal presentations to each other on the status of work on the project; they also enjoyed a party at a local swimming pool, hosted by Dr. Housworth. Brief training sessions on using LaTeXwere also given during week three. Week four featured a pool party hosted by Dr. Housworth at Bryan Park, a tour of the puzzle collection at the Lily Library, and a campuswide reception for REU programs at the IMU faculty club. In week five, they received a tour of the cyclotron facility led by Prof. Baxter. During week six, there was an intimate panel discussion on What is it like to be a mathematician? hosted by Professors Hoff, Katz, and Kirk. During week seven we hosted the Indiana Mathematics Undergraduate Research conference, which featured 20 lectures by 32 students from Rose-Hulman Institute of Technology, Wabash College, Goshen College, Valparaiso University, and Indiana University, and ended with an hourlong panel discussion on graduate school. The program concluded with a dinner at local eatery Max's Place and the submission of final reports, contained in this volume.

It took the help and support of many different groups and individuals to make the program a success.

We thank the National Science Foundation for major financial support through the REU program through grants DMS-0851852 and DMS-1156515. We thank the staff of the Department of Mathematics for support, especially Mandie McCarty for coordinating the complex logistical arrangments (housing, paychecks, information packets, meal plans, frequent shopping for snacks). We thank Indiana graduate student Anne Carter for serving as LaTeXconsultant and for compiling this volume.

Thanks to mathematics faculty Chris Judge, Kevin Pilgrim, and Matthias Weber for serving as mentors and giving lectures, and to biology faculty Michael Lynch, Emilia Martins, and Sidney Shaw for serving as mentors and giving lectures. Thanks to Jillian Hinchcliffe of the Lily Library for her personal tour of the Slocum collection. Thanks to David Baxter of the Center for Exploration of Energy and Matter (nee IU cyclotron facility) for his personal tour of the cy-

clotron facility and lecture on the physics of materials. Thanks to mathematics faculty members Larry Moss, Ciprian Demeter, Richard Bradley, Matt Bainbridge and graduate student Holly Attenborough for inspiring lectures. Thanks to undergraduate John Brown for the museum tour and to Elizbeth Housworth for the pool party.

The group put together a video montage at `http://www.youtube.com/watch?v=LTHi5zGR_1I`, and volunteered this creative photo:

KMP
November, 2012

Reconstructing Rhombic Tilings From Curve Systems

Kyla Baldwin

1 Introduction

In this paper, we focus on identifying a classification system by which periodic tilings composed exclusively of rhombi may be distinguished from one another. In particular, we consider invariants such as the number of unique rhombi used and the number of zones contained in each tiling.

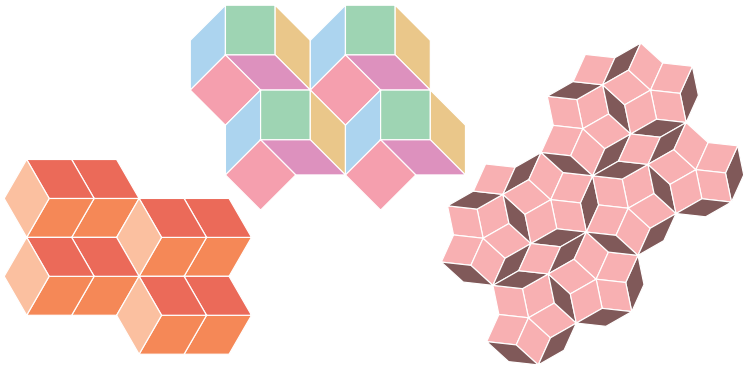


Figure 1: Periodic tilings composed exclusively of rhombi

We approach this problem systematically as follows:

1. For a given tiling, we find an associated curve system on the quotient torus of the tiling that encodes its topological nature.
2. We identify the edge vectors of the rhombi used to compose the tiling. We note that any pair of edge vectors must have a positive determinant. We refer to this as the *determinant condition*.
3. We demonstrate that any set of such curve systems and edge vectors may be used to reconstruct a periodic rhombic tiling.

Our paper first discusses general background and notation. We then provide examples illustrating the process of encoding a periodic tiling onto a curve system. We proceed to discuss the general case of reconstruction of a periodic tiling

from its given curve system and edge vectors, elaborating upon the necessary restrictions of the given data. We conclude by generalizing to a third dimension.

2 Background and Notation

We wish to examine different constructions by which a plane may be covered. We introduce the following terminology:

Definition 2.1. A *tiling* uses a set of geometric shapes repeated infinitely many times to create a two-dimensional plane without gaps or overlapping shapes.

Definition 2.2. We refer to a tiling of the plane as *periodic* if there are two independent translations under which the tiling is invariant. In our paper, we only consider periodic tilings.

Definition 2.3. Given two independent translations $\vec{v_1}$ and $\vec{v_2}$, we refer to the set $\Lambda : \{a\vec{v_1} + b\vec{v_2} \in \mathbb{Z}\}$ as the *lattice* generated by $\vec{v_1}$ and $\vec{v_2}$. If our periodic tiling is invariant under $\vec{v_1}$ and $\vec{v_2}$, we refer to Λ as the *associated periodic lattice* of $\vec{v_1}$ and $\vec{v_2}$.

We may categorize associated periodic lattices by the geometrical form that they take in the plane. For example, if our lattice is generated by $\vec{v_1} = (1, 0)$ and $\vec{v_2} = (0, 1)$, we refer to it as a square lattice (Figure 2). However, if our lattice is generated by $\vec{v_1} = (1, 0)$ and $\vec{v_2} = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right)$, we refer to it as a hexagonal lattice (Figure 2).

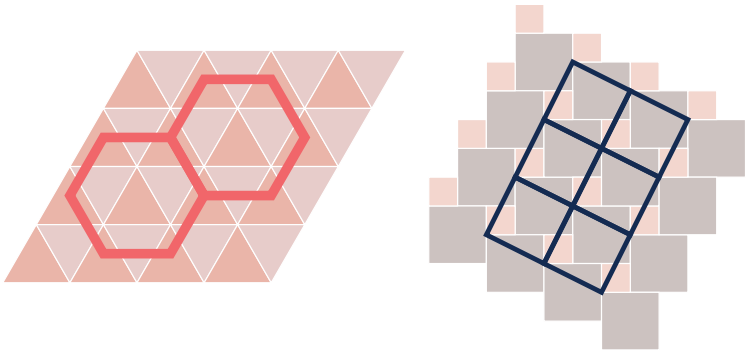


Figure 2: An example of a hexagonal lattice and a square lattice, respectively.

By definition, a periodic tiling consists of a single shape or a set of shapes repeated infinitely many times. The independent translations $\vec{v_1}$ and $\vec{v_2}$ determine the collection of shapes that is iterated infinitely many times to tile the plane. We refer to this collection as the fundamental domain of the tiling. Formally,

Definition 2.4. Given a lattice Λ, the *fundamental domain* $Q \subset \mathbb{R}^2$ is such that:

1. For every point $p \in \mathbb{R}^2$, there is a translation $\vec{v} \in \Lambda$ such that $\vec{v} + p \in Q$
2. Given a point $p \in \mathbb{R}^2$ and a translation $\vec{v} \in \Lambda$, if the translation $p + \vec{v} \in Q$, then p lies in the boundary of Q

Definition 2.5. A *zone segment* is a curve whose endpoints lie on the boundaries of two opposing edges of a rhombus. A *zone curve* is an infinite curve (without endpoints) composed of zone segments. Finally, a *zone* is composed of exactly those rhombi intersected by a given zone curve.

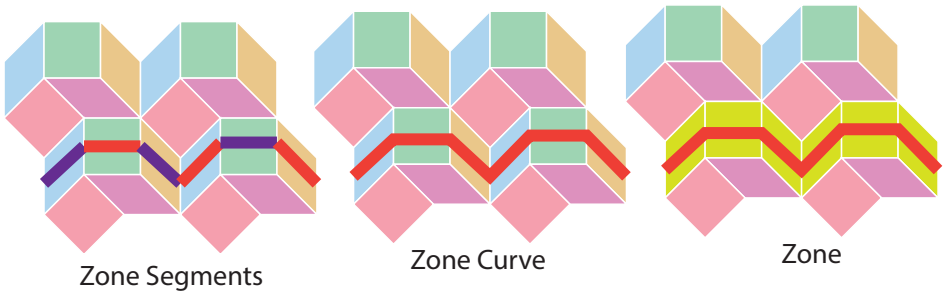


Figure 3: An example of a zone segment, a zone curve, and a zone

Definition 2.6. A *curve system* consists of the zone curves and their intersections.

Throughout this paper, we denote the n zone curves of a given periodic tiling as $(\gamma_1, \gamma_2, \ldots, \gamma_n)$, where $n \geq 2$. Furthermore, we label the intersections of pairs of zone curves (which appear as vertices on the curve system) as $\widehat{v_1}, \widehat{v_2}, \ldots, \widehat{v_m}$, where $m \geq 1$.

Note that because a periodic tiling covers the entire plane, any given zone curve is repeated infinitely many times, so, in particular, the number of zone curves in the periodic tiling is equal to the number of zone curves in the fundamental domain. We may therefore classify a periodic tiling by the number of zone curves in its fundamental domain.

3 Associating a tiling to a curve system

Consider the periodic tiling T with a fundamental domain of four squares, as shown in Figure 4. We would like to represent T by a curve system on the quotient torus.

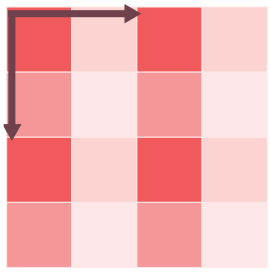


Figure 4: T.

We begin by defining the four zone curves of T.

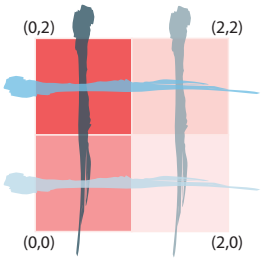


Figure 5: The zone curves of T

We refer to the zone curve intersecting $(0,0)$ and $(0,2)$ as γ_1, the zone curve intersecting $(2,0)$ and $(2,2)$ as γ_2, the zone curve intersecting $(0,2)$ and $(2,2)$ as γ_3, and the zone curve intersecting $(0,0)$ and $(2,0)$ as γ_4.

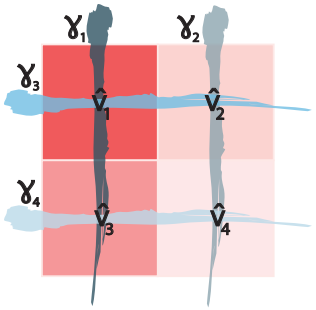


Figure 6: Labeled zone curves

We map the curve system onto a torus by identifying γ_1 with γ_2 and γ_3 with γ_4:

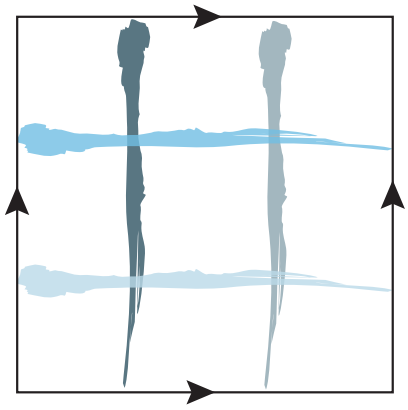


Figure 7: Mapping to quotient torus

4 Reconstruction of a Periodic Tiling From a Curve System

Considering the mapping of zone curves $(\gamma_1, \gamma_2, \ldots, \gamma_n)$ onto the quotient torus, it is clear that we may represent γ_i by a list of vertices $\widehat{v_1}, \ldots, \widehat{v_m}$ unique up to cyclic permutation, where each $\widehat{v_k}$ represents the intersection of γ_i with some other zone curve γ_j. For example, as seen in Figure 8, we may represent γ_1 as $[\widehat{v_1}, \widehat{v_3}]$.

Since a zone curve intersects rhombi with opposing edges parallel to some edge vector, it is clear that the intersection of two zone curves γ_i, γ_j defines a

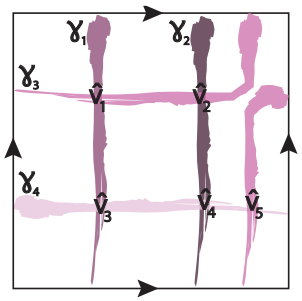


Figure 8: A curve system with zone curves $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ and intersection vertices $\widehat{v_1}, \widehat{v_2}, \widehat{v_3}, \widehat{v_4}, \widehat{v_5}$.

unique rhombus with one pair of opposing edges parallel to the vector associated with γ_i, and the other pair of opposing edges parallel to the vector associated with γ_j. In this way, we see that each intersection vertex $\widehat{v_k}$ of a curve system describes a rhombus in the periodic tiling, since $\widehat{v_k}$ represents the intersection of two zone curves.

Furthermore, since we may consider a curve system to be a dual map of a periodic tiling, each face of a curve system is uniquely associated with a vertex of the periodic tiling it represents. The number of edges bounding a given face in the curve system is equal to the degree of the corresponding vertex in the periodic tiling, so we may connect vertices in adjacent faces with segments parallel to the pertinent zone curves to determine the rhombi of the periodic tiling.

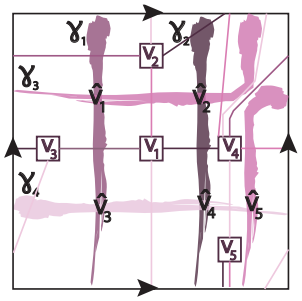
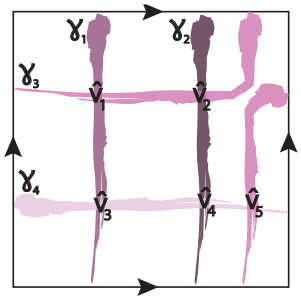


Figure 9: The curve system with faces identified with vertices, and segments connecting adjacent vertices

5 Example of a Reconstruction

We provide a concrete example of the reconstruction of a periodic tiling from a given curve system. In particular, we consider the following curve system, represented on the quotient torus:



We see that curve system contains five intersection vertices, so the periodic tiling is composed of five distinct rhombi, whose edge vectors are given by the pairs of intersecting zone curves as previously discussed.

We begin by identifying each of the five distinct faces of the curve system with a vertex of the periodic tiling, $(v_1, \ldots, v_5)$

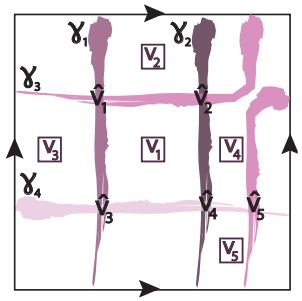


Figure 10: The five vertices of the rhombi of the periodic tiling, shown in boxes.

Then we determine how each vertex is related to another by connecting every vertex to all adjacent vertices by a segment. In a given face of the curve system, there are a certain number of edges which make up the face, and these edges must all be zone curves. When determining the number of segments extruding from a vertex of the periodic tiling contained within the face of the curve system, there must be the same number of these segments as edges of the face. In particular, every vertex of the periodic tiling must have as many segments extruding from it as the number of edges contained within the same face. An example can be seen in Figure 11.

We note that every time one of these segments crosses a zone curve, it

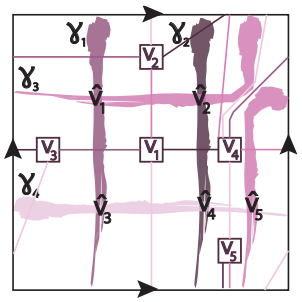


Figure 11: Curve system with segments added to detail how vertices are related to each other.

represents an edge vector in the periodic tiling belonging to whichever zone in which the edge vector is a member - i.e. when a segment crosses the γ_1 curve, in Figure 11, the segment represents an edge vector in the periodic tiling that is part of the γ_1 zone.

Therefore, if we assign edge vector to the zones and we know how each of the vertices is connected to every adjacent vertex by these edge vectors, it becomes possible for us to rebuild the fundamental domain, as well as the entire periodic tiling. In this example, we have chosen to represent the edge vectors associated with each zone curve as shown in Figure 12.

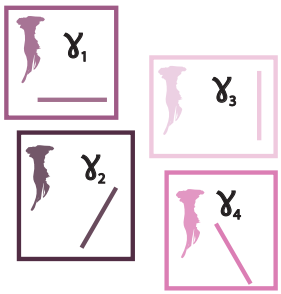


Figure 12: Edge vectors associated with each zone curve.

Now, it is possible to reconstruct the tiling. Beginning with vertex v_1, we see that it is connected to four other vertices, v_2, v_3, v_4, and to another copy of v_2.Therefore, the beginning construction of v_1 would look like Figure 13.

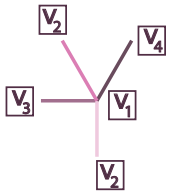


Figure 13: Every edge vector extending from v_1.

We continue this process for each of the five vertices in the curve system until the fundamental domain is complete and continue on with the periodic tiling, as seen in Figure 14.

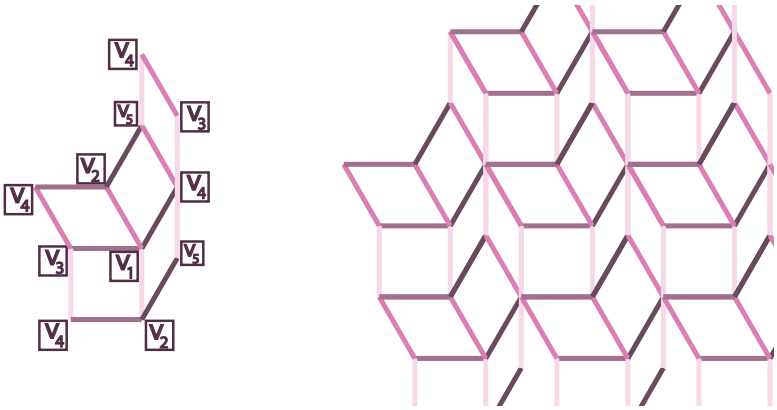


Figure 14: Completed fundamental domain and periodic tiling.

6 Generalization to the Third Dimension

Beginning with a periodic tiling in the plane, we find that these tilings can be realized in the third dimension by choosing edge vectors in $\mathbb{R}^3$. In particular, there are two different operations that allow us to move from a planar tiling to a three dimensional tiling by altering the value of the third coordinate of certain vertex pairs in the tiling.

For a given rhombus:

1. We increase the third coordinate of two adjacent vertices, raising the rhombus from the xy-plane.

2. We increase the third coordinate of a single vertex, as well as decreasing the nonadjacent vertex by the same amount.

Furthermore, we find that such tilings can be constructed using only one rhombus. For example, given the planar tiling in Figure 15, we wish to construct a spatial tiling with the same pattern of zone intersections so that it may be represented by the same curve system. Furthermore, we need four independent edge vectors that satisfy the determinant condition for area so that we maintain the same number of rhombi in the spatial tiling as in the planar version.

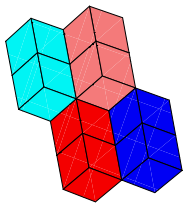


Figure 15: Planar tiling which we look to construct as a spatial tiling

To do this, we may use the four vectors that point to the four corners of the tetrahedron. However, we find that using these four vectors does not create one unique tiling. There are several orientations of such a tiling, two of which can be seen in Figure 16.

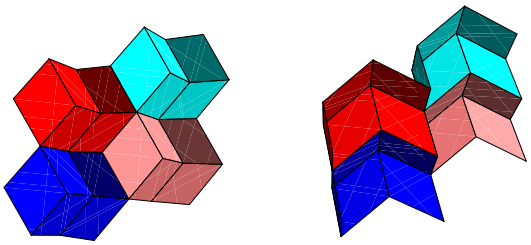


Figure 16: Different orientations of the spatial tiling created from the planar tiling in Figure 15

7 Conclusion

We see that a given periodic tiling has a unique intersection of zone curves which can be represented on the quotient torus. We also recognize that this

representation gives us a curve system for which we can classify all periodic tilings. Moreover, it is possible to not only represent a given tiling on a curve system but to reconstruct a tiling from a given curve system, as well.
Finally, we find that periodic tilings may be realized in space by choosing edge vectors in $\mathbb{R}^3$, and such tilings can even be constructed of one rhombus exclusively.

A Maximum Likelihood Approach to the Analysis of Diploid DNA Sequences

Stephen Bates

Abstract

The heterozygosity of a DNA sample from a diploid (containing two chromosomes) organism is the fraction of sites at which the 2 chromosomes match. This quantity is of great interest in the field of population genetics as it can provide information about the nature of mutations ongoing in a population. Accurate, unbiased estimation of this quantity is non-trivial because DNA sequencing methods give an incomplete picture of the data and suffer from high error rates. In this talk, we will examine a maximum likelihood approach to estimating the heterozygosity of a given DNA sequence. This approach is useful, both because it is accurate for small sampling coverage and because it provides an estimate of the sequencing error rate. We will then extend the maximum likelihood technique to the estimation of other properties of interest in a DNA sample.

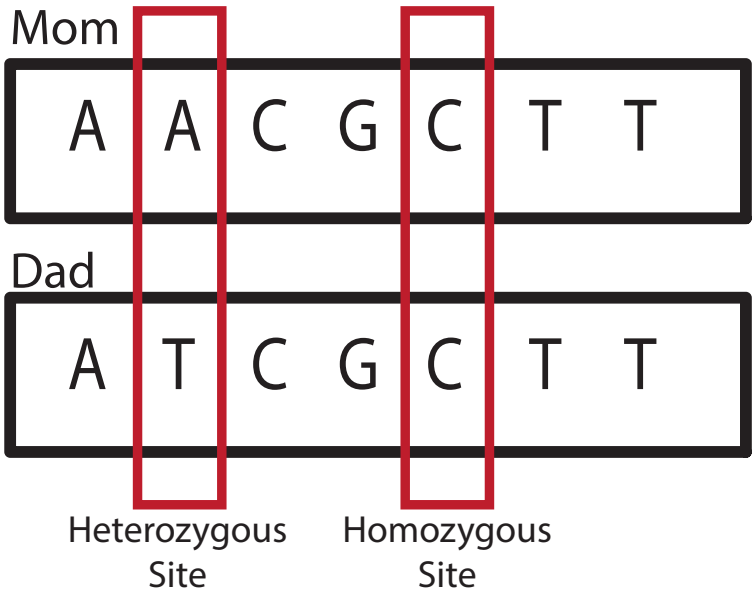
1 Background

Population genetics is a field of biology dedicated to understanding the genetic structure of populations. Mutation, recombination, natural selection, and stochastic drift all affect the genetic makeup of populations, and this field seeks a thorough understanding of these elements. Nearly all other fields of biology rely on the ideas of reproduction and the transfer of genetic information between generations, so the questions explored in this field are very central to the overall discipline of biology. Because populations can be described quantitatively in a very natural way, mathematical models and statistical techniques play a very important role in this area of study [Wak].

Recently, DNA sequencing techniques have improved dramatically. Because of this, the amount of DNA data available for analysis is continually increasing. Properties of an organisms DNA are naturally of interest in population genetics, but statistical techniques for gathering information from these sequences must be developed. The purpose of this paper is to develop a statistical method for inferring the value of two such properties of the DNA sequences: the *heterozygosity* and the *correlation of zygosity* (both to be defined shortly). These two parameters are relevant because they provided in formation about levels of genetic variation and genetic structure of the population. The approach taken here follows that of [Lyn], but with slight differences in the mathematical details.

1.1 DNA

DNA is the primary way that organisms carry and transfer information. For our purposes, a **DNA sequence** is simply a sequence of bases. In realistic biological settings, these sequences can often be millions of bases long. 4 different bases are possible at any location along this sequence, denoted by an A,C,G, or T. A **diploid** organism is one that contains two DNA sequences: one inherited from the father and one inherited from the mother. A **site** is a specific base location on the sequence; for example, the 5th base from the beginning of the sequence is a site. If we compare the two DNA sequences in a diploid organism at a given site, the two sequences will either have identical bases present, denoted **homozygous**, or have different bases, denoted **heterozygous**.



1.2 Sequencing Method

Our first goal is to estimate the fraction of sites which are heterozygous, denoted π , from sequencing data. Modern methods of DNA sequencing methods do not take a picture of the entire DNA strand, rather they chop the several DNA sequences into small pieces and the reassemble overlapping pieces. The difficulty is that we cannot determine which pieces of information come from the mother's

DNA sequence, and which come from the father's. In addition, some fraction ϵ of the sequencing reads will be errors. A priori, we do not know this epsilon and also wish to estimate this from the data.

For a given site, the only information that is observed from the sequencing process is the number of times we observed an A, C, G, or T, denoted by a quartet (n1, n2, n3, n4). Each individual read could have come from a correct read of either of the two DNA sequences or it could be an error. As a model for the sequencing process, we assume that first for each read that one of the twoDNA sequence is selected. Each has probability $\frac{1}{2}$ of being selected. It is then read correctly with probability $1 - \epsilon$, and the probability of observing any particular one of the three incorrect bases at that site is $\frac{\epsilon}{3}$ each.

2 Notation

We will use $P(A)$ to denote the probability of event A occurring and $P(A \mid B)$ to denote the probability of even A occurring given that event B has occurred.

Lowercase m1, m2, m3, m4, n1, n2, n3, n4 are the constants. They are the data observed. Capital M1, M2, M3, M4, N1, N2, N3, N4 refer to the random variable of the number of A, C, G, and T observed.

For notational convenience, we set m = m1 + m2 + m3 + m4 and n = n1 + n2 + n3 + n4. This is the **coverage** at each site.

We also denote the overall frequencies of A, C, G, and T in the sequence in question by p1, p2, p3, p4 respectively. Biologically, these frequencies will vary from sequence to sequence.

3 The Model

We begin with the problem of estimating π. There are two important difficulties estimating this parameter. First, at low coverages (n = 4 to 6 or so) there is a high probability that only one of the two chromosomes present will be sampled, making it possible to overlook heterozygous sites. Second, there are a substantial number of machine sequencing errors. Although sequencing machines come with a factory estimate of their accuracy, leaving the error rate ϵ unknown and estimating it from the sample is best, because it will vary from sample to sample. To account for these two difficulties, we will estimate the parameters π and ϵ using **maximum likelihood estimation**. This is the technique of calculating the probability of the observed data for different values of of the parameters, and then maximizing this probability for the parameter space. The values $\hat{\pi}$ and $\hat{\epsilon}$ that maximize the probability of the observed data are the maximum likelihood estimates (MLEs). Maximum likelihood estimation is well-studied statistic technique that has some very nice properties. Notably, maximum likelihood estimates are **consistent**: as the amount of data increases to infinity, the estimate will converge upon the true value of the parameter.

Our total data set will a collection of quartets: one quartet for each site.

The overall probability of the data set is simply the product of the probabilities at each site. Thus we only need to calculate the probability of observing some quartet (n1, n2, n3, n4) at a given site. To do this, we will break it into two cases: heterozygous sits and homozygous sites.

3.1 Homozygous Case

We first consider the case where the site is homozygous, i.e. the base is the same on both of the two DNA strands present. Suppose that both sites are in reality base i. In this case, for each sample taken, the probability of observing base i is simply the probability that there is no sequencing error: $1 - \epsilon$. The probability of observing any of the three other bases is $\frac{\epsilon}{3}$ each. Thus the probability of observing the quartet (n1, n2, n3, n4) follows a multinomial distribution with the above probabilities (see the appendix for details on the multinomial distribution).

To calculate the probability of observing (n1, n2, n3, n4), we must sum over the probabilities of it occurring for each of the four homozygous possibilities (the site could be A, C, G, or T). Here the $\frac{1}{p_i}$ term accounts for the probability of that particular base occurring. For instance, if over the whole sample that base 'A' occurs with frequency .9, then $\frac{1}{p_i} = .9$. Thus this term is a weighting term based on the frequencies of each of the four bases across all sites.

$$L_1(n1, n2, n3, n4, \epsilon) = \sum_{i=1}^{4} \frac{1}{p_i} [\frac{n!}{n1!n2!n3!n4!}(1-\epsilon)^{n_i}(\frac{\epsilon}{3})^{n-n_i}]$$

3.2 Heterozygous Case

Next we consider the heterozygous case, i.e. the bases present on the two DNA strands differ. Suppose one of the bases present is i and the other is j. Then the probability of observing one of the other two bases (not base i or j) is $\frac{\epsilon}{3}$ each, because these observations must arise from sequencing errors. From symmetry, we then see that the probability of observing a read of base i is $\frac{1-2/3\epsilon}{2}$ (which is of course equal to the probability of observing a j). Thus the probability of observing the quartet (n1, n2, n3, n4) follows a multinomial distribution with the above probabilities.

To calculate the probability of observing (n1, n2, n3, n3), we must sum over the probabilities of it occurring for each of the 6 possible heterozygous configurations. Here we se $S = \sum_{i=1}^{3} \sum_{j=i+1}^{4} p_i p_j$. so that the $\frac{p_i p_j}{S}$ term accounts for the weighting of each heterozygous possibility. This accounts for the difference of frequency of the four bases present.

$$L_2(n1, n2, n3, n4, \epsilon) = \sum_{i=1}^{3} \sum_{j=i+1}^{4} \frac{p_i p_j}{S} [\frac{n!}{n1!n2!n3!n4!}(\frac{1-2/3\epsilon}{2})^{n_i+n_j}(\frac{\epsilon}{3})^{n-n_i-n_j}]$$

3.3 Overall Likelihood.

We can now combine these two cases to determine the unconditional likelihood. The probability that a site is homozygous is π and the probability that a site is heterozygous is $1 - \pi$. Thus the probability of observing a quartet (n1, n2, n3, n4) at a particular site is:

$$L_{site}(n1, n2, n3, n4, \pi, \epsilon) = \pi L_2(n1, n2, n3, n4, \epsilon) + (1 - \pi)L_1(n1, n2, n3, n4, \epsilon)$$

The total likelihood over all sites is then:

$$L_{total}(\pi, \epsilon) = \prod_{all\ sites} [L_{site}(n1, n2, n3, n4, \pi, \epsilon)]$$

We then maximize this for $0 \leq \pi \leq 1$ and $0 \leq \epsilon \leq 1$ to find our estimates $\hat{\pi}$ and $\hat{\epsilon}$.

In practice, this probability will often be extremely small making it computationally difficult to deal with. Instead of maximizing L_{total}, we instead maximize

$$log(L_{total}) = \sum_{all\ sites} [log(L_{site})]$$

Because log(x) is monotonically increasing, the maximum will occur at the same position, and this sum is computationally much easier to deal with. In practice, this maximization must of course be done using a computer. There are a variety of such algorithms available. An implementation of this specific method can be found in [Hau].

Thus we have developed a method for estimating the parameters π and ϵ from the data. These estimators have been tested using simulated data sets, and the estimates prove to be very good. Even for low coverages (n = 4 to 8) the estimates are quite reliable and have bias usually less than 2 percent, which is is considered acceptable in this field.

4 The Correlation of Zygosity

We now wish to use the machinery that we developed to infer other properties of the DNA sequences from the data. One such biologically meaningful parameter is Δ: the **correlation of zygosity**. When considering pairs of sites, Δ is defined as the probability that two sites are linked: that is if one is homozygous than both are homozygous and if one is heterozygous then both are heterozygous. This parameter is biologically useful as a measurement of *linkage disequilibrium*, which loosely speaking is the non-random occurrence of some genetic material in a population.

Now for a pair of sites, there are three possible configurations: both sites are homozygous, both sites are heterozygous, and one site is homozygous and one site is heterozygous. Using our definition of Δ, the probabilities of these three cases are, respectively:

$$H_0 = \Delta(1-\pi) + (1-\Delta)(1-\pi)^2 = (1-\pi)^2 + \Delta\pi(1-\pi)$$
$$H_2 = \Delta\pi + (1-\Delta)(\pi)^2 = \pi^2 + \Delta\pi(1-\pi)$$
$$H_1 = 2(1-\Delta)\pi(1-\pi)$$

Notice $H_1 = 1 - H_0 - H_2$ so knowing the first two quantities determines the third.

Now $\pi = .5H_1 + H_2$ and from the final equation $\Delta = 1 - \frac{H_1}{2\pi(1-\pi)}$. Thus estimation of H_0 and H_2 will allow the estimation of π and Δ thanks to the following theorem (see [Bar]):

Theorem 4.1. *If $\hat{\theta}$ is the MLE of parameter θ, then $h(\hat{\theta})$ is the MLE of parameter $h(\theta)$*

In our case, π and Δ can be written as a function of H_0 and H_2.

4.1 Estimation Technique.

We will again use a maximum likelihood technique for estimating the parameters in question. The method of estimation will be very similar to our initial method of estimation of π. Because we are considering two sites, our data will now be octets (m1, m2, m3, m4, n1, n2, n3, n4). Here, m1 is the number of 'A's observed at site 1, n1 is the number of 'A's observed at site 2, m2 is the number of 'C's observed at site 2, etc. As described above, there are three possible configurations for a double site. Using the language of conditional probability:

$$\begin{aligned}
&P(m1, m2, m3, m4, n1, n2, n3, n4) = \\
&\quad P(m1, m2, m3, m4, n1, n2, n3, n4 \mid \text{double homozygous})P(\text{double homozygous}) \\
&\quad + P(m1, m2, m3, m4, n1, n2, n3, n4 \mid \text{double heterozygous})P(\text{double heterozygous}) \\
&\quad + P(m1, m2, m3, m4, n1, n2, n3, n4 \mid \text{mixed homo/heterozygous})P(\text{mixed})
\end{aligned}$$

Using the L_1 and L_2 terms calculated earlier, we find simple expressions for these terms:

$$\begin{aligned}
&P(m1, m2, m3, m4, n1, n2, n3, n4 \mid \text{double homozygous}) = L_1(m1, m2, m3, m4, \epsilon)L_1(n1, n2, n3, n4, \epsilon) \\
&P(m1, m2, m3, m4, n1, n2, n3, n4 \mid \text{double heterozygous}) = L_2(m1, m2, m3, m4, \epsilon)L_2(n1, n2, n3, n4, \epsilon) \\
&P(m1, m2, m3, m4, n1, n2, n3, n4 \mid \text{mixed homo/heterozygous}) = L_1(m1, m2, m3, m4, \epsilon)L_2(n1, n2, n3, n4, \epsilon) \\
&\quad + L_2(m1, m2, m3, m4, \epsilon)L_1(n1, n2, n3, n4, \epsilon)
\end{aligned}$$

So combining these cases, we have our likelihood function:

$$\begin{aligned}
&L_{octet}(m1, m2, m3, m4, n1, n2, n3, n4, H_0, H_2, \epsilon) = \\
&\quad L_1(m1, m2, m3, m4, \epsilon)L_1(n1, n2, n3, n4, \epsilon)H_0 + L_2(m1, m2, m3, m4, \epsilon)L_2(n1, n2, n3, n4, \epsilon)H_2 \\
&\quad + [L_1(m1, m2, m3, m4, \epsilon)L_2(n1, n2, n3, n4, \epsilon) + L_2(m1, m2, m3, m4, \epsilon)L_1(n1, n2, n3, n4, \epsilon)](1 - H_0 - H_2)
\end{aligned}$$

The overall likelihood is then the product of the likelihoods across all the different octets under consideration:

$$L_{total}(H_0, H_2, \epsilon) = \prod_{\text{all octets}} [L_{octet}(m1, m2, m3, m4, n1, n2, n3, n4, H_0, H_2, \epsilon)]$$

For computational simplicity, we again choose to instead maximize

$$log(L_{total}) = \sum_{\text{all octets}} [log(L_{octet})]$$

Maximizing this yields maximum likelihood estimates $\hat{H_0}$,$\hat{H_2}$, and $\hat{\epsilon}$. By the theorem above, $\hat{H_1} = 1 - \hat{H_0} - \hat{H_2}$ and also $\hat{\pi} = .5\hat{H_1} + \hat{H_2}$ and finally $\hat{\Delta} = 1 - \frac{\hat{H_1}}{2\pi(1-\pi)}$. Thus we have found maximum likelihood estimates for all the quantities we are interested in. Simulated data again shows that these estimators preform well; specifically, they are nearly unbiased, even for small coverages.

5 Appendix: The Multinomial Distribution

The multinomial distribution is a very common distribution from probability theory. It can be thought of as randomly dropping a ball into one of k bins n different times, where at every step the probability of a ball ending up in bin i denoted by p_i. Naturally, for a valid multinomial distribution we must have $p_1 + p_2 + ... + p_k = 1$. We denote the number of balls that end up in bin i by n_i. Thus by the definition of n, $n = n_1 + n_2 + ... + n_k$. From standard probability theory, we know the probability of observing any valid output $(n_1, n_2, ...n_k)$ is given by the following formula:

$$P(n_1, n_2..., n_k) = \frac{n}{n_1!n_2!...n_k!} p_1^{n_1} p_2^{n_2} ... p_k^{n_k}$$

There are many other interesting features of the multinomial distribution, but this is the only one necessary for the current application.

6 Acknowledgements

I would like to thank my advisor Professor Mike Lynch for all of his help and guidance this summer. I would also like to thank Professor Elizabeth Housworth for her mathematical and statistical insights. I also thank Professor Kevin Pilgrim for all of his effort organizing the REU, as well as Mandie McCarty for all of the snacks. Lastly, I thank Kyla Baldwin for her graphic design assistance and Kristin Cordwell for her Latex assistance.

References

[Bar] Bartonszynki, R. and Niewiadomska-Bugaj, M., 2008: *Probability and Statistical Inference*. 2nd ed. Wiley, 647 pp.

[Hau] Haubold, B.,Pfaffelhuber, P., Lynch, M.: mlRho - a program for estimating the population mutation and recombination rates from shotgun-sequenced diploid genomes. *Molecular Ecology*, **19** (Suppl. 1), 277-284.

[Lyn] Lynch, M.: Estimation of Nucleotide Diversity, Disequilibrium Coefficients, and Mutation Rates from High-Coverage Genome-Sequencing Project. *Mol. Biol. Evol.*, **25**(11), 2409-2419.

[Wak] Wakeley, J., 2009: *Coalescent Theory: An Introduction*. Roberts and Company, 326 pp.

Probabilistic Modeling Techniques for Microtubules

Alex Chin

Abstract

We develop a discrete time model for the behavior of microtubules, a cellular filament vital for such tasks as motility and cell division. We consider integer values for growth and shortening velocities and construct a Markov chain model for a single polymer based on the nucleation rate, catastrophe frequency, and rescue frequency. This model gives rise to a set of recurrence relations for each pair of growth and shortening velocities that can be solved using linear algebra techniques to determine a length distribution for the polymer in steady state. We consider both finite and infinite models and consider the mathematical and biological implications of each.

1 Introduction

1.1 Motivation

The behavior of microtubules has been studied extensively by biologists and mathematicians alike, and the field of microtubule dynamics lies at the intersection of mathematics and cell biology. Several models have been developed but none perfectly predict *in vivo* observations and all make key simplifying assumptions. The development of such a biologically viable model would have several immediate applications, including in the development of cancer treatment pharmaceuticals. We first present a summary of the biology behind the process of microtubule dynamics, so that the reader may better understand the mathematical processes and methods chosen.

1.2 Biological Background

Microtubules are one of three types of protein polymer filaments that compose the cellular cytoskeleton, and are critical for a variety of tasks including motility, cargo transport and cell division. Microtubules are composed of α- and β- tubulin protein dimers that are globular in shape. These individual subunits arrange longitudinally into protofilaments, and subsequently laterally into a hollow tube composed of 13 protofilaments. The parallel nature of this arrangement creates a polar filament, as α- and β- tubulins are exposed at opposite (minus and plus) ends. Microtubules may extend in length indefinitely, and in living cells may be thousands of subunits long.

At any given time, a tubulin subunit may be in one of two states. *Free* subunits float in the cellular cytoplasm and *incorporated* subunits are attached to microtubule polymers. Free subunits in the cytoplasm are bound to a guanosine triphosphate (GTP) molecule. Once these subunits attach to a microtubule, the GTP on the β monomer begins to hydrolyze, leaving a GDP-bound tubulin subunit. GTP-bound subunits have a higher bonding affinity than GDP-bound subunits; therefore, microtubules are likely to grow if the plus-end subunit is GTP-bound, and likely to shrink if it is GDP-bound. This depends on the rate of subunit addition; if it is high enough to facilitate the addition of new subunits before hydrolysis can occur (forming a GTP-cap) then the microtubule undergoes a growth phase. If the end unit is allowed to hydrolyze, then the microtubule undergoes a rapid shortening phase until a new GTP-bound subunit can be added. This protein hydrolysis gives rise to a system in which some microtubules may be growing while others are shortening, and in fact a single polymer may switch states unpredictably.

Definition 1.1. The stochastic process of switching between growth and shortening is termed *dynamic instability*.

Definition 1.2. The event that a polymer switches from a growth state to a shortening state is called *catastrophe*, and the event that a polymer switches from a shortening state to a growth state is called *rescue*.

Definition 1.3. The event that a polymer begins to grow on a free site is called *nucleation*, and the event that a shrinking polymer shortens completely is called *extinction*.

We now describe the process by which a system of microtubules comes to steady state. Consider a system consisting initially only of free tubulin subunits where the temperature and initial concentration of free subunits is fixed. The free subunit concentration will then decrease in a non-uniform manner until steady state is reached, beginning with a nucleation process that creates an initial lag in which little activity occurs. The high concentration of free subunits in the system then leads to a high polymerization rate and low catastrophe frequency, and any polymers that do undergo catastrophe are quickly rescued. If polymerization is allowed to continue for some time, however, the free dimer concentration decreases as more subunits are incorporated into the polymer system. Growth rate and rescue frequency decrease while catastrophe frequency increase; these factors slow the decrease of free subunit concentration. Throughout this process, the shortening rate of polymers that have undergone catastrophe is constant; it depends only on temperature and binding kinetics and is independent of free subunit concentration.

The system eventually reaches a steady state in which global rate of subunit polymerization equals the rate of subunit depolymerization (we do not use the term *equilibrium* because energy must constantly be fed into the system in order for the dynamic instability process to occur). This steady state can then be analyzed for a length distribution of the microtubules in the system, which will

remain constant. Note that a second steady state must also occur that relates the number of polymers undergoing nucleation and extinction; this steady state is necessary for the length distribution to remain unchanged. However, we do not address this second steady state here.

2 Mathematical Formalization

Past models have used a continuous time approach to model microtubule dynamics with differential equations techniques; instead, we use discrete time steps. We begin by formalizing some of the biological terms with respect to this approach.

Definition 2.1. The *growth velocity* is the number of units that a growing microtubule will grow in a given time step, and is denoted by V_g. The *shortening velocity* is the number of units that a shrinking microtubule will shrink in a given time step, and is denoted by V_s.

Here the unit may be chosen to be anything biologically appropriate, measured in microns, another unit of length, or even individual subunits. We will only consider positive integer values of velocities. Furthermore, we need only consider relatively prime pairs of V_g and V_s because all other cases may be scaled accordingly.

Notation. We denote the probability of catastrophe by P_{gs}, the probability of rescue by P_{sg}, and the probability of nucleation by P_N, where all three values lie in $(0, 1)$ (we do not consider the trivial endpoint probabilities).

These five parameters form the basis for our analysis. To simplify the model, we assume that all five parameters are constant and independent of the free subunit concentration. This allows us to analyze the model without recalculating the parameters at each time step. We also assume that individual polymers behave independently from each other.

We consider possible states $0, 1g, 1s, 2g, 2s, 3g, 3s, \ldots$, where the 0 state denotes a free nucleation site and every other state denotes a length coupled with whether the polymer is growing or shortening. We denote by $\pi_i(t)$ the probability that a polymer exists at state i at time t. We write $\pi_0(t), \pi_{1,g}(t), \pi_{1,s}(t), \ldots$ instead of $\pi_0, (t)\pi_{1g}(t), \pi_{1s}(t), \ldots$ to eliminate ambiguity, e.g. when dealing with such states as the shortening state of length $m + 1$.

Definition 2.2. A *Markov chain* is a sequence of random variables such that given the present state, the future and past states are independent.

We use the defined states and parameters to develop a Markov chain model for the behavior of a single microtubule polymer.

We may represent a Markov chain using a directed graph where the states are vertices and the edges are labeled with probabilities such that for every vertex, the sum of edge labels leaving the vertex is 1. For example, if $V_g = V_s = 1$, then we have the following graph:

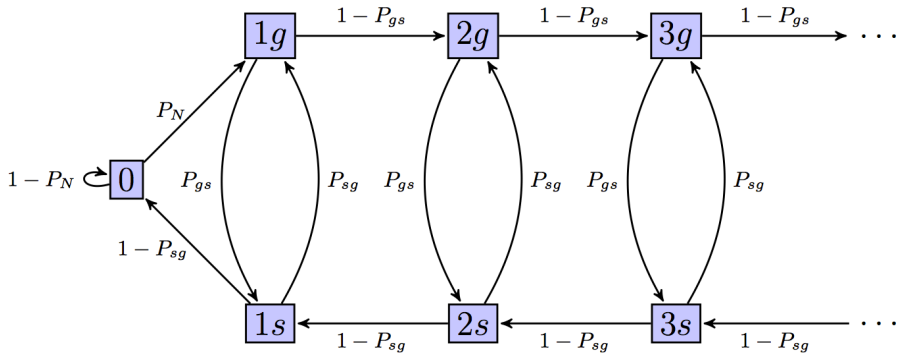


Figure 1: Infinite Markov chain for $V_g = V_s = 1$

Note that we assume that a polymer undergoing catastrophe or rescue continues this process for the entire time step and does not change length until the following time step.

A well-defined Markov chain may be described using a set of recurrence relations. Our model gives the following for all $t = 0, 1, 2, \ldots$:

$$\pi_0(t+1) = (1 - P_N)\pi_0(t) + (1 - P_{sg}) \sum_{k=1}^{V_s} \pi_{k,s}(t) \tag{1}$$

$$\pi_{V_g,g}(t+1) = P_N \pi_0(t) + P_{sg} \pi_{V_g,s}(t) \tag{2}$$

$$\pi_{m,s}(t+1) = (1 - P_{sg})\pi_{m+V_s,s}(t) + P_{gs}\pi_{m,g}(t) \qquad m = 1, 2, 3, \ldots \tag{3}$$

$$\pi_{n,g}(t+1) = (1 - P_{gs})\pi_{n-V_g,g}(t) + P_{sg}\pi_{n,s}(t) \qquad n = 2, 3, 4, \ldots \tag{4}$$

Equation (1) represents a free nucleation site by considering the sum of the probability that the site did not undergo nucleation in the previous step and the probability that all polymers shorter than V_s did not undergo rescue and instead became extinct. Equation (2) represents nucleation by considering the sums of nucleation and rescue in the previous time step. Equation (3) represents shortening and catastrophe of polymers of length m, and Equation (4) represents growth and rescue of polymers of length n.

Now we observe what happens if the system achieves steady state.

Definition 2.3. A *steady state* is a probability vector $(\pi_0, \pi_{1,g}, \pi_{1,s}, \ldots)$ such that if $\pi_i(0) = \pi_i$ for all $i = 0, 1g, 1s, \ldots$, then $\pi_i(t) = \pi_i(t+1)$ for all $t = 0, 1, 2, \ldots$.

Therefore, if we have a system at steady state, we may assume our recurrence

relations are independent of time t and may write them as follows:

$$\pi_0 = (1 - P_N)\pi_0 + (1 - P_{sg}) \sum_{k=1}^{V_s} \pi_{k,s} \tag{5}$$

$$\pi_{V_g,g} = P_N \pi_0 + P_{sg} \pi_{V_g,s} \tag{6}$$

$$\pi_{m,s} = (1 - P_{sg})\pi_{m+V_s,s} + P_{gs}\pi_{m,g} \qquad m = 1, 2, 3, \ldots \tag{7}$$

$$\pi_{n,g} = (1 - P_{gs})\pi_{n-V_g,g} + P_{sg}\pi_{n,s} \qquad n = 2, 3, 4, \ldots \tag{8}$$

This allows us to analyze the possible parameters for which the system reaches steady state, and ultimately determine a length distribution.

Definition 2.4. A *steady state length distribution (SSLD)* is a probability mass function $P : \mathbb{N} \to [0, 1]$, where $P(L)$ is defined to be the probability that a polymer in steady state has length L.

It is clear that given expressions for the probabilities, we may obtain the length distribution by

$$P(L) = \begin{cases} \pi_0 & \text{if } L = 0 \\ \pi_{L,g} + \pi_{L,s} & \text{if } L = 1, 2, 3, \ldots \end{cases}$$

Note that in steady state, $\pi_i > 0$ for all possible states i.

It is helpful to consider the system for specific values of V_g and V_s because then the recurrence relations take on a specific form. For each, we establish a set of relations that characterize the values of P_N, P_{sg}, and P_{gs} that lead to steady state.

3 The $V_g = 1$, $V_s = 1$ case

3.1 Infinite Model

We first allow for arbitrarily long polymers. Using these specific velocities allows us to simplify our recurrence relations as follows:

$$\pi_0 = (1 - P_N)\pi_0 + (1 - P_{sg})\pi_{1,s} \tag{9}$$

$$\pi_{1,g} = P_N \pi_0 + P_{sg} \pi_{1,s} \tag{10}$$

$$\pi_{m,s} = (1 - P_{sg})\pi_{m+1,s} + P_{gs}\pi_{m,g} \qquad m = 1, 2, 3, \ldots \tag{11}$$

$$\pi_{n,g} = (1 - P_{gs})\pi_{n-1,g} + P_{sg}\pi_{n,s} \qquad n = 2, 3, 4, \ldots \tag{12}$$

This provides a much more tangible set of equations to work with. We determine the set of parameters that leads to steady state.

Theorem 3.1. *Let $V_g = V_s = 1$. The system diverges if $P_{sg} \geq P_{gs}$ and converges if $P_{sg} < P_{gs}$. The steady state probabilities are*

$$\pi_0 = \frac{P_{gs} - P_{sg}}{P_{gs} - P_{sg} + 2P_N}$$

$$\pi_{m,s} = \pi_{m,g} = \frac{P_N(P_{gs} - P_{sg})}{(1 - P_{gs})(P_{gs} - P_{sg} + 2P_N)} \left(\frac{1 - P_{gs}}{1 - P_{sg}}\right)^m$$

Proof. Solving Equation (9) for $P_N\pi_0$ and substituting into Equation (10) yields

$$\pi_{1,g} = \pi_{1,s} \tag{13}$$

Let m be an arbitrary positive integer. We prove that the growth and shortening probabilities for length m are equal, and that each forms a monotone increasing subsequence of probabilities. Note that adding Equations (11) and (12) for $n = m + 1$ yields

$$\pi_{m,s} + \pi_{m+1,g} = \pi_{m,g} + \pi_{m+1,s} \tag{14}$$

A simple induction proof using Equation (14) with Equation (13) as the base case yields

$$\pi_{m,g} = \pi_{m,s} \qquad\qquad m = 1, 2, 3, \ldots \tag{15}$$

Substituting into Equation (11) gives

$$(1 - P_{gs})\pi_{m,s} = (1 - P_{sg})\pi_{m+1,s} \tag{16}$$

for all lengths m.

If $P_{sg} \geq P_{gs}$ it is clear that

$$\pi_{m,s} \leq \pi_{m+1,s} \tag{17}$$

This contradicts $\sum P_i = 1$, so this particular parameter set leads to unbounded growth.

If $P_{sg} < P_{gs}$, then we may reduce the problem to first order recurrence relations of the growth and shortening states. Since $\pi_{m,g} = \pi_{m,s}$, we need only worry about the shortening state (the growth state will have the same distribution). Equation (16) now yields a ratio of $\frac{1-P_{gs}}{1-P_{sg}}$ that is nonzero and less than one. The following recurrence relation can be solved

$$\pi_{m+1,s} = \frac{1 - P_{gs}}{1 - P_{sg}}\pi_{m,s} \tag{18}$$

and has a solution of the form

$$\pi_{m,s} = A\left(\frac{1 - P_{gs}}{1 - P_{sg}}\right)^m \tag{19}$$

for some constant A. Using Equation (9) we conclude that $A = \frac{P_N \pi_0}{1-P_{gs}}$. We now use the fact that $\sum \pi_i = 1$ to solve for π_0.

$$\begin{aligned}
\pi_0 + \sum_{m=1}^{\infty} (\pi_{m,g} + \pi_{m,s}) &= 1 \\
\pi_0 + \sum_{m=1}^{\infty} \left(2 \frac{P_N \pi_0}{1-P_{gs}} \left(\frac{1-P_{gs}}{1-P_{sg}} \right)^m \right) &= 1 \\
\pi_0 + 2 \frac{P_N \pi_0}{1-P_{gs}} \frac{1-P_{gs}}{1-P_{sg}} \sum_{m=1}^{\infty} \left(\frac{1-P_{gs}}{1-P_{sg}} \right)^{m-1} &= 1 \\
\pi_0 + \frac{2P_N \pi_0}{1-P_{sg}} \frac{1}{1-\frac{1-P_{gs}}{1-P_{sg}}} &= 1 \\
\pi_0 + \frac{2P_N \pi_0 (1-P_{sg})}{(1-P_{sg})(P_{gs}-P_{sg})} &= 1 \\
\pi_0 + \frac{2P_N \pi_0}{P_{gs}-P_{sg}} &= 1 \\
\pi_0 = \frac{1}{1+\frac{2P_N}{P_{gs}-P_{sg}}} \\
\pi_0 = \frac{P_{gs}-P_{sg}}{P_{gs}-P_{sg}+2P_N}
\end{aligned}$$

Therefore, substituting back into Equation (19) produces the desired result for $\pi_{m,s}$ and $\pi_{m,g}$. This completes the proof. □

The growth and shortening states for a given length are equal, meaning $P(L) = 2\pi_{L,s}$ for any L. Thus Theorem 3.1 gives us the following length distribution, given in terms of the parameters P_N, P_{sg}, and P_{gs}:

$$P(L) = \begin{cases} \frac{P_{gs}-P_{sg}}{P_{gs}-P_{sg}+2P_N} & \text{if } L = 0 \\ \frac{2P_N(P_{gs}-P_{sg})}{(1-P_{gs})(P_{gs}-P_{sg}+2P_N)} \left(\frac{1-P_{gs}}{1-P_{sg}} \right)^L & \text{if } L = 1, 2, 3, \ldots \end{cases} \tag{20}$$

Note that the value of P_N affects nature of the length distribution but does *not* affect whether or not the system comes to steady state. Several example plots for specific parameters are below (note the exponential nature of the distribution).

Example 3.2. These graphs show what is happening over time for unbounded and bounded growth parameters. The first chart displays unbounded growth for $P_{sg} = 0.7$; if we change this value to 0.3, the system displays a steady state.

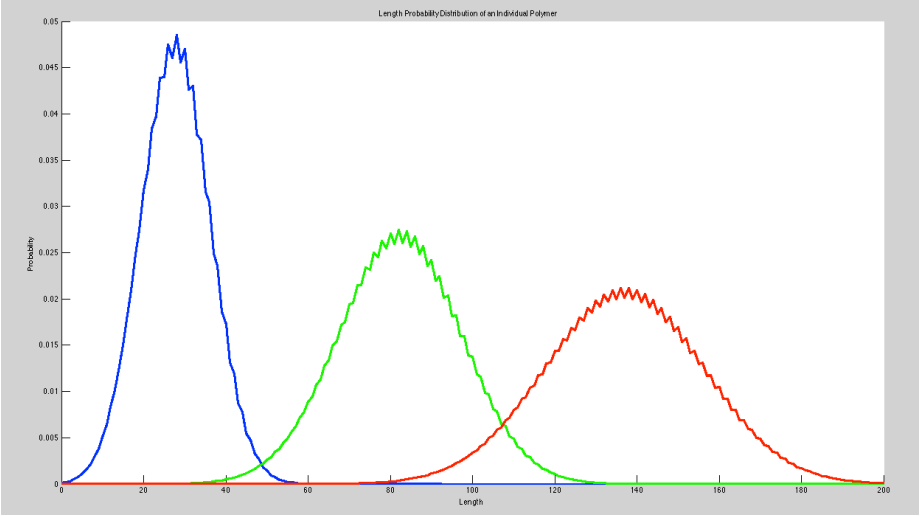


Figure 2: $P_N = 0.25, P_{sg} = 0.7, P_{gs} = 0.4$, after 100, 300, and 500 steps

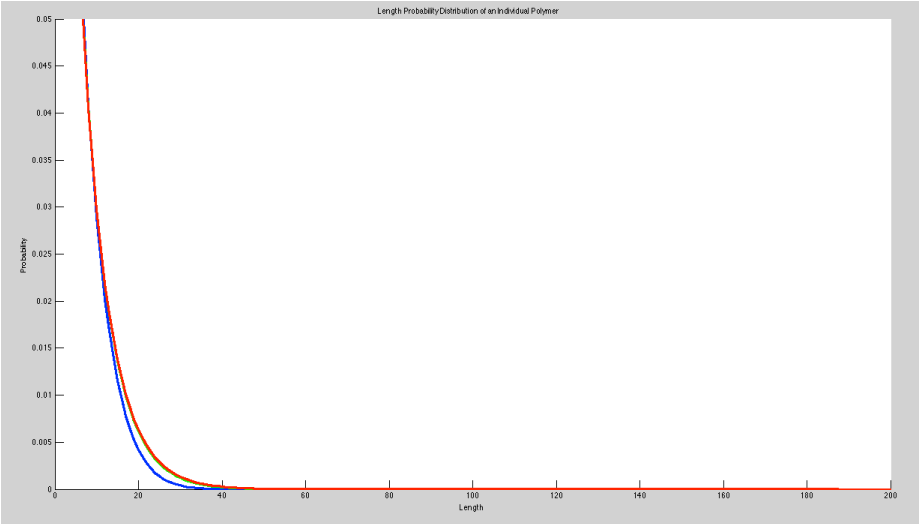


Figure 3: $P_N = 0.25, P_{sg} = 0.3, P_{gs} = 0.4$, after 100, 300, and 500 steps

3.2 Finite Model

Microtubules are stiff polymers and have an upper bound on their length, determined by the volume and dimensions of the cell. If we impose such an upper bound K on our model, the relations for $\pi_{K,s}$ and $\pi_{K,g}$ vary slightly. We place a loop on the growth state at K and let it feed itself instead of continuing to grow, and the shortening state at K is not fed from any longer shortening state since such a state does not exist. Then we obtain the following:

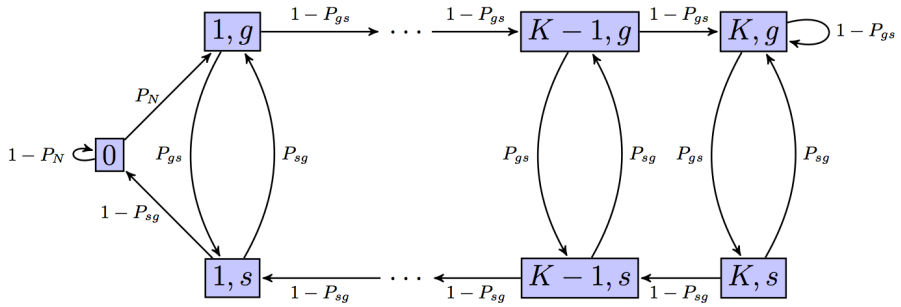


Figure 4: Finite Markov chain for $V_g = V_s = 1$

This Markov chain contains the following new relations at the K growth and shortening states:

$$\pi_{K,s} = P_{gs}\pi_{K,g} \tag{21}$$

$$\pi_{K,g} = (1 - P_{gs})\pi_{K-1,g} + (1 - P_{gs})\pi_{K,g} + P_{sg}\pi_{K,s} \tag{22}$$

These may be expressed in terms of $\pi_{K-1,g}$ as follows:

$$\pi_{K,s} = \frac{1 - P_{gs}}{1 - P_{sg}}\pi_{K-1,g} \tag{23}$$

$$\pi_{K,g} = \frac{1 - P_{gs}}{P_{gs}(1 - P_{sg})}\pi_{K-1,g} \tag{24}$$

Note that we use the same values as in the infinite case for $m = 1, \ldots, K-1$:

$$\pi_{m,s} = \pi_{m,g} = \frac{P_N \pi_0}{1 - P_{gs}} \left(\frac{1 - P_{gs}}{1 - P_{sg}} \right)^m \tag{25}$$

We may then solve for π_0 similarly:

$$\pi_0 + \sum_{m=1}^{K-1}(\pi_{m,g} + \pi_{m,s}) + \pi_{K,g} + \pi_{K,s} = 1$$

$$\pi_0 + \frac{2P_N\pi_0}{1-P_{gs}} \sum_{m=1}^{K-1} \left(\frac{1-P_{gs}}{1-P_{sg}}\right)^m + \frac{P_N\pi_0}{1-P_{gs}} \left(\frac{1-P_{gs}}{1-P_{sg}}\right)^K \left(1 + \frac{1}{P_{gs}}\right) = 1$$

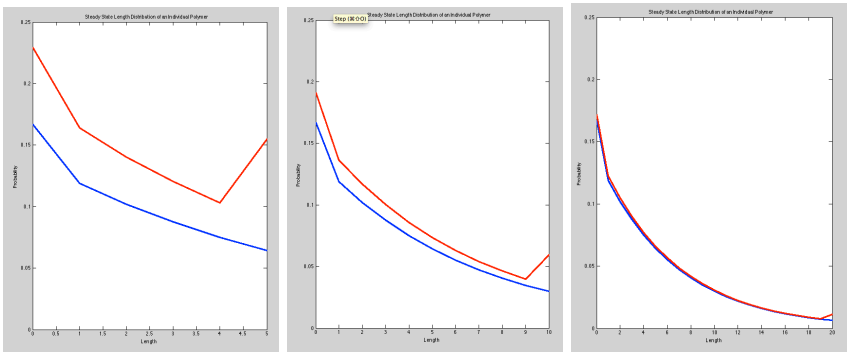
$$\pi_0 + \frac{2P_N\pi_0}{1-P_{gs}} \frac{1-P_{gs}}{1-P_{sg}} \left(\frac{1 - \left(\frac{1-P_{gs}}{1-P_{sg}}\right)^{K-1}}{1 - \frac{1-P_{gs}}{1-P_{sg}}}\right) + \frac{P_N\pi_0}{1-P_{gs}} \left(\frac{1-P_{gs}}{1-P_{sg}}\right)^K \left(1 + \frac{1}{P_{gs}}\right) = 1$$

$$\pi_0 + \frac{2P_N\pi_0}{1-P_{gs}} \left(1 - \left(\frac{1-P_{gs}}{1-P_{sg}}\right)^{K-1}\right) + \frac{P_N\pi_0}{1-P_{gs}} \left(\frac{1-P_{gs}}{1-P_{sg}}\right)^K \left(1 + \frac{1}{P_{gs}}\right) = 1$$

$$\pi_0 = \left[1 + \frac{2P_N}{1-P_{gs}} \left(1 - \left(\frac{1-P_{gs}}{1-P_{sg}}\right)^{K-1}\right) + \frac{P_N}{1-P_{gs}} \left(\frac{1-P_{gs}}{1-P_{sg}}\right)^K \left(1 + \frac{1}{P_{gs}}\right)\right]^{-1}$$

As the complexity of this equation makes it difficult to work with, we will generally stick to numerical methods.

Example 3.3. Figure 5 compares finite (in red) and infinite (in blue) models for a specific choice of parameters. Note that the finite length distribution approaches the infinite length distribution as $K \to \infty$, and in fact converges rather quickly.



(a) $K = 5$ (b) $K = 10$ (c) $K = 20$

Figure 5: finite and infinite models with parameters $P_N = 0.25, P_{sg} = 0.3, P_{gs} = 0.4$ and varying max length K

4 The $V_g = 1$, $V_s = 2$ case

4.1 Infinite Model - Unbounded Growth Parameters

Let us now consider the system with growth velocity $V_g = 1$ and shortening velocity $V_s = 2$. Then we obtain the following Markov chain:

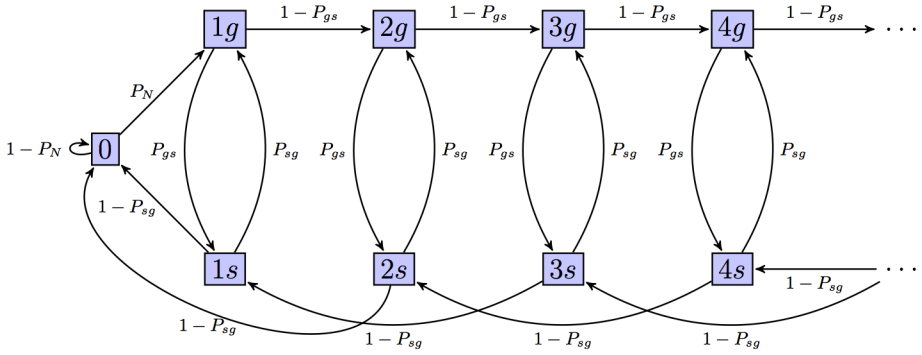


Figure 6: Markov chain for $V_g = 1$, $V_s = 2$

Note that shortening polymers now shrink by length 2 instead of 1. This gives us the following recurrence relations:

$$\pi_0 = (1 - P_N)\pi_0 + (1 - P_{sg})\pi_{1,s} + (1 - P_{sg})\pi_{2,s} \tag{26}$$

$$\pi_{1,g} = P_N\pi_0 + P_{sg}\pi_{1,s} \tag{27}$$

$$\pi_{m,s} = (1 - P_{sg})\pi_{m+2,s} + P_{gs}\pi_{m,g} \qquad m = 1, 2, 3, \ldots \tag{28}$$

$$\pi_{n,g} = (1 - P_{gs})\pi_{n-1,g} + P_{sg}\pi_{n,s} \qquad n = 2, 3, 4, \ldots \tag{29}$$

We first prove a result relating the growth and shortening probabilities of the same length.

Lemma 4.1. *Let $V_g = 1$ and $V_s = 2$. Then for every $m = 1, 2, 3, \ldots$, $\pi_{m,g} > \pi_{m,s}$ and, in particular, $\pi_{m,g} = \pi_{m,s} + (1 - P_{sg})\pi_{m+1,s}$.*

Proof. We proceed by induction. Solving Equation (26) for $P_N\pi_0$ and substituting into Equation (27) yields

$$\pi_{1,g} = \pi_{1,s} + (1 - P_{sg})\pi_{2,s} \tag{30}$$

Adding Equations (28) and (29) for $n = m + 1$ yields

$$\pi_{m,s} + \pi_{m+1,g} = \pi_{m,g} + P_{sg}\pi_{m+1,s} + (1 - P_{sg})\pi_{m+2,s} \tag{31}$$

for m = 1,2,3,.... We prove by induction that $\pi_{m,g} = \pi_{m,s} + (1 - P_{sg})\pi_{m+1,s}$ for every positive integer m. Equation (30) serves as the base case. Now suppose

that there is some $k = 1, 2, 3, \ldots$ such that $\pi_{k,g} = \pi_{k,s} + (1 - P_{sg})\pi_{k+1,s}$. Then we may substitution this equation into Equation (31) with $m = k$ to obtain

$$\pi_{k,s} + \pi_{k+1,g} = \pi_{k,s} + (1 - P_{sg})\pi_{k+1,s} + P_{sg}\pi_{k+1,s} + (1 - P_{sg})\pi_{k+2,s}$$

which simplifies to

$$\pi_{k+1,g} = \pi_{k+1,s} + (1 - P_{sg})\pi_{k+2,s}$$

This completes the induction proof. □

We may now use this lemma to obtain a result regarding unbounded growth parameters.

Theorem 4.2. *Let $V_g = 1$ and $V_s = 2$. If $P_{gs} \leq 2P_{sg} - 1$, then the system has no steady state.*

Proof. Consider an arbitrary $m = 1, 2, 3, \ldots$. We add the result of Lemma 4.1 to Equation (28) to obtain

$$\pi_{m,g} = (1 - P_{sg})(\pi_{m+2,s} + \pi_{m+1,s}) + P_{gs}\pi_{m,g}$$

Using Lemma 4.1 to substitute $\pi_{m+2,g}$ and $\pi_{m+1,g}$ for $\pi_{m+2,s}$ and $\pi_{m+1,s}$, this equation becomes

$$(1 - P_{gs})\pi_{m,g} < (1 - P_{sg})(\pi_{m+2,g} + \pi_{m+1,g})$$

Using the assumption $P_{gs} \leq 2P_{sg} - 1$ gives

$$(1 - (2P_{sg} - 1))\pi_{m,g} < (1 - P_{sg})(\pi_{m+2,g} + \pi_{m+1,g})$$

which simplifies to

$$\pi_{m,g} < \frac{1}{2}(\pi_{m+2,g} + \pi_{m+1,g})$$

So each growth term is smaller than the average of the two subsequent growth terms. This contradicts $\sum P_i = 1$, and the result is proven. □

4.2 Infinite Model - Steady State Parameters

In the $V_g = V_s = 1$ case we were able to solving a first order recurrence to obtain a closed-form expression for the length distribution (Equation (20)). We attempt a similar approach for the $V_g = 1$, $V_s = 2$ case, but this case is complicated by the fact that the growth and shortening probabilities for a given length are not equal. We proceed by obtaining homogeneous second-order recurrence relations for the growth and shortening states.

We first determine a shortening-only relation. We know that for every m (this is simply Lemma 4.1)

$$\pi_{m,g} = \pi_{m,s} + (1 - P_{sg})\pi_{m+1,s} \tag{32}$$

Substituting this equation for $\pi_{m,g}$ into equation (28) and solving for the $\pi_{m+2,s}$ term gives what is desired:

$$(1 - P_{sg})\pi_{m+2,s} = -P_{gs}(1 - P_{sg})\pi_{m+1,s} + (1 - P_{gs})\pi_{m,s} \tag{33}$$

We now determine a growth-only relation. Combining equations (28) and (29) with $n = m + 2$ gives

$$\pi_{m+2,g} - (1 - P_{gs})\pi_{m+1,g} - P_{gs}\pi_{m,g} = \pi_{m+2,s} - \pi_{m,s} \tag{34}$$

which implies the following equation by induction

$$\pi_{m+1,g} + P_{gs}\pi_{m,g} = \pi_{m+1,s} + \pi_{m,s} \tag{35}$$

Note that equation (29) with $n = m + 1$ and $n = m$ gives the following two equations:

$$\pi_{m+1,s} = \frac{1}{P_{sg}}(\pi_{m+1,g} - (1 - P_{gs})\pi_{m,g}) \tag{36}$$

$$\pi_{m,s} = \frac{1}{P_{sg}}(\pi_{m,g} - (1 - P_{gs})\pi_{m-1,g}) \tag{37}$$

Substituting into equation (35) and simplifying results in the following equation consisting of growth states only:

$$(1 - P_{sg})\pi_{m+1,g} = -P_{gs}(1 - P_{sg})\pi_{m,g} + (1 - P_{gs})\pi_{m-1,g} \tag{38}$$

We reindex with $m + 1$ instead of m for consistency and simplicity.

$$(1 - P_{sg})\pi_{m+2,g} = -P_{gs}(1 - P_{sg})\pi_{m+1,g} + (1 - P_{gs})\pi_{m,g} \tag{39}$$

We now have two useful recurrence relations, equations (35) and (39). Note that these relations are identical. Thus a solution of the system has the form

$$\pi_{m,s} = Ax_1^m + Bx_2^m \tag{40}$$

$$\pi_{m,g} = Cx_1^m + Dx_2^m \tag{41}$$

where A, B, C, and D are appropriate constants and where x_1 and x_2 are the roots of the quadratic equation $(1 - P_{sg})x^2 + P_{gs}(1 - P_{sg})x - (1 - P_{gs})x = 0$. We determine that x_1 and x_2 are given as follows:

$$x_1 = \frac{-P_{gs} + \sqrt{(P_{gs})^2 + \frac{4(1-P_{gs})}{1-P_{sg}}}}{2} \tag{42}$$

$$x_2 = \frac{-P_{gs} - \sqrt{(P_{gs})^2 + \frac{4(1-P_{gs})}{1-P_{sg}}}}{2} \tag{43}$$

MATLAB simulations seem to indicate that no steady state solution exists for these relations. The author has no idea why this is the case. We treat the finite case instead.

4.3 Finite Model

Just as in the trivial velocity case, the finite model here provides additional information that allow the system to be solved. If we place an upper bound K on the length of the polymer, we obtain the following three top end relations

$$\pi_{K-1,s} = P_{gs}\pi_{K-1,g} \tag{44}$$
$$\pi_{K,s} = P_{gs}\pi_{K,g} \tag{45}$$
$$\pi_{K,g} = (1 - P_{gs})(\pi_{K-1,g} + \pi_{K,g}) + P_{sg}\pi_{K,s} \tag{46}$$

which can be solved in terms of $\pi_{K-1,g}$:

$$\pi_{K-1,s} = P_{gs}\pi_{K-1,g} \tag{47}$$
$$\pi_{K,s} = \frac{1 - P_{gs}}{1 - P_{sg}}\pi_{K-1,g} \tag{48}$$
$$\pi_{K,g} = \frac{1 - P_{gs}}{P_{gs}(1 - P_{sg})}\pi_{K-1,g} \tag{49}$$

The $1, \ldots, K-2$ shortening states and the $1, \ldots, K-1$ growth states obey the normal geometric pattern discussed in the infinite case (Equations (40) and (41)).

We require four boundary conditions to obtain a unique solution; we use expressions for the π_0, $\pi_{1,g}$, $\pi_{K,s}$, and $\pi_{K,g}$ states (derived from Equations (26) through (29)).

$$\begin{aligned}
(1 - P_{sg})(Ax_1 + Bx_2) + (1 - P_{sg})(Ax_1^2 + Bx_2^2) &= P_N\pi_0 \\
(Cx_1 + Dx_2) - P_{sg}(Ax_1 + Bx_2) &= P_N\pi_0 \\
(Ax_1^K + Bx_2^K) &= P_{gs}(Cx_1^K + Dx_2^K) \\
P_{sg}(Ax_1^K + Bx_2^K) + (1 - P_{gs})(Cx_1^{K-1} + Dx_2^{K-1}) &= P_{gs}(Cx_1^K + Dx_2^K)
\end{aligned}$$

When grouped by A, B, C, and D terms, these equations generate the following matrix:

$$\left(\begin{array}{cccc|c}
(1 - P_{sg})(x_1 + x_1^2) & (1 - P_{sg})(x_2 + x_2^2) & 0 & 0 & P_N\pi_0 \\
-P_{sg}x_1 & -P_{sg}x_2 & x_1 & x_2 & P_N\pi_0 \\
x_1^K & x_2^K & -P_{gs}x_1^K & -P_{gs}x_2^K & 0 \\
P_{sg}x_1^K & P_{sg}x_2^K & (1 - P_{gs})x_1^{K-1} - P_{gs}x_1^K & (1 - P_{gs})x_2^{K-1} - P_{gs}x_2^K & 0
\end{array}\right)$$

The reduced matrix generates an analytic solution, but because it is too complicated of an expression to be useful, we recommend generating numerical results for specific parameter choices. The MATLAB code used to generate these expressions is included at the end of the report.

Example 4.3. A length distribution for specific parameter choices. Note that even with a rather small upper bound K, much of the mass is concentrated at small lengths.

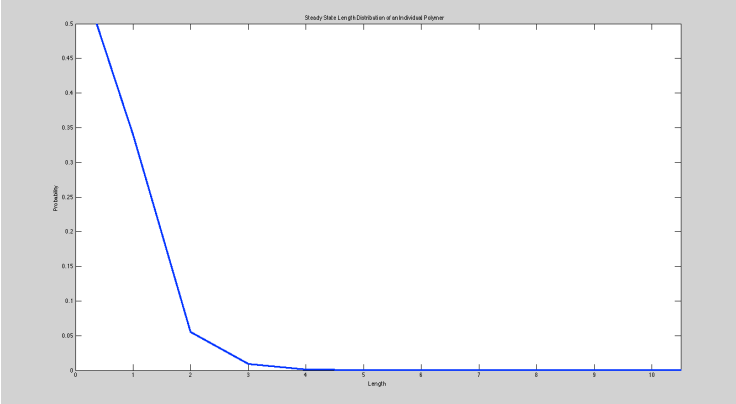


Figure 7: $P_N = 0.25, P_{sg} = 0.2, P_{gs} = 0.6$ with an upper bound $K = 20$

5 Future Work

This work represents an approach to microtubule modeling that utilizes discrete time steps and probabilistic techniques rather than continuous time differential equations models. It remains to be seen whether this approach is useful. We present several possible research directions below.

5.1 Generalized Velocities

Theorem 4.2 gave a requirement for unbounded growth if $V_g = 1$ and $V_s = 2$. A similar result may be obtained for $V_g = 1$ and $V_s = n$, in which every growth term is smaller than the average of the n subsequent growth terms. The required condition can be shown to be $P_{gs} \leq nP_{sg} - n + 1$. M. Dogterom et al. presented a model that displayed unbounded growth if $P_{gs}V_s \leq P_{sg}V_g$ [1]. As n is the ratio between V_s and V_g, we conclude that our model behaves similarly, but with the error term $-n + 1$.

Determining steady state parameters is more difficult. The set of equations for specific velocities $V_g = 1$ and $V_s = n$ generates an n-th order recurrence relation. This yields a complicated analytic solution for $n = 2$ and is more difficult to solve for larger values of n (unsolvable in fact, for $n \geq 5$ as there is no formula for roots of a general quintic formula). We reason that it is best to stick to numerical methods.

Ultimately, we seek to develop results for arbitrary growth velocities as well. As mentioned previously, we need only consider relatively prime pairs of velocities.

5.2 Non-constant Parameters

In reality, the values of V_g, P_{gs}, P_{sg}, and P_N are not constant and are instead dependent on the free subunit concentration of the system $[D_f]$. The value of V_s depends on the properties of binding kinetics and on the temperature of the system, but is independent of free concentration and can be assumed to be constant. V_g, P_{sg}, and P_N takes on higher values at higher concentrations and P_{gs} takes on higher values at lower concentration. In a Monte Carlo simulation study, Mourão et al. represented these values as varying linearly with concentration, as follows [2]:

$$\begin{aligned} P_N &= k_1[D_f] - k_2 \\ V_g &= k_3[D_f] - k_4 \\ V_s &= C_1 \\ P_{sg} &= k_5[D_f] - k_6 \\ P_{gs} &= -k_7[D_f] + k_8 \end{aligned}$$

for biologically appropriate constants k_i, C_1 (all positive). Such an assumption greatly complicates the Markov model, as the parameters change with each iteration before the system reaches steady state. It is within reach to calculate a numerical solution for specific parameter choices, but it remains to be seen whether this Markov approach can produce an analytic solution.

5.3 Multiple Polymers

We have thus far considered the behavior of a single microtubule polymer. If all polymers in the system behaved independently, then the probability length distribution for a single polymer would represent the length distribution for the entire system. However, polymer behavior is highly dependent; for example, growing polymers locally remove free subunits resulting in a lower free subunit concentration. In the past, researchers have approached this problem by considering a fixed number of nucleation sites that at any given point in time are either free or occupied. A length distribution can be determined for each site, and the aggregate provides the distribution for the overall system.

6 MATLAB Code

This section contains MATLAB code that was used for this project.

6.1 Probability Matrix - Infinite Model

This code begins with a free polymer and runs for T time steps, allowing the user to examine how the length distribution changes over time. The polymer may grow arbitrarily long. The user specifies the input parameters and the value of T. Parameters that reach steady state may be visually observed to do so,

though the code itself does not provide a proof of steady state parameters. The code recursively generates a $(T+1) \times (2V_gT+1)$ matrix P consisting of length probabilities for a single microtubule polymer. Each row consists of time steps from 0 to T and each column denotes a single Markov state. The first column is the unoccupied state with length 0, the $(2j)$-th column is the shortening state with length j, and the $(2j+1)$-st column is the growth state with length j. Each row sums to 1. To plot the length distribution, the growth and shortening states for a given length are summed together in the matrix P_{sum}.

```
clear;

% INPUTS - these can be edited

PN = .5;            %nucleation frequency
Psg = .7;           %rescue frequency
Pgs = .4;           %catastrophe frequency
Vg = 1;             %growth velocity
Vs = 1;             %shortening velocity

T = 500;            %number of steps

% DEFINING THE MATRIX

P = zeros(T+1,2*Vg*T+1);        %empty matrix framework
P(1,1) = 1;                     %initial state at time i=0
Psum = zeros(T+1,Vg*T+1);       %length matrix to plot

for i = 1:T

    %extinction
    k = 1:Vs;
    P(i+1,1) = (1-PN)*P(i,1) + sum((1-Psg)*P(i,2*k));

    %shortening
    for j = 1:T-Vs
        P(i+1,2*j) = (1-Psg)*P(i,2*j+2*Vs) + Pgs*P(i,2*j+1);
    end
    for j = T-Vs+1:T
        P(i+1,2*j) = Pgs*P(i,2*j+1);
    end

    %growth
    for j = 1:Vg-1                                      %length < Vg
        P(i+1,2*j+1) = Psg*P(i,2*j);
    end
    P(i+1,2*Vg+1) = PN*P(i,1) + Psg*P(i,2*Vg);      %length = Vg
```

```
    for j = Vg+1:i*Vg                                   %length > Vg
        P(i+1,2*j+1) = (1-Pgs)*P(i,2*j+1-2*Vg) + Psg*P(i,2*j);
    end

    %PLOT OF PROBABILITIES

    Psum(:,1) = P(:,1);
    for j = 2:T+1
        Psum(:,j) = P(:,2*j-2)+P(:,2*j-1);
    end
    x = 0:T*Vg;
    figure(1);
    plot(x,Psum(i+1,:),'LineWidth',2)
    title('Length Probability Distribution of an Individual Polymer');
    xlabel('Length');
    ylabel('Probability');
    axis([0 200 0 .05]);

end
```

6.2 Transition Matrix - Finite Model

This code generates a transition matrix P for the finite polymer model. The user specifies the input parameters, the upper bound on length K, and the number of time steps t. Any positive value of K may be specified, but note that the finite model resembles the infinite model with values as small as $K = 20$. The initial state v_0 is an empty polymer, and the code calculates the steady state vector $v_{ss} = v_0 P^t$ (t must be rather large for v_{ss} to approach steady state). The growth and shortening states are summed together and plotted as a length distribution.

```
clear;

PN = .5;            %nucleation frequency
Psg = .2;           %rescue frequency
Pgs = .6;           %catastrophe frequency
Vg = 1;             %growth velocity
Vs = 2;             %shortening velocity

K = 50;             %upper limit on length
t = 50^50;             %time steps

v0 = [1 zeros(1,2*K)];    %initial state

P = zeros(2*K+1);         %empty matrix framework
P(1,1) = 1-PN;
```

```
P(1,2*Vg+1) = PN;
for i = 1:Vs                          %early states
    P(2*i,1) = 1-Psg;
    P(2*i,2*i+1) = Psg;
    P(2*i+1,2*i) = Pgs;
    P(2*i+1,2*i+2*Vg+1) = 1-Pgs;
end

for i = Vs+1:K-Vg                     %middle states
    P(2*i,2*i-2*Vs) = 1-Psg;
    P(2*i,2*i+1) = Psg;
    P(2*i+1,2*i) = Pgs;
    P(2*i+1,2*i+2*Vg+1) = 1-Pgs;
end

for i = K-Vg+1:K                      %end states
    P(2*i,2*i-2*Vs) = 1-Psg;
    P(2*i,2*i+1) = Psg;
    P(2*i+1,2*i) = Pgs;
    P(2*i+1,2*K+1) = 1-Pgs;
end

%steady state vector
vss = v0*(P^t);
vsum(1) = vss(1);
for j = 1:K
    vsum(j+1) = vss(2*j) + vss(2*j+1);
end
x = [0:K];
figure(2);
plot(x,vsum,'LineWidth',2)
title('Steady State Length Distribution of an Individual Polymer');
xlabel('Length');
ylabel('Probability');
axis([0 K 0 .1]);
```

6.3 Finite Numerical Solution for $V_g = 1$, $V_s = 2$

This code was used to solve for specific values of matrix coefficients as discussed in Section 4.3. $V_g = 1$ and $V_s = 2$ are assumed and do not appear in the code, but the other input parameters are specified along with the upper bound on length K. The value of π_0 is first treated symbolically (`pi0`) before its numerical value is calculated (`pi`). The vector `pre` consists of the coefficients of π_0 for each length, and multiplying by π_0 gives the final length distribution.

```
clear;

PN = .25;
Pgs = .6;
Psg = .2;

K = 20;             %upper limit on length

pi0 = sym('pi0');

%roots and matrix coefficients
x1 = (-Pgs+sqrt(Pgs^2+(1-Pgs)/(1-Psg)))/2;
x2 = (-Pgs-sqrt(Pgs^2+(1-Pgs)/(1-Psg)))/2;

Mat = [(1-Psg)*(x1+x1^2) (1-Psg)*(x2+x2^2) 0 0 PN*pi0;            %pi0
    -Psg*x1 -Psg*x2 x1 x2 PN*pi0;                                %pi1g
    x1^K x2^K -Pgs*x1^K -Pgs*x2^K 0;                              %piK,s
    Psg*x1^K Psg*x2^K (1-Pgs)*x1^(K-1)-Pgs*x1^K (1-Pgs)*x2^(K-1)-Pgs*x2^K 0];
%piK,g
R=rref(Mat);

A = R(1,5)/pi0;
B = R(2,5)/pi0;
C = R(3,5)/pi0;
D = R(4,5)/pi0;

%value of pi0
for j = 1:K-2
    S(j) = A*x1^j+B*x2^j;
    G(j) = C*x1^j+D*x2^j;
    pre(j) = (A*x1^j+B*x2^j+C*x1^j+D*x2^j);
    pre(K-1) = (1+Pgs)*(C*x1^(K-1)+D*x2^(K-1));
    pre(K) = ((1-Pgs)/(1-Psg)+(1-Pgs)/(Pgs*(1-Psg)))*(C*x1^(K-1)+D*x2^(K-1));
end
pi = 1/(1+sum(pre));

pi0value = eval(pi)

%length distribution
PL(1) = pi;
for i = 1:K
    PL(i+1) = pi*pre(i);
end

sumPL = eval(sum(PL))
L = [0:length(PL)-1];
```

```
figure(3);
plot(L,PL,'LineWidth',2)
title('Steady State Length Distribution of an Individual Polymer');
xlabel('Length');
ylabel('Probability');
axis([0 length(PL) 0 .5]);
```

7 Acknowledgements

The author would like to thank Drs. Sidney Shaw, Elizabeth Housworth, and Kevin Pilgrim for their mentorship on this project. He also extends thanks to Indiana University Bloomington for facilities use and accommodation, and to the National Science Foundation for its continued support of Research Experience for Undergraduates programs. This summer's REU program was supported by NSF grant number DMS-1156515.

References

[1] M. Dogterom, S. Leibler: Physical aspects of the growth and regulation of microtubule structures. Physical Review Letters 70 (1993), no. 9, 1347–50.

[2] M. Mourão, S. Schnell, S. L. Shaw: Macroscopic simulations of microtubule dynamics predict two steady-state processes governing array morphology. Computational Biology and Chemistry 35 (2011), no. 5, 269–81.

On the Realizability of Critical Orbit Portraits

Kristin Cordwell *Selina Gilbertson* *

Abstract

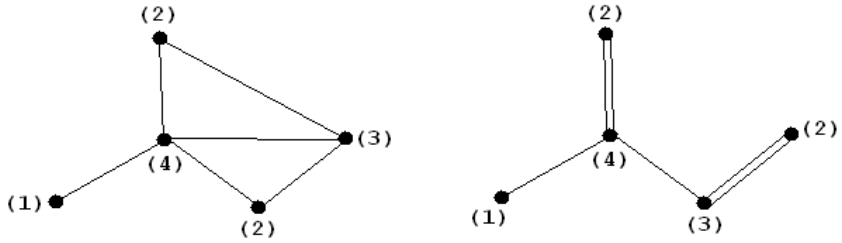
Given a rational function f with fixed critical points, the associated branch data of f refers to a set of partitions of the degree of f, with each partition determined by the local degrees of points in the preimages of a corresponding distinct critical value of f. The critical portrait of f is determined by a partition of the multiplicities of the critical points of f. It is known that a critical portrait is realizable if it may be expressed as a connected planar multigraph G, where each critical point corresponds to a vertex of G and each vertex has degree equal to the multiplicity of the associated critical point. It is also known that necessary conditions for the realizability of a critical portrait are that the associated branch data is realizable and the number of distinct critical points is at most the degree of the function. In this paper, we provide a graph-theoretic proof to conclude that these conditions are also sufficient for the realizability of a critical portrait. We also discuss code written to explore the classifications of such rational functions.

1 Introduction

A rational function $f := \frac{p(z)}{q(z)}$ where p, q are polynomials such that $\mathrm{GCD}(p, q) = 1$ has degree $d := \max(\deg(p), \deg(q))$. The iteration of rational functions gives dynamical systems on the extended complex plane, and one of the most basic cases occurs when the critical points $c \in (\mathbb{C} \cup \infty) : f'(c) = 0$ are also fixed under f. For the remainder of this paper, we will only consider rational functions with fixed critical points. In such cases, the branch data of f, which is a set of partitions of d with each partition determined by the local degrees of the critical points mapping to a distinct critical value of f, and the critical orbit portrait of f, which is given by a partition corresponding to the multiplicities of the critical points of f, encode the same information. For convenience, we note that the partition associated with a critical orbit portrait may be expressed as a Young diagram, or a union of N congruent boxes arranged into horizontal rows such that each row consists of at most the number of boxes of the row above. In particular, each critical point corresponds to a unique row of the Young diagram, and the number of boxes in the row is determined by the multiplicity of the associated critical point.

*The authors would like to thank the NSF for their support of the Indiana University REU.

We refer to a critical orbit portrait as realizable if this partition corresponds to data from a rational function f. We consider a multigraph G to be a finite collection of vertices and edges such that two vertices may be connected by arbitrarily many edges, but one vertex may not be connected to itself by an edge. A multigraph G may be mapped to a partition, which may in turn be regarded as a potential critical orbit portrait as previously described, such that each vertex of G corresponds to a unique critical point and the degree of a vertex corresponds to the multiplicity of the associated critical point. Note that two multigraphs that are not isomorphic in the plane through an orientation-preserving homeomorphism may map to the same critical orbit portrait. For example, both of the following graphs correspond to a critical orbit portrait that is a partition of 12 into $(1, 2, 2, 3, 4)$ (here, $d = 7$).



Note that the degree sequence of a connected planar multigraph with $d-1$ edges gives a partition of $2d-2$, and the multiplicities of a rational function of degree d also give a partition of $2d-2$. In particular, is known that each such multigraph partition coincides with the partition of a rational function [3, Section 5].

Previous results from complex dynamical systems have shown that several conditions are necessary for the realizability of a critical orbit portrait. First, the multiplicity of each critical point of f may be at most $d-1$. Second, by the Riemann-Hurwitz condition, the sum of the multiplicities of all of the critical points must be $2d-2$. Finally, f may have at most d fixed critical points. This last is a consequence of the Holomorphic Index Formula, which states that given a rational map of degree d with $d+1$ distinct fixed points, we have $\sum_{z \in \mathrm{Fix}(f)} \frac{1}{1-f'(z)} = 1$ [2].

We prove the following theorem, answering a recent question posed by Rafe Jones and Michelle Manes.

Theorem 1.1. *Any partition of $2d-2$ with at most d elements, each of size at most $d-1$, is realizable by a rational function.*

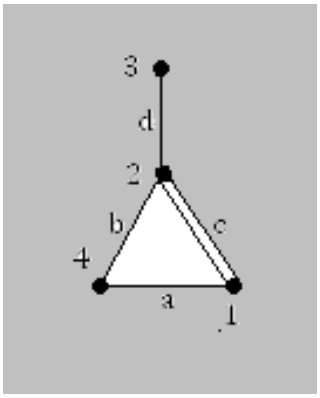
We also consider a topological approach dealing with the classification of such rational functions. In particular, we would like to determine when, given

$f, g : S^2 \to S^2$ branched coverings (orientation-preserving morphisms of degree ≥ 2), we may send f to g through some braiding of the critical points of f. We consider the invariants given by the Hurwitz data of f, which is given by:

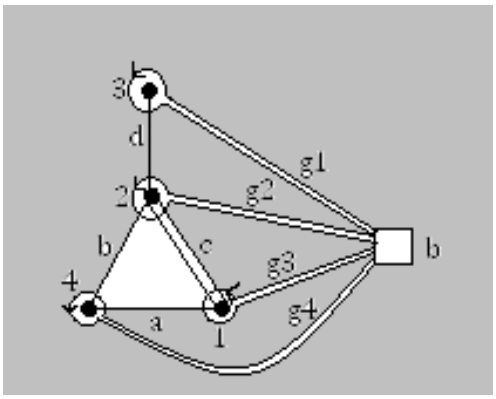
1. A labeling of critical values $\{v_1, v_2, \ldots, v_n\}$
2. A chosen base point $b \in S^2 \setminus V$
3. For each critical value v_i, the choice of a ray r_i from b to v_i such that:
(i) $r_i \cap r_j = \{b\}$ for all $i \neq j \in \{1, \ldots, n\}$
(ii) the cyclic ordering on rays given by $r_i < r_j$ for $i < j$ corresponds to the cyclic ordering on rays given by the counterclockwise ordering of the plane
4. A bijection $f^{-1}(b) \leftrightarrow \{1, 2, \ldots, d\}$

The following example illustrates how a connected planar multigraph yields a rational map. We begin with a topological description of the rational map, recalling that this data is sufficient to determine a unique rational map, up to Moebius conjugacy [3, Section 5]. From this topological description, we retrieve the ordered list of permutations determined by the Hurwitz data of our rational function f.

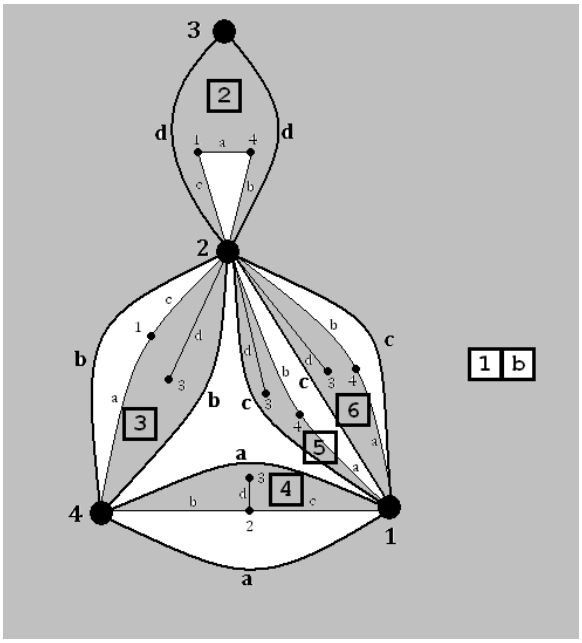
Suppose we have the critical orbit portrait of f associated with G a connected planar multigraph such that $V(G) = \{1, 2, 3, 4\}$ and each edge with distinct endpoints is labeled with a different letter, as shown. Since we have 10 edges, $d = 6$. Note that, for later convenience, we color the distinct planar regions defined by the graph.



We first choose a base point and a cyclic ordering of rays as below. We find generators of the fundamental group $\pi_1(S^2 \setminus V, b)$ by following each ray from the basepoint to the pertinent vertex, looping counterclockwise, and returning. Notice that each loop drawn is uniquely determined by the sequence of edges that it crosses. Thus, the sequences for g_1, g_2, g_3, g_4 are respectively d, dbc, ca, ab, read from left to right.



We may now topologically define the branched covering associated to our original multigraph G. The figure below shows the inverse image of each region, edge, and numbered vertex of G, with labeled objects mapping to like labeled objects. Since we chose our original basepoint to lie in the shaded region defined by G, each shaded region in the figure below contains exactly one preimage of the basepoint, and each region is labeled with a distinct element of $\{1, \ldots, d\}$. Following this figure, we find the ordered list of permutations given by a path-lifting of each generator, as $(1,2), (1,2,3,5,6), (1,6,5,4), (1,4,3)$.



It is known that f and g are Hurwitz equivalent if their corresponding lists of permutations, up to some set of elementary moves, are simultaneously conjugate in S_d. It is also known that the Hurwitz equivalence class of any branched

covering f with exactly one critical point in the preimage of each critical value is determined by the branch data of f [1].

The second half of this paper discusses code written to computationally explore this idea of Hurwitz equivalence classes. In particular, the code provides a quick verification that two non-isomorphic connected planar multigraphs corresponding to the same partition are indeed Hurwitz equivalent.

2 Realizability of a Critical Orbit Portrait

We will refer to the underlying simple graph of G by SG, where $V(SG) = V(G)$ and $v, w \in V(SG)$ are connected by exactly one edge if v, w are connected by at least one edge in G. We denote the degree of $v \in v(G)$ in G by $\deg_G(v)$, and the degree of v in SG by $\deg_{SG}(v)$. We consider a face of a simple planar graph SG to be a connected region of the plane that is bounded by edges of SG.

Definition 2.1. Given $n, d \in \mathbb{N}$, if $d \geq 3$ and $d \geq n$, we say that (n, d) has an associated admissible partition $k_1 + \ldots + k_n = 2d - 2$, with $k_i \leq d - 1$ for all $i \in \{1, 2, \ldots, n\}$. Consider a Young diagram associated with an admissible partition of (n, d) to have n rows and $2d-2$ boxes, where row i has k_i boxes.

Theorem 2.2. *Given any Young diagram associated with an admissible partition of (n, d), we may find a corresponding connected planar multigraph G such that each Young diagram row corresponds to a unique vertex of G and the number of boxes in a row is equal to the degree of the associated vertex.*

Note that the restriction $n \leq d$ is necessary. For instance, if $d = 3$, $n = 4$, then we have $2 \cdot (3-2) = 4$ boxes and 4 rows, so each row must have exactly one square. But clearly there is no way to obtain a connected planar multigraph with 4 vertices each of degree 1.

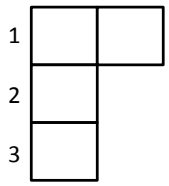
Proof. By induction on d.

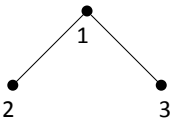
Note that, given a Young diagram associated with an admissible partition of $(2, d)$, we may find a corresponding connected planar multigraph G that consists of vertices v_1 and v_2 connected by $d - 1$ edges. For the remainder of the proof, we assume $n \geq 3$.

Base Case:

Since $n, d \geq 3$ and $n \leq d$, we first must show that this holds for $n = d$. We do so by induction on n.

If $n = d = 3$, there is only one possible Young diagram, as below. We see that the only possible corresponding multigraph is a path of length 2, which we note is connected and planar.





Assume that we may find a connected planar multigraph G for any Young diagram associated with an admissible partition of $(k-1, k-1)$. Since $|V(G)| > 2$ and G is connected, we may find v_1 and $v_2 \in V(G)$ two adjacent vertices connected by edge $e \in E(G)$.

We now must show that we may find a connected planar multigraph G' for any Young diagram associated with an admissible partition of (k, k). We note that we may construct any of the Young diagrams associated with an admissible partition of (k, k) (with $2k-2$ boxes) from one of the Young diagrams associated with an admissible partition of $(k-1, k-1)$ (with $2k-4$ boxes) by adding one row and two new boxes. Let the vertex corresponding to the added row be $v' \in V(G')$. We consider two cases.

Case I: We add one new row with two boxes.

In this case, we may construct G' from G by deleting $e \in E(G)$ and adding $e_1, e_2 \in E(G')$ such that e_1 connects v_1 and v', and e_2 connects v_2 and v'. Note that the degree of v_1 and v_2 has not changed, and v' has degree 2 in G'. Furthermore, since G was connected and planar by our inductive assumption, G' is clearly connected and planar.

Case II: We add one new row consisting of one box and also add one other box to some existing row (say the row corresponding to v_1).

In this case, we may construct G' from G by adding v' as a leaf of v_1. Note that we have increased the degree of v_1 by 1, and v' has degree 1 in G'. Again, since G was connected, G' is clearly connected. Furthermore, since we may add a leaf to any vertex of a planar multigraph and the resulting multigraph will also be planar, G planar implies that G' is planar.

Thus, given $n = d$, we may find a connected planar multigraph for any Young diagram associated with an admissible partition of (n, d). □

Now, consider $d > n$. For the purposes of induction on d, we assume that we may find a connected planar multigraph G for any Young diagram associated with an admissible partition of (n, d). We now must show that we may find such a connected planar multigraph G' for any Young diagram associated with an admissible partition of $(n, d+1)$. We note that we may construct all of the Young diagrams associated with an admissible partition of $(n, d+1)$ from the Young diagrams associated with an admissible partition of (n, d) by adding two

new boxes. We proceed by two cases.

Case I: We add one new box to each of two distinct rows of a Young diagram associated with an admissible partition of (n, d). Recall that, by assumption, $n \geq 3$. We note that this corresponds to increasing the degree of two vertices of G by 1.

Lemma 2.3. *Let G be a connected planar multigraph with $|V(G)| \geq 3$. Given $v_1, v_2 \in V(G)$ adjacent and $\deg_{SG}(v_1) \geq 2$, we may find a connected planar multigraph G' with $V(G) = V(G')$ such that*

1. $\deg_{G'}(v) = \deg_G(v)$ *for all* $v \in V(G)$, $v \neq v_1, v_2$
2. $\deg_{G'}(v_1) = -1 + \deg_G(v_1)$
3. $\deg_{G'}(v_2) = 1 + \deg_G(v_2)$.

Proof. G is a planar multigraph with $|V(G)| \geq 3$, so since SG contains no multiple edges or loops, the boundary of each face of SG contains at least three distinct vertices. Since v_1 and v_2 are adjacent, they lie on the boundary of some face F of SG. Furthermore, since $\deg_{SG}(v_1) \geq 2$, there exists $w \in V(G)$ adjacent to v_1, $w \neq v_2$, such that w also lies on ∂F.

Definition 2.4. We refer to w as a buffer vertex of v_1, v_2.

Let G' be the graph obtained from G by deleting edge v_1w and adding edge wv_2. By construction, Conditions 1 - 3 hold. Since w and v_2 are both on ∂F, we may add edge wv_2 without crossing any other edges, so G' is planar. Since G is connected, G' is connected by construction. □

The proof of Case I then follows from the following proposition:

Proposition 2.5. *Let G be a connected planar multigraph with $|V(G)| \geq 3$. Given arbitrary $s, t \in V(G)$, we may find a connected planar multigraph G' with $V(G) = V(G')$ such that*

1. $\deg_{G'}(v) = \deg_G(v)$ *for all* $v \in V(G)$, $v \neq s, t$
2. $\deg_{G'}(s) = 1 + \deg_G(s)$
3. $\deg_{G'}(t) = 1 + \deg_G(t)$.

Proof. If s and t are adjacent in G, we may construct G' by simply adding another edge between them. Assume s, t not adjacent in G.

Since G is connected, we may find $P_1 = s, a_1, a_2, \ldots, a_m, t$ a path of minimal length between s and t. Add edge sa_1, increasing the degrees of s and a_1 each by 1. Call the resulting graph H, and note that H is a connected planar multigraph with $V(G) = V(H)$.

Since a_1 is adjacent to s and a_2, $\deg_{SH}(a_1) \geq 2$. Furthermore, since P_1 is a path of minimal length, a_1 is not adjacent to any other vertices in P_1. Therefore, there exists $w \in V(H)$ adjacent to a_1 such that $w \notin V(P_1) - \{s\}$ and w, a_1, a_2 lie on the boundary of some face F of SH. We apply Lemma 1 with w as our buffer vertex of a_1, a_2 to obtain connected planar multigraph H' with $V(H) = V(H')$ such that

1. $\deg_{H'}(v) = \deg_H(v)$ for all $v \in V(H)$, $v \neq a_1, a_2$

2. $\deg_{H'}(a_1) = -1 + \deg_H(a_1)$
3. $\deg_{H'}(a_2) = 1 + \deg_H(a_2)$.

Note that we now have $\deg_{H'}(a_1) = \deg_G(a_1)$. So, if $a_2 = t$, we are done. Otherwise, since H' is connected, we may find $P_2 = a_2, b_1, \ldots, b_n, t$ a path of minimal length from a_2 to t. Note that since the path $a_1, a_2, a_3, \ldots, a_m, t \in H'$, the length of P_2 is strictly less than the length of P_1. We redefine $a_1 := a_2$, $a_2 := b_1$ and repeat the argument. Since the length of our minimal path strictly decreases with each iteration, we will eventually obtain $a_2 = t$, thus proving our claim. □

Case II: We add two new boxes to one row of a Young diagram associated with an admissible partition of (n, d). Following the given constraints, this case only pertains to rows of size $\leq d - 2$.

Suppose that our row corresponds to $v_1 \in V(G)$. Since $n, d \geq 3$ and G is planar, we may find $v_2, v_3 \in V(G)$ such that v_2 and v_1 are connected by edge e_{12}, v_2 and v_3 are connected by edge e_{23}, and v_1, v_2, v_3 lie along the boundary of a face F of SG. Now, to increase the degree of v_1 by 2 without changing the degrees of any of the other vertices of G, we may simply delete e_{23} and add an edge from v_3 to v_1 and another edge from v_2 to v_1 to form G'. Note that since G is connected, G' is connected. Furthermore, as previously argued, we may connect v_1 and v_3 without violating planarity, since they are both on ∂F.

Thus, if we can find a connected planar multigraph G for any Young diagram associated with an admissible partition of (n, d), we may find a connected planar multigraph for any Young Diagram associated with an admissible partition of $(n, d + 1)$, so our induction is complete. As desired. □

3 Hurwitz Equivalency Programs

We provide brief descriptions of the programs written to explore Hurwitz equivalency classes. The specific code may be found in Appendix A.

The function *isAdmissible* takes a list of permutations and determines whether the data satisfies the restrictions imposed by the Hurwitz data invariants. In particular, it checks whether the group generated by the permutations is transitive, whether one less than the sum of the cycle lengths is equal to $2d-2$, and whether the product of the permutations is equal to the identity.

The function *areSimulConj* takes two lists of permutations, $g1$ and $g2$, and determines whether there exists some $x \in S_d$ such that each permutation $g1[i]$ is equal to the corresponding permutation $g2[i]$ conjugated with x (that is, $g2[i] \wedge x = x^{-1} \circ g2[i] \circ x$).

The function *elementaryMove* takes g a list of permutations and performs an elementary move on some ith element, in which $g[i]$ is replaced with $g[i+1]$,

and $g[i+1]$ is replaced with $g[i] \wedge g[i+1]$.

The function *orbit* takes g a list of permutations and determines the associated Schreier graph, whose vertices consist of lists of permutations accessible from g through some sequence of elementary moves, and whose directed edges join two lists of permutations $g1, g2$ iff $g2$ is obtained from $g1$ with a single elementary move. The edges of Schreier graph are returned in the form [tail, label, head], where the label of the edge indicates which elementary move was performed on the tail to obtain the head.

The function *HC* takes two lists of permutations and determines whether they are in the same Hurwitz equivalence class by generating the orbit of the first list and checking whether the second list appears, up to simultaneous conjugation.

References

[1] Liu, F. & Osserman, B. (2008). The irreducibility of certain pure-cycle Hurwitz spaces. American Journal of Mathematics 130(6), 1687-1708.

[2] Milnor, J. (2006). Dynamics in one complex variable. Third edition. Annals of Mathematics Studies, 160. Princeton University Press, Princeton, NJ. viii+304 pp. ISBN: 978-0-691-12488-9; 0-691-12488-4.

[3] Pilgrim, K. & Tan, L. (1998). Combining rational maps and controlling obstructions. Ergodic Theory & Dynamical Systems. 18, 221-245.

A Appendix: Code

```
#isAdmissible
#Kristin Cordwell, Selina Gilbertson
#July 2012
#this function determines whether a set of generators is admissible data
#admissibility if all three conditions print "true"
######################
#
# calling sequence: isAdmissible(d, g) where g is a list of permutations in the
#  symmetric group on d symbols
# output is boolean (true or false)

isAdmissible := function(d, g)

#declare local variables
local G, length, cycleLengths, sum, prod, i;

#Group generated by permutations
G:=Group(g);

#length of list g
length := Length(g);;

######################

#first admissibility criterion: check if generated group is transitive
if not (IsTransitive(G)) then
Print("false trans");
return false;
fi;

######################

#second admissibility criterion: check Riemann-Hurwitz condition

#initialize
cycleLengths:=[];;

#determine one less than cycle length of a given permutation, add it to list
for i in [1..length] do
   cycleLengths[i] := (NrMovedPointsPerm(g[i])-1);;
od;

#check sum of elements of cycleLengths equals 2d - 2
sum:=Sum(cycleLengths);
```

```
if not sum=(2*d-2) then
   Print("false Sum");
   return false;
fi;
Print("\n");

######################

#third admissibility criterion: check product of generators is the identity

#initialize
prod:=[()];;

#compute product
for i in [1..length] do
   prod:=prod*g[i];
od;

#check
if not prod=[()] then
Print("false prod");
   return false;
fi;

return true;

end;

#areSimulConj
#Kristin Cordwell, Selina Gilbertson
#July 2012
#this function determines whether two sets of generators are simultaneously conjugate
######################
#
# calling sequence: areSimulConj(d, g1, g2), where g1, g2 are lists of permutations in the
# symmetric group on d symbols
# output is boolean (true or false)

#d the size of the symmetric group, g1 and g2 lists of permutations
areSimulConj := function(d, g1, g2)

#declare local variables
local SG, length, xinv, x, i, centr, coset, S;

#symmetric group of size d
SG := SymmetricGroup(d);;
```

```
#length of list g1
length := Length(g1);;

#initialize
x:=[];;
centr:=[];;
coset:=[];;

#check if g1 and g2 are identical
if g1 = g2 then
return true;
fi;


for i in [1..length] do

#check that g1[i] is the same cycle size as g2[i]
if RepresentativeAction(SG, g2[i], g1[i],OnPoints) = fail then
return false;
else
#find the element of SG such that g2[i] = xinv^-1 g1[i] xinv
xinv:=RepresentativeAction(SG, g2[i], g1[i],OnPoints);;
fi;

#find the element of SG such that g1[i] = x^-1 g2[i] x
x[i]:=xinv^(-1);;

#find the centralizer of g1[i]
centr[i] := Centralizer(SG, g1[i]);;

#find the right coset centr*x
coset[i] := RightCoset(centr[i], x[i]);;

#on the first iteration, initialize S
if i=1 then
S := coset[1];;
fi;

#while S nonempty, intersect coset[i] with previous elements in the list
if not Size(S)=0 then
S := Intersection(coset[i], S);;
fi;

#if S becomes empty, return false
if Size(S) = 0 then
```

```
return false;
fi;
od;

#if S nonempty
return true;

end;

#elementaryMove
#Kristin Cordwell, Selina Gilbertson
#July 2012
#this function performs an elementary move on a permutation belonging to the
# symmetric group on d elements
######################
#
# calling sequence: elementaryMove(d, g, i), where g is a list of permutations in the
# symmetric group on d symbols and i and i+1 are the elements affected
# output is the resulting permutation list

elementaryMove := function(d, g, i)

#declare local variables
local gNew, length, j;

gNew := [];;

#length of given list
length := Length(g);;

#set gNew equal to g
for j in [1..length] do
gNew[j] := g[j];;
od;

#move i, i+1 elements of g
if i < length then
   gNew[i] := g[i+1];;
   gNew[i+1] := g[i]^g[i+1];;
fi;

if i = length then
   gNew[length] := g[1];;
   gNew[1] := g[length]^g[1];;
fi;
```

```
return gNew;

end;

#orbit
#Kristin Cordwell, Selina Gilbertson
#July 2012
#this function takes a list of permutations and generates the associated orbit
# (Hurwitz class), stored as edges
######################
#
# calling sequence: orbit(d, g), where g is a list of permutations in the
# symmetric group on d symbols
# output is list of edges of the form [tail, label, head], where tail is a list of
# permutations, label is an integer dictating the element of tail on which an
# elementary move will be performed, and head is the resulting list of permutations

#d the size of the symmetric group, g a list of permutations
orbit := function(d, g)

#declare local variables
local edges, length, gAdj, h, i, j, k, l, found, orbitLength, heads;

#initialize local variables
length := Length(g);
#edges of the Hurwitz graph of the orbit
edges := [[]];;
#heads a list of lists of heads used and the number of times they appear as heads
heads := [[[]]];;
found := false;;

#first element of heads is g, appearing once
heads[1][1] := g;;
heads[1][2] := 1;;
i := 1;;
#length of heads
l := 1;;
#count number of elements in the orbit
orbitLength := 0;;

#for each element of heads
while i <= l do
Print("found", i, "\n");
for j in [1..length] do
#each possible elementary move
h := elementaryMove(d,heads[i][1],j);;
```

```
#check not already simul conj to an element of heads
for k in [1..l] do
#only check elements of heads that are not saturated
if heads[k][2] > length then
continue;
else
if areSimulConj(d, heads[k][1], h) then

found := true;;

#increment the number of times heads[k] is a head
heads[k][2] := heads[k][2] + 1;

#add new edge
Add(edges, [heads[i][1], j, heads[k][1]]);;
orbitLength := orbitLength + 1;;

break;
fi;
fi;
od;
if not found then
#add to heads
Add(heads, [h,1]);
#increment length of heads
l := l+1;;

#add new edge
Add(edges, [heads[i][1], j, h]);;
orbitLength := orbitLength + 1;;
fi;
found := false;
od;

#move to next element of heads
i := i+1;

od;

#delete the first element of edges, which is the empty set
Remove(edges, 1);;

return edges;

end;
```

```
#HurwitzClass
#Kristin Cordwell, Selina Gilbertson
#July 2012
#this function takes a list of permutations and determines whether another list is
# in the orbit of the first
######################
#
# calling sequence: hc(d, g1, g2), where g1 and g2 are lists of permutations in the
# symmetric group on d symbols
# output is true if g2 is in the orbit of g1, false otherwise

#d the size of the symmetric group, g a list of permutations
HC := function(d, g1, g2)

#declare local variables
local length, h, i, j, k, l, found, heads;

#initialize local variables
length := Length(g1);
#heads a list of lists of heads used and the number of times they appear as heads
heads := [[[]]];;
found := false;;

#check if g1, g2 are simultaneously conjugate
if areSimulConj(d,g1,g2) then
return true;
fi;

#first element of heads is g, appearing once
heads[1][1] := g1;;
heads[1][2] := 1;;
i := 1;;
#length of heads
l := 1;;

#for each element of heads
while i <= l do
Print("found", i, "\n");
for j in [1..length] do
#each possible elementary move
h := elementaryMove(d,heads[i][1],j);;
#check not already simul conj to an element of heads
for k in [1..l] do
#only check elements of heads that are not saturated
if heads[k][2] > length then
continue;
```

```
else
if areSimulConj(d, heads[k][1], h) then

found := true;;

#increment the number of times heads[k] is a head
heads[k][2] := heads[k][2] + 1;

break;
fi;
fi;
od;
if not found then

#check if g2 is simulconj
if areSimulConj(d,h,g2) then
return true;
fi;

#add to heads
Add(heads, [h,1]);

#increment length of heads
l := l+1;;

fi;
found := false;

od;

#move to next element of heads
i := i+1;

od;

return false;

end;
```

3-Dimensional Gluings of Cubes

Lindsay Martin

Abstract

A 3-dimensional gluing of cubes is constructed by gluing together n unit cubes. We glue the top of each cube to a bottom of another, the left of each cube to the right of another, and the front of each cube to the back of another. If we label the cubes 1 through n, the gluing pattern corresponds to a triple of permutations of {1, ..., n}. Given a triple of permutations, we provide a method for determining whether or not the corresponding 3-dimensional gluing of cubes is a manifold.

1 Introduction

Our goal is to study properties of 3-dimensional gluings of cubes. In this paper we focus on determining whether or not a 3-dimensional gluing of cubes is a 3-manifold. The 2-D analogue, a square-tiled surface, is always a 2-manifold. However, that is not the case in 3-D. In section 2, we formally define square-tiled surfaces, 3-dimensional gluings of cubes, and what it means for a 3-dimensional gluing of cubes to be a manifold. In section 3, we use Euler's characteristic to reduce the problem to counting the vertices, edges, and faces in a 3-dimensional gluing of cubes. Section 4 shows us how to algorithmically count the vertices, edges, and faces. Once we have computed the number of vertices, edges, and faces, the problem is solved. An example of 3-dimensional gluing of cubes that is a manifold is given in section 5.

Remark 1.1. In order to be consistent, we will always multiply permutations from left to right in this paper. For example, let σ =(1 2 4) and τ =(2 5 4). Then, $\sigma\tau$ = (1 2 4)(2 5 4) = (1 5 4).

2 Preliminary Definitions

2.1 Square-Tiled Surfaces and 3-Dimensional Gluings of Cubes

Definition 2.1. Let $S_i = [2i, 2i+1] \times [0, 1]$. Let Z be the disjoint union of the squares $S_1, S_2, S_3, \ldots, S_n$. Let σ, τ be permutations of $\{1, \ldots, n\}$. For each $i \in \{1, \ldots, n\}$, let

$$A_{\sigma,i} = \{((2i, y), (2\sigma(i) + 1, y)) \mid y \in [0, 1]\}$$
$$A_{\tau,i} = \{((2i + x, 1), (2\tau(i) + x, 0)) \mid x \in [0, 1]\}.$$

Let $A \subset Z \times Z$ be the equivalence relation generated by the union

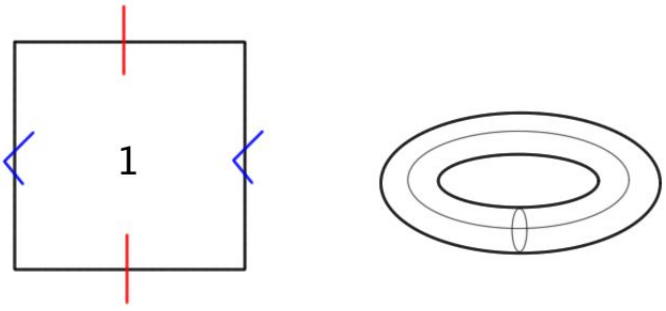
$$\bigcup_{i=1}^{n} A_{\sigma,i} \cup A_{\tau,i}$$

A *square-tiled surface* $X(\sigma, \tau)$, i.e., the gluing space, is the set of equivalence classes.

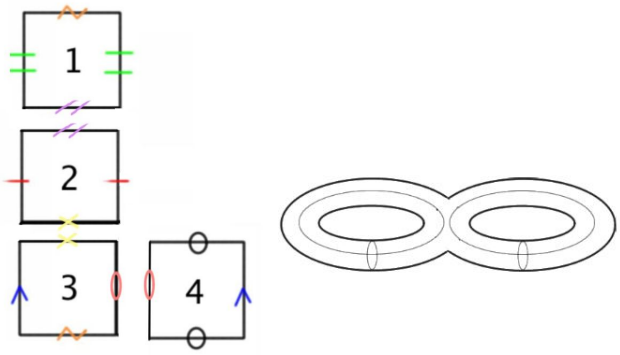
A vertex in $X(\sigma, \tau)$ is an equivalence class that contains a vertex of one of the n squares. An edge in $X(\sigma, \tau)$ is an equivalence class that contains an edge of one of the n squares.

Basically, a square-tiled surface, $X(\sigma, \tau)$, is the result of gluing together n unit squares. We identify each left edge with a right edge and each top edge with a bottom edge. If we label the squares $1, \ldots, n$, the gluing pattern corresponds to a pair of permutations, (σ, τ), of $\{1, \ldots, n\}$.

Example 2.2. $X((1), (1))$, i.e. a torus



Example 2.3. $X(\sigma, \tau)$, where $\sigma = (3\ 4)$ and $\tau = (1\ 3\ 2)$., i.e., a double torus



Definition 2.4. Let $C_i = [2i, 2i+1] \times [0, 1] \times [0, 1]$. Let Z be the disjoint union of the cubes $C_1, C_2, C_3, \ldots, C_n$. Let σ, τ, ω be permutations of $\{1, 2, \ldots, n\}$.

For each $i \in \{1, \ldots, n\}$, let

$$A_{\sigma,i} = \{((2i, y, z), (2\sigma(i) + 1, y, z) \mid y, z \in [0, 1]\}$$
$$A_{\tau,i} = \{((2i + x, y, 1), (2\tau(i) + x, y + 1, 0) \mid x, y \in [0, 1]\}$$
$$A_{\omega,i} = \{((2i + x, 0, z), (2\omega(i) + x, 1, z) \mid x, z \in [0, 1]\}.$$

Let $A \subset Z \times Z$ be the equivalence relation generated by the union

$$\bigcup_{i=1}^{n} A_{\sigma,i} \cup A_{\tau,i} \cup A_{\omega,i}$$

A *3-dimensional gluing of cubes* $X(\sigma, \tau, \omega)$, i.e., the gluing space, is the set of equivalence classes.

A vertex in $X(\sigma, \tau, \omega)$ is an equivalence class that contains a vertex of one of the n cubes. An edge in $X(\sigma, \tau, \omega)$ is an equivalence class that contains an edge of one of the n cubes. A face in $X(\sigma, \tau, \omega)$ is an equivalence class that contains a face of one of the n cubes.

Basically, a 3-dimensional gluing of cubes, $X(\sigma, \tau, \omega)$, is the result of gluing together n unit cubes. We identify each left face with a right face, each top face with a bottom face, and each front face to a back face. If we label the cubes $1, \ldots, n$, the gluing pattern corresponds to a triple of permutations, (σ, τ, ω), of $\{1, \ldots, n\}$.

2.2 Locally Euclidean 3-Dimensional Gluings of Cubes

Definition 2.5. A 3-dimensional gluing of cubes is said to be *locally Euclidean*, and therefore a manifold, if and only if for each point in the space, there is a neighborhood of the point which is homeomorphic to Euclidean space[1].

Remark 2.6. The rest of the paper will give a systematic way of determining whether or not a 3-dimensional gluing of cubes is a manifold.

3 Euler's Characteristic for 3-Dimensional Gluings of Cubes

We will use the Euler's characteristic of a 3-dimensional gluing of cubes in order to determine whether or not $X(\sigma, \tau, \omega)$ is a manifold.

First, we have a theorem about 3-dimensional gluings of tetrahedra.

Definition 3.1. The Euler characteristic of a 3-dimensional gluing of tetrahedra is

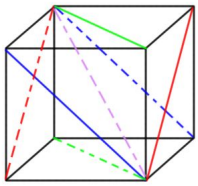
$$\chi(\text{ 3-dimensional gluing of tetrahedra}) = v - e + f - t$$

where v, e, f, and t are the number of vertices, edges, faces, and tetrahedra in the 3-dimensional gluing of tetrahedra, respectively [2].

Theorem 3.2. *A three-dimensional gluing of tetrahedra is a three-manifold if and only if its Euler characteristic is zero.*

Proof. See [2]. □

We can divide a cube into 6 tetrahedra.



Thus, we can apply theorem 3.2 to a 3-dimensional gluing of cubes.

Proposition 3.3. *The Euler characteristic of $X(\sigma, \tau, \omega)$ is*

$$\chi(X(\sigma, \tau, \omega)) = v - e + f - n$$

where v, e, f and n are the number of vertices, edges, faces, and cubes in $X(\sigma, \tau, \omega)$, respectively.

Proof. Let v, e, f, n be the number of vertices, edges, faces, and cubes in $X(\sigma, \tau, \omega)$, respectively. Divide each cube in $X(\sigma, \tau, \omega)$ into 6 tetrahedra as above. We will have the same gluing space except it will be divided into tetrahedra instead of cubes. Let v', e', f', t be the number of vertices, edges, faces, and tetrahedra in the gluing of tetrahedra. We can see that $v' = v$, $e' = e + f + n$, $f' = 2f + 6n$, and $t = 6n$. So by definition 3.1,

$$\begin{aligned}\chi(X(\sigma, \tau, \omega)) &= v' - e' + f' - t \\ &= v - (e + f + n) + (2f + 6n) - 6n \\ &= v - e + f - n.\end{aligned}$$

□

Theorem 3.4. *A three-dimensional gluing of cubes is a three-manifold $X(\sigma, \tau, \omega)$ is a three-manifold if and only if its Euler characteristic is zero.*

Proof. Divide each cube into 6 tetrahedra, and apply theorem 3.2. □

Thus, if we can compute v, e, and f, then we can determine whether or not $X(\sigma, \tau, \omega)$ is a manifold by computing its Euler characteristic.

4 Computing $v, e,$ and f in $X(\sigma, \tau, \omega)$

4.1 Computing f

First, we compute f. Each cube has 6 faces, and since the faces are glued in pairs,

$$f = \frac{6n}{2} = 3n.$$

4.2 Computing e

We want to compute the number of edges in a 3-dimensional gluing $X(\sigma, \tau, \omega)$. Of course, we can count the edges directly for $X(\sigma, \tau, \omega)$ for small n. However, we would like an algorithm for computing the number of edges given σ, τ, and ω for any n.

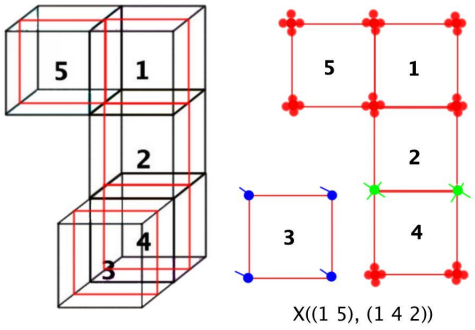
Proposition 4.1. *The number of edges in $X(\sigma, \tau, \omega)$ is equal to the sum of of the number of vertices in $X(\sigma, \tau), X(\tau, \omega), X(\sigma, \omega)$.*

Proof. Let $X(\sigma, \tau, \omega)$ be a 3-dimensional gluing of cubes. Let square-tiled surface $X(\sigma, \tau)$ is naturally identified with the subset of $X(\sigma, \tau, \omega)$ corresponding to the union $\bigcup_{i=1}^{n} \widetilde{S}_i$ where

$$\widetilde{S}_i = [2i, 2i+1] \times \{\frac{1}{2}\} \times [0, 1].$$

The edges in $X(\sigma, \tau, \omega)$ parallel to the vector (1, 0, 0) are in one to one correspondence with the vertices of $X(\sigma, \tau)$. Similarly, the edges of $X(\sigma, \tau, \omega)$ parallel to the vector (0, 0, 1) are in one-to-one correspondence with the vertices of $X(\sigma, \omega)$, and the edges parallel to (0, 1, 0) correspond to the vertices of $X(\tau, \omega)$. We must count the vertices in the $X(\sigma, \tau), X(\sigma, \omega)$ and $X(\tau, \omega)$ to count all the edges in $X(\sigma, \tau, \omega)$. Thus, the number of edges in $X(\sigma, \tau, \omega)$ is equal to the sum of of the number of vertices in $X(\sigma, \tau), X(\tau, \omega), X(\sigma, \omega)$. □

Example 4.2. $X(\sigma, \tau, \omega)$, where $\sigma = (1\ 5)$, $\tau = (1\ 4\ 2)$, and $\omega = (1\ 4\ 3)$

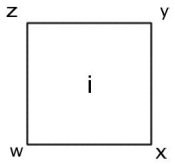


Thus, in example 4.2, there are 3 vertices in $X(\sigma, \tau)$. Therefore, these three vertices account for 3 of the edges in $X(\sigma, \tau, \omega)$.

We want to be able to compute the number of vertices in a square-tiled surface, so that we can count the number of edges in a 3-dimensional gluing of cubes.

4.2.1 Counting Vertices in a Square-Tiled Surface

For notational purposes, we will label the vertices of a square as follows:



The left bottom vertex is labeled w_i, right bottom vertex is labeled x_i, right top vertex is labeled y_i, and left top vertex is labeled z_i. So the set of vertices of the squares is $\{v_i \mid v \in \{w, x, y, z\} \text{ and } i \in \{1, \ldots, n\}\}$.

First, we begin by defining an equivalence relation generated by a set B.

Definition 4.3. Let B be any subset $X \times X$. We define *the equivalence relation generated by* B to be the smallest equivalence relation in $X \times X$ that contains B [1].

Proposition 4.4. *[1] Let $B \subset V \times V$ be a relation. Let $A \subseteq V \times V$ be the set of ordered pairs (x, y) such that there exists a sequence $x_1, \ldots, x_n$ with $x = x_1$, $y = x_n$ and for each $i = 1, \ldots, n-1$ we have either:*

1. *(x_i, x_{i+1}) belongs to B,*
2. *(x_{i+1}, x_i) belongs to B, or*
3. *$x_{i+1} = x_i$*

A is the equivalence relation generated by B.

Proof. To prove this claim, we need to show

- $A \supset B$
- A is an equivalence relation
- If A' is any other equivalence relation that contains A, then $A \subset A'$.

Let $(x, y) \in B$, $x_1 = x$, and $x_2 = y$. Thus, there exists a sequence x_1, x_2 such that $(x_1, x_2) \in B$. $\Rightarrow (x, y) \in A$.

Let $(x, x) \in V \times V, x_1 = x$, and $x_2 = x$. Thus, there exists a sequence x_1, x_2 such that $x_1 = x_2$. $\Rightarrow (x, x) \in A$.

Let $(x, y) \in A$. Then there exists a sequence $x_1, \ldots, x_n$ such that $x = x_1$, $y = x_n$ and for each $i = 1, \ldots, n-1$ either (1), (2), or (3) above is true. There is a sequence $x'_{j_n}, \ldots, x'_{j_1}$ such that $x'_{j_n} = y$ and $x'_{j_1} = x$ and for each $j_i = 1, \ldots, n-1$ either (1), (2), or (3) above is true if we let $j_i = n + 1 - i$. Thus, $(y, x) \in A$.

Let $(x, y) \in A$ and $(y, z) \in A$. Then there exist sequences $x_1, \ldots, x_n$ such that $x = x_1$, $y = x_n$ and for each $i = 1, \ldots, n-1$either (1), (2), or (3) above is true and $w_1, \ldots, w_m$ such that $y = w_1$, $z = w_m$ and for each $j = 1, \ldots, m-1$ either (1), (2), or (3) above is true. If we let each $w_j = x_{j+n}$, then there is a sequence $x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+m}$ such that $x = x_1$ and $z = x_{n+m}$ and for each $l = 1, \ldots, (n+m)-1$ either (1), (2), or (3) above is true since $x_n = x_{n+1} = y$. Thus, $(x, z) \in A$.

Suppose that $(x, y) \in A$. Then there exists a sequence $x_1, \ldots, x_n$ $l = 1, \ldots, n-1$ with $x = x_1$, $x_n = y$ and for each $i = 1, \ldots, n-1$ such that either (1), (2), or (3) above is true. Since $B \subset A'$, and since A' is an equivalence relation and hence symmetric and reflexive, we have $(x_i, x_{i+1}) \in A'$ for each i. Since A' is transitive and by induction we find that $(x, y) \in A'$. $\square$

The equivalence relation, $\langle B \rangle$, on the set of vertices of the n squares is the equivalence relation generated by B, where $B = \{(w_i, x_j) \mid \sigma(i) = j\} \cup \{(x_i, y_j) \mid \tau^{-1}(i) = j\} \cup \{(y_i, z_j) \mid \sigma^{-1}(i) = j\} \cup \{(z_i, w_j) \mid \tau(i) = j\}$. The equivalence classes of $\langle B \rangle$ are the vertices of $X(\sigma, \tau)$.

The equivalence relation on the set of vertices of the n squares gives us multiple "paths" between two equivalent vertices. The following proposition defines one of the "paths" between equivalent vertices.

Definition 4.5. $\Theta := \sigma\tau^{-1}\sigma^{-1}\tau$

Proposition 4.6. *1.* $w_i \sim w_j \Leftrightarrow \exists m \in \mathbb{Z}$ *such that* $\Theta^m(i) = j$.

2. $w_i \sim x_j \Leftrightarrow \exists m \in \mathbb{Z}$ *such that* $\Theta^m\sigma(i) = j$.

3. $w_i \sim y_j \Leftrightarrow \exists m \in \mathbb{Z}$ *such that* $\Theta^m\sigma\tau^{-1}(i) = j$.

4. $w_i \sim z_j \Leftrightarrow \exists m \in \mathbb{Z}$ *such that* $\Theta^m\tau^{-1}(i) = j$.

5. $x_i \sim x_j \Leftrightarrow \exists m \in \mathbb{Z}$ *such that* $\sigma^{-1}\Theta^m\sigma(i) = j$. .

6. $x_i \sim y_j \Leftrightarrow \exists m \in \mathbb{Z}$ *such that* $\sigma^{-1}\Theta^m\sigma\tau^{-1}(i) = j$.

7. $x_i \sim z_j \Leftrightarrow \exists m \in \mathbb{Z}$ *such that* $\sigma^{-1}\Theta^m\tau^{-1}(i) = j$.

8. $y_i \sim y_j \Leftrightarrow \exists m \in \mathbb{Z}$ *such that* $\tau\sigma^{-1}\Theta^m\sigma\tau^{-1}(i) = j$.

9. $y_i \sim z_j \Leftrightarrow \exists m \in \mathbb{Z}$ *such that* $\tau\sigma^{-1}\Theta^m\tau^{-1}(i) = j$.

10. $z_i \sim z_j \Leftrightarrow \exists m \in \mathbb{Z}$ *such that* $\tau\Theta^m\tau^{-1}(i) = j$.

Note: We can use symmetry to show that for each $v_i \sim v_j'$, $v_j' \sim v_i$ implies the same result as $v_i \sim v_j'$.

Proof. Suppose $w_i \sim w_j$. Then by proposition 4.4, there exists a sequence $v_1, \ldots, v_l$ with $v_1 = w_i$ and $v_l = w_j$ such that for each $k = 1, \ldots, l-1$ either (v_k, v_{k+1}) belongs to B, (v_{k+1}, v_k) belongs to B, or $v_{k+1} = v_k$. If for any $k = 1, \ldots, l-1$ $v_k = v_{k+1}$, then we can alter the sequence by deleting v_{k+1} from the sequence. $v_1, \ldots, v_k, v_{k+2}, \ldots, v_l$ still satisfies the necessary properties because if $v_k = v_{k+1}$, then either $(v_k, v_{k+2}) \in B, (v_{k+2}, v_k) \in B$, or $v_k = v_{k+2}$.

So now we consider the case where $(w_i, v_2) \in B$. $\Rightarrow v_2 = x_{\sigma(i)}$. Then either $(x_{\sigma(i)}, v_3) \in B$ or $(v_3, x_{\sigma(i)}) \in B$. If $(v_3, x_{\sigma(i)}) \in B$, then $v_3 = w_i$ which lead us back to the starting vertex and we would repeat the previous process (backtrack). So it is enough to only consider the case $(x_{\sigma(i)}, v_3) \in B$. $\Rightarrow v_3 = y_{\sigma\tau^{-1}(i)}$. Next, since we don't want to consider backtracking, $(y_{\sigma\tau^{-1}(i)}, v_4) \in B$. $\Rightarrow v_4 = z_{\sigma\tau^{-1}\sigma^{-1}(i)}$. $\Rightarrow (z_{\sigma\tau^{-1}\sigma^{-1}(i)}, v_5) \in B$. $\Rightarrow v_5 = w_{\sigma\tau^{-1}\sigma^{-1}\tau(i)}$. If $w_{\sigma\tau^{-1}\sigma^{-1}\tau(i)} = w_j$, then we are done and we have $\sigma\tau^{-1}\sigma^{-1}\tau(i) = \Theta(i) = j$. Otherwise, we repeat the process until we have $\Theta^m(i) = j$ for some $m \in \mathbb{Z}$.

Next, suppose that $(v_2, w_i) \in B$. we can analogously repeat the process above to find that $\Theta^{-m}(i) = j$.

Thus, $\exists m \in \mathbb{Z}$ such that $\Theta^m(i) = j$.

Conversely, suppose $\exists m \in \mathbb{Z}$ such that $\Theta^m(i) = j$. Then there is a sequence

$$w_i, x_{\sigma(i)}, y_{\sigma\tau^{-1}(i)}, z_{\sigma\tau^{-1}\sigma^{-1}(i)}, w_{\sigma\tau^{-1}\sigma^{-1}\tau(i)}, \ldots, w_{\Theta^m} = w_j$$

that satisfies the condition $(v_k, v_{k+1}) \in B$ $\forall k = 1, \ldots l-1$. By proposition 4.4, $w_i \sim w_j$.

An analogous argument applies to the other 9 cases. □

Remark 4.7. This theorem gives a way to count the number of vertices in a square-tiled surface only given σ and τ.

Theorem 4.8. *There is a bijective correspondence between the vertices in* $X(\sigma, \tau)$ *and the orbits of the action of* $\langle\Theta\rangle$ *on* $\{1, \ldots, n\}$.

Proof. Define $\varphi : \{v_i \mid v \in \{w, x, y, z\} \text{ and } i \in \{1, \ldots, n\}\} \mapsto \{\langle\Theta\rangle j \mid j \in \{1, \ldots, n\}\}$ by $v_i \to \langle\Theta\rangle j$ where

$$j = \begin{cases} i & \text{if } v = w, \\ \sigma^{-1}(i) & \text{if } v = x, \\ \tau\sigma^{-1}(i) & \text{if } v = y, \\ \tau(i) & \text{if } v = z. \end{cases}$$

We must show that φ is well defined, injective, and surjective. Generally, we will say that $v_i \mapsto \langle\Theta\rangle j$ where $j = h(i)$ and $h \in H = \{(1), \sigma^{-1}, \tau\sigma^{-1}, \tau\}$. Note that h depends on v.

To show that φ is well defined, we must that if $v_i \sim v'_{i'}$, then $\varphi(v_i) = \varphi(v'_{i'})$. So assume $v_i \sim v'_{i'}$. $\varphi(v_i) = \langle\Theta\rangle j$, and $\varphi(v'_{i'}) = \langle\Theta\rangle j'$. By proposition 4.6, there exists an r such that $r = s\Theta^m t$ and $r(i) = i'$ where $s, t^{-1} \in H$ and $m \in \mathbb{Z}$. We know $\varphi(v_i) = \langle\Theta\rangle j = \langle\Theta\rangle h(i)$ and $\varphi(v'_{i'}) = \varphi(v'_{r(i)}) = \langle\Theta\rangle j' = \langle\Theta\rangle h'(i')$. Note that $s = h$ and $t = (h')^{-1}$. So $r(i) = s\Theta^m t(i) = h\Theta^m(h')^{-1}(i) = i'$. $\Rightarrow$ $h\Theta^m(i) = h'(i')$. $\Rightarrow$ $\langle\Theta\rangle h(i) = \langle\Theta\rangle h'(i')$. $\Rightarrow$ $\langle\Theta\rangle j = \langle\Theta\rangle j'$. Thus, φ is well defined.

Next, we will show the φ is injective.

Assume $\varphi(v_i) = \varphi(v'_{i'})$.$\varphi(v_i) = \langle\Theta\rangle j = \langle\Theta\rangle h(i)$ and $\varphi(v'_{i'}) = \langle\Theta\rangle j' = \langle\Theta\rangle h'(i')$. Thus, $\langle\Theta\rangle h(i) = \langle\Theta\rangle h'(i')$. $\Rightarrow \exists m \in \mathbb{Z}$ such that $h\Theta^m(i) = h'(i')$. $\Rightarrow$ $h\Theta^m(h')^{-1}(i) = i'$. Propostion 4.6 implies $v_i \sim v'_{i'}$.

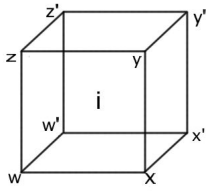
Finally, we will show that φ is surjective. Let $i = j$ and $v = w$, then $\forall j \in \{1, \ldots, n\}$ $\varphi(w_j) = O_j$. Thus, φ is surfective.

□

4.3 Computing the number of vertices, v, in $X(\sigma, \tau, \omega)$

Now, we have a way to compute f and e. Thus, once we compute v, then we will be able to determine whether or not $X(\sigma, \tau, \omega)$ is a manifold by computing its Euler characteristic. First, we want to define the equivalence relation on the set of vertices.

For notational purposes, we will label the vertices of a cube as follows:



So the set of vertices of the cubes is $\{v_i \mid v \in \{w, x, y, z, w', x', y', z'\}$ and $i \in \{1, \ldots, n\}\}$.

The equivalence relation, $\langle B\rangle$, on the set of vertices in of the n cubes is the equivalence relation generated by B, where

$$\begin{aligned} B =& \{(w_i, x_j) \mid \sigma(i) = j\} \cup \{(x_i, x'_j) \mid \omega(i) = j\} \cup \{(x'_i, y'_j) \mid \tau^{-1}(i) = j\} \\ & \cup \{(y'_i, y_i) \mid \omega^{-1}(i) = j\} \cup \{(y_i, x_j) \mid \tau(i) = j\} \cup \{(x_i, w_j) \mid \sigma^{-1}(i) = j\} \\ & \cup \{(x_i, y_j) \mid \tau^{-1}(i) = j\} \cup \{(y_i, z_j) \mid \sigma^{-1}(i) = j\} \cup \{(z_i, w_j) \mid \tau(i) = j\} \\ & \cup \{(w_i, z_j) \mid \tau^{-1}(i) = j\} \cup \{(z_i, y_j) \mid \sigma(i) = j\} \cup \{(y_i, y'_j) \mid \omega(i) = j\} \\ & \cup \{(y'_i, z'_j) \mid \sigma^{-1}(i) = j\} \cup \{(z'_i, z_j) \mid \omega^{-1}(i) = j\} \cup \{(z_i, z'_j) \mid \omega(i) = j\} \\ & \cup \{(z'_i, w'_j) \mid \tau(i) = j\} \cup \{(w'_i, w_j) \mid \omega^{-1}(i) = j\} \cup \{(w_i, w'_j) \mid \omega(i) = j\} \\ & \cup \{(w'_i, x'_j) \mid \sigma(i) = j\} \cup \{(x'_i, x_j) \mid \omega^{-1}(i) = j\}. \end{aligned}$$

The following definitions for the γ_i's are analogous to our definition of Θ in the square-tiled surface.

Definition 4.9.

$$\gamma_1 = \sigma\omega\tau^{-1}\omega^{-1}\tau\sigma^{-1}.$$
$$\gamma_2 = \sigma\tau^{-1}\sigma^{-1}\tau.$$
$$\gamma_3 = \sigma\tau^{-1}\omega\sigma^{-1}\omega^{-1}\tau.$$
$$\gamma_4 = \tau^{-1}\omega\tau\omega^{-1}.$$
$$\gamma_5 = \omega\sigma\omega^{-1}\sigma^{-1}.$$

The equivalence relation on the set of vertices gives us multiple "paths" between two equivalent vertices. The following proposition defines one of the "paths" between equivalent vertices in $X(\sigma, \tau, \omega)$.

Proposition 4.10. *Let* $G = \langle \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5 \rangle$.

1. $w_i \sim w_j \Leftrightarrow \exists g \in G$ *such that* $g(i) = j$.
2. $w_i \sim x_j \Leftrightarrow \exists g \in G$ *such that* $g\sigma(i) = j$.
3. $w_i \sim y_j \Leftrightarrow \exists g \in G$ *such that* $g\sigma\tau^{-1}(i) = j$.
4. $w_i \sim z_j \Leftrightarrow \exists g \in G$ *such that* $g\tau^{-1}(i) = j$.
5. $w_i \sim w'_j \Leftrightarrow \exists g \in G$ *such that* $g\omega(i) = j$.
6. $w_i \sim x'_j \Leftrightarrow \exists g \in G$ *such that* $g\sigma\omega(i) = j$.
7. $w_i \sim y'_j \Leftrightarrow \exists g \in G$ *such that* $g\sigma\omega\tau^{-1}(i) = j$.
8. $w_i \sim z'_j \Leftrightarrow \exists g \in G$ *such that* $g\tau^{-1}\omega(i) = j$.
9. $x_i \sim x_j \Leftrightarrow \exists g \in G$ *such that* $\sigma^{-1}g\sigma(i) = j$.
10. $x_i \sim y_j \Leftrightarrow \exists g \in G$ *such that* $\sigma^{-1}g\sigma\tau^{-1}(i) = j$.
11. $x_i \sim z_j \Leftrightarrow \exists g \in G$ *such that* $\sigma^{-1}g\tau^{-1}(i) = j$.
12. $x_i \sim w'_j \Leftrightarrow \exists g \in G$ *such that* $\sigma^{-1}g\omega(i) = j$.
13. $x_i \sim x'_j \Leftrightarrow \exists g \in G$ *such that* $\sigma^{-1}g\sigma\omega(i) = j$.
14. $x_i \sim y'_j \Leftrightarrow \exists g \in G$ *such that* $\sigma^{-1}g\sigma\omega\tau^{-1}(i) = j$.
15. $x_i \sim z'_j \Leftrightarrow \exists g \in G$ *such that* $\sigma^{-1}g\tau^{-1}\omega(i) = j$.
16. $y_i \sim y_j \Leftrightarrow \exists g \in G$ *such that* $\tau\sigma^{-1}g\sigma\tau^{-1}(i) = j$.
17. $y_i \sim z_j \Leftrightarrow \exists g \in G$ *such that* $\tau\sigma^{-1}g\tau^{-1}(i) = j$

18. $y_i \sim x'_j \Leftrightarrow \exists g \in G$ *such that* $\tau\sigma^{-1}g\sigma\omega(i) = j$.

19. $y_i \sim y'_j \Leftrightarrow \exists g \in G$ *such that* $\tau\sigma^{-1}g\sigma\omega\tau^{-1}(i) = j$.

20. $y_i \sim z'_j \Leftrightarrow \exists g \in G$ *such that* $\tau\sigma^{-1}g\tau^{-1}\omega(i) = j$.

21. $z_i \sim z_j \Leftrightarrow \exists g \in G$ *such that* $\tau g\tau^{-1}(i) = j$.

22. $z_i \sim x'_j \Leftrightarrow \exists g \in G$ *such that* $\tau g\sigma\omega(i) = j$.

23. $z_i \sim y'_j \Leftrightarrow \exists g \in G$ *such that* $\tau g\sigma\omega\tau^{-1}(i) = j$.

24. $z_i \sim z'_j \Leftrightarrow \exists g \in G$ *such that* $\tau g\tau^{-1}\omega(i) = j$.

25. $w'_i \sim w'_j \Leftrightarrow \exists g \in G$ *such that* $\omega^{-1}g\omega(i) = j$.

26. $w'_i \sim x'_j \Leftrightarrow \exists g \in G$ *such that* $\omega^{-1}g\sigma\omega(i) = j$.

27. $w'_i \sim y'_j \Leftrightarrow \exists g \in G$ *such that* $\omega^{-1}g\sigma\omega\tau^{-1}(i) = j$.

28. $w'_i \sim z'_j \Leftrightarrow \exists g \in G$ *such that* $\omega^{-1}g\tau^{-1}\omega(i) = j$.

29. $x'_i \sim x'_j \Leftrightarrow \exists g \in G$ *such that* $\omega^{-1}\sigma^{-1}g\sigma\omega(i) = j$.

30. $x'_i \sim y'_j \Leftrightarrow \exists g \in G$ *such that* $\omega^{-1}\sigma^{-1}g\sigma\omega\tau^{-1}(i) = j$.

31. $x'_i \sim z'_j \Leftrightarrow \exists g \in G$ *such that* $\omega^{-1}\sigma^{-1}g\tau^{-1}\omega(i) = j$.

32. $y'_i \sim y'_j \Leftrightarrow \exists g \in G$ *such that* $\tau\omega^{-1}\sigma^{-1}g\sigma\omega\tau^{-1}(i) = j$.

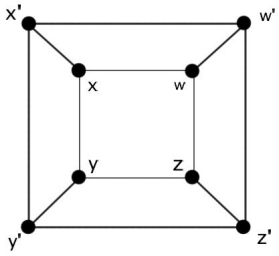
33. $y'_i \sim z'_j \Leftrightarrow \exists g \in G$ *such that* $\tau\omega^{-1}\sigma^{-1}g\tau^{-1}\omega(i) = j$.

34. $z'_i \sim z'_j \Leftrightarrow \exists g \in G$ *such that* $\omega^{-1}\tau g\tau^{-1}\omega(i) = j$.

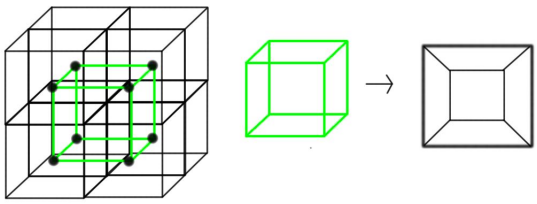
Proof. Suppose $w_i \sim w_j$. Then by proposition 4.4, there exists a sequence $v_1, \ldots, v_l$ with $v_1 = w_i$ and $v_l = w_j$ such that for each $k = 1, \ldots, l-1$ either $(v_k, v_{k+1}) \in B, (v_{k+1}, v_k) \in B$, or $v_k = v_{k+1}$. Our goal is to show that given any sequence starting with w_i and ending with w_j, we can alter the sequence to make a new equivalent sequence that is a word in G.

We will to represent the sequence in terms of the cycles beginning at w in the following cubic graph.

Note: We will allow the cycle to return to w multiple times.



This cubic graph is a dual graph of the gluing space of a 3-dimensional gluing of cubes.



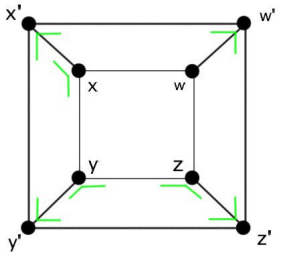
Here is how we will define the sequence $w_i, \ldots, w_j$ in terms of cycles in the cubic graph. By the properties of this sequence, there is a word $r \in \langle \sigma, \tau, \omega \rangle$ such that $r(i) = j$. We want to take the simplest form of this word, i.e., we simplify r such that after the simplification no section can be simplified to the identity permutation.Now we have a simplified version of r, call it r'. r' represents an equivalent sequence of shorter or the same length as $w_i, \ldots, w_j$. The sequence representing r' can be represented in terms of elements in $V = \{w, w', x, x', y, y', z, z'\}$. If a cycle returns to w before the end we will insert another w whenever this occurs. For example,we defined to be $\gamma_1 = \sigma\omega\tau^{-1}\omega^{-1}\tau\sigma^{-1}$. The sequence for γ_1 is $w_i, x_{\sigma(i)}, x'_{\sigma\omega(i)}, y'_{\sigma\omega\tau^{-1}(i)}, y_{\sigma\omega\tau^{-1}\omega^{-1}(i)}$, $x_{\sigma\omega\tau^{-1}\omega^{-1}\tau(i)}, w_{\sigma\omega\tau^{-1}\omega^{-1}\sigma^{-1}(i)}$. Thus, $\gamma_1 = w, x, x', y', y, x, w$ when represented by cycles of the cubic graph. The rest of the γ_i's are as follows:

$$\gamma_2 = w, x, y, z, w$$
$$\gamma_3 = w, x, y, y', z', zw$$
$$\gamma_4 = w, z, z', w', w$$
$$\gamma_5 = w, w', x', x, w$$

Now, we define $C \subset V \times V \times V$ to be

$$C := \{(x', x, y), (y, x, x'), (w', x', y'), (y', x', w'), (x', y', z'), (z', y', x'), (y', y, z),$$
$$(z, y, y'), (y, z, z'), (z', z, y), (y', z', w'), (w', z', y')(x', w', z')(z', w', x')\}$$

In the following figure, each element of C is represented by a green arc.



Thus, if a cycle, $v_1, \ldots, v_m$, follows a green part on the following graph then there exists an $i \in \{1, \ldots, m-2\}$ such that $(v_i, v_{i+1}, v_{i+2}) \in C$.

Lemma 4.11. *Let $l = v_1, \ldots, v_m$ be the cycle that represents a sequence given by proposition 4.4. If $\forall i \in \{1, \ldots, m-2\}, (v_i, v_{i+1}, v_{i+2}) \notin C$, then the cycle is a word in $G = \langle \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5 \rangle$.*

Proof. Assume $\forall i \in \{1, \ldots, m-2\}, (v_i, v_{i+1}, v_{i+2}) \notin C$. The cycle must begin and end with w. Thus, $v_1 = v_m = w$.

Case 1: $v_2 = x$.

Case 1a. $v_3 = x'$. Then either $v_4 = w' \Rightarrow v_5 = w \Rightarrow v_1, \ldots, v_5 = {\gamma_5}^{-1}$, or $v_4 = y' \Rightarrow v_5 = y. \Rightarrow v_6 = x \Rightarrow v_7 = w \Rightarrow v_1, \ldots, v_7 = \gamma_1$.

Case 1b. $v_3 = y$. Then either $v_4 = y'$ or $v_4 = z$. If $v_4 = y$, then either $v_5 = x' \Rightarrow v_6 = x \Rightarrow v_7 = w \Rightarrow v_1, \ldots, v_7 = {\gamma_1}^{-1}$ or $v_5 = z' \Rightarrow v_6 = z \Rightarrow v_7 = w \Rightarrow v_1, \ldots, v_7 = \gamma_3$. If $v_4 = z$, then $v_5 = w. \Rightarrow v_1, \ldots, v_5 = \gamma_2$.

Case 2: $v_2 = z$.

Case 2a. $v_3 = y. \Rightarrow v_4 = x. \Rightarrow v_5 = w. \Rightarrow v_1, \ldots, v_5 = {\gamma_2}^{-1}$.

Case 2b. $v_3 = z'$. Then either $v_4 = y' \Rightarrow v_5 = y \Rightarrow v_6 = x \Rightarrow v_7 = w \Rightarrow v_1, \ldots, v_7 = {\gamma_3}^{-1}$ or $v_4 = w' \Rightarrow v_5 = w \Rightarrow v_1, \ldots, v_5 = \gamma_4$.

Case 3: $v_2 = w'$.

Case 3a. $v_3 = x'. \Rightarrow v_4 = x. \Rightarrow v_5 = w. \Rightarrow v_1, \ldots, v_5 = \gamma_5$.

Case 3b. $v_3 = z'. \Rightarrow v_4 = z. \Rightarrow v_5 = w. \Rightarrow v_1, \ldots, v_5 = \gamma_4$.
If $k \neq 5$ or 7, then we repeat this process until we have a word in G.

□

Proposition 4.12. *Given any cycle $l = v_1, \ldots, v_m$ that represents a sequence given by proposition 4.4, we can make a finite number of insertions of "backtracks" into l, such that the new cycle, l' is a word in $G = \langle \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5 \rangle$ and equivalent to l.*

Proof. This is how we will make the insertions:

- If for some $i \in \{1, \ldots, m-2\}$, $(v_i, v_{i+1}, v_{i+2}) = (x', x, y)$ or (y, x, x'), then we insert w, w, x into the cycle after v_{i+1}.
- If for some $i \in \{1, \ldots, m-2\}$, $(v_i, v_{i+1}, v_{i+2}) = (w', x', y)$ or $(y, x'w')$, then we insert x, w, w, x, x' into the cycle after v_{i+1}.
- If for some $i \in \{1, \ldots, m-2\}$, $(v_i, v_{i+1}, v_{i+2}) = (x', y', z')$ or (z', y', x'), then we insert y, x, w, w, x, y, y' into the cycle after v_{i+1}.
- If for some $i \in \{1, \ldots, m-2\}$, $(v_i, v_{i+1}, v_{i+2}) = (y', y, z)$ or (z, y, y'), then we insert x, w, w, x, y into the cycle after v_{i+1}.
- If for some $i \in \{1, \ldots, m-2\}$, $(v_i, v_{i+1}, v_{i+2}) = (y, z, z')$ or (z', z, y), then we insert w, w, z into the cycle after v_{i+1}.
- If for some $i \in \{1, \ldots, m-2\}$, $(v_i, v_{i+1}, v_{i+2}) = (y', z', w')$ or (w', z', y'), then we insert z, w, w, z, z' after v_{i+1}.
- If for some $i \in \{1, \ldots, m-2\}$, $(v_i, v_{i+1}, v_{i+2}) = (x', w', z')$ or (z', w', z'), then we insert w, w, w' into the cycle after v_{i+1}.

Note: When thinking of the cycle as as a word in $\langle \sigma, \tau, \omega \rangle$, the insertions are equivalent to the identity permutations. Thus, adding them in does not change the cycle.

Once we have made the insertions, we have a new equivalent cycle, $v_1, \ldots, v_q$, such that $\forall i \in \{1, \ldots, q-2\}$, $(v_i, v_{i+1}, v_{i+2}) \notin C$. By lemma 4.11, $v_1, \ldots, v_q$ is a word in G. □

So $w_i \sim w_j \Rightarrow$ there exists a sequence $v_1, \ldots, v_l$ with $v_1 = w_i$ and $v_l = w_j$ such that for each $k = 1, \ldots, l-1$ either $(v_k, v_{k+1}) \in B, (v_{k+1}) \in B$, or $v_k = v_{k+1}$. We represent this sequence in terms of cycles of the cubic graph which we have proven can be written as a word $g \in G$. When we write g in terms of elements in $\langle \sigma, \tau, \omega \rangle$, $g(i) = j$.

Conversely, suppose $\exists g \in G$ such that $g(i) = j$. Then there is a sequence

$$w_i, \ldots, w_g(i) = w_j$$

that satisfies the property $(v_k, v_{k+1}) \in B \ \forall k = 1, \ldots, l-1$. By proposition 4.4, $w_i \sim w_j$.

An analogous argument applies to the other 33 cases listed in proposition 4.10. □

Remark 4.13. The following theorem gives us a way to count the number of vertices in a 3-dimensional gluing of cubes given σ, τ, and ω.

Theorem 4.14. *There is a bijective correspondence between the vertices in $X(\sigma, \tau, \omega)$ and the orbits of the action of $G = \langle \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5 \rangle$ on $\{1, \ldots, n\}$.*

Proof. Define $\phi : \{v_i \mid v \in \{w, x, y, z\} \text{ and } i \in \{1, \ldots, n\}\} \mapsto \{Gj \mid j \in \{1, \ldots, n\}\}$ by $v_i \to Gj$ where

$$j = \begin{cases} i & \text{if } v = w, \\ \sigma^{-1}(i) & \text{if } v = x, \\ \tau\sigma^{-1}(i) & \text{if } v = y, \\ \tau(i) & \text{if } v = z. \\ \omega^{-1}(i) & \text{if } v = w' \\ \omega^{-1}\sigma^{-1}(i) & \text{if } v = x' \\ \tau\omega^{-1}\sigma^{-1}(i) & \text{if } v = y' \\ \omega^{-1}\tau(i) & \text{if } v = z' \end{cases}$$

We must show that ϕ is well defined, injective, and surjective. Generally, we will say that $v_i \mapsto Gj$ where $j = h(i)$ and $h \in H = \{(1), \sigma^{-1}, \tau\sigma^{-1}, \tau, \omega^{-1}, \omega^{-1}\sigma^{-1}, \tau\omega^{-1}\sigma^{-1}, \omega^{-1}\tau\}$. Note that h depends on v.

To show ϕ is well defined, we must show that if $v_i \sim v'_{i'}$, then $\phi(v_i) = \phi(v'_{i'})$. So assume $v_i \sim v'_{i'}$. $\phi(v_i) = Gj$, and $\phi(v'_{i'}) = Gj'$. By proposition 4.10, there exists an r such that $r = sgt$ and $r(i) = i'$ where $s, t^{-1} \in H$ and $g \in G$. We know $\phi(v_i) = Gj = Gh(i)$ and $\phi(v'_{i'}) = \phi(v'_{r(i)}) = Gj' = Gh'(i')$. Note that $s = h$ and $t = (h')^{-1}$. So $r(i) = sgt(i) = hg(h')^{-1}(i) = i'$. $\Rightarrow hg(i) = h'(i')$. $\Rightarrow Gh(i) = Gh'(i')$. $\Rightarrow Gj = Gj'$. Thus, ϕ is well defined.

Next, we will show that ϕ is injective.

Assume $\phi(v_i) = \phi(v'_{i'})$. $\phi(v_i) = Gj = Gh(i)$ and $\phi(v'_{i'}) = Gj' = Gh'(i')$. Thus, $Gh(i) = Gh'(i')$. $\Rightarrow \exists g \in G$ such that $hg(i) = h'(i')$. $\Rightarrow hg(h')^{-1}(i) = i'$. Propostion 4.10 $v_i \sim v'_{i'}$.

Finally, we will show that ϕ is surjective. Let $i = j$ and $v = w$, then $\forall j \in \{1, \ldots, n\}$ $\phi(w_j) = Gj$. Thus, ϕ is surjective.

□

Thus, we have a way to compute v, e, and f in $X(\sigma, \tau, \omega)$.

5 An Example

Example 5.1. Here is an example of a 3-dimensional gluing of cubes that is a manifold. Let σ =(2 3 5), τ =(3 4), and ω =(1 2). Let $X(\sigma, \tau, \omega)$ is a 3-dimensional gluing of cubes. Since $n = 5$, $f = 15$.

Next, we find Θ for $X(\sigma, \tau), X(\sigma, \omega)$, and $X(\tau, \omega)$ in order to compute e.

For $X(\sigma, \tau)$,

$$\Theta = (2\ 3\ 4).$$

There are 3 orbits in the group action of Θ on $\{1, \ldots, 5\}$. So by theorem 4.8, there are 3 vertices in $X(\sigma, \tau)$.

For $X(\sigma, \omega)$

$$\Theta = (1).$$

There are 5 orbits in the group action of Θ on $\{1, \ldots, 5\}$. So by theorem 4.8, there are 5 vertices in $X(\sigma, \omega)$.

For $X(\tau, \omega)$,

$$\Theta = (1\ 5\ 2).$$

There are 3 orbits in the group action of Θ on $\{1, \ldots, 5\}$. So by theorem 4.8, are 3 vertices in $X(\tau, \omega)$. By proposition 4.1, $e = 11$.

Lastly, we compute v by using theorem 4.14. For $X(\sigma, \tau, \omega)$,

$$\begin{aligned}
\gamma_1 &= (1) \\
\gamma_2 &= (2\ 3\ 4) \\
\gamma_3 &= (1\ 5\ 2) \\
\gamma_4 &= (1) \\
\gamma_5 &= (1\ 2\ 5).
\end{aligned}$$

Thus, there is one orbit in the group action of $\langle \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5 \rangle$ on $\{1, \ldots, 5\}$. By theorem 4.14, $v = 1$.

$\chi(X(\sigma, \tau, \omega)) = v - e + f - n = 1 - 11 + 15 - 5 = 0$. Theorem 3.4 implies $X(\sigma, \tau, \omega)$ is a manifold.

References

[1] J. Lee, Introduction to Topological Manifolds, Second Edition, Springer Science+Business Media LLC, New York, NY, 2011.

[2] W. Thurston, 3-D Geometry and Topology, Princeton University Press, Princeton, NJ, 1997.

Developing a Phylogenetic ANCOVA: Analyzing Multiple Variables in the Evolution of Continuous Traits

Annie Murphy and Ashley Weber

Abstract

The relationship between a particular trait and a factor where it occurs is a relevant issue in evolutionary biology; however, this kind of relationship rarely happens in isolation. In addition to analyzing the relationship between a continuous and a categorical variable, researchers are usually interested in assessing the effect of a concomitant continuous variable. Data transformations (both phylogenetic and non-phylogenetic) have been commonly used to deal with this kind of data, but such approach usually involves splitting the analysis in several steps. The phylogenetic version of the analysis of covariance (ANCOVA) allows modeling the evolutionary relationships of this kind of data explicitly. It has been used before by means of simulations or to address specific hypotheses about the relationships of the variables, but the former case is particularly linked to hypothesis testing and the latter gives place for some improvement in terms of model fitting. Besides such improvement, a phylogenetic ANCOVA would be more informative by its combination with new comparative approaches, allowing addressing questions about adaptation and phenotypic radiation. Here we present a phylogenetic ANCOVA framework that takes those features into account. To address phenotypic radiation questions, our framework allows for heteroscedasticity according to phylogenetic regions determined by the categorical variable. To address causal relationships we combine this framework with the adaptation-inertia model, where the categorical variable is presented as the weighted, summed, time each species has evolved under each evolutionary regime (i.e. each category of the discrete variable mapped on the phylogeny). The approach we present here has a potential advantage in terms of model fitting, where the relevance of all the parameters involved in the model (not necessarily linked to particular hypotheses) can be determined by iterative procedures.

1 Background

A phylogeny is a diagram of the relationships of ancestry and descent among a group of species. Mathematically, a phylogeny is a tree. The phylogenies we will be using are rooted, so we will be talking about rooted trees. The tips of the tree will correspond to a species within our set. A phylogeny also contains clades, or subtrees. Sometimes we will need to refer to clades within the phylogeny to

separate some species. We use the phylogenies to correct for independence in the comparative analysis model. The phylogenies we work with are ultrametric, meaning that all the tips are lined up and the branch lengths from the root to any tip sums to one. It is important to keep in mind that in a phylogeny, the branch lengths are proportional to time.

We are interested in looking at traits of species and find a relationship between the traits. There are two types of traits: categorical and continuous traits. Categorical traits have a finite number of states or possibilities. These can be considered qualitative traits. Gender is a categorical trait with the states as male or female. The categorical traits we work with usually can be considered present or absent, so there are two distinct states. Some examples that we worked directly with are sociality in canids where the states are social or not social, diet in canids where the states are carnivore or omnivore, and belly color in lizards where the states are blue or white. Sometimes we call one state the novel state, meaning that state showed up in the phylogeny closer to the present. The other state is called the ancestral state, meaning that is the state that the root species is considered to have. Continuous traits are measurable traits and in theory can take on any value. Some examples of continuous traits that we worked with directly are size as according to weight, litter size of canids, or the number of head bobs lizards make when signaling. Statistical methods can be used to analyze the relationship between traits of a given set of species.

1.1 Regression

ANCOVA is a statistical method that analyzes the relationship between one categorical trait and two continuous traits. The categorical trait and one of the continuous traits are used the predict the second continuous trait. The first continuous trait (used in the predicting) is also called the covariate. The second continuous trait is also called the predicted variable or trait. In this method, two regression lines are given, a line for the relationship between the two continuous traits for each state of the categorical trait. The equation in matrix format for any general linear model, and so for ANCOVA as well, is:

$$Y = X\beta + \varepsilon \tag{1}$$

Where Y is a column vector of the value of a continuous trait, X is the design matrix, β is a column of parameters, and ε is the error term. Written out fully, equation (1) becomes:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_1 & 0 \cdot x_1 \\ 1 & 1 & x_2 & 1 \cdot x_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & x_n & 1 \cdot x_n \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} \tag{2}$$

In this equation, n is the number of species in the data set. In the design matrix, the first column is all 1's for the intercept. The second column is for the

categorical trait, where the two states are coded with either a 0 or 1. The third column is the value of the covariate. And the fourth column is for an interaction between the categorical trait and the covariate, which is just the second column multiplied by the third column. In the β matrix we have 4 parameters. b_0 is the intercept for the 0 state, b_1 is the difference between the intercepts, b_2 is the slope for the 0 state, and b_3 is the difference between slopes. So, the intercept for the 1 state is $b_0 + b_1$ and the slope for the 1 state is $b_2 + b_3$. To estimate β we can use the following equation, called the normal equation:

$$\beta = (X^t X)^{-1} X^t Y \tag{3}$$

1.2 Variance Likelihood Equation

Sometimes with a given phylogeny and a continuous trait there are two variances across the continuous trait. We say one variance is for those species in one state of the categorical trait and the other variance is for the second categorical trait. We can consider branches of the phylogeny in one variance state or another depending on the state of the categorical trait in that branch. O'Meara et al. (2006) presents a likelihood equation to test whether one variance or two variances are appropriate for a given phylogeny [5].

$$\log(L) = \log\left(\frac{-\frac{1}{2}[X - E(X)]^t V^{-1}[X - E(X)]}{\sqrt{(2\pi)^N \times \det(V)}}\right) \tag{4}$$

In this equation, X is the column vector of the observations, $E(X)$ is a column matrix of expected tip values, and V is a matrix that is given by the phylogeny. Details on the V matrix will be discussed later. Our model does not use this X or $E(X)$, so refer to the [5] for more details.

2 Our Likelihood Equation

We expanded on equation (4) to include the regression model as well. We wanted to have one likelihood equation that we could use to maximize that gives us both the regression parameters and whether one variance or two variances are appropriate. Equation (4) became

$$\log(L) = \log\left(\frac{\exp\left(-\frac{1}{2}[Y - X\beta]^t V^{-1}[Y - X\beta]\right)}{\sqrt{(2\pi)^N \times \det(V)}}\right) \tag{5}$$

In equation (5) Y, X, and β are all from the regression model explained in Section 1.1. N is the number of species in the data set. V is considered a weight matrix that is given by the phylogeny. We define V as:

$$V = \sigma^2 G + \tau^2 C \tag{6}$$

Here, σ^2 is the variance for the 0 state of the categorical trait and τ^2 is the difference in the variances. G and C are both matrices that change depending

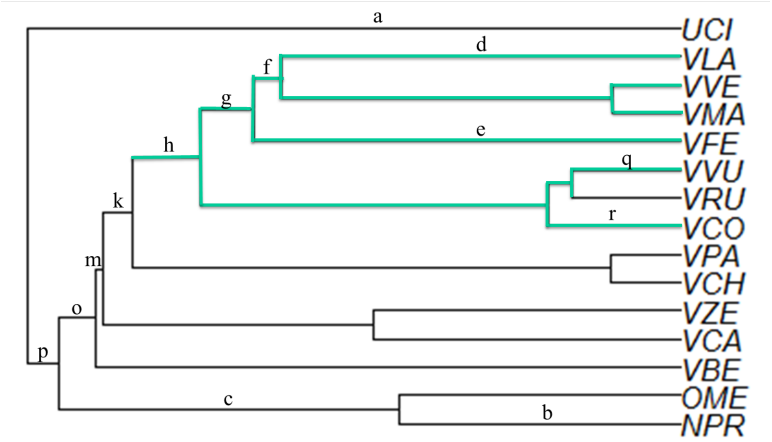


Figure 1: A phylogeny of some canid species. The green branches represent the branches considered to be in the 1 state or to have the second variance. The letter above some of the branches represent branch lengths.

on what biological process one wants to model. The details of these two matrices are explained below.

2.1 Brownian Motion

Brownian motion is one process used in evolutionary biology to model traits. Essentially, Brownian Motion is a random walk. The error term, ε is considered to have a mean of 0 and a variance of $\sigma^2 t$, where t is time. In this case, the variance of the error term will be the same as the variance of the predicted continuous trait. As suggested by the variance equation, under Brownian Motion as time increases the variance of the trait increases as well. We use this fact, and the fact that the branches in a phylogeny are proportional to time to calculate the matrices G and C that are in equation (6).

2.1.1 The G Matrix

The G matrix in equation (6) takes into account the phylogeny in order to account for the dependence of species. Under Brownian Motion, this can be called a similarity matrix or a variance-covariance matrix. Each entry in the G matrix is given by

$$G_{ij} = t_{\text{ra}} \tag{7}$$

where t_{ra} is the time from the root to the most recent common ancestor of species i and j. This means that the diagonals of G are 1 because all of our phylogenies are required that the branch lengths from the root to the tip sum to 1.

Example 2.1. For the phylogeny in Figure 1, $G_{\text{UCI,NPR}} = 0$ because their most recent common ancestor is the root itself.

Example 2.2. If we look at the species VLA and VFE then $G_{\text{VLA,VFE}} = g + h + k + m + o + p$.

2.1.2 The C Matrix

The C matrix in equation (6) is to take into account the extra variance the species that are in state 1 might have. To make the C matrix, first all the clades that only contain species in the 1 state must be identified. In Figure 1 there are three clades. One is the subtree containing VFE, VMA, VVE, and VLA; the second clade contains only one species, VCO; and the third clade contains VVU. Then we have to decide how much of the branch directly preceding the clade is considered to be in the second variance state. For the clade with VFE, VMA, VVE, and VLA the branch directly preceding it is labeled g. For clades with VCO or VVU, we look at the branches r and q, respectively. This introduces a new parameter t, which is the proportion of the branch directly preceding the clade that is to be considered part of the second variance state. So if $t = 0$ then none of the branch is considered, but if $t = 1$ the whole branch is considered part of the second variance state. So now we can fill in the C matrix. We only want to consider the relationship between species that are in the same clade, so many of the entries in the C matrix are 0. However, if species i and j are in the same clade, then

$$C_{ij} = t_{\text{sa}} \tag{8}$$

where t_{sa} is the time from where the second variance state starts to the most recent common ancestor of species i and j.

Example 2.3. The following are some entries in C when we consider Figure 1 as the phylogeny.

$$\begin{aligned} C_{\text{UCI,NPR}} &= 0 \\ C_{\text{VVU,VCO}} &= 0 \\ C_{\text{VCO,VCO}} &= t \cdot r \\ C_{\text{VLA,VFE}} &= t \cdot g \\ C_{\text{VVE,VLA}} &= f + t \cdot g \end{aligned}$$

Because the entries in C are dependent on the parameter t, we can consider C as a function of t.

2.2 Ornstein-Uhlenbeck Process

The Ornstein-Uhlenbeck process is the second process that we modeled using our likelihood equation. Ornstein-Uhlenbeck can be said to be Brownian Motion under a constraint. In the Ornstein-Uhlenbeck process the error term, ε is considered to have mean 0 and variance $e^{-2\alpha t}\left(\frac{e^{2\alpha t}-1}{2\alpha}\right)$, where t is time and α is a constraint parameter. Again, the variance of the error term will be the same as the variance of the predicted continuous trait. Under Ornstein-Uhlenbeck

process, has time increases the variance reaches a steady state, meaning that after a certain time t, the variance is the same. The Ornstein-Uhlenbeck process is a stationary processes, where Brownian Motion is not stationary. However, if $\alpha = 0$ then Ornstein-Uhlenbeck and Brownian Motion are the same. The α parameter determines how fast the stationary state is reached. For our model, we assume the the traits are already stationary. In biological terms, this accounts for any outside conditions, like environmental conditions, that might pull a trait to an optimum. Like in Brownian Motion, we use the variance to calculate the G and C matrix in equation (6).

2.2.1 The G Matrix

The G matrix in equation (6) for the Ornstein-Uhlenbeck process is calculated in two steps. First, a divergence matrix must be calculated, call this matrix D. Each entry in D is given by

$$D_{ij} = t_{ij} \tag{9}$$

where t_{ij} is the time separating species i and j. This means that the diagonals of D are 0 because there is no time separating a species from itself.

Example 2.4. For the phylogeny in Figure 1, $D_{\text{UCI,NPR}} = a + p + c + b = 2$, it equals 2 because we require that our branch lengths from the tip to the root is 1, and UCI and NPR separated at the root.

Example 2.5. If we look at the species VLA and VFE then $D_{\text{VLA,VFE}} = d+f+e$.

We use the D matrix to calculate G. The entries of G are given by

$$G_{ij} = \exp(-\alpha D_{ij}) \tag{10}$$

So the diagonals of G are 1.

Example 2.6. $G_{\text{UCI,NPR}} = \exp(-\alpha(a+p+c+b))$ and $G_{\text{VLA,VFE}} = \exp(-\alpha(d+f+e))$.

Because G depends on α, we can think of G as being a function of α. α becomes one of the parameters we must estimate.

2.2.2 The C Matrix

In the C matrix, which is in equation (6), we again just want to compare species that are in the same clade and in state 1 to account for any extra variance. Because we are looking at the divergence of species in the Ornstein-Uhlenbeck process, we do not need the parameter t since that parameter represented something before the common ancestor of species in the same clade. To account for the potential of extra variance is the 1 state, if species i and j are in the same clade then

$$C_{ij} = G_{ij} \tag{11}$$

where G_{ij} is defined as in equation (10). As with Brownian Motion, if species i and j are not in the same clade then the entry is 0. This makes the C matrix mostly zeros.

Example 2.7. $C_{\text{UCI,NPR}} = 0$ and $C_{\text{VLA,VFE}} = \exp(-\alpha(d+f+e))$ since UCI and NPR are not in the same clade, but VLA and VFE are in the same clade.

2.3 Hansen's Method

Hansen's method, described in [2] describes a new regression method that performs the regression on the amount of weighted time each species spent evolving in each different environment, where the environment is the state of the categorical trait of interest, either 0 or 1. Hansen's method allows for the evolution of traits toward two separate optimum, one for each state of the categorical variable. Using Hansen's method, we can determine whether the presence of one particular categorical state influences the evolution of the continuous trait of interest, allowing us to infer causation. In Hansen's method, the categorical trait is transformed into a continuous trait according to the time each species spent evolving in a particular state, and this information is stored in the design matrix.

We were not able to implement a complete version of Hansen's method, however code was developed to calculate both the method matrix and the design matrix described in Hansen's paper [2], and the code is detailed below.

2.3.1 Method Matrix

The first matrix described in Hansen's method is the method matrix. The method matrix has a row corresponding to each branch in the tree, and four columns. The first two columns list the beginning and ending node for each branch in the tree, with the beginning node being the node that is closest to the root. The third column lists the branch length. The fourth lists Hansen's number for each branch, where Hansen's number is defined to be

$$H = \exp(-\alpha \cdot t_e) - \exp(-\alpha \cdot t_b) \tag{12}$$

Where t_e is the distance from the tips to the end of the branch closest to the tips, and t_b is the distance from the tips to the end of the branch closest to the root. For any branch connected to the root, t_b is assumed to be infinite, and $\exp(-\alpha \cdot t_b) = 0$.

2.3.2 Design Matrix

The second matrix described in Hansen's method is the design matrix. The design matrix lists, for each species, the amount of time the species evolved in the ancestral state, and the novel state. It has a row corresponding to each species and two columns for each species, the first for the amount of time spent in the novel state and the second for the amount of time spent in the ancestral state. This describes the design matrix when looking at one continuous trait and one categorical trait.

To implement Hansen's Method using two continuous traits and one categorical trait, like the ANCOVA, one has to add two more columns in the design

matrix. The first column added (which is the third column in the design matrix) would be the value of the second continuous (or covariate) trait. And the second column would be the interaction between the covariate and categorical trait, as described in Section 1.1. Note that this is slightly different than what Hansen describes in [2]. The method described in the paper uses ancestor trait values as well, but we just want to use the tip values. We were not able to code the additional two columns into the design matrix.

3 The Code

3.1 Current Code

3.1.1 Brownian Motion

Our current Brownian motion code loads the "ctv", "ape", "geiger", "nlme" and "phangorn" libraries in R.

It reads in alphabetical data that contains the species and character values from a .csv file, assuming that each row of the .csv contains a species and each column contains either species names or characters. It identifies the categorical and covariate variables within that .csv file, and reads in the phylogeny, a .tre file.

It then proceeds to a section of code referred to as Fast Clades, which will be discussed in further detail in section 3.2.1. FastClades identifies the species that possess a 1 value for the categorical trait of interest, which is also referred to as the novel state of the categorical trait. It places the numbers corresponding to those species on the tree (the node labels) in a matrix called clade matrix, the species name abbreviations in a matrix called name matrix, and the numbers corresponding to the appropriate species rows in the data frame read in from the .csv file in row matrix.

The code then calculates the G similarity matrix using the cophenetic function, and scales that matrix to have a maximum value of 1. G is ordered alphabetically by species name along both the rows and columns.

The code moves on to a function called Time Ranges, which calculates the location of the starting and ending points of the branch directly proceeding the monophyletic clades and single species that have a categorical variable state of 1 (the novel species), and uses these locations to calculate the C similarity matrix which will describe the novel species in which additional variance occurs. The C similarity matrix assumes that changes in the novel groups occur along the branches determined in TimeRanges, according to a parameter t which is to be estimated.

Two functions that will calculate the likelihood equation based on an estimation are declared, Alt and Null. Alt calculates the V matrix according to (6) and uses the V matrix to calculate the likelihood according to (5). Null feeds the correct parameters to the Alt function.

A standard version of Powell's method is laid out, and bounds are set for the parameters that are to be estimated using Powell's method.

Where x_0 is the covariate in the ancestral state (value of 0), x_1 is the covariate in the novel state (value of 1) and y is the predicted variable, we allow the upper bounds on the four β parameters to be estimated to be

$$\beta_0 = y_{max} - (\beta_{2max} \cdot x_{0,max}) \tag{13}$$

$$\beta_1 = y_{max} - \beta_{0,min} - (\beta_{2,min} \cdot x_{1,min}) - (\beta_{3,max} \cdot x_{1,min}) \tag{14}$$

$$\beta_2 = ((y_{min} - y_{max})/(x_{0,max} - x_{0,min})) \tag{15}$$

$$\beta_3 = ((y_{min} - y_{max} - \beta_{0,min} + \beta_{0,max} - (\beta_{2,min} \cdot x_{1,min}) + (\beta_{2,max} \cdot x_{1,max}))/(x_{1,max} - x_{1,min})) \tag{16}$$

and the lower bounds to be

$$\beta_0 = y_{min} - (\beta_{2,min} \cdot x_{0,max}) \tag{17}$$

$$\beta_1 = y_{min} - \beta_{0,max} - (beta_{2,max} \cdot x_{1,max}) - (\beta_{3,min} \cdot x_{1,max}) \tag{18}$$

$$\beta_2 = (y_{min} - y_{max})/(x_{0,min} - x_{0,max})) \tag{19}$$

$$\beta_3 = (y_{min} - y_{max} - \beta_{0,max} - \beta_{0,min} - (\beta_{2,max} \cdot x_{1,max}) + (\beta_{2,min} \cdot x_{1,min}))/(x_{1,min} - x_{1,max})) \tag{20}$$

We allow the lower bounds for σ^2, τ^2 and t to be 0, and the upper bounds to be the following

$$\sigma^2 = (x_{max} - x_{min})^2 \tag{21}$$

$$\tau^2 = (x_{max} - x_{min})^2 \tag{22}$$

$$t = 1 \tag{23}$$

,

where x is the covariate in either the ancestral or novel state.

Powell's method is then used to estimate the parameters of interest, and the likelihood according to (5) is returned.

3.1.2 Ornstein-Uhlenbeck Process

Our current Ornstein-Uhlenbeck process code loads the same R libraries as the Brownian motion code. It reads in the same type of .csv data and phylogeny files.

It reads in data that contains the species and character values from a .csv file that is alphabetized by row, assuming that each row of the .csv contains a species and associated traits and each column contains either species names or characters. It identifies the categorical and covariate variables within that .csv file, and reads in the phylogeny, a .tre file.

It then proceeds to the same Fast Clades section, which will be discussed in further detail in section 3.2.1, placing the numbers corresponding to the novel species on the tree (the node labels) in a matrix called clade matrix, the species name abbreviations in a matrix called name matrix, and the numbers corresponding to the appropriate species rows in the data frame read in from the .csv file in row matrix.

The code then calculates the G divergence matrix, described in further detail in section 3.2.3.

The code moves on to the same Time Ranges function , which calculates the location of the starting and ending points of the branch directly proceeding the monophyletic clades and single species that have a categorical variable state of 1 (the novel species), and uses these locations to calculate the C divergence matrix, which contains the divergence distances for only the novel species . The C divergence matrix is described in greater detail in section 3.2.3.

Both matrices are transformed according to the Ornstein-Uhlenbeck process, described in greater detail in Section 2.2, and for which code is discussed in Section 3.2.3.

The same Alt and Null functions are used as described in Section 3.1.1.

A standard version of Powell's method is laid out, and bounds are set for the parameters that are to be estimated using Powell's method. The bounds are the same as those described in section 3.1.1. Powell's method is then used to estimate the parameters of interest, and negative two times the likelihood according to (5) is returned.

3.1.3 Hansen's Method

While we didn't have time to implement a full version of Hansen's method, as described in Section 2.3, we were able to develop code to calculate both the method matrix and the design matrix for a set of data and phylogeny.

The Method Matrix: We formed the first three columns of the method matrix by binding together existing fields of information from our tree.

```
method_matrix<-cbind(tree$edge, tree$edge.length)
```

.

We then calculated the distance from each node in the tree to its first tip descendant, and listed these values in the vector tip2node. We assumed that the tree was ultrametric.

```
for(i in 1:length(tip2node)){
  descendants<-Descendants(tree, i, type="tips")[[1]]
  tip_descendant<-descendants[1]
  tip2node[i]<-dist.nodes(tree)[i, tip_descendant]
}
```

We changed the distance from the root node to its tip descendant to -1, which we allow to represent infinity.

```
depths<-node.depth(tree)
max_depth<-max(depths)
root<-which(depths== max_depth)
tip2node[root]<--1

hansens_number<-rep(0,nrow(method_matrix))
t_e<-0
t_b<-0
alpha<-0.5

for(i in 1:nrow(method_matrix)){
  t_b<-tip2node[method_matrix[i,1]]
  t_e<-tip2node[method_matrix[i,2]]

  if(tip2node[method_matrix[i,1]]==-1){
    hansens_number[i]<-(exp(-1*alpha*t_e))
  }else {
    hansens_number[i]<-(exp(-1*alpha*t_e))-(exp(-1*alpha*t_b))
  }
}

method_matrix<-cbind(method_matrix,hansens_number)
```

The Design Matrix: To calculate the design matrix, we began by assigning space for the matrix and listing the species that evolved in the novel state of the categorical variable and those that did not. The nodes and names of the species were stored in ones, names1, zeroes and names0.

```
design_matrix<-mat.or.vec(num_tips,2)
zeroes<-0
rm(ones)
ones<-0
names0<-'xxx'
```

```
names1<-'xxx'

for(i in 1:nrow(data)){
  if(categ[i]==0){
    add<-as.character(data[i,1])
    names0<-c(names0,add)
  }
  if(categ[i]==1){
    add<-as.character(data[i,1])
    names1<-c(names1,add)
  }
}

for(i in 1:nrow(data)){
  for(j in 1:length(names0)){
    if(tip_labels[i]==names0[j]){
      zeroes[j]<-i
    }
  }
}

for(i in 1:nrow(data)){
  for(j in 1:length(names1)){
    if(tip_labels[i]==names1[j]){
      ones[j]<-i
    }
  }
}
zeroes<-zeroes[-1]
ones<-ones[-1]
names0<-names0[-1]
names1<-names1[-1]
```

Then, for the species listed in the vector zeroes, that evolved in the ancestral state, we listed the names of their ancestors in the vector edge progression, and used the values stored in the method matrix to add the Hansen's numbers of each branch described by their edge progression nodes. We then filled in the design matrix's first column with the sum of these species' Hansen's numbers (which summed to 1) and the design matrix's second column with a zero, because the species spent no time evolving in the second (novel) environment.

```
for(i in 1:length(zeroes)){
  edge_progression<-Ancestors(tree,zeroes[i])
  edge_progression<-sort(edge_progression, decreasing=FALSE)
  edge_progression<-c(edge_progression, zeroes[i])
  edge_progression<-edge_progression[-1]
```

```
  hansens_sum<-0

  for(j in 1:length(edge_progression)){
    for(k in 1:nrow(method_matrix)){
      if(method_matrix[k,2]==edge_progression[j]){
        hansens_sum<-hansens_sum+method_matrix[k,4]
      }
    }
  }

  design_matrix[zeroes[i],1]<-hansens_sum
  design_matrix[zeroes[i],2]<-0

}
```

We listed all the descendants of the nodes stored in the vector clade markers (the common ancestors of monophyletic clades and the single species that displayed the novel version of the categorical trait). We stored these descendants in the vector ones nodes, and eliminated repeated values.

```
ones_nodes<-unlist(c(clade_markers, Descendants(tree,clade_markers,type="all")))
ones_nodes<-unique(ones_nodes)
```

We then listed the edge progression for each species in the vector ones. For each branch described by the edge progression, we asked if its endpoint closest to the root was in the vector ones nodes. If it was, we added that branch's Hansen's number to the second column of the design matrix. If it wasn't, we added it to the first, thus recording the weighted time each species spent evolving in each environment.

```
for(i in 1:length(ones)){
  edge_progression<-Ancestors(tree,ones[i])
  edge_progression<-sort(edge_progression, decreasing=FALSE)
  edge_progression<-c(edge_progression, ones[i])
  edge_progression<-edge_progression[-1]

  hansens_sum0<-0
  hansens_sum1<-0

  for(j in 1:length(edge_progression)){
    for(k in 1:nrow(method_matrix)){
      if(method_matrix[k,2]==edge_progression[j]){
        if(method_matrix[k,1]%in%ones_nodes==TRUE){
          hansens_sum1<-hansens_sum1+method_matrix[k,4]
        }else {
          hansens_sum0<-hansens_sum0+method_matrix[k,4]
```

```
        }
      }
    }
  }

  design_matrix[ones[i],1]<-hansens_sum0
  design_matrix[ones[i],2]<-hansens_sum1
}
```

3.2 Components of Code

3.2.1 Fast Clades

FastClades is a piece of code that was developed to identify monophyletic clades and single species in the phylogenetic tree that display the novel state of the categorical trait of interest. It begins by determining the names of the species associated with a categorical trait value of 1 and placing them in a vector called names.

```
for(i in 1:nrow(data)){
  if(categ[i]==1){
    add<-as.character(data[i,1])
    names<-c(names,add)
  }
}
```

Taking the names, the code compares the names from the data frame to the tip labels of the phlyogenetic tree and puts them into vector ones.

```
for(i in 1:nrow(data)){
  for(j in 1:length(names)){
    if(tip_labels[i]==names[j]){
      ones[j]<-i
    }
  }
}
```

The code determines which of the species' whose node labels are in the vector ones have ancestors with at least two descendants in ones and stores them in vector ancestors.

```
while(n!=max_depth){

  ancestors<-0

  for(i in 1:length(ones)){
    for(j in 1:nrow(edge)){
```

```
      if(edge[j,2]==ones[i]){
        ancestors[i]<-edge[j,1]
      }
    }
  }

  for(i in 1:length(ancestors)){
    count<-0
    for(j in 1:length(ancestors)){
      if(ancestors[i]==ancestors[j]){
        count<-count+1
      }
    }

    index<-max(0,which(ones==ancestors[i]))
    if(index==0 && count==2){
      ones<-c(ones,ancestors[i])
    }
  }
  n<-n+1
}
```

The code then removes those nodes contained in vector ancestors that have descendants that are not in ones.

```
remove<-0

for(i in 1:length(ancestors)){
  node<-ancestors[i]
  descendants<-Descendants(tree, node, type="all")

  for(j in 1:length(descendants)){
    if(max(which(ones==descendants[j]),0)==0){
      remove<-c(remove,ancestors[i])
    }
  }
}

remove<-remove[-1]
remove<-unique(remove)

for(i in 1:length(remove)){
  element<-remove[i]
  index<-which(ancestors==element)
  ancestors<-ancestors[-index]
}
```

We take those in the vector ancestors that are unique, and assign them to vector clade markers.

```
clade_markers<-unique(ancestors)
```

The clade markers vector describes the common ancestors of each monophyletic clade within the group of species that display the novel state of the categorical variable.

If the clade markers vector is non-empty, the tip numbers for the species are added back in to the clade markers.

If the clade markers vector is non-empty, the matrix clade matrix is filled with the descedants of those nodes listed in clade markers, with a row corresponding to each monophyletic clade or isolated species.

If the clade markers vector is empty, then the matrix clade matrix is filled with the species contained in ones.

3.2.2 Time Ranges

In an effort to replicate the distance ranges returned by the original Time Ranges function, a piece of code was developed based on the newly developed Fast Clades code to calculate the distance from the root to the beginning and ending points of each branch preceding a monophyletic clade displaying the novel state of the categorical trait, and each single species displaying the same novel state. The code uses the vector clade markers used in the Fast Clades code described in section 3.2.1, and determines each entry's parent, placing them in corresponding spots in a vector the same length as clade markers called preceding clade markers.

Then, using the following code, the root of the tree is determined, and the distance from the root to each end of the branch marked by preceding clade markers and clade markers is recorded in an $i \times 2$ matrix called TimeRanges.

```
root2_prec_clade_markers<-0
root2_clade_markers<-0
TimeRanges<-mat.or.vec(length(clade_markers),2)

max_depth<-max(node_depth)
root<-which(node_depth==max_depth)

for(i in 1:length(clade_markers)){
  TimeRanges[i,1]<-dist.nodes(tree)[root,prec_clade_markers[i]]
  TimeRanges[i,2]<-dist.nodes(tree)[root,clade_markers[i]]
}
```

3.2.3 G and C Matrices

The G Divergence Matrix: In order to calculate the G divergence matrix, we form two matrices, each with a row and column corresponding to the species

depicted in the tree. These matrices are the divergence matrix and the common ancestor matrix. We then fill the upper triangle of the common ancestor by allowing each cell to be the node label corresponding to the node that is the most recent common ancestor of the species given by the row number and column number.

```
#get common ancestors, fill upper triangle of common ancestor matrix
for(i in 1:num_tips){
  for(j in i:num_tips){
    common_anc_matrix[i,j]<-(mrca(tree)[i,j])
  }
}
```

We then calculate the distance from most recent common ancestor to each of the tips given by the row number and the column number of the cell, and add those to get the total divergence. The corresponding cell in the divergence matrix is then filled with this value.

```
for(i in 1:num_tips){
  for(j in i:num_tips){
    mrca_to_tip1<-dist.nodes(tree)[common_anc_matrix[i,j],i]
    mrca_to_tip2<-dist.nodes(tree)[common_anc_matrix[i,j],j]
    divergence_matrix[i,j]<-mrca_to_tip1 + mrca_to_tip2
  }
}
```

We then assign row and column names, scale the G matrix and order G alphabetically according to the species names.

```
rownames(divergence_matrix)<-tree$tip.label
colnames(divergence_matrix)<-tree$tip.label
G<-divergence_matrix

G = G/(max(G)/2)
G <- as.matrix(G)

G <- G[order(dimnames(G)[[1]], na.last=NA) , ]
G <- G[,sort(colnames(G))]
```

The C Divergence Matrix: In order to create the C divergence matrix, which describes the divergence of the species listed in the clade matrix, row matrix and name matrix given by FastClades (see Section 3.2.1 for more details), we first create a matrix with a row for each species, and a column for each species. We initially fill all the cells of that matrix with -1.

```
fill_ones<-as.matrix(rep(-1,num_tips))
c_divergence<-fill_ones[,rep(1,num_tips)]
```

For each clade in FastClades, we look at the species contained in it and determine the common ancestor for each two species. We then sum the distance from that common ancestor to its two descendants of interest and place that divergence distance in the C matrix, in the cell corresponding to the two species.

```
for(i in 1:nrow(row_matrix)){
  clade<-0
  count<-1
  for(j in 1:ncol(row_matrix)){
    if(row_matrix[i,j]!=0){
      clade[count]<-row_matrix[i,j]
      count<-count+1
    }
  }

  for(n in 1:length(clade)){
    for(m in 1:length(clade)){
      mrca<-mrca(tree)[clade[n],clade[m]]
      div<-dist.nodes(tree)[mrca,clade[n]] + dist.nodes(tree)[mrca,clade[m]]
      c_divergence[clade[n],clade[m]]<-div
      c_divergence[clade[m],clade[n]]<-div
    }
  }
}
```

We then assign row and column names to C and alphabetize C in a manner consistent with G.

```
rownames(c_divergence)<-tree$tip.label
colnames(c_divergence)<-tree$tip.label
C<-(c_divergence/(max(G)/2))

C <- C[order(dimnames(C)[[1]], na.last=NA) , ]
C <- C[,sort(colnames(C))]
```

We note that for each entry $C_{ij} \neq -1$, $C_{ij} = G_{ij}$.

Transforming the G and C Matrices According to the Ornstein-Uhlenbeck Process: Each entry in the G divergence matrix is transformed according to the Ornstein-Uhlenbeck process described in (10), using the following function:

```
OU<-function(alpha, G){
  for(i in 1:nrow(G)){
    for(j in 1:ncol(G)){
```

```
      G[i,j]<-exp(-1*alpha*G[i,j])
    }
  }

  G
}.
```

The fields in the C matrix corresponding to species that do not display the novel state of the categorical trait of interest are marked with -1, and so these are changed to zeroes when the matrix is transformed. The remaining entries are transformed according to the same equation described above, using the following function:

```
OUC<-function(alpha,C){
  for(i in 1:nrow(C)){
    for(j in 1:ncol(C)){
      if(C[i,j]==-1){
        C[i,j]<-0
      }else{
        C[i,j]<-exp(-1*alpha*C[i,j])
      }
    }
  }

  C
}.
```

4 Results

We ran our code assuming no phylogeny, Brownian Motion, and Ornstein-Uhlenbeck process. We had three main data sets that we worked with and ran our code with. The first data set is a canid phylogeny, with the categorical trait as diet, the covariate as cranial length, and the predicted as palatal width. In this case, carnivore is considered the novel state or the 1 state and omnivore is considered the ancestral state or the 0 state. The second data set is a similar canid phylogeny, with the categorical trait as sociality, the covariate as size, and the predicted as litter size. In this case, social is considered the novel state and solitary is considered the ancestral state. The third data set is a lizard phylogeny, with belly color as the categorical trait, number of head bobs (HBT), which is a visual cue, as the covariate, and femoral pores (FP), which is a chemical cue, as the predicted. In this case, white is considered the novel state and blue is considered the ancestral state. The following are the numbers we got and interpretation of them.

4.1 No Phylogeny

The first thing that we needed to check was that our code, maximizing our likelihood equation, would give the same results for the regression parameters as the least square regression way. Since we didn't need any phylogeny to check this, we set G equal to the identity matrix and set τ^2 equal to zero. The results of our code are shown in Table 1 and Table 3. The results of the the linear regression are shown in Table 2 and Table 4. Comparing the tables with the same data we see that in fact we do get the same numbers for the regression parameters. So we can correctly estimate the regression parameters without a phylogeny.

4.2 Brownian Motion

Next we ran our code with the G and C matrices being coded as described in Section 3.2.3. Tables 5, 6, and 7 show our results. The regression parameters all seem to fit the data well, which can be seen in the Figures 2, 3, and 4. The variances for the first canid set is reasonable and a good estimate. The variances for the second canid set are slightly high, but not so high to be too concerned. The variances for the lizard set is very high however. As you can see, $\sigma^2 = 13$ which isn't that high given the data set. On the other hand, $\tau^2 = 177$ which is very high given the data. The data shows little difference between the variances however the simulation suggests very high difference between the two variances. τ^2 does not fit the data at all. There is obviously something wrong somewhere in our code, and we have not been able to as of yet figure out why our simulation is putting τ^2 so high.

4.3 Ornstein-Uhlenbeck Process

We also ran our code with the G and C matrix coded as explained in Section 3.2.3. Tables 8, 9, and 10 show our results. The fact that α is going to the maximum allowed number for the first canid and the lizard data set is a little worrying. When looking at the likelihood of the data sets, setting α closer to 0, the likelihood is higher than when α is set to 15. This was done with setting all other parameters to their maximum likelihood when G was equal to the identity matrix. So more investigation into why α is going off to the maximum is needed. For the second canid data set and the lizard data set the variances are more appropriate than in the Brownian Motion model. The regression parameters all seem to fit the data appropriately, this can be seen in Figures 5, 6, and 7.

5 Acknowledgments

We would like to acknowledge the contributions of the graduate student on our team, Jesualdo A. Fuentes, from the Indiana University Biology Department for his work on the project, as well as the abstract he provided for this paper. We would also like to acknowledge our faculty mentor, Professor Emília Martins,

from the Indiana University Biology Department, as well as Professor Elizabeth Housworth from the Indiana University Mathematics Department for her contributions to our initial version of the code that was developed.

Furthermore, we would like to thank the entire Martins Lab for providing us with a comfortable working environment for the summer.

Finally, we would like to thank the other Indiana University Math REU students, our program director, Professor Kevin Pilgrim from the Mathematics Department and the National Science Foundation.

References

[1] Joseph Felsenstein: Phylogenies and the Comparative Method. The American Naturalist, vol. 125, No. 1 (Jan 1985), 1–15

[2] Thomas F. Hansen: Stabilizing Selection and the Comparative Analysis of Adaptation. Evolution. Vol. 51, No. 5 (Oct 1997), 1341–1351.

[3] Emilia P. Martins and Thomas F. Hansen: Phylogenies and the Comparative Method: A General Approach to Incorporating Phylogenetic Information into the Analysis of Interspecific Data. The American Naturalist, Vol. 149, No. 4 (Apr 1997), 646–667

[4] Emilia P. Martins and Elizabeth A. Housworth: Phylogeny Shape and the Phylogenetic Comparative Method. Systematic Biology. Vol 51 (Aug 2002), 873–880

[5] Brian C. O'Meara, Cecile Ane, Michael J. Sanderson, Peter C. Wainwright: Testing for Different Rates of Continuous Trait Evolution Using Likelihood. Evolution. Vol 60, No. 5 (May 2006), 922–933

Parameter	Estimates	Std Error
β_0	0.04883	3.73460
β_1	−9.31656	5.72461
β_2	0.22416	0.02903
β_3	0.08220	0.04003

Table 1: Results of our code with the first canid data set.

Parameter	Estimates	Std Error
β_0	0.04883	3.73460
β_1	-9.31656	5.72461
β_2	0.22416	0.02903
β_3	0.08220	0.04003

Table 2: Results of the linear regression with the first canid data set.

Parameter	Estimates	Std Error
β_0	13.76471	4.82301
β_1	0.37045	5.21600
β_2	-0.35294	0.53003
β_3	-0.01156	0.58150

Table 3: Results of our code with the lizard data set

Parameter	Estimates	Std Error
β_0	13.76471	4.82301
β_1	0.37045	5.21600
β_2	-0.35294	0.53003
β_3	-0.01156	0.58150

Table 4: Results of the linear regression with the lizard data set

Parameter	Estimates
β_0	-1.91141779
β_1	-8.18273501
β_2	0.24380374
β_3	0.05851228
σ^2	20.32750116
τ^2	82.14189284
t	0.19678674

Table 5: Results under Brownian Motion for the first canid data set.

Parameter	Estimates
β_0	4.14618885
β_1	0.80167733
β_2	0.05715093
β_3	0.02903409
σ^2	3.58700887
τ^2	0.67301755
t	0.02100741

Table 6: Results under Brownian Motion for the second canid data set.

Parameter	Estimates
β_0	12.446887630
β_1	0.726860591
β_2	-0.263412550
β_3	0.186805728
σ^2	13.080796297
τ^2	177.860775362
t	0.005066908

Table 7: Results under Brownian Motion for the lizard data set.

Parameter	Estimates
β_0	1.2306484
β_1	-12.4193078
β_2	0.2160755
β_3	0.1000723
σ^2	5.2344894
τ^2	18.0756395
α	14.9999992

Table 8: Results under Ornstein-Uhlenbeck Process for the first canid data set.

Parameter	Estimates
β_0	3.92667130
β_1	0.16753446
β_2	0.01338394
β_3	0.08459182
σ^2	1.06176301
τ^2	0.72237481
α	8.96256396

Table 9: Results under Ornstein-Uhlenbeck Process for the second canid data set.

Parameter	Estimates
β_0	12.30247
β_1	1.666286
β_2	-0.2479737
β_3	-0.008568642
σ^2	16.06123
τ^2	7.490528×10^{-12}
α	15

Table 10: Results under Ornstein-Uhlenbeck Process for the lizard data set.

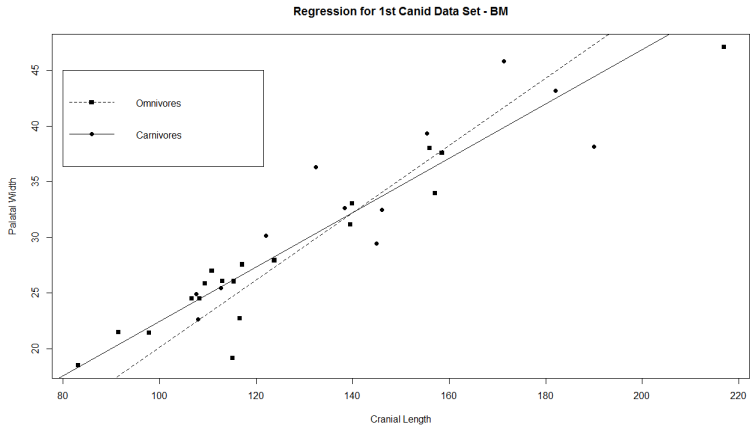


Figure 2: Regression for the first canid data set given by the four β parameters under Brownian Motion. The graph shows that there is a significant difference in the intercepts and slopes between omnivores and carnivores. The lines follow the data well.

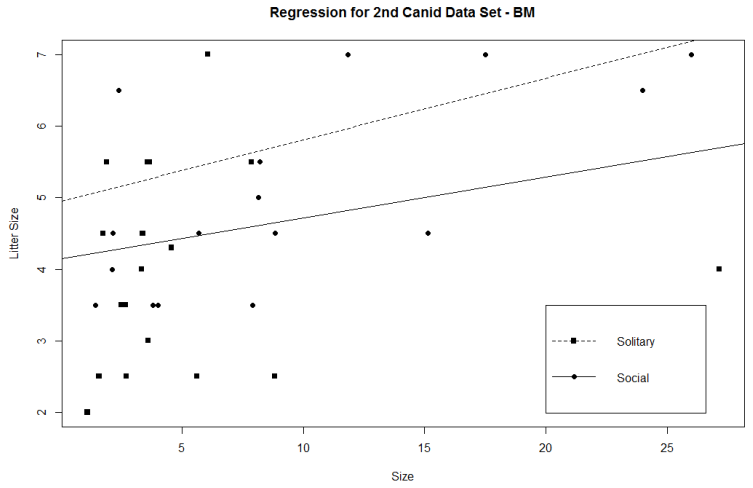


Figure 3: Regression for the second canid data set given by the four β parameters under Brownian Motion. The graph shows that there is a significant difference in the intercepts between solitary and social canids, but there is not much difference in the slopes.

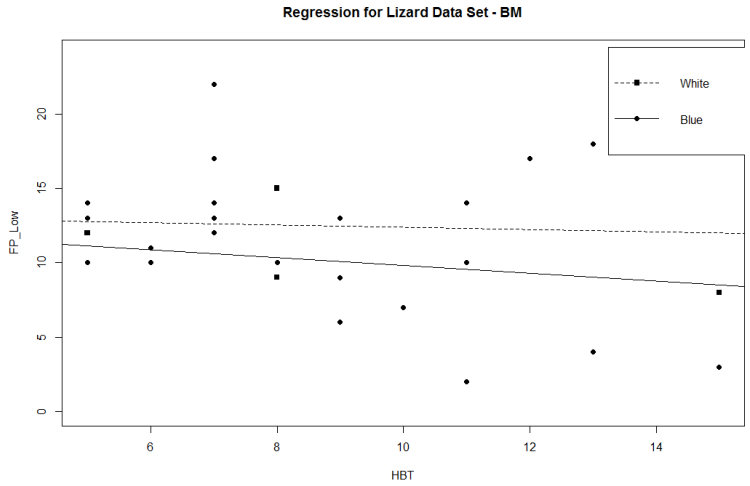


Figure 4: Regression for the lizard data set given by the four β parameters under Brownian Motion. The graph shows that there is a slight difference in the intercepts and slopes between blue and white bellied lizards.

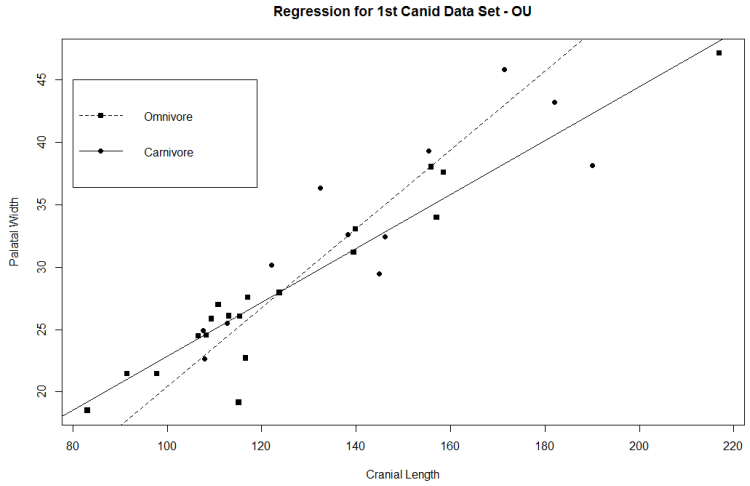


Figure 5: Regression for the first canid data set given by the four β parameters under Ornstein-Uhlenbeck process. The graph shows that there is a significant difference in the intercepts and slopes between omnivores and carnivores. The lines follow the data well.

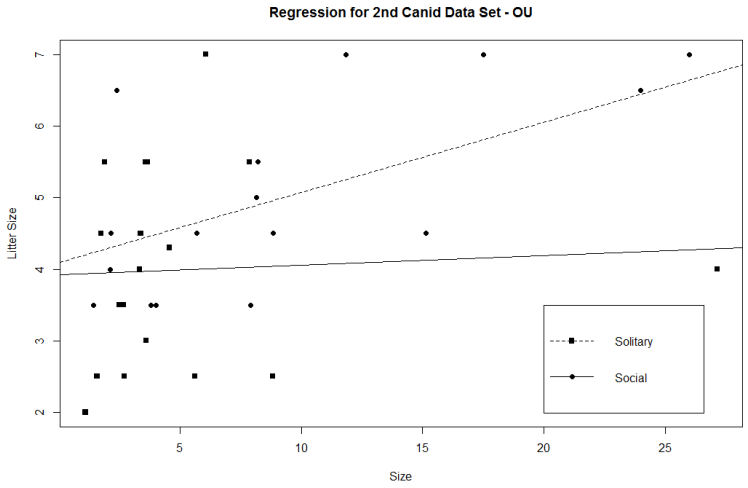


Figure 6: Regression for the second canid data set given by the four β parameters under Ornstein-Uhlenbeck process. The graph shows that there is a slight difference in the intercepts and a large difference in the slopes between social and solitary canids.

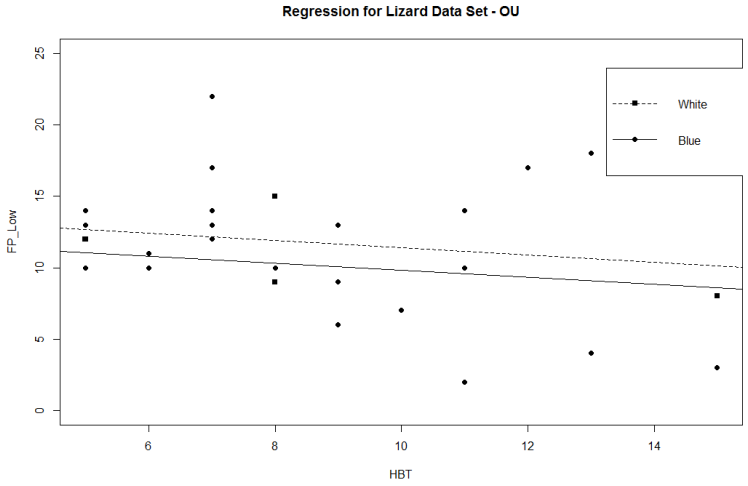


Figure 7: Regression for the lizard data set given by the four β parameters under Ornstein-Uhlenbeck process. The graph shows that there is a slight difference in the intercepts and almost no difference in the slopes between blue and white bellied lizards.

Combinatorics of Curves on Closed Surfaces

Drew Reisinger

Let S be an oriented, closed (but not necessarily connected) surface.

Definition 0.1. A *topological curve system* on S is a finite list of regular, oriented simple closed curves $\Sigma = (\gamma_1, \gamma_2, \ldots, \gamma_m)$ on S, $m \geq 2$, such that

1. all intersections between curves are double points,
2. every curve intersects at least one other curve, and
3. each component of $S \backslash \Sigma$ is a topological disk (here Σ is identified with the trace of its curves).

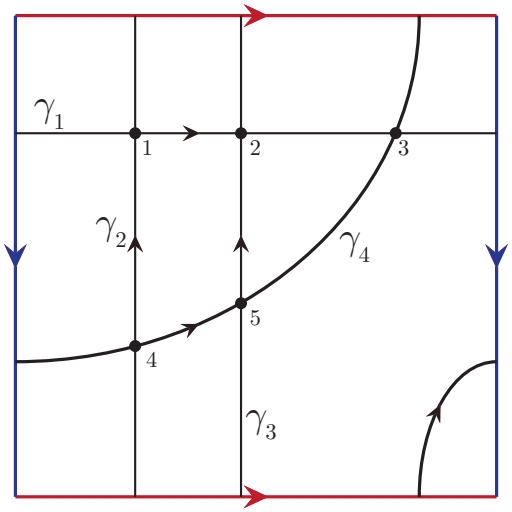


Figure 1: An example of a curve system on a torus

Figure 1 depicts a curve system on a torus consisting of four curves. Note that each of the system's intersection points involves exactly two curves; we will explain the integer labels on these intersection points shortly.

Our goal is a combinatorial description of the *oriented* intersections between curves in a curve system. Label the intersection points of Σ with positive integers $k_1, \ldots, k_n$, $n \geq 1$. For each $\gamma_i \in \Sigma$, let

$$a_i = (a_i^1, a_i^2, \ldots, a_i^{m_i}),$$

$a_i^j \in \{\pm k_1, \pm k_2, \ldots, \pm k_n\}$, be the vector of signed intersection points on γ_i ordered by the curve's orientation where the sign of each entry is determined by intersection number[1] of γ_i with the other curve at that point. We identify these vectors up to cyclic permutation so that this correspondence is well-defined.

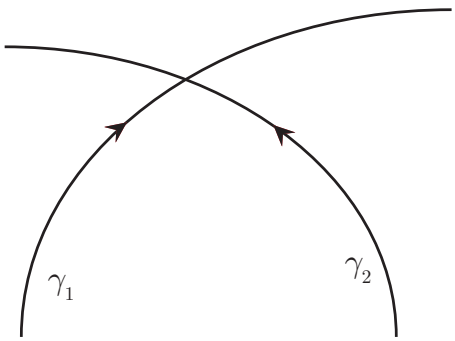


Figure 2: The intersection number of γ_1 with γ_2 at their intersection is $+1$, whereas the intersection number of γ_2 with γ_1 is -1.

For example, the encoding of the system in Figure 1 is

$$\begin{aligned} a_1 &= (1, 2, 3) \\ a_2 &= (-1, -4) \\ a_3 &= (-2, -5) \\ a_4 &= (-3, 4, 5). \end{aligned}$$

We have thus defined a procedure for describing a topological curve system Σ on a surface S with intersection labels $k_1, \ldots, k_n$ with a list of integer vectors, which we call the *encoding* of the tuple $(S, \Sigma, (k_1, \ldots, k_n))$. We now define these lists of vectors more precisely.

[1]The intersection number of a curve γ_1 with another curve γ_2 at a point is $+1$ if γ_2 crosses from the right side of γ_1 to the left side at that point (where "left" and "right" are defined by the orientation of S); otherwise, the intersection number is -1. See Figure 2 for an example.

Definition 0.2. A *combinatorial curve system* on positive integers $k_1, \ldots, k_n$, $n \geq 1$ is a list of vectors $A = (a_1, \ldots, a_m)$, $m \geq 2$, with elements from the set $\{\pm k_1, \ldots, \pm k_n\}$ and identified up to cyclic permutation that satisfies the following conditions:

1. Each element of $\{\pm k_1, \ldots, \pm k_n\}$ appears exactly once in some vector a_i, and
2. $+k_j$ and $-k_j$ never appear in the same vector.

We have already demonstrated a method for associating to each tuple

$$(S, \Sigma, (k_1, \ldots, k_n))$$

—where S is a surface, Σ is a topological curve system on S, and $(k_1, \ldots, k_n)$ is a list of positive integer labels for intersection points of Σ—a combinatorial curve system A on $k_1, \ldots, k_n$. We will now demonstrate that there exists an inverse to this map, thus establishing a one-to-one correspondence between topological and combinatorial curve systems.

Notation. If a_i is a vector in a combinatorial curve system A, and k is an integer in a_i, we denote the cyclic predecessor of k in a_i by k^- and the cyclic successor by k^+. For example, if $a_1 = (1, -2, 3)$ is a vector in some combinatorial curve system A, then $1^+ = -2$, $1^- = 3$, $3^+ = 1$, and $3^- = -2$ in this system.

Theorem 0.3. *Let A be a combinatorial curve system on $k_1, \ldots, k_n$ with $n \geq 1$. Then there exists a surface S and a curve system Σ on S such that the encoding of $(S, \Sigma, (k_1, \ldots, k_n))$ is A.*

We first discuss the intuitive motivation for the construction behind this theorem before we proceed with its formal proof, which can seem cryptic without its geometric context.

One way to understand a topological curve system is as a graph on the surface S whose vertices correspond to the intersection points of the system and whose edges correspond to the sections of curves between intersections (see Figure 3). By the definition of a topological curve system, this graph divides the surface into faces that are homeomorphic to disks. The essence of the proof of Theorem 0.3 is that a combinatorial curve system provides enough information to traverse the edges that bound each face in counter-clockwise order. Once we specify the face boundaries, we can topologically attach disks to insides of these loops. Hence gluing these faces along the appropriate edges will reconstruct the original surface.

In the course of this proof, we introduce the formalism of *segments*. One way to conceptualize segments is as the left and right "sides" of edges in the graph described above. More concretely, consider the edge between k_i and its successor k_i^+. We will let the segment $[k_i, k_i^+]$ represent the left side of this edge while $-[k_i, k_i^+]$ will represent the right side (see Figure 4).

Proof. For each k appearing in some vector in A, let $[k, k^+]$ and $-[k, k^+]$ be disjoint closed unit intervals [0, 1], which we call *segments* at k. Let $\mathcal{S}$ be the

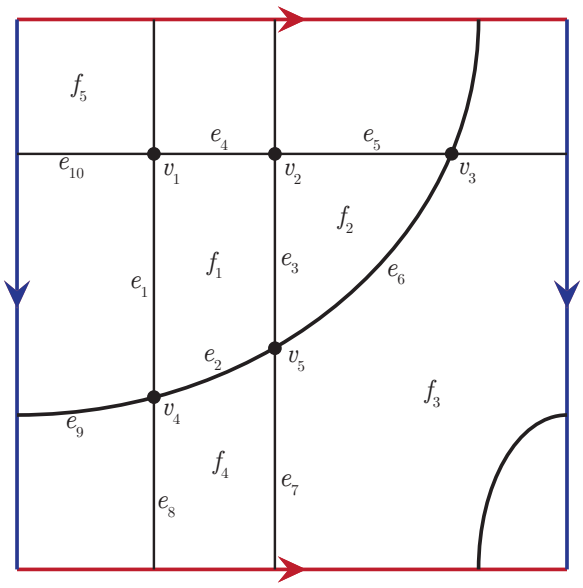


Figure 3: The system from Figure 1 as a graph with its vertices, edges, and faces labeled. Note that some of the edges and faces continue over the identified edges of the square in this diagram.

set of all such disjoint segments. For each segment in $\mathcal{S}$, we define its *successor* based on which of the following four forms the segment takes.

- If a segment is of the form $[k_i^-, k_i]$, then its successor is $[-k_i, (-k_i)^+]$.
- If a segment is of the form $[(-k_i)^-, -k_i]$, then its successor is $-[k_i^-, k_i]$.
- If a segment is of the form $-[k_i, k_i^+]$, then its successor is $-[(-k_i)^-, -k_i]$.
- If a segment is of the form $-[-k_i, (-k_i)^+]$, then its successor is $[k_i, k_i^+]$.

Note that each segment also takes exactly one of the successor forms listed above, so we can also define the *predecessor* of a segment as the inverse of the above operation. Intuitively, the successor operation represents turning left at a vertex while traversing the boundary of a face in the counter-clockwise direction (see Figure 5).

Let

$$U = \bigcup_{s \in \mathcal{S}} s$$

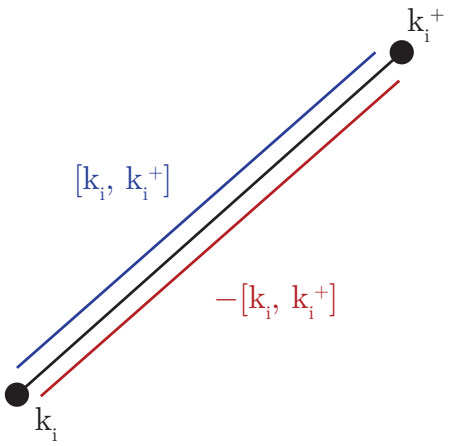


Figure 4: Segments represent the two sides of an edge

be the disjoint union of all these segments, and let L be the quotient space of U formed by identifying the 1 endpoint of each segment with the 0 endpoint of its successor. The space L is then composed of a finite disjoint union of circles, which we call *loops* (see Figure 6). To see this, consider linking the successors of a given segment s. As there are only finitely many segments in $\mathcal{S}$, the end of some successor segment must eventually attach to the beginning of a segment that is already in this loop. But since the predecessor of each segment is unique, only two segments can meet at any one point. It follows that the loop must close up at the beginning of the original segment s and hence is homeomorphic to a circle. Since every segment has a successor, these loops partition L.

For each loop in L, define its *face* to be a closed unit disk, and let $\mathcal{F}$ be the disjoint union of all such faces. Define F to be the quotient space of $L \cup \mathcal{F}$ obtained by identifying each loop with the boundary of its face (see Figure 7).

Finally, we construct S as a quotient space of F by identifing each segment $[k, k^+]$ with its negative counterpart $-[k, k^+]$ such that the 0 endpoint of each segment is attached to the 1 endpoint of its negative; in essence, we are joining the left and right sides of each edge with opposite orientation. Formally, we refer to the images of segment pairs under this quotient map as *edges*. We denote the edge formed by the segments $[k, k^+]$ and $-[k, k^+]$ with e_k.

We now show that S is, in fact, a surface by exhibiting a neighborhood of each point in S that is homeomorphic to an open disk. By definition, points on the interior of some face have such a neighborhood. Now consider a point p in the interior of some edge. On each bordering face, the point is contained in a half-disk; in the quotient, these two half-disks meet to form an open disk

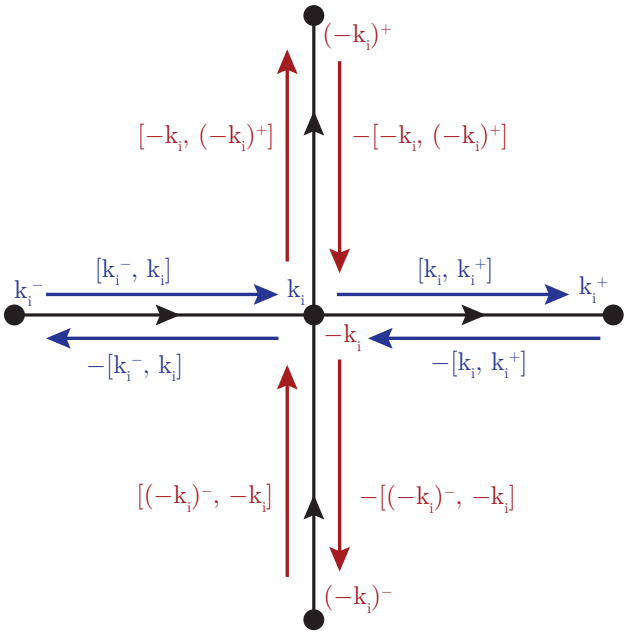


Figure 5: The successor operation represents turning left at a vertex.

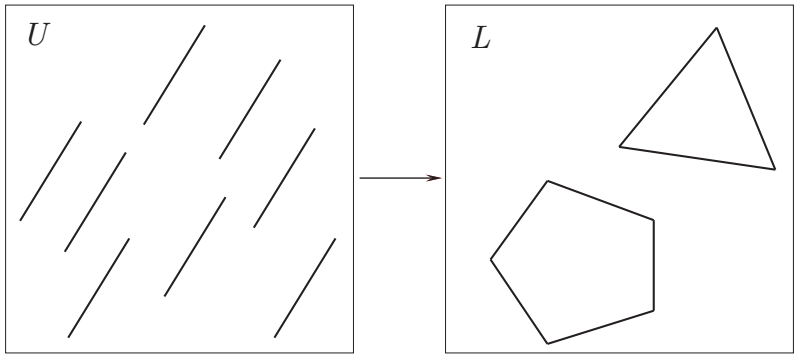


Figure 6: The space L is formed from U by connecting each segment to its successor.

around p. Finally, consider a point p at the intersection of edges. In each of the four adjacent faces, p is contained in a sector of an open disk; in the quotient, these sectors meet along their edges to form an open disk around p. The latter two cases are illustrated in Figure 8.

As S is the quotient space of a finite union of closed disks, which is compact, it is also compact, and since each boundary component of a face in F is identified with some other boundary component, it follows that S is a closed surface.

Each disk in F inherits an orientation from the natural orientation of its loop, which runs in the direction from a segment to its successor. Since each segment is identified with its negative counterpart in the *opposite* direction, it follows that the orientations of the adjacent faces are preserved across the edge (see Figure 9). Thus S is an oriented surface.

Finally, we construct the curve system $\Sigma = \{\gamma_1, \ldots, \gamma_n\}$ on S. For each vector $a_i = (a_i^1, \ldots, a_i^{m_i})$ in A, let γ_i consist of the edges $e_{a_i^j}$ oriented so that γ_i traces these edges in the cyclic order specified by a_i, that is, in the order $e_{a_i^1}$, $e_{a_i^2}$, $e_{a_i^3}$, and so on. If $k \in a_i$, then $-k \in a_j$ for some $i \neq j$, and hence γ_i intersects γ_j; thus Σ satisfies condition 1 of Definition 0.1. Condition 2 is guaranteed by the requirement that if k appears in some a_i, then $-k$ is not in a_i. Finally, condition 3 follows from the construction of S from closed disks and that the curves in Σ comprise the boundaries of these disks. By construction, A is the encoding of $(S, \Sigma, (k_1, \ldots, k_n))$. □

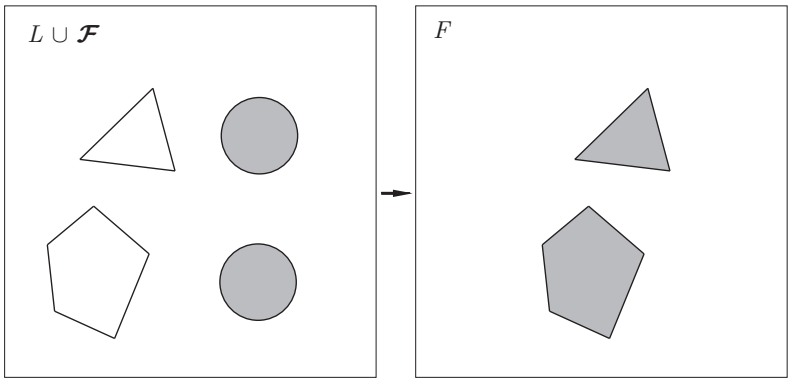


Figure 7: F is constructed from $L \cup \mathcal{F}$ by "filling in" each loop with a closed disk.

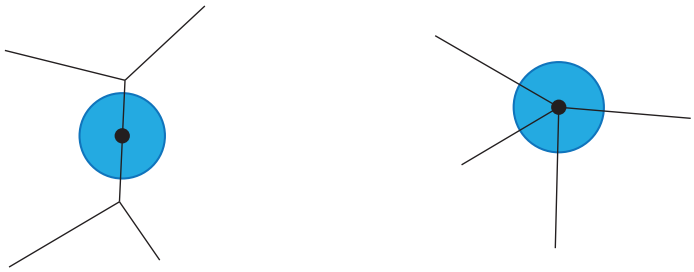


Figure 8: The two cases of points on the boundary of some face. In both cases, the neighborhoods of the point in each face meet to form an open disk in the quotient.

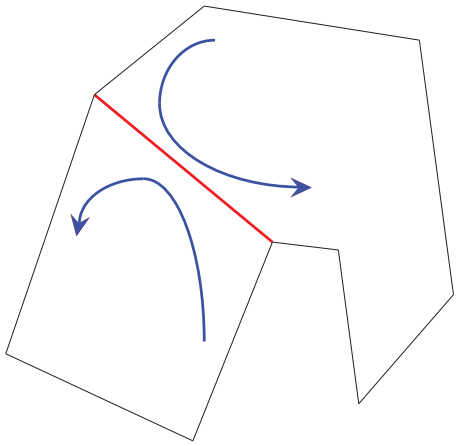


Figure 9: Two faces attached along the highlighted edge. The blue arrows define "left" on each face, and because the faces are being attached in the opposite direction, the new surface inherits this orientation.