Research Experience for Undergraduates Program
Research Reports

Indiana University, Bloomington

Summer 2017

Contents

Preface

During the summer of 2017 five students participated in the Undergraduate Research Experience program in Mathematics at Indiana University. This program was sponsored by the National Science Foundation through the Research Experience for Undergraduates grant #1461061 and the Department of Mathematics at Indiana University, Bloomington. The program ran for eight weeks, from June 5 through July 30, 2017. Five faculty served as research advisers to the students from Indiana University:

- Corrin Clarkson worked with Rebecca Rohrlich.
- Chris Connell worked with Anabel Tassoni.
- Mike Jolly worked with Anthony Isenberg.
- Larry Moss worked with Charlotte Raty.
- Kevin Pilgrim and Dylan Thurston worked with Jackson Morris and Matthew Powell.
- Carmen Rovi worked with Matthew Schoenbauer.
- Amr Sabry worked with Sivva Somayyajula.

Following the introductory pizza party, students began meeting with their faculty mentors and continued to do so throughout the next eight weeks. The students also participated in a number of social events and educational opportunities and field trips.

Individual faculty gave talks throughout the program on their research, about two a week. Students also received LaTeX training in a series of workshops. Other opportunities included the option to participate in a GRE and subject test preparation seminar. Additional educational activities included tours of the library, the Slocum puzzle collection at the Lilly Library and the IU cyclotron facility, and self guided tours of the art museum. Students presented their work to faculty mentors and their peers at various times. This culminated in their presentations both in poster form and in talks at the statewide Indiana Summer Undergraduate Research conference which we hosted at the Bloomington campus of IU.

On the lighter side, students were treated to a reception by the graduate school as well as the opportunity to a fun filled trip to a local amusement park. They were also given the opportunity to enjoy a night of “laser tag” courtesy of the Department of Mathematics.

The summer REU program required the help and support of many different groups and individuals to make it a success. We foremost thank the Indiana University and the Department of Mathematics for major financial support for this bridge year between two National Science Foundation grants. We especially thank our staff member Mandie McCarty for coordinating the complex logistical arrangments (housing, paychecks, information packets, meal plans, frequent shopping for snacks). Additional logistical support was provided by the Department of Mathematics and our chair, Elizabeth Housworth. We are in particular thankful to Jeff Taylor for the computer support he provided. Thanks to those faculty who served as mentors and those who gave lectures. Thanks to David Baxter of the Center for Exploration of Energy and Matter (nee IU cyclotron facility) for his personal tour of the LENS facility and lecture. Thanks to Andrew Rhoda for his tour of the Slocum Puzzle Collection.

Chris Connell
September, 2017

Figure 1: REU Participants, from left to right: Charlotte Raty, Anabel Tassoni, Siva Somayyajula, Anthony Isenberg, Chris Connell, Matthew Powell, Morris Jackson and Matthew Shoenbauer. Not Shown: Rebecca Rohrlich.

Examining the Heegaard Floer Homology of Torus Bundles

Dr. Corrin Clarkson and Rebecca Rohrlich

July 28, 2017

1 Abstract

The objective of this project is to determine whether or not the Heegaard Floer homology is a complete topological invariant of torus bundles. Although we do not answer this question definitively, we do show that the Heegard Floer homology distinguishes members of certain families of torus bundles. Our work relies heavily on results discovered by John A. Baldwin, Peter Ozvath and Zoltan Szabo, and Louis Funar.

2 Introduction

2.1 Torus bundles

To understand torus bundles, we first must state a few definitions.

Definition 1. Let X be a topological space and let $f : X \to X$ be a homeomorphism. Then a *mapping torus* is the quotient space $(X \times I)/\big((x, 0) \sim (f(x), 1)\big)$, where $I = [0, 1]$.

Definition 2. A *torus bundle* is a mapping torus created with a homeomorphism $f : T^2 \to T^2$.

Hence, we have the following surjective map:

$$\text{homeomorphisms of the 2-torus} \quad \longrightarrow \quad \text{torus bundles}$$

One example of a torus bundle is the 3-torus.

Example 1. The 3-torus is a three-manifold of the form $S^1 \times S^1 \times S^1$ (a product of three circles). We can construct it in our minds by imagining taking a cube and gluing each pair of its opposite sides together. After the first gluing, we have a solid torus; after the second gluing, we have a solid torus with a hollow tube running through it; and the last gluing merges the inside wall of this hollow tube to the outside wall of the solid torus, a step which cannot be performed in 3-space.

From our definition of torus bundles and our bijection, it is clear that to understand tours bundles we must understand the automorphism group, or "mapping class group," of the torus. Since the torus is the product of two circles, any point on the torus can be identified by an ordered pair (θ, ϕ), where θ and ϕ are each reals describing an angle on one of the circles. Since $\theta = \theta + 2\pi n$ and $\phi = \phi + 2\pi n \ \forall n \in \mathbb{Z}$, we see that the torus can be regarded as the quotient $\mathbb{R}^2/\mathbb{Z}^2$. Hence, any automorphism of the torus can be regarded as a linear combination of basis vectors

$$m = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^5, l = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

where m and l represent longitude (circle around the torus "doughnut hole"), and longitude (other circle) respectively. Since any linear map of m and l can be represented as a 2×2 invertible matrix, we see that the mapping class group of the torus, $MCG(T^2)$, is $\mathrm{Aut}(\mathbb{Z} \oplus \mathbb{Z}) \cong GL_2(\mathbb{Z})$. Restricting ourselves to orientation-preserving automorphisms on the torus, we only examine $SL_2(\mathbb{Z}) \subset GL_2(\mathbb{Z})$ – namely, the the set of 2×2 invertible matrices with determinant 1.

2.2 Dehn twist factorization

From a theorem of Dehn and Lickorish, we know that any automorphism of the torus can be created by a series of twists, called Dehn twists. Although Dehn twists can be performed on any closed curves on the torus, the two kinds of Dehn twists that are linearly independent (and thus can create all the automorphisms of the torus) are longitudinal twists and meridional twists.

Definition 3. Let m be a meridian on the torus. Then we can split m into two curves (the upper and lower halves of the circle), and we know that either of these curves c lies in a neighborhood homeomorphic to an annulus. We see that c lies along some ray extending from the center of the annulus and has endpoints on both of the annulus's rings. A *meridional twist* on m can be described by the continuous map $f : m \to T^2$ such that for any $(r, \theta) \in m$, where r is the radius length from the center of the annulus, we have $f(r, \theta) = (r, \theta + 2\pi r)$. A *longitudinal twist* on a longitude of the torus behaves is constructed identically, but with a longitude l.

The longitudinal twist and meridional twist can be represented by the monodromies x and y, where

$$x = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, y = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$$

Hence, any monodromy has a factorization into x's and y's (although this factorization may be not be unique), and conversely, every factorization gives rise to a monodromy.

2.3 *Spin*^{c} structures and d-invariants

The technical definition of $Spin^c$ structures uses advanced material from algebra and topology that are beyond the author's knowledge. However, the set of $Spin^c$ structures for a 3-manifold are in bijection with the abelianization of the manifold's fundamental group (the quotient of the fundamental group group by the commutator subgroup), and we can think of a given $Spin^c$ structure for the manifold as an orientation. A $Spin^c$ it is an extra piece of information that, together with the 3-manifold, provides the necessary input for computation of Heegaard Floer homology.

The d-invariant allows us to identify $Spin^c$ different structures in a simpler way. There is a function $d : Spin^c(M^3) \to \mathbb{Q}$, and, in a slight abuse of notation, we refer to the output of this function as d as well.

2.4 Heegaard Floer Homology

The objective of this project is to determine whether or not the Heegaard Floer (HF) homology is a complete topological invariant on torus bundles. If HF homology is indeed proven to be a complete invariant, it would be quite useful, because the algorithm to compute HF homology for a given monodromy (the matrix corresponding to a torus bundle) is relatively quick and efficient.

Let $(M^3, \mathfrak{s})$ be a closed, oriented 3-manifold together with a $Spin^c$ structure $\mathfrak{s}$. HF homology associates to any $(M^3, \mathfrak{s})$ some $\mathbb{Q}$-graded $\mathbb{Z}[U]$-module. The abelianization in bijection with the set of $Spin^c$ structures for the manifold is always of the form $\mathbb{Z} \times H$, where $H = \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_n}$

is a finite polycycic group. Let $i \times (x_1, \dots, x_n) \in \mathbb{Z} \times H$. From John Baldwin's paper, we know that $\forall i \neq 0$, we have $HF^+\big(M^3, (i \times (x_1, \dots, x_n))\big) = 0$. Therefore, since we need only worry about non-trivial HF homologies, we only examine a finite number of $Spin^c$ structures for any given torus bundle.

3 Methods

3.1 Algorithm to determine the x, y factorization

As we stated previously, there is not necessarily a unique factorization for a given monodromy. However, there is an algorithm to find one such factorization. The algorithm was originally implemented in C by other developers, and re-implemented by the author in Python [4]. The algorithm proceeds as follows:

$$\text{Given a monodromy } M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \text{ while } M \neq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}:$$

If $a \leq c$ and $b \leq d$, then set

$$M = \begin{pmatrix} a & b \\ c-a & d-b \end{pmatrix} \text{ and add an } x \text{ to the factorization.}$$

Otherwise, set

$$M = \begin{pmatrix} a-c & b-d \\ c & d \end{pmatrix} \text{ and add a } y^{-1} \text{ to the factorization.}$$

By the time M becomes the identity matrix, the algorithm terminates.

3.2 Black Graph and Q

We can regard any x, y factorization as a three-stranded braid, where x moves the first strand over the second strand and y moves the third strand over the second strand. From this braid, we color the interior regions black, place vertices on the black regions, and place edges over the folds in the black surface. (This graph will always be connected.) Over this graph, we draw a spanning tree. Then, we construct a square matrix Q as follows.

Each row and column of Q corresponds to an edge on the black graph which is not in the spanning tree; call this set of edges $E^* = e_1, \dots, e_n$. For each edge $e \in E^*$, find the cycle that would be created by adding e to the graph and orient the cycle. Each entry with the coordinates (e_i, e_i) (a.k.a, the edges along the diagonal), is filled with the length of the cycle that would be created by adding e_i to the graph. For all other coordinates (e_i, e_j), with corredponding cycles C_i and C_j, the entry is filled with $-1 \cdot |C_i \cap C_j|$ if the cycles are orient their shared edges in the same direction, and $|C_i \cap C_j|$ otherwise [2]. The author wrote a Python script to create Q for any of Baldwin's case 1 monodromies, in his Theorem 6.4 [1].

3.3 Computation of d

The matrix Q is necessary for the computation of the d-invariant, which (as the reader may recall) identifies the Torus bundle's $Spin^c$ structure.

Definition 4. Let $m_1, ..., m_n$ be the entries along the diagonal of Q. Let

$$\vec{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \text{ and let } \begin{array}{rcl} v_1 & \in & \{-|m_1|, -|m_1|+2, \ldots, -|m_1|+2i, \ldots, |m_1|-2i, |m_1|-2, |m_1|\} \\ \vdots & & \\ v_n & \in & \{-|m_n|, -|m_n|+2, \ldots, -|m_n|+2i, \ldots, |m_n|-2i, |m_n|-2, |m_n|\} \end{array}$$

Then, $\vec{v}$ is a *characteristic vector* of Q.

Definition 5. Let $\vec{v}$ be a characteristic vector of the matrix Q. Then the *norm* of $\vec{v}$ is the scalar $k = \vec{v} \cdot Q^{-1} \cdot \vec{v}$.

There is a certain equivalence relation on the characteristic vectors of Q such that we have a function

$$d : \{\text{equivalence classes of characteristic vectors of } Q\} \to \mathbb{Q} \tag{1}$$

such that

$$d = \frac{k_{max} + b}{4},$$

where k_{max} is the maximum norm of all vectors in the set of characteristic vectors.

However, clearly if the maximum of the d-outputs for one monodromy is larger than the maximum of the d-outputs for another monodromy, then the two monodromies cannot have the same HF homology. So far, we have found it sufficient to compare only these global maximum d's for different monodromies in order to tell whether or not they are distinct, and have not found it necessary to examine the characteristic vector equivalence classes [2].

3.4 Application of Funar's Theorem 1.3

If the HF-homology is indeed a complete invariant on Torus bundles, it would be most useful to distinguish tori that are not distunguished by other invariants, such as TQFT invariants. Given an Anosov matrix [3]

$$A = \begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{pmatrix}$$

the corresponding fundamental group is the polycyclic group [3]

$$\Gamma = \langle t, a, b | ab = ba, tat^{-1} = \alpha_{11}\alpha_{12}, tbt^{-1} = \alpha_{21}\alpha_{22} \rangle \tag{2}$$

In Funar's Theorem 1.3, Funar provides an equation $4a^2 + b^2 = u^2 + 4$, and he states that, for a given u, all a, b solutions to the above equation correspond to a matrix of the form

$$\begin{pmatrix} \frac{1}{2}(u^2 - b \cdot u + 2) & a \cdot u \\ a \cdot u & \frac{1}{2}(u^2 + b \cdot u + 2) \end{pmatrix} [3]$$

Funar shows that matrices of this form, with a given u, satisfy two properties:

1. They all have the same first homology group, and thus the same set of $Spin^c$ structures.
2. Their HF homologies are isomorphic as modules (not taking into account the grading shift).

Thus, the HF homologies of the monodromies in Funar's u-families were already similar in almost every aspect – they could only be different monodromies if they had different grading shift. For this reason, we found Funar's u-families to be promising families of matrices in which to search for counterexamples to HF homology being a complete invariant on torus bundles.

4 Results

Using Funar's Theorem 1.3, we computed all monodromies for a given u for all values of u between 1 and 50. Furthermore, we computed the factorizations of all these monodromies, and computed d-invariants for the first 34 of them. These results are summarized in the following table. The first column of the table is the 2×2 monodromy, the second column is d, and the third is the list of powers of y^{-1} describing the x, y factorization. (For example, the list $[2, 1, 0, 1]$ would correspond to the factorization $xy^{-2}xy^{-1}xy^{0}xy^{-1}$).

```
************
GROUP: u = 4
******************
[[ 1  4]
 [ 4 17]] 1.0 [0,0,0,4]
[[ 5  8]
 [ 8 13]] 0.5 [1,1,1]
```

```
************
GROUP: u = 6
******************
[[ 1  6]
 [ 6 37]] 1.5 [0,0,0,0,0,6]
[[13 18]
 [18 25]] 0.8648648648648649 [2,1,0,1]
```

```
************
GROUP: u = 8
******************
[[ 1  8]
 [ 8 65]] 2.0 [0,0,0,0,0,0,0,8]
[[25 32]
 [32 41]] 1.0 [3,1,0,0,1]
```

```
************
GROUP: u = 9
******************
[[ 1  9]
 [ 9 82]] 2.2222222222222223 [0,0,0,0,0,0,0,0,9]
[[10 27]
 [27 73]] 1.0136986301369864 [0,1,0,2,2]
```
9

```
************
GROUP: u = 10
*******************
[[  1  10]
 [ 10 101]] 2.5 [0,0,0,0,0,0,0,0,0,10]
[[41 50]
 [50 61]] 1.3 [4,1,0,0,0,1]



************
GROUP: u = 11
*******************
[[  1  11]
 [ 11 122]] 2.727272727272727 [0,0,0,0,0,0,0,0,0,0,11]
[[34 55]
 [55 89]] 0.9090909090909092 [1,1,1,1,1]



************
GROUP: u = 12
*******************
[[  1  12]
 [ 12 145]] 3.0 [0,0,0,0,0,0,0,0,0,0,0,12]
[[61 72]
 [72 85]] 1.5 [5,1,0,0,0,0,1]



************
GROUP: u = 14
*******************
[[  1  14]
 [ 14 197]] 3.5 [0,0,0,0,0,0,0,0,0,0,0,0,0,14]
[[ 29  70]
 [ 70 169]] 1.36 [0,2,0,2,0,2]
[[ 85  98]
 [ 98 113]] 1.7857142857142858 [6,1,0,0,0,0,0,1]



************
GROUP: u = 21
*******************
[[  1  21]
```

```
 [ 21 442]] 5.238095238095238 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,21]
[[106 189]
 [189 337]] 1.4202247191011237 [1,0,0,1,1,3,1]



************
GROUP: u = 22
*******************
[[  1  22]
 [ 22 485]] 5.5 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,22]
[[221 242]
 [242 265]] 2.75 [10,1,0,0,0,0,0,0,0,0,0,1]



************
GROUP: u = 23
*******************
[[  1  23]
 [ 23 530]] 5.739130434782608 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,23]
[[185 253]
 [253 346]] 1.515370705244123 [2,2,1,0,1,0,1]



************
GROUP: u = 24
*******************
[[  1  24]
 [ 24 577]] 6.0 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,24]
[[ 73 192]
 [192 505]] 1.5 [0,1,1,0,2,1,2]
[[ 97 216]
 [216 481]] 2.0 [0,4,0,2,0,0,0,2]
[[265 288]
 [288 313]] 3.0 [11,1,0,0,0,0,0,0,0,0,0,0,1]



************
GROUP: u = 25
*******************
[[  1  25]
 [ 25 626]] 6.24 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,25]
[[ 26 125]
 [125 601]] 1.9853300733496333 [0,0,0,1,0,0,0,4,4]
```

```
************
GROUP: u = 26
*******************
[[  1  26]
 [ 26 677]] 6.5 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,26]
[[ 53 182]
 [182 625]] 1.8823529411764706 [0,0,2,0,0,3,0,3]
[[157 286]
 [286 521]] 1.7080291970802919 [1,0,0,0,1,1,4,1]
[[313 338]
 [338 365]] 3.25 [12,1,0,0,0,0,0,0,0,0,0,0,0,1]
```

```
************
GROUP: u = 28
*******************
[[  1  28]
 [ 28 785]] 7.0 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,28]
[[365 392]
 [392 421]] 3.5 [13,1,0,0,0,0,0,0,0,0,0,0,0,0,1]
```

```
************
GROUP: u = 29
*******************
[[  1  29]
 [ 29 842]] 7.241379310344827 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,29]
[[146 319]
 [319 697]] 2.206896551724138 [0,5,0,2,0,0,0,0,2]
[[233 377]
 [377 610]] 1.310344827586207 [1,1,1,1,1,1,1]
```

```
************
GROUP: u = 30
*******************
[[  1  30]
 [ 30 901]] 7.5 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,30]
[[421 450]
 [450 481]] 3.75 [14,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
```

```
************
GROUP: u = 31
******************
[[  1  31]
 [ 31 962]] 7.741935483870968 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,31]
[[218 403]
 [403 745]] 1.9201637666325486 [1,0,0,0,0,1,1,5,1]



************
GROUP: u = 32
******************
[[   1   32]
 [  32 1025]] 8.0 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32]
[[481 512]
 [512 545]] 4.0 [15,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]



************
GROUP: u = 34
******************
[[   1   34]
 [  34 1157]] 8.5 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,34]
[[ 137  374]
 [ 374 1021]] unknown [0,1,0,1,0,2,2,2]
[[205 442]
 [442 953]] unknown [0,6,0,2,0,0,0,0,0,2]
[[545 578]
 [578 613]] 4.25 [16,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]



************
GROUP: u = 36
******************
[[   1   36]
 [  36 1297]] 9.0 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,36]
[[  37  216]
 [ 216 1261]] unknown [0,0,0,0,1,0,0,0,0,5,5]
[[ 109  360]
 [ 360 1189]] unknown [0,0,3,0,0,3,0,0,3]
[[ 289  540]
 [ 540 1009]] unknown [1,0,0,0,0,0,1,1,6,1]
[[433 612]
 [612 865]] unknown [2,0,2,1,0,2,0,1]
[[613 648]
```

13

```
 [648 685]] 4.5 [17,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]


************
GROUP: u = 38
*******************
[[   1   38]
 [  38 1445]] 9.5 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,38
[[685 722]
 [722 761]] 4.75 [18,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]


************
GROUP: u = 39
*******************
[[   1   39]
 [  39 1522]] 9.743589743589743 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
[[ 274  585]
 [ 585 1249]] unknown [0,7,0,2,0,0,0,0,0,0,2]
[[586 741]
 [741 937]] unknown [3,3,1,0,0,1,0,0,1]


************
GROUP: u = 40
*******************
[[   1   40]
 [  40 1601]] 10.0 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
[[761 800]
 [800 841]] 5.0 [19,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]


************
GROUP: u = 41
*******************
[[   1   41]
 [  41 1682]] 10.24390243902439 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
[[ 370  697]
 [ 697 1313]] unknown [1,0,0,0,0,0,0,1,1,7,1]


************
GROUP: u = 42
```

```
******************
[[   1   42]
 [  42 1765]] 10.5 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
[[  85  378]
 [ 378 1681]] unknown [0,0,0,2,0,0,0,4,0,4]
[[ 505  798]
 [ 798 1261]] unknown [1,2,1,1,1,0,1,1]
[[841 882]
 [882 925]] 5.25 [20,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]



************
GROUP: u = 43
******************
[[   1   43]
 [  43 1850]] 10.744186046511627 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
[[ 130  473]
 [ 473 1721]] unknown [0,0,1,1,0,0,3,1,3]



************
GROUP: u = 44
******************
[[   1   44]
 [  44 1937]] 11.0 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
[[ 221  616]
 [ 616 1717]] unknown [0,1,0,0,1,0,2,3,2]
[[ 353  748]
 [ 748 1585]] unknown [0,8,0,2,0,0,0,0,0,0,0,2]
[[ 925  968]
 [ 968 1013]] 5.5 [21,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]



************
GROUP: u = 46
******************
[[   1   46]
 [  46 2117]] 11.5 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
[[ 185  598]
 [ 598 1933]] unknown [0,0,4,0,0,3,0,0,0,3]
[[ 461  874]
 [ 874 1657]] unknown [1,0,0,0,0,0,0,0,1,1,8,1]
[[1013 1058]
 [1058 1105]] 5.75 [22,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
```

```
************
GROUP: u = 48
*******************
[[   1   48]
 [  48 2305]] 12.0 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
[[1105 1152]
 [1152 1201]] 6.0 [23,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
```

```
************
GROUP: u = 49
*******************
[[   1   49]
 [  49 2402]] 12.244897959183673 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
[[  50  343]
 [ 343 2353]] unknown [0,0,0,0,0,1,0,0,0,0,0,6,6]
[[ 442  931]
 [ 931 1961]] unknown [0,9,0,2,0,0,0,0,0,0,0,0,0,2]
[[ 785 1127]
 [1127 1618]] unknown [2,0,0,2,1,0,3,0,1]
```

```
************
GROUP: u = 50
*******************
[[   1   50]
 [  50 2501]] 12.5 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
[[1201 1250]
 [1250 1301]] 6.25  [24,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
```

Theorem 1. *For any even integer $u \geq 4$, let $n = \frac{u}{2} - 1$. Then $\exists$ a monodromy M,*

$$M = (xy^{-n})(xy)x^{n-1}(xy)$$

such that both M and the monodromy A corresponding to the given u (see Funar, Thm. 1.3) have the same abelianization, but different maximum d-invariants.

Proof. **4.1 Same Abelianization for M and A**

We use the following matrices for x and y:

$$x = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}^{16}, y = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$$

Thus, we have

$$M = \begin{pmatrix} 2n^2+2n+1 & 2n^2+4n+2 \\ 2n^2+4n+2 & 2n^2+6n+5 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(u^2-2u+2) & (n+1)u \\ (n+1)u & \frac{1}{2}(u^2+2u+2) \end{pmatrix}$$

Clearly, M follows the format of the matrices described in Funar's Theorem 1.3 if we let $a = n+1$ and $b = 2$. Furthermore, we see that

$$4a^2 + b^2 = 4(n+1)^2 + 2^2 = 4n^2 + 8n + 8 = (2n+2)^2 + 4 = u^2 + 4$$

and thus Funar's number theoretic condition is satisfied. Funar and Clarkson already showed that the A matrices obey these constraints for any u. Therefore, for any even u, M and A have the same abelianization.

4.2 Matrices Q and Q^{-1}

The knot and black graph corresponding to M give rise to the following matrix Q:

$$\begin{pmatrix} -(n+2) & -1 & -(n+1) & 0 & \dots & \dots & 0 \\ -1 & -3 & -2 & -1 & \dots & \dots & -1 \\ -(n+1) & -2 & -(n+3) & -1 & \dots & \dots & -1 \\ 0 & -1 & -1 & -2 & -1 & \dots & -1 \\ \vdots & \vdots & \vdots & -1 & \ddots & & \vdots \\ \vdots & \vdots & \vdots & \vdots & & \ddots & -1 \\ 0 & -1 & -1 & -1 & \dots & -1 & -2 \end{pmatrix}$$

Thus, Q^{-1} is of the form

$$\frac{1}{2n+2} \cdot \begin{pmatrix} -(n+1) & 0 & n & -1 & \dots & \dots & -1 \\ 0 & -(n+1) & 1 & 1 & \dots & \dots & 1 \\ n & 1 & -(n+1) & 1 & \dots & \dots & 1 \\ -1 & 1 & 1 & -2n & 2 & \dots & 2 \\ \vdots & \vdots & \vdots & 2 & \ddots & & \vdots \\ \vdots & \vdots & \vdots & \vdots & & \ddots & 2 \\ -1 & 1 & 1 & 2 & \dots & 2 & -2n \end{pmatrix}$$

4.3 Characteristic Vectors

Let v be any characteristic vector of Q and v_i be the i^{th} coordinate of v. Clearly, v has length $n+2$ and each v_i is defined as follows:

$$\begin{aligned} v_1 &\in \{-(n+2), -n, \dots, n, n+2\} \\ v_2 &\in \{-3, -1, 1, 3\} \\ v_3 &\in \{-(n+3), -(n+1), \dots, n+1, n+3\} \\ v_4, \dots, v_{n+2} &\in \{-2, 0, 2\} \end{aligned}$$

4.4 Formula for d

The d corresponding to v is

$$d = \frac{v \cdot Q^{-1}(v) + \dim(Q)}{4} = \frac{v \cdot Q^{-1}(v) + (n+2)}{4}$$

Thus, to find the maximum d, d_{max}, we must find the characteristic vector v_{max} corresponding to the maximum norm.

The general formula for the norm, $v \cdot Q^{-1}(v)$, is

$$v \cdot Q^{-1}(v) = 2 \cdot (-v_1 + v_2 + v_3) \sum_{i=4}^{n+2} v_i + 2 \sum_{i=4}^{n+2} v_i \left(\sum_{j=4}^{n+2} v_j \right) - (2+2n) \sum_{j=4}^{n+2} {v_j}^2 - (n+1) \sum_{i=1}^{3} {v_i}^2 + 2v_3(nv_1 + v_2)$$

Using cancelation, we have

$$v \cdot Q^{-1}(v) = 2 \cdot (-v_1 + v_2 + v_3) \sum_{i=4}^{n+2} v_i + 4 \sum_{i=4}^{n+2} \left(\sum_{j=i+1}^{n+2} v_i v_j \right) - 2n \sum_{j=4}^{n+2} {v_j}^2 - (n+1) \sum_{i=1}^{3} {v_i}^2 + 2v_3(nv_1 + v_2)$$

4.5 Lower Bound on d_{max}

We will construct a lower bound on d_{max}. The possibilites for each v_i were shown in section 1. Since the options for v_1 and v_3 will differ depending on whether n is odd or even, we examine each case separately.

Suppose n is odd. Then we can have

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ \vdots \\ v_{n+2} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

which gives $v \cdot Q^{-1}(v) = -1$, and thus

$$d = \frac{-1 + (n+2)}{2} = \frac{n+1}{4} = \frac{u}{8}.$$

Suppose n is even. Then we can have

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ \vdots \\ v_{n+2} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

which gives $v \cdot Q^{-1}(v) = \frac{-2n}{2n+2} = -\frac{n}{n+1}$, and thus

$$d = \frac{-\frac{n}{n+1} + (n+2)}{2} = \frac{n^2 + 2n + 2}{4(n+1)} = \frac{u}{8} + \frac{1}{2u}.$$

Therefore, we have

$$\begin{aligned} d_{max} &\geq \tfrac{u}{8} + \tfrac{1}{2u} && \text{if } n \text{ is even,} \\ d_{max} &\geq \tfrac{u}{8} && \text{if } n \text{ is odd.} \end{aligned}$$

4.6 Upper Bound on d_{max}

To find an upper bound on d_{max}, we will break up the summands in the formula for $v \cdot Q^{-1}(v)$ into three groups, and maximize each group.

The first group is comprised of $4 \sum_{i=4}^{n+2} \left(\sum_{j=i+1}^{n+2} v_i v_j \right)$ and $-2n \sum_{j=4}^{n+2} {v_j}^2$. Since $-2n$ is expected to cause the second term grow more quickly than the first term, we want to minimize $v_4 \ldots v_{n+2}$. Clearly, this means we set $v_4 \ldots v_{n+2}$ equal to 0.

The second group is comprised of the terms $-(n+1) \sum_{i=1}^{3} {v_i}^2$ and $2v_3(nv_1 + v_2)$, which, combined, simplify to $-n(v_1 - v_3)^2 - (v_2 - v_3)^2 - n{v_2}^2 - {v_1}^2$. Each of these terms is negative and involves a square, so to maximize the expression we set v_1, v_2, and v_3 to values as close to 0 as possible. These sets of values differ depending on whether n is even or odd (see end of section 1). Specifically, we set

$$\begin{array}{lcl} v_1 & = & \pm 1 \\ v_2 & = & \pm 1 \\ v_3 & = & 0 \end{array} \text{ when } n \text{ is even, and } \begin{array}{lcl} v_1 & = & 0 \\ v_2 & = & \pm 1 \\ v_3 & = & \pm 1 \end{array} \text{ when } n \text{ is odd.}$$

Thus, we have

$$-(n+1) \sum_{i=1}^{3} {v_i}^2 + 2v_3(nv_1 + v_2) = \begin{cases} -2n - 2 & \text{if } n \text{ is odd} \\ -2n & \text{if } n \text{ is even} \end{cases}$$

The third group is simply the term $2(-v_1 + v_2 + v_3) \sum_{i=4}^{n+2} v_i$. To maximize this term, we set the following equalities:

$$\begin{array}{rcl} v_1 & = & -(n+2) \\ v_2 & = & 3 \\ v_3 & = & n+3 \\ v_4 \ldots v_{n+2} & = & 2 \end{array}$$

which give

$$2(-v_1 + v_2 + v_3) \sum_{i=4}^{n+2} v_i = 8n^2 + 16n - 24.$$

Finally, now that the three groups have been individually maximized, we take their sum. Since the maximized first group evaluated to 0, we are really just summing the second and third groups. Once again, we have two cases. In n is even, our sum is

$$v \cdot Q^{-1}(v) = \frac{(8n^2 + 16n - 24) + (-2n - 2)}{2n + 2} = \frac{4n^2 + 7n - 13}{n + 1}, \tag{3}$$

whereas if n is odd, our sum is

$$v \cdot Q^{-1}(v) = \frac{(8n^2 + 16n - 24) + (-2n)}{2n + 2} = \frac{4n^2 + 7n - 12}{n + 1}. \tag{4}$$

Recall that our objective is to compare $d_{max}(A)$ for Funar's A matrix (given an even u) to $d_{max}(M)$ for the corresponding "$n = \frac{u}{2} - 1$" matrix. We showed in another proof that for even u, $d_{max}(A) = \frac{u}{4}$. Using substitution, we see that

$$d_{max}(M) < d_{max}(A) \textbf{ if and only if } v_{max} \cdot Q^{-1}(v_{max}) < n.$$

Evaluating (1) and (2), we see that, no matter whether n is even or odd, $d_{max}(M) < d_{max}(A)$ $\forall n > 1$. We easily compute that, for $n = 1$, $d_{max}(A) = 1$ and $d_{max}(M) = 0.5$. Hence, the inequality holds for all n. $\square$

5 Conclusion

In conclusion, we have not yet determined whether or not HF homology is a complete invariant on torus bundles. Nevertheless, we have found a few infinite families of torus bundles for which HF homology is an invariant, and we have computed the HF homology for a substantial number of torus bundles. Future work would likely involve examining this data set more closely to see if other infinite families are present, and whether the we can prove that HF homology is invariant for them as well.

6 Code

6.1 Check for isomorphic abelianizations - GAP

```
#******************************************
# Author: Rebecca Rohrlich
# Date created: 21 June 2017
#
# Examines the pi-1 group presentations for two
# torus bundles and finds whether these torus
# bundles are isomorphic.
#
# Parameters: The coordinates of the monodromy,
# where the positions are as follows:
# v11 v12
# v21 v22
#******************************************

createFundGroup := function(v11, v12, v21, v22)
    local f, g;
    f := FreeGroup("x","y");
    g := f/ [f.1*f.2*(f.1^-1)*(f.2^-1), f.1^(v11-1)*f.2^v12, f.1^(v21)*f.2^(v22-1)];
    return g;
end;

checkIfFundIso := function(v11, v12, v21, v22, w11, w12, w21, w22)
    local group_1, group_2;
    group_1 := createFundGroup(v11, v12, v21, v22);
    group_2 := createFundGroup(w11, w12, w21, w22);
    Display(group_1);
    Display(Size(group_1));
    Display(group_2);
    Display(Size(group_2));
    return IsomorphismGroups(group_1, group_2);
```

```
end;

createAbGroup := function(v11, v12, v21, v22)
    local f, g;
    f := FreeGroup("x","y");
    g := f/ [f.1*f.2*(f.1^-1)*(f.2^-1), f.1^(v11-1)*f.2^v12, f.1^(v21)*f.2^(v22-1)];
    return g;
end;

checkIfAbIso := function(v11, v12, v21, v22, w11, w12, w21, w22)
    local group_1, group_2;
    group_1 := createAbGroup(v11, v12, v21, v22);
    group_2 := createAbGroup(w11, w12, w21, w22);
    Display(group_1);
    Display(Size(group_1));
    Display(group_2);
    Display(Size(group_2));
    return IsomorphismGroups(group_1, group_2);
end;
```

6.2 Build monodromy from factorization - Python

```
import sys
import numpy as np

'''
***********************************************
author: Rebecca Rohrlich
last updated: 14 June 2017
***********************************************
'''

# set verbose to True to see monodromies and shifts used
# in computations
VERBOSE = True

'''
Start the testing by looking for collisions between
cases 1 and 2.
'''
def testCases_1_2(list_of_a_terms, m):
    mono_1, shift_1 = getMonodromy_Case1(list_of_a_terms)
    mono_2, shift_2 = getMonodromy_Case2(m)

    foundCollision = (shift_1 == shift_2)

    if VERBOSE == True:
        print( "\n", foundCollision, "\n\nCASE 1:" )
```

```
        print(mono_1)
        print("shift:", shift_1)
        print("\nCASE 2:")
        print(mono_2)
        print("shift:", shift_2, "\n")

    return foundCollision

'''
Create matrix for case 1 of Baldwin's paper, with
input specifications.

NOTE: We are assuming that d is odd, because if d is even then
there is no confusion about injectivity. (Uses figure-eight knot
rather than trefoil.)
'''
def getMonodromy_Case1(list_of_a_terms):
    h_d = np.array( [[-1,0], [0,-1]] )
    mono = h_d  # "mono" for monodromy
    sum_a_terms = 0
    for a in list_of_a_terms:
        #print("mono case 1:\n", mono)
        sum_a_terms += a
        matrix = np.array( [[1,a], [1,a+1]] )
        mono = mono.dot(matrix)
    n = len(list_of_a_terms)
    shift = (n + 4 - sum_a_terms)/4
    return mono, shift

'''
Create matrix for case 2 of Baldwin's paper, with
input specifications.
'''
def getMonodromy_Case2(m):
    mono = np.array( [[-1,m], [0,-1]] )
    shift = (m + 4)/4
    return mono, shift

'''
Create matrix for case 2 of Baldwin's paper, with
input specifications.

NOTE: Assuming d is odd to avoid left-hand trefoil.
'''
def getMonodromy_Case3(m):
    mono = np.array([[-1,0], [0,-1]]).dot(np.array([[1,0], [m,1]])).dot(np.array([[1,1], [0,1]
    shift = (m + 3)/4
    return mono, shift
```

```
'''
For now, main method simply reads input from commandline
to test cases 1 and 2.

run with:
python3 surveyBaldwin.py listOfAterms m

example:
python3 ../surveyBaldwin.py '[0, 1]' 0
'''
def main(a_terms_text, m):
    list_of_a_terms = eval(a_terms_text)
    m = eval(m)
    #print( list_of_a_terms )
    testCases_1_2(list_of_a_terms, m)

main(sys.argv[1], sys.argv[2])
```

6.3 Decompose monodromy into factorization - Python

```
import sys

'''
**********************************
Author: Rebecca Rohrlich
Date created: 29 June 2017

Finds the x,y factorization for symmetric
matrices from Baldwin's case 1.

NOTE: skips conjugation and other initial
steps from original SnapPy implementation.
**********************************
'''

def divisionAlg(a,b,c,d):
    factors = '' # factors is a string of x's and Y's (capital Y to denote y^-1)
    while a != 1 or d != 1 or b != 0  or c != 0:
        if a <= c and b <= d:
            c = c-a
            d = d-b
            factors += 'x'
        else: # a >= c and b >= d
            a = a-c
            b = b-d
            factors += 'Y'
    return factors
```

```
'''
Pass the four entries of a matrix,
(a b)
(d c),
on the commandline in the form
python3 factorize.py a b c d

Example:
python3 factorize.py 1 4 5 2
'''
def main(a, b, c, d):
    a = int(a); b = int(b); c = int(c); d = int(d)
    factors = divisionAlg(a,b,c,d)
    print("NOTE: Y denotes y^-1.\n\n"+factors)
```

6.4 Compute d-invariant - Python

```
import sys
import numpy as np
from itertools import product, combinations
from fractions import Fraction

'''
************************************
Author: Rebecca Rohrlich
Date created: 20 June 2017

Computes the d invariants for matrices of
diverse spin^c structures for Baldwin's
cases 1 and 2.
************************************
'''

'''
AString is a string of of space-separated numbers representing the a-exponents in
Baldwin's case 1
'''
def compute_Q_case1(AString):
    # collect A-terms
    AStrList = AString.split()
    allSpokesSets = [int(i) for i in AStrList]
    if sum(allSpokesSets) < 1:
        print("Error: all entries in a-terms list must be >= 0, and at least one term must be
        sys.exit(1)

    # numPolyWalls: number of walls (edges) around exterior of the wheel/polygon
    numPolyWalls = len(allSpokesSets)
```

```
# sum of edges not in spanning tree; i.e., A-term spokes
sumExtraEdges = sum(allSpokesSets) + 1 - 1 # add 1 for extra edge on rim, subtract 1 for sp

# create blank, square array with sumExtraEdges side lengths
Q = np.zeros(shape = (sumExtraEdges, sumExtraEdges))
if sum(allSpokesSets) == 1 and len(allSpokesSets) == 1:
    Q[0][0] = 1
    allSpokesSets = []
    sumExtraEdges = 0

# lengths of each cycle formed by adding an extra edge
indexNonzero = next((i for i, x in enumerate(allSpokesSets) if x), None)
allSpokesSets[indexNonzero] = allSpokesSets[indexNonzero] - 1 # add one spoke to span tree
hasTwoCycle = False
if allSpokesSets[indexNonzero] > 1:
    #print("has two-cycle!!! indexNonzero is: "+str(indexNonzero))
    hasTwoCycle = True
while allSpokesSets[0] < 1:
    allSpokesSets.append(allSpokesSets.pop(0))

# lengths of each cycle formed by adding an extra edge
print("allSpokesSets are:",allSpokesSets)
allCycles = []
rimCycle = numPolyWalls
allCycles.append(rimCycle)
Q[0][0] = -1*rimCycle
count = 1
if hasTwoCycle:
    numTwoCycles = allSpokesSets.pop(0)
    for i in range(numTwoCycles):
        allCycles.append(2)
        Q[count][count] = -2
        count += 1
cycleLength = 3 # cycle length is incremented
for spokeSet in allSpokesSets:
    for i in range(spokeSet):
        Q[count][count] = -1*cycleLength
        count += 1
        allCycles.append(cycleLength)
    cycleLength += 1

# calculate cycle interactions, fill matrix
combos = list(combinations(range(sumExtraEdges), 2))
for combo in combos:
    index_1 = combo[0]
    index_2 = combo[1]
    cycle_1 = allCycles[index_1]
```

```
        cycle_2 = allCycles[index_2]
        if index_1 == 0 :
            entry = -1*(cycle_2 - 2) # subtract the two spokes from non-poly to get overlap
        elif index_2 == 0:
            entry = -1*(cycle_1 - 2)
        else:
            entry = -1*(min([cycle_1, cycle_2])-1)
        Q[index_1][index_2] = entry
        Q[index_2][index_1] = entry

    return Q

'''
Computes the Q matrix for Baldwin's case 2.
n is the exponent for monodromy y.
'''
def compute_Q_case2(n):
    Q = None
    val = -2#+(10**-6) # ought to be just -2; added tiny value to make Q non-singular...?
    if n > 0:
        Q = np.zeros(shape = (n+5, n+5))
        for i in range(n+5):
            for j in range(n+5):
                if i == j:
                    Q[i][j] = val
                elif (i == 2 and (j < 2 or j == 3)) or (j == 2 and (i < 2 or i == 3)):
                    Q[i][j] = 1
                elif (i == n+2 and (j > i  or j == i-1)) or (j == n+2 and (i > j  or i == j-1)
                    Q[i][j] = 1
                elif (i > 2 and i < n+2 and i == j-1) or (j > 2 and j < n+2 and j == i-1):
                    Q[i][j] = 1
                else:
                    Q[i][j] = 0
    elif n < 0:
        Q = np.array([[val,0,1,0,0,0,0],
                      [0,val,1,0,0,0,0],
                      [1,1,-1,1,0,0,0],
                      [0,0,1,n,1,0,0],
                      [0,0,0,1,-1,1,1],
                      [0,0,0,0,1,val,0],
                      [0,0,0,0,1,0,val]])
    else:
        Q = np.array([[val,1,1,1,1],
                      [1,val,0,0,0],
                      [1,0,val,0,0],
                      [1,0,0,val,0],
                      [1,0,0,0,val]])
```

```
    return Q;

'''
Compute the d-invariant for a given charecteristic vector
'''
def compute_d(Q, charVect):
    dim = Q.shape[0]
    Q_inv = np.linalg.inv(Q) # take the inverse of Q
    k = charVect
    k_norm = (np.atleast_2d(k)).dot(Q_inv).dot(np.atleast_2d(k).T)
    d = (k_norm + dim)/4
    return d

'''
Compute the list of characteristic vectors
'''
def getListsToBeMult(Q):
    dim = Q.shape[0] # Q is a square matrix
    listsToBeMult = [] # a list of all the lists we will take the Cart prod of
    for i in range(dim):
        k_i = int(Q[i][i])
        list = []
        step = 2
        end = k_i + 1
        if k_i < 0:
            step = -1*step
            end = k_i - 1
        if k_i != 0: # check if k_i is nonzero
            for l in range(-1*k_i, end, step):
                list.append(l)
        else:
            list.append(0)
        listsToBeMult.append(list)
        #print("next list:",list)
    #charVectors = [np.asarray(p) for p in product(*listsToBeMult)]
    #print(listsToBeMult)
    return listsToBeMult

def cartProdRecurse(Q, listsToBeMult, i, currentTuple, max_d):
    if len(currentTuple) < len(listsToBeMult):
        for j in range(len(listsToBeMult[i])):
            newTuple = currentTuple + [listsToBeMult[i][j]]
            max_d = cartProdRecurse(Q, listsToBeMult, i+1, newTuple, max_d)
    else:
        #print("current tuple:",currentTuple)
        charVect = np.asarray(currentTuple)
        d = compute_d(Q, charVect)
        max_d = max([d, max_d])
```

```
        if max_d == d:
            print("max tuple:",currentTuple)
    return max_d

'''
Finds the maximum d out of all characteristic vectors of Q.
'''
def get_max_d(Q):
    listsToBeMult = getListsToBeMult(Q)
    max_d = cartProdRecurse(Q, listsToBeMult, 0, [], 0) # initialize the recursion
    return max_d

'''
Main method takes two arguments.

Arg 1: the number 1 or 2 (for case 1 or 2)

Arg 2:
Baldwin Case 1) a list of a-exponents separated by spaces, all enclosed in quote marks.
Baldwin Case 2) the n exponent for monodromy y

Examples:
python3 compute_d.py 1 "1 42 5 4 90"
python3 compute_d.py 2 4
'''
def main(arg1, arg2):
    Q = None
    if int(arg1) == 1:
        Q = compute_Q_case1(arg2)
    else:
        Q = compute_Q_case2(int(arg2))
    print("Q is:\n",Q)
    print(np.linalg.det(Q))
    max_d = get_max_d(Q)
    print("max_d is:",max_d)
    #d = compute_d(Q)
    #print("d is:", d)
```

6.5 Solving Funar's equation for a given u - Mathematica

```
(* Solves the equations for Funar's special family of matrices *)
SolveFunar[u_] := Module[{solns},
   solns =
    Solve[4 a^2 + b^2 ==  u^2 + 4 && a > 0 && b > 0, {a, b},
     Integers ];
   Return[solns]];
```

28

```
 FunarMatrixEntries[u_, a_, b_] := Module[{list, e1, e2, e3},
```

```
e1 = .5*(u^2 - b*u + 2);
e2 = a*u;
e3  = .5*(u^2 + b*u + 2);
list = List[e1, e2, e2, e3];
Return[list]
]
```

References

[1] Baldwin, J. A. (2008). Heegaard Floer homology and genus one, one-boundary component open books. Journal of Topology, 1(4), 963-992.

[2] Ozsváth, P., Szabó, Z. (2005). On the Heegaard Floer homology of branched double-covers. Advances in Mathematics, 194(1), 1-33.

[3] Funar, L. (2013). Torus bundles not distinguished by TQFT invariants. Geometry Topology, 17(4), 2289-2344.

[4] Culler, Marc and Dunfield, Nathan M. and Goerner, Matthias and Weeks, Jeffrey R. SnapPy, a computer program for studying the geometry and topology of 3-manifolds. Available at http://snappy.computop.org (22/07/2017)

Optimizing Convex Hull Surface Areas and Volumes

Chris Connell and Anabel Tassoni

July 27, 2017

Abstract

We conjecture that of all closed curves (in three-dimensional space) of length 1, the circle maximizes the surface area of the convex hull. Though we do not solve the general case, we work with specific cases. We also investigate the problem of maximizing convex hull volumes and surface areas of polygonal curves.

1 Maximizing Convex Hull Volume of Length-1 Closed Curves

The convex hull of a set P is the smallest superset of P such that between any $p_1, p_2 \in P$, there exists a geodesic (in the 2-dimensional and 3-dimensional cases, this amounts to a straight line) between p_1 and p_2 that is completely contained in P. In 2-dimensional space, the circle is the closed curve of maximal convex hull area (this is known as the Isoperimetric Problem in two dimensions)–the intuitive extension of the problem to three dimensions, then, is the conjecture that the circle with doubled-disk area maximizes the convex hull surface area of a three-space curve (the area is doubled to allow for the fact that if any perturbation occurred, the circle would split and we would again have a three-dimensional problem). In this section, we investigate several approaches (some more fruitful than others) to the three-dimensional isoperimetric problem. When it is convenient, we normalize curve length to be one in order to avoid tedium with scaling.

The planar isoperimetric inequality tells us that

$$L^2 \geq 4\pi A \tag{1}$$

where L and A stand for length and area, respectively. Translating this inequality into the three-dimensional case, then, we see that our conjecture becomes the claim that for a three-dimensional curve α with length L,

$$L^2 \geq 4\pi Area(CH(\alpha)) \tag{2}$$

(where CH(α) stands for "convex hull of α") , and that there is equality exactly when α is a round circle with a doubled-disk's area. For convenience, we rewrite this inequality conjecture as

$$Area(2CH(\alpha)) \leq \frac{L^2}{2\pi} \tag{3}$$

noting of course that twice the area of a convex hull is equivalent to the area of two duplicate convex hulls.

1.1 Polygonal Approximation

In order to work in a more discrete context, it is useful to think first in terms of polygonal approximation (this is because any closed curve can be approximated more and more accurately by polygonal curves of smaller and smaller edge length; the supremum over all such approximations yields the original curve). Our first strategy was to think explicitly about the tetrahedra formed by the convex hull of a closed curve in three-dimensional space. It might seem reasonable to claim that the convex hull of a polygonal curve (piecewise linear curve) takes the form of a simplicial complex. This would mean that the tetrahedra formed by the convex hull do not have any three-dimensional overlap. If such a claim were true, it would lead to a second, possibly very helpful, conjecture, which is that the convex hull of any three-dimensional polygonal curve is exactly the union of all tetrahedra formed by edge pairs in the curve.

This second conjecture, unfortunately, is definitely not true. One counterexample is the following figure:

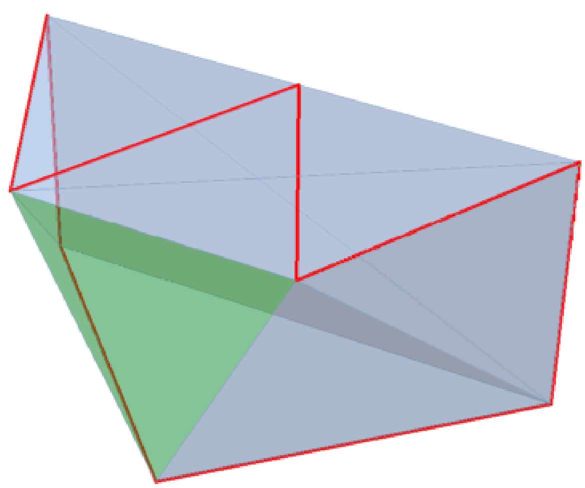


Figure 1: Counterexample

1.2 Working with Symmetrization

Another approach to the 3-D isoperimetric problem for polygonal curves is to attempt to come up with an analog to Steiner Symmetrization in order to arrive at the optimal convex hull surface area for a curve of given length. Ideally, this procedure would begin with some curve α, keep the length of α constant, and non-decrease the area of the boundary of the convex hull. Because the actual fixed curve length is unimportant except for the issue of scaling, we can assume that α is of length 1 without losing vital information; that is, we can assume that we have

$$\alpha : [0,1] \to \mathbb{R}^3 \tag{4}$$

Again, we begin by thinking about the polygonal case. For a piecewise linear α, we consider local deformations that preserve total length. That is, we can imagine taking a vertex (call it v) of α and moving it slightly. Only the edge lengths of the two edges incident to v will be affected, but we want our procedure to keep length constant; thus we want the sum of the lengths of these two edges not to change. We also want to non-decrease area, so we want

$$\sum_i F_i$$

(where each F_i is a triangular boundary face formed by an edge and a vertex) never to decrease.

We can see, though, that because we are working with optimizing the isoperimetric constant, we actually do not need to force the length to remain constant and the area to non-decrease–we only need for

$$\frac{A}{L^2} \tag{5}$$

(where A represents twice the area of the convex hull boundary and L represents the length of α)to non-decrease.

Our hypothesis is that the circle maximizes the convex hull; in the circular case (when radius $r = \frac{L}{2\pi}$), $A = \frac{L^2}{2\pi}$, which means that the ratio $\frac{A}{L^2}$ is equal to $\frac{1}{2\pi}$.

This, in turn, means that our conjecture is that

$$2\pi A \leq L^2 \tag{6}$$

with equality when and only when the curve in question is a round circle with doubled area.

Our conjecture leads us to the following question: consider a closed curve c and its convex hull boundary S. Is it always possible to find some plane P through S such that when we project S and c onto P (call these projections $\hat{S}$ and $\hat{c}$),

$$4\pi \frac{Area(\hat{S})}{L(\hat{c})^2} \geq 2\pi \frac{Area(S)}{L(c)^2}? \tag{7}$$

(We have the doubled scalar factor of π on the left in order to account for the same doubled-disk issue discussed above.)

At first glance, the problem might seem trivial; perhaps, for example, we could pick any plane as our P. But a fairly simple counterexample shows us that we cannot simply choose a plane at random.

Counterexample: Consider a round circle of diameter 1, and let P be the plane that cuts through the diameter.

In this case, $\hat{c}$ and— $\hat{S}$ are both simply a doubled line segment of length 1, which has area 0 and total length 2. This gives us

$$\frac{Area(\hat{S})}{L(\hat{c})^2} = 0$$

.

But, of course, the length and area of the original circle are positive, so the inequality breaks. Thus we must be a bit more discriminating in how we choose our plane P.

1.3 Projection with a Box

For a general curve, and even for an arbitrary polygonal curve, this problem seems daunting. However, we can gain some interesting insight from looking at the case of a curve c that traverses the edges of a box (shown below in Figure 2).

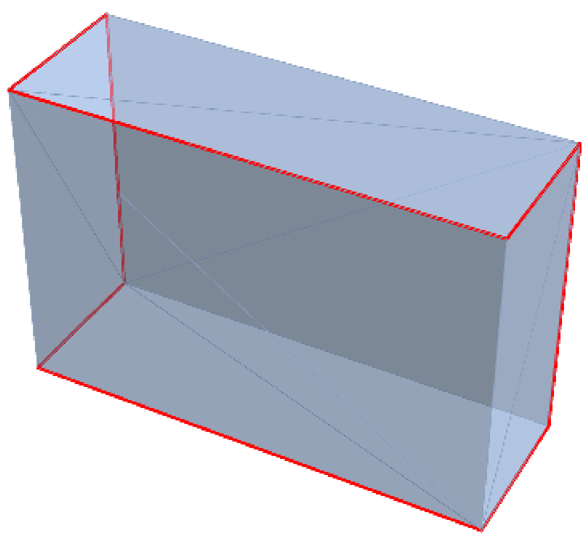


Figure 2: Box

Note that the box along which c travels has length l, width w, and height h, and also that the the convex hull of c is the entire box.

First we can try projecting the curve into the plane that runs through the bottom of the box.

Given our l, w, and h, we can set to work finding known quantities: In this case the area of S is simply the surface area of the box, which is $2hl+2wl+2hw$. $L(c)$ is $2h+4w+2l$. The computations for the projected quantities $\hat{S}$ and $\hat{c}$ are not much more difficult: projection collapses w, and so our $Area(\hat{S})$ is simply wl and our $L(\hat{c})$ is $2(2w+l)$ (the doubling happens because we must account for the fact that the collapsed projection curve $\hat{c}$ actually retraces its course exactly once).

This gives us the following quantities:

$$4\pi\frac{Area(\hat{S})}{L(\hat{c})^2} = 4\pi\frac{(wl)}{4(2w+l)^2} = \frac{\pi wl}{(2w+l)^2}$$

(we can call the above quantity $\mathrm{Iso}(\hat{c})$)
and

$$2\pi\frac{Area(S)}{L(c)^2} = 2\pi\frac{2hl+2wl+2hw}{(2h+4w+2l)^2} = \frac{\pi(hl+wl+hw)}{(h+2w+l)^2}$$

(we can call the above quantity $\mathrm{Iso}(c)$).
We want to know when $\mathrm{Iso}(\hat{c}) \geq \mathrm{Iso}(c)$–that is, we need to see when

$$\frac{\pi wl}{(2w+l)^2} \geq \frac{\pi(hl+wl+hw)}{(h+2w+l)^2}$$

.

We can rearrange numerators and denominators in order to see that this expression is equivalent to

$$\frac{wl}{wl+h(l+w)} \geq \frac{(2w+l)^2}{(h+2w+l)^2}$$

With a bit more simplification, we can see that the inequality does not always hold for all possible parameter values (for example, when h is very small). So this plane, though it makes for simple computations, is not ideal for what we want to solve.

Our goal is to find a plane through (it will turn out to be a skew plane) that forces our inequality to hold in the *general case* of parameters w, h and l.

First, it might be useful to set the box in coordinate 3-space; we can say that it has corners at (0,0,0) and (w, l, h)

Our goal is to find the unit vector that is normal to the plane that we want. We can name this vector $N = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$, where

$||N|| = ||N||^2 = n_x^2 + n_y^2 + n_z^2 = 1$.

Now we can try using a Lagrange multiplier with the constraint that

$n_x^2 + n_y^2 + n_z^2 = 1$. A bit of arithmetic is needed in order to reach a formula for $\hat{S}$ and $\hat{c}$ in the general case; we can use the formula for projection to conclude that

$$Iso(\hat{c}) = \frac{\pi(hwn_x + hln_y + wln_z)}{(l\sqrt{1-n_x^2} + 2w\sqrt{1-n_y^2} + h\sqrt{1-n_z^2})^2}$$

.

If we maximize this quantity (in terms of the fixed, unknown parameters) subject to the given constraint, we can see if the resulting unit vector forces Iso($\hat{c}$) to be larger than or equal to Iso(c), which is the same as in the parallel projection case.

A Lagrange multiplier gives us the equation

$$\nabla Iso(\hat{c}) = \lambda\nabla(n_x^2 + n_y^2 + n_z^2)$$

$$= 2\lambda \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

This gives us a seemingly manageable system of four equations and four variables, but solving it turns out to be extremely difficult (we did not have much luck).

Fortunately, the programming language Mathematica can give us some good computational insight about how these equations operate, though we still cannot solve the problem completely for the general case.

To start, we have the following variables for the isoperimetric quantities that we are working with:

```
Iso = Pi (h w + h l + w l)/(l + 2 w + h)^2

IsoP = Pi (h w nx + h l ny +
    w l nz)/(l Sqrt[1 - nx^2] + 2 w Sqrt[1 - ny^2] +
     h Sqrt[1 - nz^2])^2;
```

(IsoP stands in for Iso($\hat{c}$)). Although the general-case system of equations is difficult to solve, we can fairly easily find the maximizer for a cube of side length one by parametrizing the unit vector with parameters s and t:

```
  IsoP2 = FullSimplify[
IsoP /. {nx -> Cos[t] Cos[s], ny -> Cos[t] Sin[s], nz -> Sin[t]},
Assumptions -> {0 < s < Pi/2, 0 < t < Pi/2}]


  NMaximize[IsoP2 /. {h -> 1, l -> 1, w -> 1}, {s, t}]

  {0.785398, {s -> 1.5708, t -> 1.61807*10^-8}}
```

Thus in the cubic case, our optimal isoperimetric constant for the projected curve is .785398, and it occurs when $s = 1.5708$ and t is zero. Now we find the corresponding normal vector:

```
 {Cos[t] Cos[s], Cos[t] Sin[s], Sin[t]} /. {s -> 1.5707963086718582`,
t -> 1.6180734060878987`*^-8}

{1.8123*10^-8, 1., 1.61807*10^-8}
```

This optimizing plane, then, is not skew (because $\begin{bmatrix} 1.8123 * 10^{-}8 \\ 1 \\ 1.61807 * 10^{-}8 \end{bmatrix}$ is simply $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ with a Mathematica rounding error). Furthermore, we can see that the initial isoperimetric constant for the curve is

```
N[Iso /. {h -> 1, l -> 1, w -> 1}]

0.589049
```

Thus we can see that the plane that we found does in fact yield a higher isoperimetric constant (.785398) than the initial isoperimetric constant for the curve (.589049). So the problem of always being able to find a plane that improves isoperimetric constant upon projection is solved for the cube.

The problem for a thin box with height 1, length 2, and width .01 yields a skew plane as the optimizer. We have

```
opt = NMaximize[IsoP2 /. {h -> 1, l -> 2, w -> .01}, {s, t}]

{0.778713, {s -> 0.527022, t -> 0.00667402}}
```

which gives us a projected curve isoperimetric constant of .778713 occurring when $s = 0.527022$ and $t = 0.00667402$. We can also see from the following code that this constant does in fact beat the initial isoperimetric constant of 0.699249:

```
Iso /. {h -> 1, l -> 2, w -> .01}


0.699249
```

When we use these values to find the corresponding vector, we get the following:

```
{Cos[t] Cos[s], Cos[t] Sin[s], Sin[t]} /. opt[[2]]

{0.86429, 0.50295, 0.00667397}
```

This normal vector, $\begin{bmatrix} .86429 \\ .50295 \\ .00667397 \end{bmatrix}$, shows that our optimizing plane is skew.

Efforts to solve the case of unknown parameters were generally unsuccessful. One tactic, for example, was to take the gradient of the parametrized isoperimetric function of the projected box, to normalize width (w) to 1, and to ask Mathematica to find the maximum. As can be seen below, Mathematica was unable to give a satisfying answer.

```
         gIsoP2 = Simplify[D[IsoP2, {{s, t}}],
  Assumptions -> {0 < s < Pi/2, 0 < t < Pi/2}]

  Solve[gIsoP2 == {0, 0}, {s, t}]
  $Aborted

  newIsoP =
 Simplify[gIsoP2 /. {Sqrt[1 - Cos[s]^2 Cos[t]^2] -> a,
    Sqrt[1 - Cos[t]^2 Sin[s]^2] -> b, Cos[t] -> Sqrt[2 - (a^2 + b^2)],
     Sin[t] -> Sqrt[a^2 + b^2 - 1],
    Cos[s] -> Sqrt[1 - a^2]/Sqrt[2 - (a^2 + b^2)],
    Sin[s] -> Sqrt[1 - b^2]/Sqrt[2 - (a^2 + b^2)]},
  Assumptions -> {a > 0, b > 0}]

  newIsoP =
 FullSimplify[{h (Sqrt[2 - a^2 - b^2] h + a l +
       2 b w) (Sqrt[1 - a^2] l - Sqrt[1 - b^2] w) + (
     2 Sqrt[1 - a^2] Sqrt[
      1 - b^2] (-b l + 2 a w) (Sqrt[1 - b^2] h l + Sqrt[1 - a^2] h w +
         Sqrt[-1 + a^2 + b^2] l w))/(a b),
    2 Sqrt[-1 + a^2 +
       b^2] (a b Sqrt[2 - a^2 - b^2] h + (-1 + a^2) b l +
        2 a (-1 + b^2) w) (Sqrt[1 - b^2] h l + Sqrt[1 - a^2] h w +
        Sqrt[-1 + a^2 + b^2] l w) +
     a b (Sqrt[2 - a^2 - b^2] h + a l +
        2 b w) (-(-2 + a^2 + b^2) l w -
        Sqrt[-1 + a^2 + b^2]
          h (Sqrt[1 - b^2] l + Sqrt[1 - a^2] w))} /. {w -> 1}, {a > 0,
    b > 0}]

    Solve[newIsoP == {0, 0}, {a, b}]
    $Aborted
```

Although the problem with general parameters is still a mystery, the insight on specific instances was helpful in allowing us to better understand the box case of projection.

1.4 Local Deformations About a Vertex

Another way to gain insight into the polygonal problem is to look at vertex deformations in a vector sense. We first consider some polygonal curve c_0 with vertices v_0 through v_k, some deformation vector $\mathbf{w}$, and some very small $\epsilon > 0$ (a picture is shown below).

Let's say that we are going to deform v_1 by ϵ in the direction of $\mathbf{w}$–we can call the new curve c_ϵ. First we note that the deformation that we are going to make only changes the lengths of the edges that are incident to v (for ease, let's call them v_0 and v_2).

Given our setup, then, we can say the following:

$$l(c_0) = \sum_{i=1}^{k} ||\mathbf{v_i} - \mathbf{v_{i-1}}||$$

and

$$l(c_\epsilon) = l(c_o) - ||\mathbf{v_1} - \mathbf{v_0}|| - ||\mathbf{v_1} - \mathbf{v_2}|| + ||\mathbf{v_1} + \epsilon\mathbf{w} - \mathbf{v_0}|| + ||\mathbf{v_1} + \epsilon\mathbf{w} - \mathbf{v_2}||$$

.

For convenience, we can let $a = ||\mathbf{v_1} - \mathbf{v_0}|| + ||\mathbf{v_1} - \mathbf{v_2}||$ and let $b(\epsilon) = ||\mathbf{v_1} + \epsilon\mathbf{w} - \mathbf{v_0}|| + ||\mathbf{v_1} + \epsilon\mathbf{w} - \mathbf{v_2}||$, which allows us to write the previous equation as

$$l(c_\epsilon) = l(c_0) - a + b(\epsilon)$$

.

Thinking now about convex hulls, we can say that

$$Area(\delta(CH(c_0))) = \sum_{i=1}^{k} Area(CH(edge \cup point))$$

and that

$$Area(\delta(CH(c_\epsilon))) = Area(\delta(CH(c_0))) + \sum -Area(T_i = \{v_{i-1}, v_1, v_i\}) + \sum Area(\{v_{i-1}, v_i, v_1 + \epsilon\mathbf{w}\})$$

which, again for convenience, we can rewrite as

$$Area(\delta(CH(c_\epsilon))) = Area(\delta CH(c_0)) - A + B(\epsilon)$$

We can now write the isoperimetric constant for c_ϵ as follows:

$$Iso(c_\epsilon) = \frac{2\pi Area(\delta CH(c_\epsilon))}{l(c_\epsilon)^2}$$

$$= \frac{2\pi(Area(\delta CH(c_0)) - A + B(\epsilon))}{(l(c_0) - a + b(\epsilon))^2}$$

Now we can take the derivative with respect to ϵ and evaluate that quantity at $\epsilon = 0$ to see whether an arbitrarily small change is an improvement to the isoperimetric constant:

$$\frac{\delta}{\delta\epsilon}\bigg|_{\epsilon=0} (Iso(c_\epsilon)) = \frac{2\pi B'(0)}{(l(c) - a - b(0))^2} - 2 * \frac{2\pi(Area(\delta CH(c_0)) - A + B(0)) * b'(0)}{(l(c_0) - a + b(0))^3}$$

$$= \frac{2\pi B'(0)}{l(c_0)^2} - \frac{4\pi Area(\delta CH(c_0)) * b'(0)}{l(c_0)^3}$$

.

Now, recalling that

$$B'(0) = \frac{\delta}{\delta\epsilon}\bigg|_{\epsilon=0} \sum Area(\{\mathbf{v_{i-1}}, \mathbf{v_i}, \mathbf{v_1} + \epsilon\mathbf{w}\})$$

and that

$$b'(0) = \frac{\delta}{\delta\epsilon}\bigg|_{\epsilon=0} (||\mathbf{v_1} + \epsilon\mathbf{w} - \mathbf{v_0}|| + ||\mathbf{v_1} + \epsilon\mathbf{w} - \mathbf{v_2}||)$$

,

we can rewrite our previous equation as

$$\frac{\delta}{\delta\epsilon}\bigg|_{\epsilon=0} (Iso(c_\epsilon)) = 2\pi * \frac{B'(0) * l(c) - 2 * Area(\delta CH(c_0) * b'(0)}{l(c)^3}$$

$$= 2\pi * (\frac{B'(0)}{l(c)^2} - \frac{2Iso(c_0) * b'(0)}{l(c)})$$

So now we can try to understand the conditions under which the isoperimetric constant improves (that is, under which $\frac{\delta}{\delta\epsilon}\big|_{\epsilon=0} (Iso(c_\epsilon)) \geq 0$).

A positive derivative at zero would require that $\frac{B'(0)}{l(c)^2} \geq \frac{2Iso(c_0)*b'(0)}{l(c)}$. Of course, because $l(c)$ is positive, this is the same thing as requiring that

$$B'(0) \geq 2 * Iso(c_0) * l(c) * b'(0)$$

(note that this follows no matter the signs of $B'(0)$ and $b'(0)$).

We might now conjecture that moving $\mathbf{v_1}$ toward the line segment between $\mathbf{v_0}$ and $\mathbf{v_2}$ (inward, essentially) might always work to improve the isoperimetric constant. But not much progress was made past this point, and so this particular angle on the problem turned out not to be particularly fruitful.

1.5 A Return to Projections

We now examine another attempt to work with the isoperimetric problem through the lens of projection.

First we consider N, the unit normal to plane P (the plane that preserves the greatest area upon projection). We also remember our curve c and its projection $\hat{c}$.

So we have that

$$Area(\delta CH(\hat{c})) = \sum |\langle N_T, N\rangle| * Area(T)$$

(a sum over all boundary triangles T, with N_T denoting the unit normal to boundary triangle T).

Remembering that N and N_T are unit vectors and thus have length one, we can rewrite our previous equation as

$$Area(\delta CH(\hat{c})) = \sum |cos(\sphericalangle N_T, N)| * Area(T)$$

,

where $cos(\sphericalangle N_T, N)$ denotes the cosine of the smallest angle between vectors N_T and N.

We can also say that

$$l(\hat{c}) = \sum ||\mathbf{e} - \langle \mathbf{e}, \mathbf{N}\rangle \mathbf{N}||$$

over all curve edges $\mathbf{e}$.Again, because N is a unit vector, we can rewrite this as

$$l(\hat{c}) = \sum_{edges} l(\mathbf{e})|sin(\sphericalangle \mathbf{e}, \mathbf{N})|$$

,

where $sin(\sphericalangle \mathbf{e}, \mathbf{N})$ represents the smallest angle between edge $\mathbf{e}$ and unit vector $\mathbf{N}$.

We claim that for all polygonal curves except for the triangle, there are at most twice as many edges as there are boundary triangles.

Proof : Each edge must necessarily belong to two triangles, and each triangle can have no more than two edges, so there must be no more than twice as many edges as triangles.

□

Of course, the triangle has three edges and is itself only one triangle, so we must exclude it.

So from before, we have

$$Iso(\hat{c}) = \frac{2\pi Area(\delta CH(\hat{c}))}{l(\hat{c})^2}$$

,

which we can now translate into

$$Iso(\hat{c}) = 2\pi \frac{\sum_{triangles} Area(T) * |cos(\sphericalangle \mathbf{N_T}, \mathbf{N})|}{(\sum_{edges} l(\mathbf{e}|sin(\sphericalangle \mathbf{e}, \mathbf{N})|))^2}$$

Again, maximization would be helpful but is difficult to do. One first step might be to try to maximize the total area in the numerator by choosing a preliminary $\mathbf{N}$ that the weighted average of the cosines of the angles between the $\mathbf{N_T}$ vectors and $\mathbf{N}$ is maximized.

One potential strategy is again to use Lagrange multipliers, with which we can set up the equations

$$\nabla_N(Iso(\hat{c})) = \lambda \nabla g$$

where

$$g = n_x^2 + n_y^2 + n_z^2 - 1$$

(with n_x, n_y, and n_z being the components of the vector N).

Our next step might be to get rid of the inconvenient absolute values in our $Iso(\hat{c})$ equation in order to make differentiation easier. We can do this by separating the boundary triangles into "top convex hull boundary triangles" (the unit normals of these triangles dotted with $\mathbf{N}$ will be positive), and "bottom convex hull boundary triangles" (the unit normals of these triangles dotted with $\mathbf{N}$ will be negative, but the actual areas are positive–we can fix this problem by dotting these triangle unit normals with $-\mathbf{N}$ instead of $\mathbf{N}$). We can also take care of the problem of absolute values in the denominator by converting $\mathbf{e}$ to a unit vector $\hat{\mathbf{e}}$ with the equation

$$\hat{\mathbf{e}} = \frac{\mathbf{e}}{l(\mathbf{e})}$$

.

So we can now rewrite our equation as

$$Iso(\hat{c}) = 2\pi \frac{\sum_{TopT} Area(T) * \langle \mathbf{N_T}, \mathbf{N} \rangle + \sum_{BottomT} Area(T) * \langle \mathbf{N_T}, -\mathbf{N} \rangle}{\sum_{edges} l(\mathbf{e}) \sqrt{1 - \langle \hat{\mathbf{e}}, \mathbf{N} \rangle}^2}$$

,

which is free of absolute values.

Now we do some derivative computations: note that in the following equations, "Denom" refers to the denominator of the previous equation and "Num" refers to the numerator of the previous equation.

$$\frac{\delta}{\delta n_x}(Iso(\hat{c})) = \frac{\frac{d}{dn_x}(Num)}{Denom} - Num * \frac{\frac{d}{dn_x}(Denom)}{(Denom)^2}$$

and

$$\frac{\delta}{\delta n_y}(Iso(\hat{c})) = \frac{\frac{d}{dn_y}(Num)}{Denom} - Num * \frac{\frac{d}{dn_y}(Denom)}{(Denom)^2}$$

and

$$\frac{\delta}{\delta n_z}(Iso(\hat{c})) = \frac{\frac{d}{dn_z}(Num)}{Denom} - Num * \frac{\frac{d}{dn_z}(Denom)}{(Denom)^2}$$

We can note here that

$$\frac{d}{dn_x}\langle \mathbf{N_T}, \mathbf{N}\rangle = N_{T,x}$$

We can make further simplifications by noting that

$$\frac{d}{dn_x}(Num) = \sum_{Top} Area(T)N_{T,x} - \sum_{Bottom} Area(T)N_{T,x}$$

and that

$$\frac{d}{dn_x}(Denom) = \sum -l(\mathbf{e})\frac{\langle \hat{\mathbf{e}}, \mathbf{N}\rangle N_{T,x}}{\sqrt{1-\langle \hat{\mathbf{e}}, \mathbf{N}\rangle}^2}$$

(the numerator and denominator derivatives are similar with respect to n_y and n_z).

With these computations done, we can rewrite our gradient equation as

$$\nabla Iso(\hat{c}) =$$

$$\frac{1}{(Denom)^2} * \begin{bmatrix} Denom * (\sum_{Top} Area(T)N_{T,x}) - (\sum_{Bottom} Area(T)N_{T,x}) + Num * (\sum_{AllT} l(\mathbf{e})\frac{\langle \hat{\mathbf{e}}, \mathbf{N}\rangle N_{T,x}}{\sqrt{1-\langle \hat{\mathbf{e}}, \mathbf{N}\rangle}^2}) \\ Denom * (\sum_{Top} Area(T)N_{T,y}) - (\sum_{Bottom} Area(T)N_{T,y}) + Num * (\sum_{AllT} l(\mathbf{e})\frac{\langle \hat{\mathbf{e}}, \mathbf{N}\rangle N_{T,y}}{\sqrt{1-\langle \hat{\mathbf{e}}, \mathbf{N}\rangle}^2}) \\ Denom * (\sum_{Top} Area(T)N_{T,z}) - (\sum_{Bottom} Area(T)N_{T,z}) + Num * (\sum_{AllT} l(\mathbf{e})\frac{\langle \hat{\mathbf{e}}, \mathbf{N}\rangle N_{T,z}}{\sqrt{1-\langle \hat{\mathbf{e}}, \mathbf{N}\rangle}^2}) \end{bmatrix}$$

$$= \lambda \nabla g$$

$$= 2\lambda \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

Solving this system of equations, however, proved just as difficult as solving the system discussed earlier in this paper, and not much progress was made past this point.

1.6 Working With Convex Hull Volumes

While we continue to investigate convex hulls of polygonal curves, we now turn our attention to their volumes rather than their surface areas. It will be useful for us to remember that the convex hull of a polygonal curve is a union of tetrahedra that have vertices in the curve.

1.6.1 The Case of a Curve With Four Edges

First, we consider a polygonal curve with four edges. The convex hull of such a curve is a single tetrahedron –four of this tetrahedron's edges are in the curve and two of them are not. We can call the vertices of the tetrahedron v_1, v_2, v_3, and v_4. Figure 5 below illustrates the situation:

We will refer to d_{ij} as the line segment connecting v_i and v_j.

We first examine the case in which all four edges of the curve have the same length , which we will assume to be one (meaning that $d_{12} = d_{23} = d_{14} = d_{34} = 1$). Our goal is to find the lengths of the other two edges of the tetrahedron that maximize its volume (so we are solving for d_{13} and d_{24}); recall that this is the same as finding the edge lengths that maximize the volume of the curve's convex hull.

The Mathematica formula for the volume of a tetrahedron given the edge lengths is shown below:

```
TVol[d12_, d13_, d14_, d23_, d24_, d34_] :=
  Sqrt[1/288  Det[{{0, d12^2, d13^2, d14^2, 1 }, { d12^2, 0, d23^2,
       d24^2, 1 }, { d13^2, d23^2, 0, d34^2, 1 }, { d14^2, d24^2,
       d34^2, 0, 1 }, { 1, 1, 1, 1, 0}}]];
```

We can substitute our values for the four sides of the curve to get the following:

```
TVol[1, d13, 1, 1, d24, 1]

Sqrt[8 d13^2 d24^2 - 2 d13^4 d24^2 - 2 d13^2 d24^4]/(12 Sqrt[2])
```

Now that we have our volume of $\frac{\sqrt{8d_{13}^2d_{24}^2-2d_{13}^4d_{24}^2-2d_{13}^2d_{24}^4}}{12\sqrt{2}}$, we can differentiate with respect to both d_{13} and d_{24} and set both of those equations equal to zero in order to get a critical point (in this case, the critical point will be the maximizer).

Differentiating and setting our equations equal to zero gives us the following equations:

$$4d_{13}d_{24}^2 - d_{13}d_{24}^4 - 2d_{13}^3d_{24}^2 = 0$$

$$4d_{13}^2d_{24} - d_{13}^4d_{24} - 2d_{13}^2d_{24}^3 = 0$$

Wolfram gives us the following solutions:

```
d13 = 0

d24 = 0

d13 = -2/sqrt(3), d24 = -2/sqrt(3)

d13 = -2/sqrt(3), d24 = 2/sqrt(3)

d13 = 2/sqrt(3), d24 = -2/sqrt(3)

d13 = 2/sqrt(3), d24 = 2/sqrt(3)
```

Of course, in this context we cannot have zero or negative solutions, so our solution is

$$d_{13} = d_{24} = \frac{2}{\sqrt{3}}$$

If we substitute these values back into the original equation, we get the following:

```
TVol[1, 2/3^(1/2), 1, 1, 2/3^(1/2), 1]

2/(9 Sqrt[3])
```

So given a polygonal curve with four edges all of length one, we have a maximum tetrahedral volume of $\frac{2}{9\sqrt{3}}$ that occurs when the other two edges are of length $\frac{2}{\sqrt{3}}$.

In some ways, this might seem like a counterintuitive result–perhaps we would guess that given four edges of length one, the lengths for the other two edges that maximize tetrahedral volume might also be one rather than a radical term.

1.6.2 The Case of a Curve With 5 Edges

We can now turn our attention to the case of a polygonal curve with five edges. We first note that the convex hull of a five-edge curve consists of two tetrahedra that share an edge. We can set the total length of the curve to be L. Our goal is to find the lengths of the tetrahedral edges in the curve that maximize total volume. We can do this by maximizing the volumes of each tetrahedron separately.

The Mathematica code used to solve this problem is shown below:

```
obj = d12 d23 d34/6 + d34 d45 d15/6
cond = d12 + d23 + d34 + d45 + d15 - L

eq = Join[Grad[obj - t cond, {d12, d23, d34, d45, d15}], {cond}]

sol5 = Solve[eq == {0, 0, 0, 0, 0, 0}]

{{d34 -> 0, d45 -> -((d12 d23)/d15),
  L -> (d12 d15 + d15^2 - d12 d23 + d15 d23)/d15, t -> 0}, {d12 -> 0,
  d15 -> 0, d34 -> 0, L -> d23 + d45, t -> 0}, {d15 -> 0, d23 -> 0,
  d34 -> 0, L -> d12 + d45, t -> 0}, {d12 -> 0, d15 -> 0, d23 -> 0,
  d45 -> 0, L -> d34, t -> 0}, {d12 -> d34/2, d15 -> d34/2,
  d23 -> d34/2, d45 -> d34/2, L -> 3 d34, t -> d34^2/12}}

  obj /. sol5

  {0, 0, 0, 0, d34^3/12}

  sol5[[5]]

  {d12 -> d34/2, d15 -> d34/2, d23 -> d34/2, d45 -> d34/2, L -> 3 d34,
 t -> d34^2/12}
```

Thus our final conclusion is that the maximum volume occurs when the shared edge is length $\frac{L}{3}$ and all other curve edges are $\frac{L}{6}$–this yields a volume of $\frac{L^3}{324}$.

1.6.3 The Case of a Curve With 6 Edges

The six-edge case turns out to be significantly more difficult than the two cases discussed above, and less progress was made on it than was made on the previous two. However, it was still possible to gain some insight under the assumption that all six edges of the curve were of length one. In this case, we found the following figure to be the maximizer:

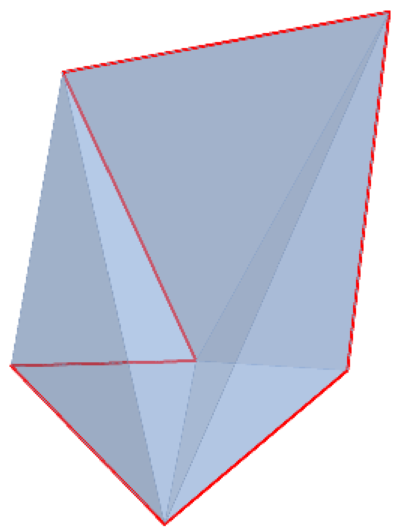


Figure 3: Maximizer in the case of all six edges being length one

It should be noted that this problem becomes progressively more computationally complex with each additional edge (for example, the number of critical points starts to increase dramatically–there were already 59 critical points in the six-edge case). For this reason, the approach used for the cases of four, five, and six edges discussed above would not be a reasonable technique for solving an arbitrary n-edge case.

2 Conclusion

While many aspects of the initial conjecture remain a mystery, we feel that the work done in specific polygonal cases gives interesting insight into the problem. We hope that more work is done in the future toward solving the general conjecture.

Many thanks to Indiana University Bloomington and to the National Science Foundation for supporting this project, and to Purdue University for providing us with a forum to present our results.

References

[Conn 1] R. Connelly, E. D. Demaine and G. Rote. Straightening polygonal arcs and convexifying polygonal cycles. Discrete Comput. Geom. 30(2):205?239, 2003.

[Gho 1] M. Ghomi, Open Problem in Geometry of Curves and Surfaces. Georgia Tech University, 1993

[Hur 1] A. Hurwitz, Sur le problme des isoprimtres, Comptes Rendus, 132 (1901), 401-403; also in Math. Werke I, 490-491.

[Krabb 1] N. Krabbenhoeft. The isoperimetric problem for the convex hull of a space curve. Indiana University Bloomington REU Student Reports, 2006.

[Schn 1] R. Schneider. Convex bodies: the Brunn-Minkowski theory. Cambridge University Press, Cambridge, 1993.

[Scho 1] I.J Schoenberg :An isoperimetric inequality for closed curves convex in even-dimensional euclidean spaces Acta Math., Institut Mittag-Leffler, 1954, 91, 143-164

[Weil 1] A. Weil. Sur les surfaces a courbure negative. C. R. Acad. Sci. Paris, 182:10691071, 1926.

[Zalg 1] . V. A. Zalgaller. Extremal problems on the convex hull of a space curve. Algebra i Analiz, 8(3):1–13, 1996.

VERTICALLY AVERAGED VELOCITY IN RAYLEIGH-BÉNARD CONVECTION: A COMPUTATIONAL STUDY

ANTHONY ISENBERG

ABSTRACT. We present a numerical analysis of the vertically averaged velocity in 3D Rayleigh-Bénard convection, which results in a 2D Navier-Stokes system with a body force depending on the 3D flow. Using pseudo-spectral methods found in a Python suite, Dedalus, we compute the 3D flow, from which we extract the vertically averaged velocity, the corresponding force, as well as time-averaged physical quantities. We then compare them to theoretical bounds to see if the features of 2D turbulence arise.

1. INTRODUCTION

There are fundamental differences between 2D and 3D turbulence. Many simulations have been run using an arbitrarily chosen body force to test characteristics of turbulent flow. We instead use a force naturally imposed by the 3D Rayleigh-Bénard problem on the vertical average of the velocity. In experiments concerning the Bénard problem, one observes vertical hot and cold plumes (for instance, see [5]) which have an important role in the Bénard convection. This suggests that the vertical averages may be helpful in understanding the problem. Our treatment is mathematical, but to have an eye on the physical phenomenon, we avoid nondimensionalization. The question, then, is whether this averaged velocity exhibits the characteristics of 2D turbulence.

Toward an answer to our question, we extract from a numerically computed solution to the 3D Rayleigh-Bénard problem both the vertically averaged velocity as well as the corresponding body force and quantities relevant to 2D turbulence. From these simulations, we find that the averaged velocity does indeed display some features of 2D turbulence. We concluded this by computing the energy spectrum, transfer of enstrophy, and bounding expressions for the dissipation length scale of this 2D velocity for a range of Rayleigh numbers. We mention here that in [11] there is a discussion of 2D cascade in 3D atmospheric flows, mostly inverse energy cascade. Our focus and the methods are different.

We additionally examine various theorems from [2]. We see that the bounds in (3.28) and (5.1) hold and find the lower bounds in these inequalities to be tighter. We also find the expected constant value of the pseudo-flux found in [8] holds. However, energy spectra associated with 2D flow are not consistent with expected results for large Rayleigh numbers. Figures 1 and 2 provide heuristic images of what we expected for the spectrum.

Date: July 28, 2017.

2010 *Mathematics Subject Classification.* 35Q30, 76F02, 76F25.

Key words and phrases. Navier-Stokes equations, turbulence, enstrophy cascade.

This work is supported by NSF REU Grant # 1461061.

We refer the reader to ([1], [8]) for background on 2D turbulence theory from the same angle as in this work.

2. Problem Statement

The 3D Rayleigh-Bénard problem consists of the equations

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = -g(1-\alpha\delta_\theta\theta)\mathbf{e}_3 \tag{2.1}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2.2}$$

$$\frac{\partial\theta}{\partial t} - \beta\Delta\theta + (\mathbf{u}\cdot\nabla)\theta = 0 \tag{2.3}$$

in $\Omega = (-L/2, L/2)^2 \times (0,h)$ with the boundary conditions

$$\mathbf{u} = 0 \text{ at } z = 0, h \tag{2.4}$$

$$\theta(0) = 1, \theta(h) = 0 \tag{2.5}$$

$$p, \mathbf{u}, \theta \text{ periodic in the horizontal directions } x, y.$$

Here, ν is the viscosity, α is the volume expansion coefficient, δ_θ is the actual difference in temperature between $z = 0$ and $z = h$, β is the heat conduction coefficient, g the acceleration due to gravity, and $\mathbf{e}_3 = (0,0,1)$.
We consider the vertically averaged velocity

$$\bar{u} = \int_0^h u(x,y,z)dz \tag{2.6}$$

$$\bar{v} = \int_0^h v(x,y,z)dz. \tag{2.7}$$

Note that we have 2D incompressibility from (2.2) and $\mathbf{u} = 0$ at the boundary in the z-direction

$$\int_0^h u_x + v_y + w_z dz = \bar{u}_x + \bar{v}_y + [w(h) - w(0)] = \bar{u}_x + \bar{v}_y = 0. \tag{2.8}$$

We denote $\bar{\mathbf{u}} = (\bar{u}, \bar{v})$. We can then write the vertical average of the horizontal components of the 3D momentum equation, (2.1), as the 2D Navier-Stokes equation (NSE)

$$\bar{\mathbf{u}}_t + (\bar{\mathbf{u}}\cdot\nabla)\bar{\mathbf{u}} - \nu\Delta\bar{\mathbf{u}} + \nabla\bar{p} = \mathbf{F}^\partial(\mathbf{u}) + \mathbf{F}^R(\mathbf{u}) \tag{2.9}$$

$$\nabla\cdot\bar{\mathbf{u}} = 0 \tag{2.10}$$

in $\bar{\Omega} = (-L/2, L/2)^2$, where $\mathbf{F}^\partial$ is a boundary shear force and $\mathbf{F}^R$ is a Reynolds stress force.

2.1. Towards a Final Expression for $\mathbf{F}^\partial$ and $\mathbf{F}^R$.

We begin with the boundary shear. This can simply be written as

$$\mathbf{F}^\partial = \nu\Delta\overline{\mathbf{u}_{zz}} = \nu\begin{pmatrix} \frac{\partial u}{\partial z}(x,y,h) - \frac{\partial u}{\partial z}(x,y,0) \\ \frac{\partial v}{\partial z}(x,y,h) - \frac{\partial v}{\partial z}(x,y,0) \end{pmatrix}. \tag{2.11}$$

We then define the Reynolds force as

$$\mathbf{F}^R = (\bar{\mathbf{u}}\cdot\nabla)\bar{\mathbf{u}} - \overline{(\mathbf{u}\cdot\nabla)\mathbf{u}} \tag{2.12}$$

To derive a simpler expression for the Reynolds force, we first note that integration by parts and 3D incompressibility yields

$$\overline{wu_z} = -\overline{uw_z} = \overline{u(u_x + v_y)}. \tag{2.13}$$

We can insert this expression into the first component of the Reynolds stress, noting $\bar{u}_x = -\bar{v}_y$

$$\begin{aligned} F_1^R &= \bar{u}\bar{u}_x + \bar{v}\bar{u}_y - \overline{uu_x + vu_y + wu_z} \\ &= \bar{u}\bar{u}_x + \bar{v}\bar{u}_y - 2\overline{uu_x} - \overline{vu_y} - \overline{uv_y} \\ &= 2\bar{u}\bar{u}_x + \bar{v}\bar{u}_y + \bar{u}\bar{v}_y - 2\overline{uu_x} - \overline{vu_y} - \overline{uv_y}. \end{aligned} \tag{2.14}$$

Using the identities

$$\begin{aligned} \overline{[(u-\bar{u})^2]_x} &= 2\overline{(u-\bar{u})(u_x - \bar{u}_x)} \\ &= 2(\overline{uu_x} - \bar{u}\bar{u}_x - \bar{u}\bar{u}_x + \bar{u}\bar{u}_x) \\ &= 2\overline{uu_x} - 2\bar{u}\bar{u}_x, \end{aligned} \tag{2.15}$$

$$\begin{aligned} \overline{[(u-\bar{u})(v-\bar{v})]_y} &= \overline{(u_y - \bar{u}_y)(v-\bar{v})} - \overline{(u-\bar{u})(v_y - \bar{v}_y)} \\ &= \overline{vu_y} - \bar{v}\bar{u}_y + \overline{uv_y} - \bar{u}\bar{v}_y, \end{aligned} \tag{2.16}$$

we find that

$$F_1^R = -\overline{[(u-\bar{u})^2]_x} - \overline{[(u-\bar{u})(v-\bar{v})]_y}. \tag{2.17}$$

A symmetric derivation yields the second component

$$F_2^R = -\overline{[(v-\bar{v})^2]_y} - \overline{[(u-\bar{u})(v-\bar{v})]_x}. \tag{2.18}$$

Note that $\mathbf{F}^R$ has zero two-dimensional spatial mean, but $\mathbf{F}^\partial$ might not. Analysis for the NSE with periodic boundary conditions is enabled by subtracting the mean flow, so we take $\bar{\mathbf{u}} = \bar{\mathbf{u}}_2 + \bar{\mathbf{u}}_0$, where

$$\bar{\mathbf{u}}_0 = \iint_{\overline{\Omega}} \bar{\mathbf{u}} \; d\bar{\mathbf{x}}. \tag{2.19}$$

We can then use (2.1) to write

$$\frac{\partial \bar{\mathbf{u}}_2}{\partial t} + (\bar{\mathbf{u}}_2 \cdot \nabla)\bar{\mathbf{u}}_2 - \nu\Delta\bar{\mathbf{u}}_2 + \nabla\bar{p} = \bar{\mathbf{F}} \tag{2.20}$$

with

$$\overline{\mathbf{F}} = \mathbf{F}^\partial + \mathbf{F}^R - \iint_{\overline{\Omega}} \mathbf{F}^\partial \; d\overline{\mathbf{x}} - (\bar{\mathbf{u}}_0 \cdot \nabla)\bar{\mathbf{u}}_0. \tag{2.21}$$

3. Important Features of Turbulence

Since $\bar{\mathbf{u}}_2$ is periodic, it is natural to write it as a Fourier series

$$\bar{\mathbf{u}}_2 = \sum_{\mathbf{k}\in\mathbb{Z}^2/\{0\}} \hat{\mathbf{u}}_{\mathbf{k}} e^{i\kappa_0 \mathbf{k}\cdot\mathbf{x}}, \qquad \kappa_0 = \frac{2\pi}{L}.$$

More generally, we define a *wavenumber* as $\kappa = \kappa_0 |\mathbf{k}|_{\mathbb{R}^d}$ for $d = 2, 3$.
In 2D, Parseval's identity reads as

$$\|\bar{\mathbf{u}}_2\|^2_{L^2(\overline{\Omega})} = L^2 \sum_{\mathbf{k}\in\mathbb{Z}^2/\{0\}} |\hat{\mathbf{u}}_{\mathbf{k}}|^2_{\mathbb{C}^2}, \tag{3.1}$$

which we will utilize throughout our computations.
We denote the *time average* by

$$\langle \phi \rangle = \lim_{T\to\infty} \frac{1}{T} \int_0^T \phi(\mathbf{u}(t))\ dt \tag{3.2}$$

where ϕ is an appropriately chosen function, such as the norm from a function space, e.g. the L^2 norm. Strictly speaking, this limit should be a Hahn-Banach extension of the classical limit (see [8]). Naturally, in our numerical study we take T finite, but sufficiently large. The effect of this approximation is analyzed in [7], [9].
We now consider the functional form of the 2D NSE,

$$\frac{d\bar{\mathbf{u}}_2}{dt} + \nu A\overline{\mathbf{u}}_2 + B(\overline{\mathbf{u}}_2, \overline{\mathbf{u}}_2) = \bar{\mathbf{f}}, \tag{3.3}$$

where $A = -\Delta$ is the Stokes operator, $B(u,v)$ is the bilinear operator $\mathcal{P}(u\cdot\nabla)v$, $\mathcal{P}$ is the Helmholtz-Leray projection onto divergence-free functions and $\bar{\mathbf{f}} = \mathcal{P}\overline{\mathbf{F}}$.
Physical quantities in turbulent flow are most relevant when taken as averages per unit mass. Accordingly, in 3D, we define the energy as

$$\frac{1}{|\Omega|}\langle \|\mathbf{u}\|^2_{L^2(\Omega)}\rangle = \frac{1}{L^2 h}\Big\langle \int_\Omega |\mathbf{u}|^2_{\mathbb{R}^3}\ d\mathbf{x}\Big\rangle \tag{3.4}$$

and in 2D the energy, *enstrophy* and *palinstrophy* are given as

$$\frac{1}{L^2}\langle \|\overline{\mathbf{u}}\|^2_{L^2(\overline{\Omega})}\rangle \qquad \frac{1}{L^2}\Big\langle \left\|A^{1/2}\overline{\mathbf{u}}\right\|^2_{L^2(\overline{\Omega})}\Big\rangle \qquad \frac{1}{L^2}\langle \|A\overline{\mathbf{u}}\|^2_{L^2(\overline{\Omega})}\rangle, \tag{3.5}$$

respectively. Here, powers of A are defined through its eigenvalues.
We can define the *discrete time averaged energy per unit mass* (over a range of length scales) as

$$\mathsf{e}_{\kappa_1,\kappa_2} = \Big\langle \sum_{\kappa_1 \le |\mathbf{k}|_{\mathbb{R}^d} < \kappa_2} |\hat{\mathbf{u}}_{\mathbf{k}}|^2_{\mathbb{C}^d}\Big\rangle. \tag{3.6}$$

We define the *mean enstrophy dissipation rate* (per unit mass) in 2D as

$$\eta = \frac{\nu}{L^2}\langle \|A\mathbf{u}\|^2_{L^2(\overline{\Omega})}\rangle \tag{3.7}$$

and the *mean energy dissipation rate* in 3D as

$$\epsilon = \frac{\nu}{L^2 h}\Big\langle \left\|A^{1/2}\mathbf{u}\right\|^2_{L^2(\Omega)}\Big\rangle. \tag{3.8}$$

The *energy spectrum* is a function $\mathcal{E}(s)$ defined for all real numbers $s \ge \kappa_0$. It is connected to the discrete time averaged energy, $\mathsf{e}_{\kappa,2\kappa}$ as a Riemann sum (as $\kappa_0 \to 0$)

$$\mathsf{e}_{\kappa,2\kappa} \sim \int_\kappa^{2\kappa} \mathcal{E}(s)\ ds. \tag{3.9}$$

Kraichnan's heuristic derivation of $\mathcal{E}(s) \sim \eta^{2/3}\kappa^{-3}$ for 2D turbulence is then consistent with $\mathsf{e}_{\kappa,2\kappa} \sim \eta^{2/3}\kappa^{-2}$.
We first consider the case where the force is limited to a range of wavenumbers

$$\mathbf{F} = \sum_{\underline{\kappa}\ \le \kappa_0|\mathbf{k}| <\ \overline{\kappa}} \hat{\mathbf{f}}_{\mathbf{k}} e^{i\kappa_0 \mathbf{k}\cdot\mathbf{x}}. \tag{3.10}$$

The two major differences between three-dimensional and two-dimensional turbulence are the expected energy spectrum and *cascade*. In three dimensions, the cascade refers to the transfer of energy towards the smaller scales, or larger wavenumbers, being roughly constant over the *inertial range*, where the Kolmogorov spectrum $\mathcal{E}(\kappa) \approx \kappa^{-5/3}$ is expected to hold. In two dimensions, we instead expect the transfer of *enstrophy* to be roughly constant and $\mathcal{E}(\kappa) \approx \kappa^{-3}$ in the inertial range. There is also an inverse cascade of energy toward small wavenumbers in 2D. These features are depicted in Figures 1 and 2. A cartoon sketch of the spectra are shown in a log-log scale.

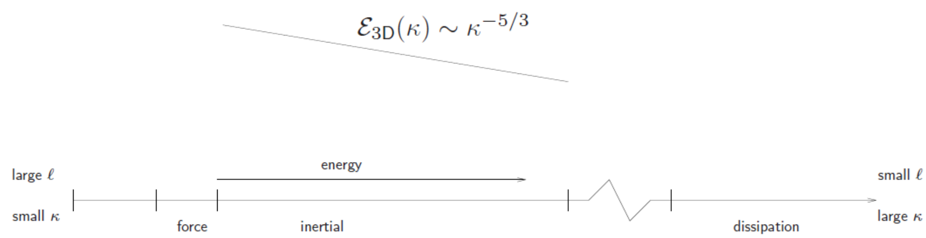


FIGURE 1. 3D Turbulence

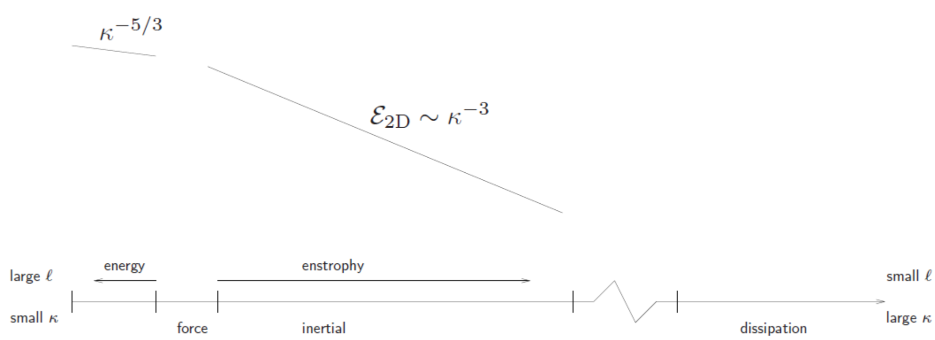


FIGURE 2. 2D Turbulence

The left end corresponds to small wavenumbers or equivalently large scales, whereas the right end is the opposite.
Heuristic arguments suggest that the dissipation range starts around $\kappa_\eta = (\eta/\nu^3)^{1/6}$ in 2D and $\kappa_\epsilon = (\epsilon/\nu^3)^{1/4}$ in 3D. At these scales, viscous effects dominate, and enstrophy and energy are dissipated as heat.

3.1. **Enstrophy Transfer Function.** Let $\mathbf{p}_\kappa$ and $\mathbf{q}_\kappa$ represent the low and high modes, respectively, i.e.,

$$\mathbf{q}_\kappa = \sum_{\kappa < |\mathbf{k}|_{\mathbb{R}^d}} \hat{\mathbf{u}}_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{x}}, \qquad \mathbf{p}_\kappa = \sum_{|\mathbf{k}|_{\mathbb{R}^d} \le \kappa} \hat{\mathbf{u}}_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{x}} \tag{3.11}$$

Taking the L^2 inner product of the 2D NSE with $A\mathbf{q}_\kappa$ gives the *enstrophy balance*

$$\frac{1}{2}\frac{d}{dt}\left\|A^{1/2}\mathbf{q}_\kappa\right\|^2_{L^2(\overline{\Omega})} + \nu\|A\mathbf{q}_\kappa\|^2_{L^2(\overline{\Omega})} = \kappa_0^{-2}\mathfrak{E}_\kappa + (f, A\mathbf{q}_\kappa), \tag{3.12}$$

where the *enstrophy transfer function* is

$$\mathfrak{E}_\kappa = \mathfrak{E}_\kappa^{\rightarrow} - \mathfrak{E}_\kappa^{\leftarrow} \tag{3.13}$$

for

$$\begin{aligned}\mathfrak{E}_\kappa^{\rightarrow}(\mathbf{u}) &= -\kappa_0^2(B(\mathbf{p}_\kappa, \mathbf{p}_\kappa), A\mathbf{q}_\kappa)\\ \mathfrak{E}_\kappa^{\leftarrow}(\mathbf{u}) &= -\kappa_0^2(B(\mathbf{q}_\kappa, \mathbf{q}_\kappa), A\mathbf{p}_\kappa).\end{aligned}$$

Note that if $f = f_{\underline{\kappa},\overline{\kappa}}$, then $(f, A\mathbf{q}_\kappa) = 0$ for $\kappa > \overline{\kappa}$. So, by the fundamental theorem of calculus and the fact that $\left\|A^{1/2}\mathbf{u}\right\|_{L^2(\overline{\Omega})}$ is bounded for all time,

$$0 \le \langle \mathfrak{E}_\kappa \rangle \text{ for } \kappa > \overline{\kappa} \tag{3.14}$$

which implies the mean flux is towards the higher wavenumbers.
A rigorous sufficient condition for an enstropy cascade is found in [8]:

Theorem 3.1.

$$1 - \left(\frac{\kappa}{\kappa_\sigma}\right)^2 \le \frac{\langle \mathfrak{E}_\kappa \rangle}{\eta} \le 1 \textit{ for } \overline{\kappa} \le \kappa \le \kappa_\sigma,$$

where

$$\kappa_\sigma^2 = \frac{\left\langle\ \|A\bar{\mathbf{u}}\|^2_{L^2(\overline{\Omega})}\ \right\rangle}{\left\langle\ \left\|A^{1/2}\bar{\mathbf{u}}\right\|^2_{L^2(\overline{\Omega})}\ \right\rangle} \tag{3.15}$$

From this theorem, we see

$$\overline{\kappa} \le \kappa \ll \kappa_\sigma \Rightarrow \langle \mathfrak{E}_\kappa \rangle \approx \eta \tag{3.16}$$

meaning the mean enstrophy transfer rate across wavenumber κ is nearly constant for sufficiently large κ. A similar result holds for the transfer of energy in terms of

$$\kappa_\tau^2 = \frac{\left\langle\ \left\|A^{1/2}\bar{\mathbf{u}}\right\|^2_{L^2(\overline{\Omega})}\ \right\rangle}{\left\langle\ \|\bar{\mathbf{u}}\|^2_{L^2(\overline{\Omega})}\ \right\rangle}. \tag{3.17}$$

The fact that $\kappa_\tau \le \kappa_\sigma$ is consistent with the expectation that in 2D a (direct) cascade of enstrophy towards larger wavenumbers is more prominent than a direct cascade of energy. An unresolved question is what, if any, kind of finite mode force results in $\overline{\kappa} \ll \kappa_\sigma$. This is mainly why we consider the 2D force naturally imposed in the vertically averaged velocity of the Rayleigh-Bénard flow.

3.2. **Pseudo-Flux.** If the force f defined in (3.12) has arbitrarily many modes, as is expected in the case of the vertically averaged velocity, we consider a *pseudo-transfer function*,

$$\kappa_0^{-2}\mathfrak{F}_\kappa = \mathfrak{E}_\kappa^{\rightarrow} - \mathfrak{E}_\kappa^{\leftarrow} + (f, A\mathbf{q}_\kappa). \tag{3.18}$$

The difference between (3.18) and (3.13) is the $(f, A\mathbf{q}_\kappa)$ term. For the pseudo-transfer function, this need not be zero. This causes the sign of $\langle \mathfrak{E}_\kappa \rangle$ to be uncertain. All we can say is

$$0 \le \langle \mathfrak{F}_\kappa(\mathbf{u}) \rangle \quad \text{for } \kappa \ge \kappa_0 \tag{3.19}$$

We know the viscous term has the same sign for all κ, but the force f may have a mixing effect similar to the nonlinear term.
We can state a theorem analogous to Theorem 3.1 for the pseudo-transfer function:

Theorem 3.2. [1]

$$1 - \left(\frac{\kappa}{\kappa_\sigma}\right)^2 \le \frac{\langle \mathfrak{F}_\kappa \rangle}{\eta} \le 1$$

From this, we can get another similar sufficient condition for a direct enstrophy "cascade," namely that if $\kappa_0 \ll \kappa_\sigma, \langle \mathfrak{F}_\kappa \rangle \approx \eta$ for $\kappa_0 \le \kappa \ll \kappa_\sigma$.

3.3. **The Grashof Number.** We define the Grashof number for a time independent force f as

$$G = \frac{|f|}{\nu^2 \kappa_0^2} \tag{3.20}$$

This dimensionless number controls the dissipation scale. The following estimate is valid for a force with an arbitrary number of modes.

Theorem 3.3. [10]

$$G^{1/6} \lesssim \frac{\kappa_\eta}{\kappa_0} \le G^{1/3} \tag{3.21}$$

In the finite mode case, the gap in (3.21) closes if κ_σ is large enough (see [6])

$$\kappa_\sigma \sim \kappa_\eta \Rightarrow \frac{\kappa_\eta}{\kappa_\sigma} \sim G^{1/4}. \tag{3.22}$$

The relation (3.22) is up to a logarithmic constant.

The Grashof number also controls the Reynolds number.

Theorem 3.4. [3]

$$\frac{G^{1/2}}{(\log G)^{1/4}} \lesssim \mathrm{Re} \le G$$

where Re is the Reynolds number, defined in 2D as

$$\mathrm{Re} = \frac{\langle \|\bar{\mathbf{u}}\|^2 \rangle^{1/2}}{\nu}. \tag{3.23}$$

The Reynolds number for the vertically averaged velocity of the Rayleigh-Bénard force satisfies the following bound:

Theorem 3.5.

$$\mathrm{Re} \lesssim \min\left\{ \frac{h}{L}\mathrm{Pr}\mathrm{Ra}, \quad \left(\frac{\mathrm{Ra}}{\mathrm{Pr}}\right)^{1/2} \left(\frac{\mathrm{Nu}-1}{\mathrm{Pr}}\right)^{1/2} \right\} \tag{3.24}$$

[1] All otherwise uncited theorems are from [2].

where Nu is the Nusselt number defined as

$$\mathrm{Nu} = -\frac{h}{L^2}\int_0^L\int_0^L \frac{\partial\langle\theta\rangle}{\partial z}\bigg|_{z=h} dxdy, \tag{3.25}$$

Pr is the Prandtl number defined as

$$\mathrm{Pr} = \frac{\nu}{\beta}, \tag{3.26}$$

and Ra is the Rayleigh number

$$\mathrm{Ra} = \frac{\delta_\theta \alpha g L^2}{h}\beta\nu. \tag{3.27}$$

3.3.1. *Additional Grashof Relations.* A bound for κ_η/κ_0 is given in terms of a modified Grashof number:

Theorem 3.6.

$$cG_*^{1/3} \le \frac{\kappa_\eta}{\kappa_0} \le G^{*1/3} \tag{3.28}$$

where

$$G_* = \frac{\|\langle\ \bar{\mathbf{f}}\ \rangle\|}{\nu^2\kappa_0^2} \quad G^* = \frac{\langle\ \|\bar{\mathbf{f}}\|^2\rangle^{1/2}}{\nu^2\kappa_0^2}.$$

The constant c is $O(1)$ for $\mathrm{Re} \le 1$ and can be taken $O\big((\mathrm{Pr}/\mathrm{Ra})^{1/3}\big)$ otherwise.

There is an alternative formulation for this inequality utilizing the Reynolds number instead of the Rayleigh number

$$\frac{G_*^{1/3}}{(\mathrm{Re}+1)^{1/3}} \lesssim \frac{\kappa_\eta}{\kappa_0} \le G^{*1/3}. \tag{3.29}$$

Numerically verifying these inequalities is one of our primary goals.

4. Computational Methods

To compute all of the following quantities, we used pseudo-spectral methods found in the Python suite, Dedalus (see [4] for more details). All simulations were run on a $512 \times 512 \times 16$ grid with a de-alias factor of $3/2$, using Ω as in (2.1) with $L = 2\pi$ and $h = 1$ (so $\kappa_0 = 1$). The horizontal directions were expanded with a Fourier basis. The vertical was expanded with a Chebyshev basis. All time averages were computed using an Euler step at every iteration of the Dedalus solver. For example, to compute $\langle\phi_\kappa\rangle$, where

$$\phi_\kappa = \sum_{|\mathbf{k}|=\kappa} |\hat{\mathbf{u}}_\mathbf{k}|^2, \tag{4.1}$$

we advanced at the j^{th} time step with

$$\phi_\kappa^j = \phi_\kappa^{j-1} + \Delta t \sum_{|\mathbf{k}|=\kappa} |\hat{\mathbf{u}}_\mathbf{k}|^2, \tag{4.2}$$

to approximate the solution to

$$\frac{d\phi}{dt} = \sum_{|\mathbf{k}|=\kappa} \big|\hat{\mathbf{u}}_\mathbf{k}(s)\big|^2 ds. \tag{4.3}$$

At the end of the run, we then divide by the total simulation time T to obtain the approximation to

$$\langle \phi_\kappa \rangle = \frac{1}{T}\int_0^T \sum_{|\mathbf{k}|=\kappa} \left|\hat{\mathbf{u}}_{\mathbf{k}}(s)\right|^2 ds. \tag{4.4}$$

4.1. **A Change of Variables.** When computing the solution, we used a different formulation from (2.1) - (2.3). The following change of variables was used.
We first note $\delta_\theta = \theta_0 - \theta_1$, where $\theta(0) = 1$ and $\theta(1) = 0$. We then set $\tilde{p} = p + gz$, and are then left with

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla\tilde{p} = g\alpha\delta_\theta\theta\mathbf{e}_3. \tag{4.5}$$

We then let

$$\tilde{\theta} = \delta_\theta\theta + \theta_1 = \begin{cases} \theta_0 & \text{if } \theta = 1 \\ \theta_1 & \text{if } \theta = 0 \end{cases} \tag{4.6}$$

(4.5) then becomes

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla\tilde{p} = g\alpha(\tilde{\theta} - \theta_1)\mathbf{e}_3 \tag{4.7}$$

with the new boundary conditions $\tilde{\theta}(0) = \theta_0$ and $\tilde{\theta}(1) = \theta_1$. We finally set the *buoyancy*, b, to be $g\alpha(\tilde{\theta} - \theta_1)$ and are left with our final expression

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla\tilde{p} = b\mathbf{e}_3 \tag{4.8}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{4.9}$$

$$\frac{\partial b}{\partial t} - \beta\Delta b + (\mathbf{u}\cdot\nabla)b = 0. \tag{4.10}$$

4.2. **Pseudo-Spectral Method.** We begin by discussing the Fast-Fourier Transform, and then display its use in showing how one can numerically compute the solution of Burgers equation.

4.2.1. *Fast-Fourier Transform.* The framework for a pseudo-spectral method is the *Fast-Fourier Transform*. We will discuss the one-dimensional case for simplicity. Let our space domain be $[0, 2L]$ (for convenience, we will take $L = \pi$). We then discretize our domain into $2M + 1$ grid points, with $x_0 = 0$ and $x_{2M+1} = 2\pi$. More generally,

$$x_m = \frac{2\pi m}{2M+1}$$

We then note the following three facts:

(1) $\xi_j = e^{ix_j}$ are the $(2M+1)^{\text{th}}$ roots of unity.
(2) $\xi_j = \xi_1^j$
(3) $\xi_j^k = \xi_k^j, \xi_j^{-1} = \bar{\xi}_j$.

Then, for a function

$$v(x) = \sum_{m=-M}^{M} c_m e^{imx},$$

we seek to find the Fourier coefficients, c_m. Using the above facts, we can instead say

$$v(x) = \sum_{m=-M}^{M} c_m e^{imx} = \sum_{m=-M}^{M} c_m \xi_j^m = v(x_j) = v_j.$$

If we then define the matrix $[T]_{j,m} = \xi_j^m$ and the vectors

$$\mathbf{c} = \begin{bmatrix} c_{-M} \\ \vdots \\ c_M \end{bmatrix} \qquad \mathbf{v} = \begin{bmatrix} v(0) \\ \vdots \\ v(x_{2M}) \end{bmatrix},$$

finding our coefficients becomes a matrix multiplication problem. Additionally, T is easy to invert: $[T^{-1}]_{j,m} = \frac{1}{2M+1}\xi_m^{-j}$. This allows us to go back and forth between the physical space and the coefficient space by matrix multiplication.
This general approach is the *Discrete Fourier Transform*, which runs in $O(M^2)$. A divide and conquer approach, which is the Fast-Fourier Transform, was developed that runs in $O(M \log M)$.

4.2.2. *Burger's Equation Using the Fast-Fourier Transform.* Consider Burger's equation,

$$u_t - \nu u_{xx} + u u_x = 0.$$

Using separation of variables, we can write our solution in terms of its Fourier series. Taking the complex version, we define

$$S_M = \sum_{m=-M}^{M} c_m e^{imx} \qquad c_{-m} = \bar{c}_m$$

We can approximate u_x and uu_x by differentiating term by term for both terms and then multiplying the two partial sums for the latter. Using a discrete convolution, we arrive at

$$uu_x \approx S_M S_M' = \sum_{m=-2M}^{2M} \Big(\sum_{k=-M}^{M} ik c_{m-k} c_k \Big) e^{imx}.$$

We can then write an ODE approximation of Burgers as

$$\frac{dc_m}{dt} - \nu m^2 c_m + \sum_{k=-M}^{M} ik c_{m-k} c_k = 0.$$

The Fast-Fourier Transform allows us to find the coefficients of $v = S_M S_M'$ and therefore approximate the solution. We first take the coefficients of u and u_x and apply the inverse transform to get our grid values, $u(x_m)$. We can then multiply at each x_m to get $u(x_m)u_x(x_m)$, and then perform a forward transform to get our coefficients for the non-linear term.
The extension to two or three-dimensions is natural. For the Rayleigh-Bénard

problem, we can split $\mathbf{u} = (u, v, w)$ into its three components

$$\frac{\partial u}{\partial t} - \nu \Delta u + (\mathbf{u} \cdot \nabla) u + \nabla p = 0$$
$$\frac{\partial v}{\partial t} - \nu \Delta v + (\mathbf{u} \cdot \nabla) v + \nabla p = 0$$
$$\frac{\partial w}{\partial t} - \nu \Delta w + (\mathbf{u} \cdot \nabla) w + \nabla p = b.$$
$$\frac{\partial b}{\partial t} - \beta \Delta b + (\mathbf{u} \cdot \nabla) b = 0$$

where $(\mathbf{u} \cdot \nabla)u$ is shorthand for $uu_x + vu_y + wu_z$, and construct a separate matrix for each dimension as above to perform Fast-Fourier Transforms as necessary for each component. The overall time complexity for higher dimensions is then given by $O(M^n(\log M)^n)$. In our case, $n = 3$.

4.3. **Basis Choices.** For the horizontal directions, we used a Fourier basis because of the periodic boundary conditions. For the vertical, a Chebyshev basis allows for more of the grid points to be concentrated towards the end points, which is where we are expecting most of our activity in that direction. By using the trigonometric form for a Chebyshev polynomial in $|z| < 1$, which is given by $T_n(z) = \cos(n \arccos z)$, we see that it is possible to use an FFT in this direction as well.

5. Simulation Results

Our approach for examining the results will be the same for each Rayleigh number. We begin by looking at the computed values for all named constants, as well as examine the effect of the Rayleigh number on the inequality found in Theorem 3.6. We then look at the velocity fields of both $\bar{\mathbf{u}}$ and $\mathbf{u}$, as well as the 2D vorticity. In these images, "midplane" refers to $u(x, y, 1/2)$. It is analogous for v and w. We then look at the energy spectra for the energy, the Reynolds force (see (2.17) and (2.18)) and the shear force (see (2.11)) of the 2D system. In the case of the total energy, we then look at an interpolated line to see if dissipation occurs at the expected rate found in Figures 1 and 2. We next look at the total energy for the Reynolds and shear forces, as well as the total energy of the entire 3D system. We finally compare the computed results for the various special wavenumbers and constants found in Sec. 3 to the bounds also found in that section.
Our simulations were run with differing Rayleigh numbers for each. We take $\nu = 1 = \beta$, as well as $\mathrm{Pr} = 1$ for all. We again note $L = 2\pi$ so that $\kappa_0 = 1$.

5.1. Composite Results.

5.1.1. *Values.*

Ra	5×10^5	10^6	2.5×10^6
κ_η	8.1974	12.837	14.143
κ_σ	10.832	16.298	18.509
κ_τ	4.252	5.261	6.294
η	3.03×10^5	4.48×10^6	8.00×10^6
G_*	1.77×10^4	3.76×10^4	9.52×10^4
G^*	1.30×10^8	5.12×10^8	3.64×10^9

5.1.2. *Theorem 3.6 Bounds vs. Rayleigh Number.* For additional Rayleigh numbers, each run was performed for .1 simulated seconds. We additionally look at another bound, also given in [2]

Theorem 5.1. *Assuming $G_* \leq c(\kappa_\tau \kappa_\sigma / \kappa_0^2)$, where $c > 1$ is some constant, we have*

$$\frac{\kappa_\eta}{\kappa_0} \gtrsim \left(\frac{\kappa_\tau \kappa_\sigma}{\kappa_0^2}\right)^{1/6} G_*^{1/6} \tag{5.1}$$

In the following plot, we use the form of the inequality found in (3.29) as opposed to (3.28), as it is easy to see from the above table that the latter will hold. We do, however, look at the change in the lower bound constant from (3.28) in later sections. Looking at the plot which shows (3.29), (5.1), and $G^{*1/3}$ compared to κ_η, we can see the constant on the lower bounds become smaller, and therefore the bound tighter, as the Rayleigh number increases.

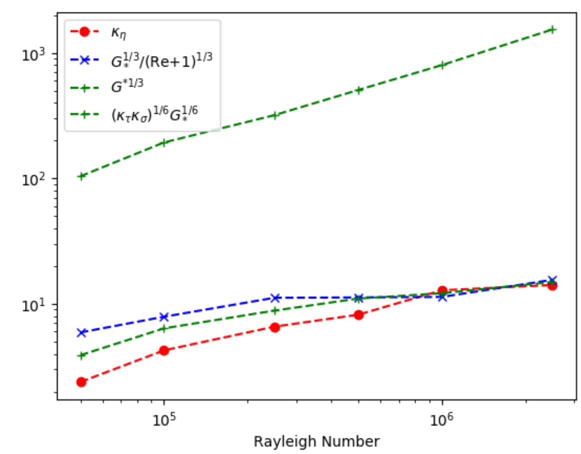


FIGURE 3. Bounds v. Rayleigh Number

5.2. **Ra** $= 5 \times 10^5$**.** This simulation was run for a total of .15 simulated seconds.

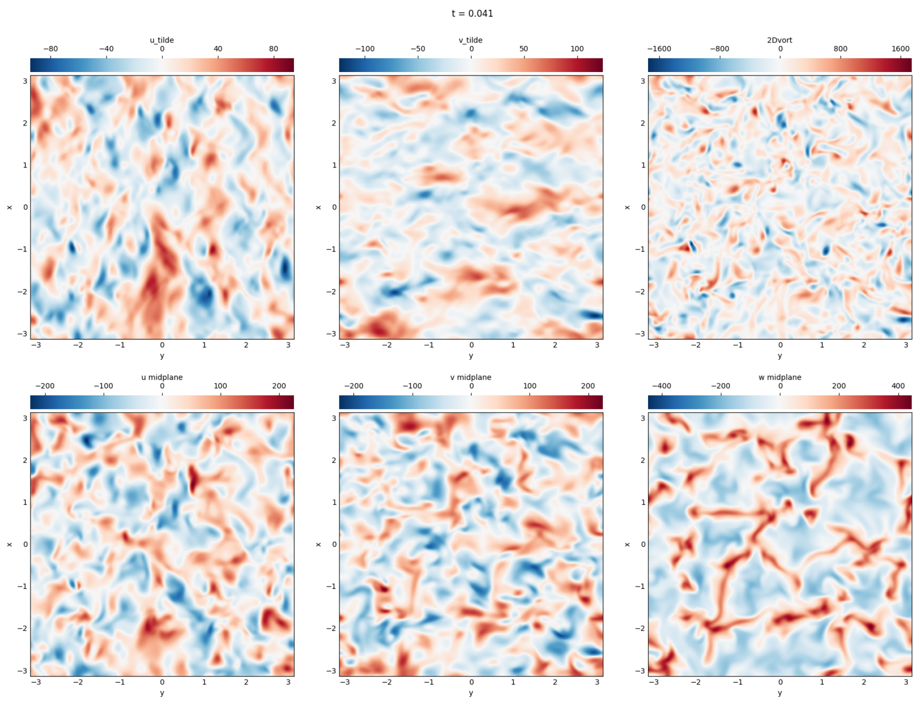


Figure 4. Velocity Fields for Ra $= 5 \times 10^5$

5.2.1. *Field Comparisons.* Comparing the horizontal midplanes to the vertically averaged velocities, we see the flows differ primarily towards the inside of our domain. However, the flow in the vertically averaged case is weaker than in the midplane. We also see some vortices forming in the vertically averaged velocity.

5.2.2. *Spectra.* We now look at an interpolated line beginning at approximately κ_η.

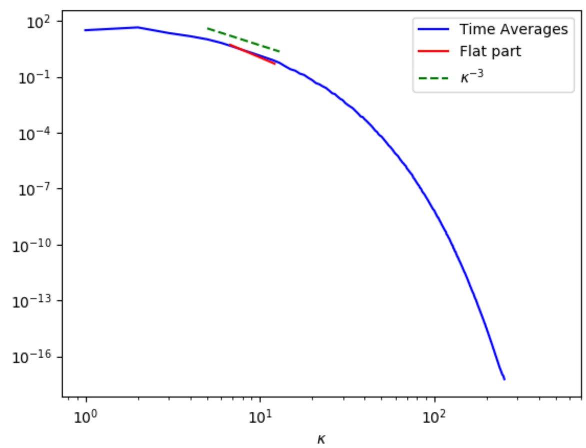


FIGURE 5. Energy Spectrum (Ra $= 5 \times 10^5$)

Looking at our interpolated line, we see our enstrophy dissipation rate is approximately κ^{-3}, as derived by Kraichnan. However, we believe the simulation did not converge quickly enough to draw definitive conclusions from the energy spectrum. Figure 6 shows the averages of the Fourier coefficients for the force terms $\mathbf{F}^R$ and $\mathbf{F}^\partial$. They indicate that indeed, all modes are active, though $\mathbf{F}^R$ dominates $\mathbf{F}^\partial$.

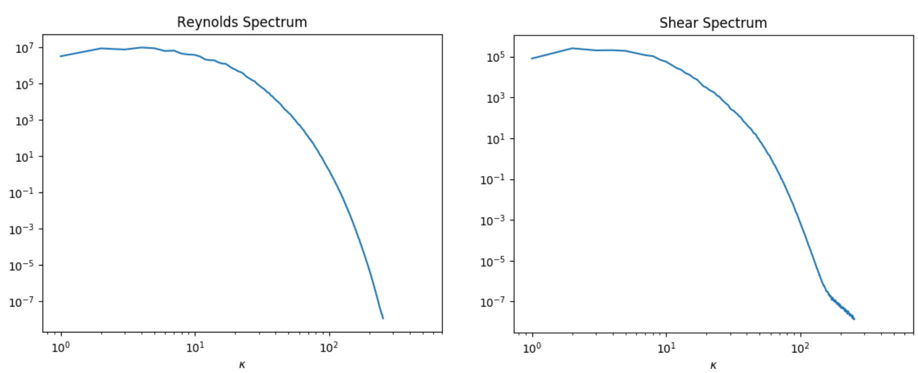


FIGURE 6. Shear and Reynolds Spectra (Ra $= 5 \times 10^5$)

5.2.3. *Total Energies.* The instantaneous energies (without the time average) were computed, after an initial transient period, using the L^2 norm of $\bar{\Omega}$ (in the case of the shear and Reynolds energy) or the L^2 norm of Ω (for the Total 3D energy). From the total energy plot, we see a large amount of the energy for the system comes from the vertical direction.

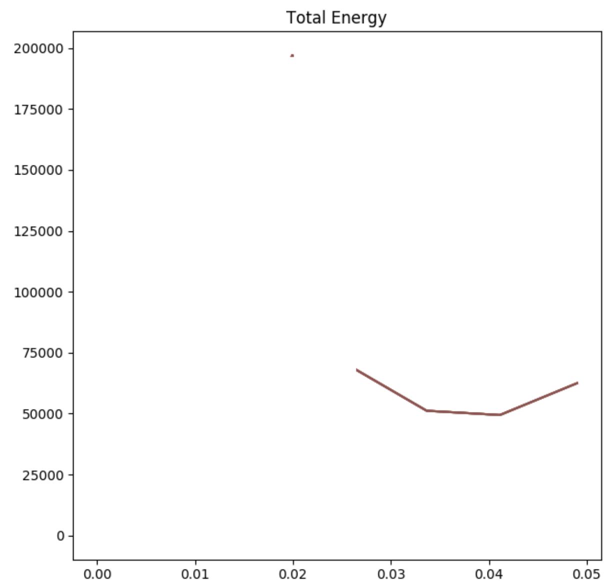


FIGURE 7. Total Energy of 3D System (Ra $= 5 \times 10^5$)

Looking at the shear and Reynolds energy, they are as expected based on the spectra in Figure 6.

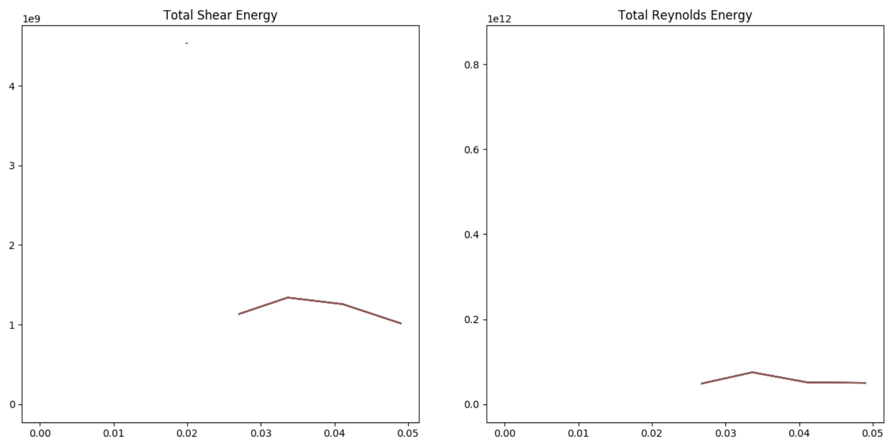


FIGURE 8. Shear and Reynolds Total Energy (Ra $= 5 \times 10^5$)

5.2.4. *Computed v. Expected Values.* From the outset, we see that κ_η and the beginning of the enstrophy dissipation in Figure 5 approximately line up. We can then look at results related to Theorem 3.2. From Figure 9, we see the pseudo-flux is constant up to κ_σ, as well as nearly equal to η.

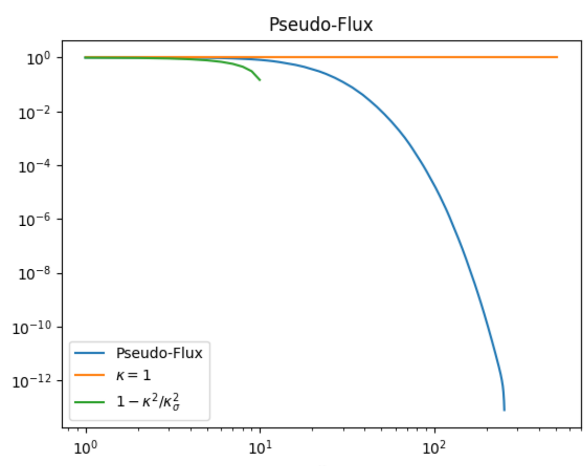


FIGURE 9. Theorem 3.2 Bounds (Ra $= 5 \times 10^5$)

The curve corresponding to $1 - (\kappa/\kappa_\sigma)^2$ is plotted only for the positive values. We in fact see the pseudo-flux remains closer to its upper bound until κ_σ.

We now come to Theorem 3.6. From the table in Sec. 5.1.1, we already see $G_* <$

Ra $< G^*$, which bodes well for our inequality. Our values for the inequality are

$$\frac{G_*^{1/3}}{\mathrm{Ra}^{1/3}} = .33851 \le \kappa_\eta = 10.507 \le G^* = 585.629$$

Here, we can take our constant for our lower bound to be about 30.

5.3. **Ra** $= 10^6$**.** This simulation was run for a total of .1 simulated seconds.

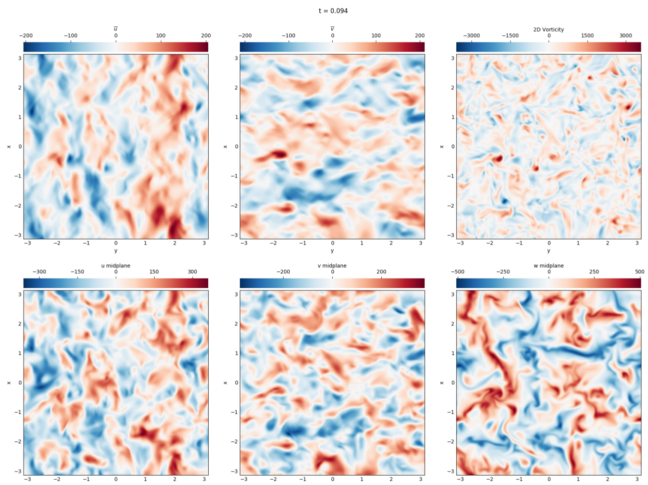


FIGURE 10. Velocity fields for Ra $= 10^6$

5.3.1. *Field Comparisons.* For this higher Rayleigh number, we see more apparent differences between $\bar{\mathbf{u}}$ and $\mathbf{u}$. We see larger pieces of positive velocities and negative velocities concentrated together in $\bar{\mathbf{u}}$, whereas our 3D field has a much more fractured texture. This is prominent in the v fields in $[-1, 1] \times [-\pi, 1]$. The weaker flow in the averaged case is also demonstrated at this higher Rayleigh number. We also see the vorticies are about twice as strong as in the run at Ra $=5 \times 10^5$.

5.3.2. *Spectra.* At Ra $= 10^6$ the energy spectrum exhibits a flat section which fits better with an inverse energy cascade ($\kappa^{-5/3}$) rather than a direct enstrophy cascade. As in Sec. 5.2.2, we do not believe convergence was quick enough.

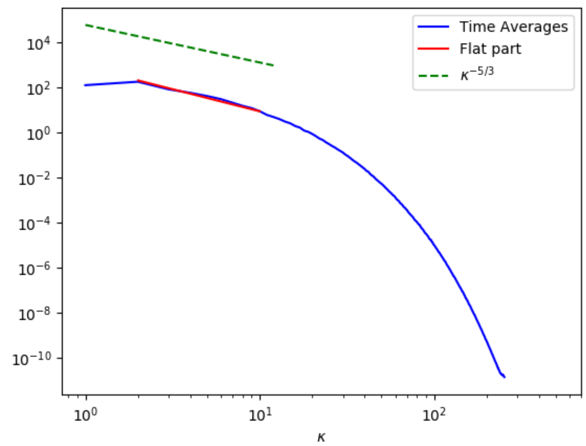


FIGURE 11. Energy Spectrum (Ra = 10^6)

Again, the shear and Reynolds spectra maintain the same shape and the Reynolds force still dominates the shear force.

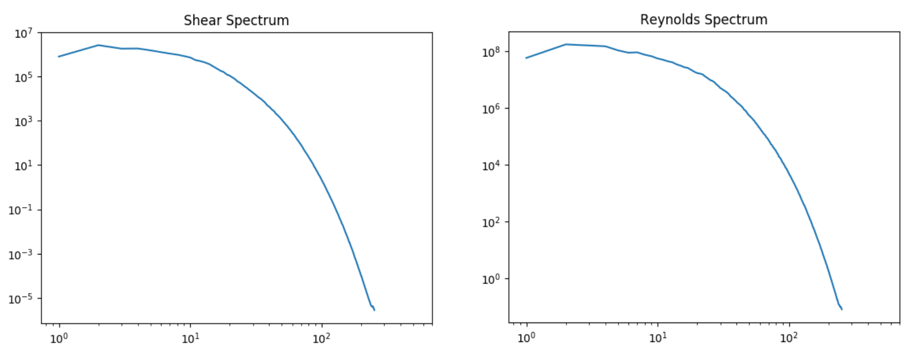


FIGURE 12. Shear and Reynolds Spectra (Ra = 10^6)

5.3.3. *Total Energies.* From the total 3D energy of the system, we in fact see it increasing as time elapses. The initial peak is surpassed towards the end of our simulation and we see that the system has not yet settled. We again see that the vertical direction does contribute significantly to the total energy.

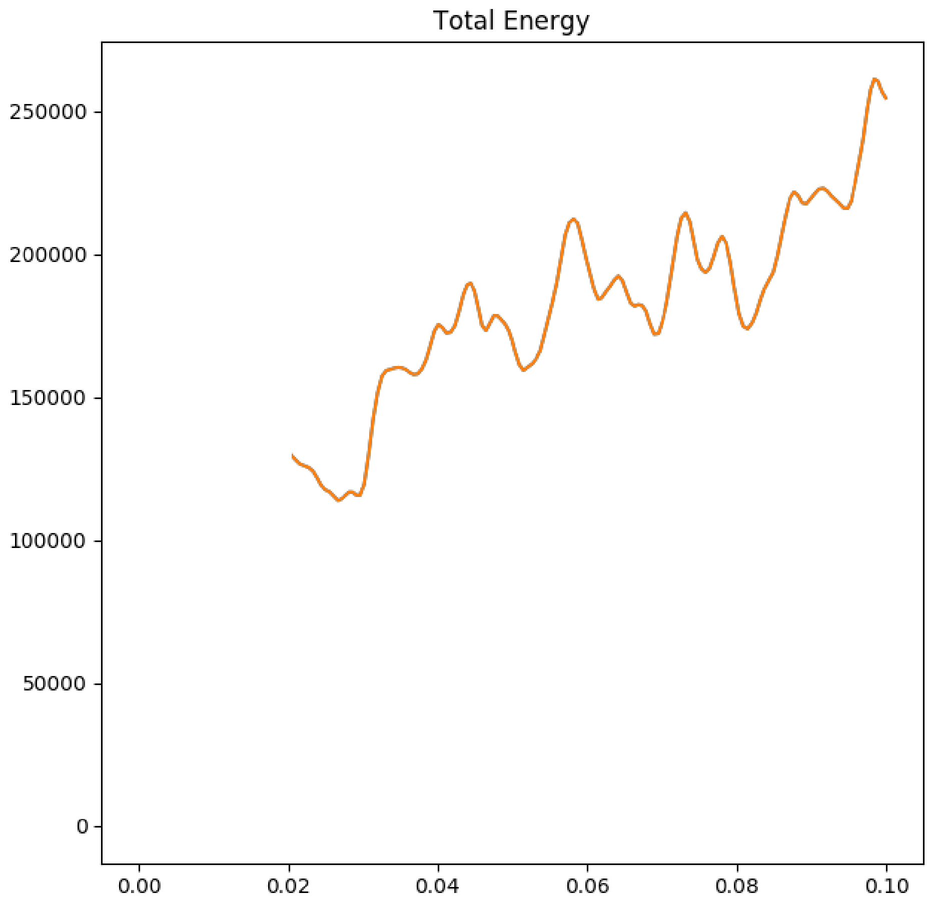


FIGURE 13. Total Energy of 3D System (Ra $= 10^6$)

The Reynolds and shear energies further show the contribution of the vertical direction to the total energy since these two forces only have horizontal components.

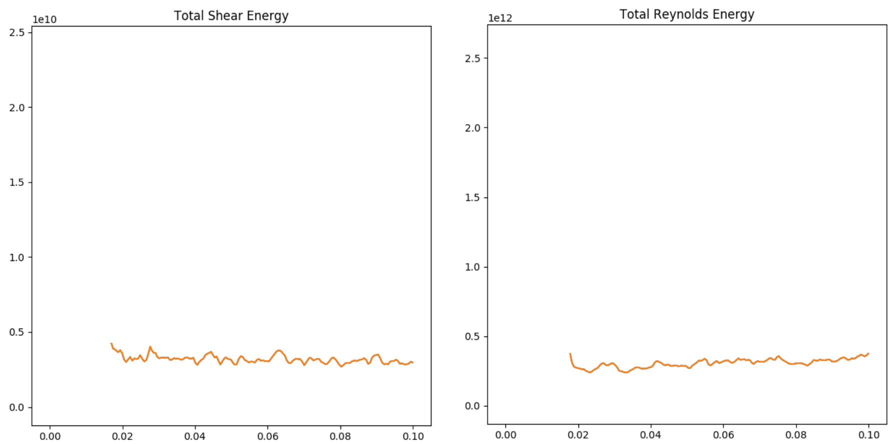


FIGURE 14. Shear and Reynolds Total Energy (Ra $= 10^6$)

5.3.4. *Computed v. Expected Values.* We see all constants increased from Ra $=$ 5×10^5, as expected.

From Figure 11, it is harder to see if the dissipation begins with κ_η. What we believe to be the inverse cascade does terminate prior to it, however. The pseudo-flux is more revealing. We still see the same nearly constant rate up to κ_σ. Additionally, the bounds found in Theorem 3.2 still hold.

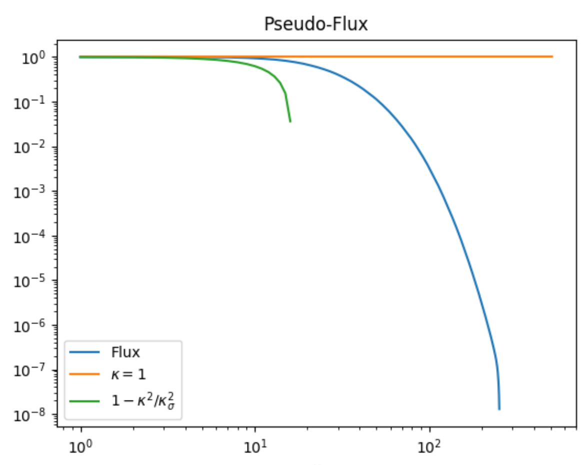


FIGURE 15. Theorem 3.2 Bounds (Ra $= 10^6$)

Looking now at Theorem 3.6, we again still see the inequality holds:

$$\frac{G_*^{1/3}}{\mathrm{Ra}^{1/3}} = .335 \leq \kappa_\eta = 12.837 \leq G^{*1/3} = 799.749.$$

Here, we can take our constant for our lower bound to be approximately 40.

5.4. $\mathbf{Ra} = 2.5 \times 10^6$**.** This simulation ran for a total of .05s of simulated seconds.

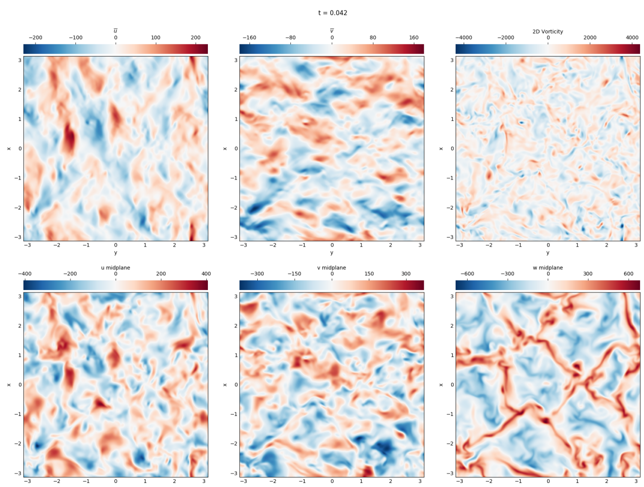


FIGURE 16. Velocity Fields for Ra $= 2.5 \times 10^6$

5.4.1. *Field Comparisons.* For this Rayleigh number, we see more similarity between $\bar{\mathbf{u}}$ and $\mathbf{u}$. Like 5×10^5 and 10^6, we see the areas where the velocity is positive are in around the same area for both the averaged and the normal. Again, we see more pockets of difference in $\mathbf{u}$, as well as more extremes and fractures. We also see stronger vorticies, but not as large of an increase as between 5×10^5 and 10^6.

5.4.2. *Spectra.* We see the same shape as the other Rayleigh numbers again. Like Sec. 5.3.2, we see a more apparent inverse energy cascade than enstrophy cascade. However, we do not see the same level of accuracy as in above, which we again believe is caused by the rate of convergence.

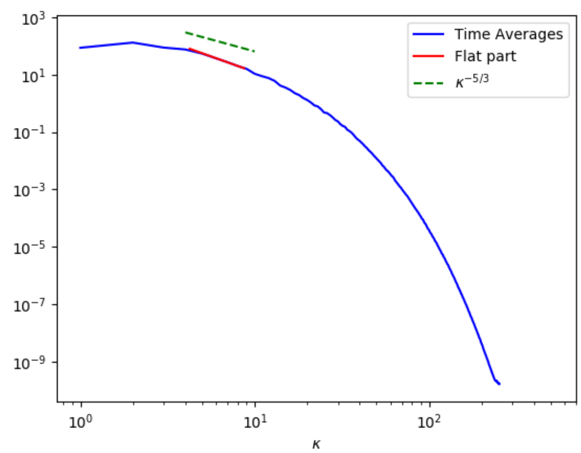


FIGURE 17. Energy Spectrum (Ra $= 2.5 \times 10^6$)

The shear and Reynolds spectrum become rougher than the preceding Rayleigh numbers, but still maintain the same contour. We also see a change in concavity towards the higher wavenumbers, as well as much less decay. This supports our claim that the simulation, in the case of the spectra, did not converge quickly enough.

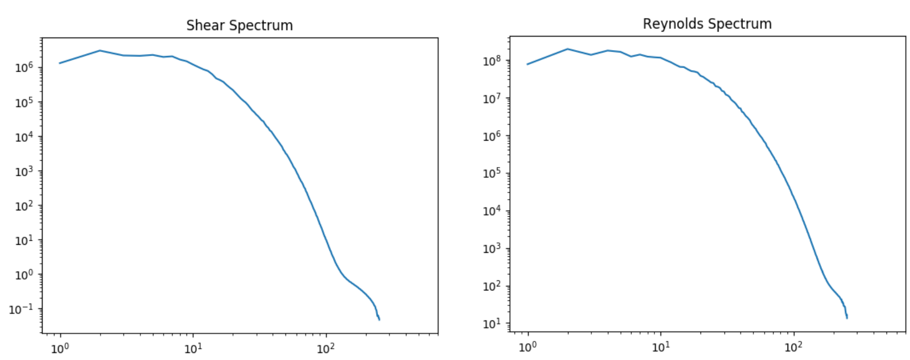


FIGURE 18. Shear and Reynolds Spectra (Ra $= 2.5 \times 10^6$)

5.4.3. *Total Energies.* The results for the total energy of the 3D system are analogous to the other examined wavenumbers.

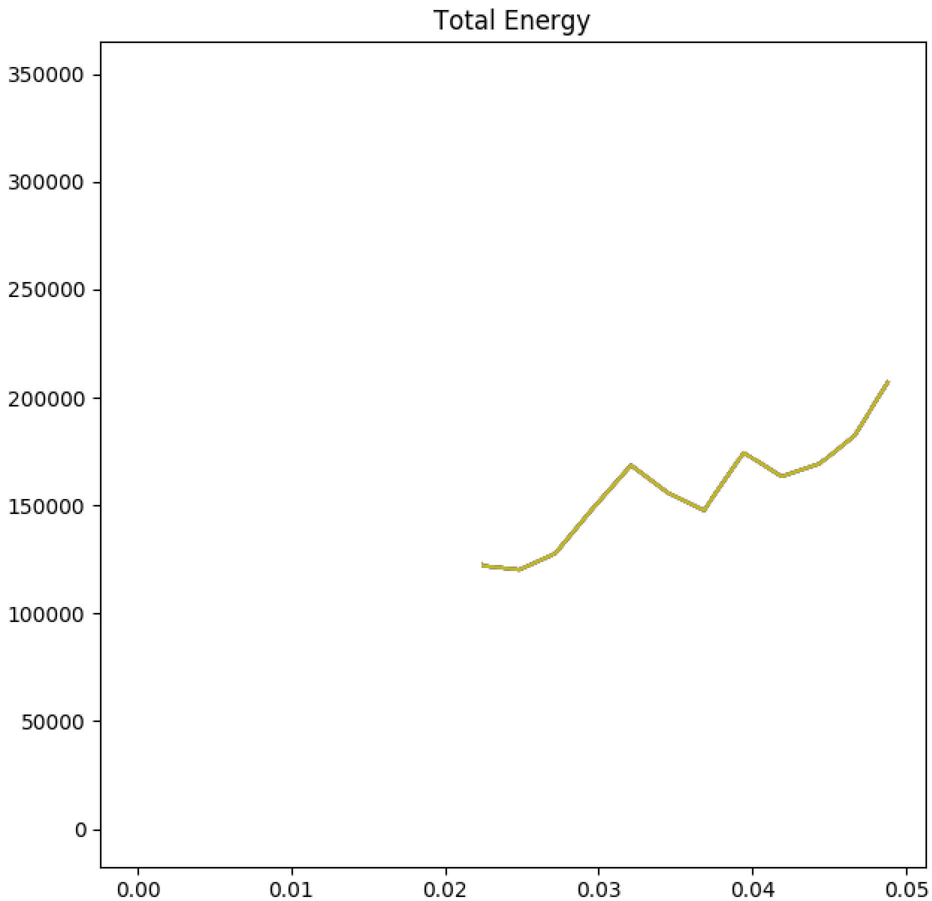


FIGURE 19. Total Energy of 3D System (Ra $= 2.5 \times 10^6$)

Similarly for the shear and Reynolds forces.

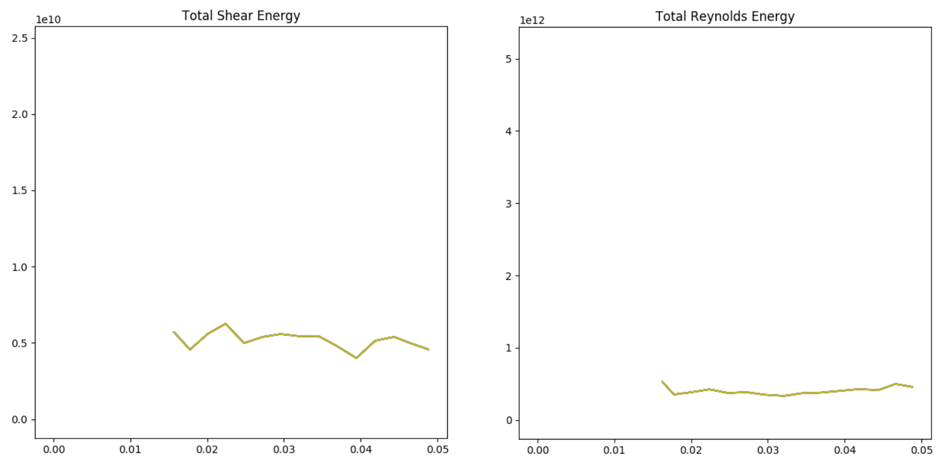


FIGURE 20. Shear and Reynolds Total Energy (Ra $= 2.5 \times 10^6$)

5.4.4. *Computed v. Expected Values.* All constants grew, as expected. Looking now at the pseudo-flux, we see our same expected result, namely nearly constant for $\kappa \leq \kappa_\sigma$. We also see that it is nearly equal to η, as well.

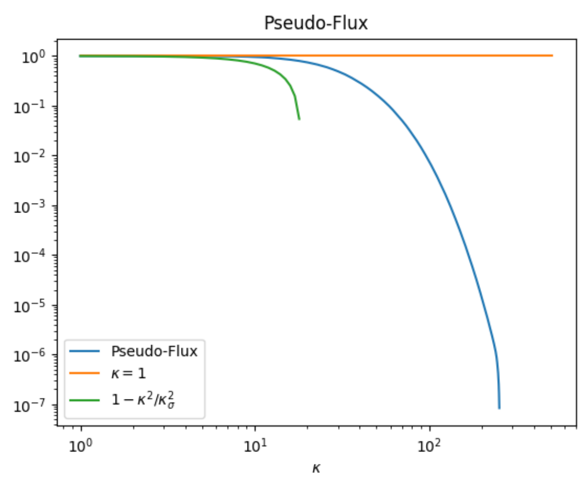


FIGURE 21. Theorem 3.2 Bounds (Ra $= 2.5 \times 10^6$)

Additionally, Theorem 3.6 still holds:

$$\frac{G_*^{1/3}}{\mathrm{Ra}^{1/3}} = .336 \leq \kappa_\eta = 14.143 \leq G^{*1/3} = 1537.624.$$

Here, we can take our constant for the lower bound to be about 45. Unlike the bounds in Figure 3, we see that the effect of this constant must increase as the Rayleigh number increases. This can be explained by $G_*^{1/3}/\mathrm{Ra}^{1/3}$ remaining nearly constant, regardless of Rayleigh number.

Acknowledgments

We thank Jeremy Fischer for his assistance in setting up Dedalus and the scaling of our simulations on Jetstream, which was made possible through the XSEDE Extended Collaborative Support Service (ECSS) program. We also extend an immense amount of gratitude to M.S. Jolly for all of his assistance and support.

References

[1] N. Balci, C. Foias, and M. Jolly. Generalized forces for 2-d turbulence. 2008.

[2] N. Balci, C. Foias, and M. S. Jolly. Vertical averages for benard convection and turbulence. (In preparation).

[3] N. Balci, C. Foias, and M. S. Jolly. 2-D turbulence for forcing in all scales. *J. Math. Pures Appl. (9)*, 94(1):1–32, 2010.

[4] K. J. Burns, G. M. Vasil, D. Oishi, B. P. Lecoanet, and E. Quataert. Dedalus: A flexible framework for spectrally solving differential equations. (In preparation).

[5] H. Castaing and et al. Scaling of hard thermal turbulence in rayleigh-bénard convection. *J. Fluid Mech.*, 204:1–30, 1989.

[6] R. Dascaliuc, C. Foias, and M. S. Jolly. Some specific mathematical constraints on 2D turbulence. *Phys. D*, 237(23):3020–3029, 2008.

[7] C. Foias, M. S. Jolly, and O. P. Manley. Kraichnan turbulence via finite time averages. *Comm. Math. Phys.*, 255(2):329–361, 2005.

[8] C. Foias, M. S. Jolly, O. P. Manley, and R. Rosa. Statistical estimates for the Navier-Stokes equations and the Kraichnan theory of 2-D fully developed turbulence. *J. Statist. Phys.*, 108(3-4):591–645, 2002.

[9] C. Foias, M. S. Jolly, O. P. Manley, R. Rosa, and R. Temam. Kolmogorov theory via finite-time averages. *Phys. D*, 212(3-4):245–270, 2005.

[10] C. Foias, O. P. Manley, and R. Temam. Bounds for the mean dissipation of 2-D enstrophy and 3-D energy in turbulent flows. *Phys. Lett. A*, 174(3):210–215, 1993.

[11] Robert M. Kerr. Is there a 2D cascade in 3D convection? In *Advances in wave interaction and turbulence (South Hadley, MA, 2000)*, volume 283 of *Contemp. Math.*, pages 41–50. Amer. Math. Soc., Providence, RI, 2001.

[12] Craig A. Stewart, Timothy M. Cockerill, Ian Foster, David Hancock, Nirav Merchant, Edwin Skidmore, Daniel Stanzione, James Taylor, Steven Tuecke, George Turner, Matthew Vaughn, and Niall I. Gaffney. Jetstream: A self-provisioned, scalable science and engineering cloud environment. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, XSEDE '15, pages 29:1–29:8, New York, NY, USA, 2015. ACM.

[13] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazelwood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, and Nancy Wilkins-Diehr. Xsede: Accelerating scientific discovery. *Computing in Science and Engineering*, 16(5):62–74, Sept., Oct. 2014.

(A. Isenberg) Courant Institute of Mathematical Sciences, New York University,, New York, NY, USA

E-mail address: `ai939@nyu.edu`

Logic with "Most" and Cardinality Comparisons

Charlotte Raty

July 28, 2017

Abstract

This paper expands upon previous work to create a logic using statements with "most", i.e. *Most x are y* as well as comparisons of the sizes of sets, i.e. *There are more x than y* and *There are at least as many x as y*. We then state the rules of the logic and prove its soundness. Next, we show the logic's completeness by giving an algorithm to construct a model for any finite set of consistent sentences in the logic. We follow this with an example, and conclude by discussing possible future work. [1]

1 Introduction

This paper discusses a logic comprised of statements about set cardinality (*There are more x than y*, written $\exists^{>}(x,y)$ and *There are at least as many x as y*, written $\exists^{\geq}(x,y)$) as well as statements of the form *Most x are y*, written $M(x,y)$. This logic also allows negations of these statements. Such a logic is interesting because we would like to know if any sound conclusions about the sizes of sets or their relations to other sets can be drawn from sets of sentences in this logic. We will provide a set of rules for sound inferences in this logic, and show that this set is complete.

A model $\mathcal{M}$ is a pair $(U, [\![]\!])$, where U is a finite set and $[\![]\!]$ is the interpretation function. The set of all variables used in Γ is the set of atoms P, and $[\![]\!]$ assigns to each $x \in P$ a subset $[\![x]\!]$ of U. We interpret sentences in $\mathcal{M}$ as

$$\begin{array}{ll} \mathcal{M} \models M(x,y) & \text{iff } |[\![x]\!] \cap [\![y]\!]| > \frac{p}{q}|[\![x]\!]| \\ \mathcal{M} \models \exists^{>}(x,y) & \text{iff } |[\![x]\!]| > |[\![y]\!]| \\ \mathcal{M} \models \exists^{\geq}(x,y) & \text{iff } |[\![x]\!]| \geq |[\![y]\!]| \end{array}$$

Where p and q are natural numbers with $0 < p < q$. Note that rather than treating the "most" graph as a special case of a proportionality $\frac{p}{q}$-digraph where $p = 1$ and $q = 2$ (i.e. interpreting "most" to mean "more than half"), we will discuss the general case, where p and q are any natural numbers such that $0 < p < q$.

Here, we give an algorithm for constructing a model for any consistent set of sentences Γ, and prove that the resulting model satisfies Γ.

This paper builds heavily off of a paper by Tri Lai, Jörg Endrullis, and Lawrence Moss that characterizes majority digraphs as graphs with no one-way cycles and proves the completeness of a logic using only statements of the form *Most x are y* by building a model from such a digraph. Here, we tweak this model to encompass a logic with a wider range of statements. We also use several key facts previously proved in that work.

We begin by laying out the rules of inference of this logic system and giving examples of proofs. We also prove the system's soundness. Then, we prove completeness by constructing a model of any consistent set of sentences Γ.

1.1 Rules

This logic uses the following set of rules:

[1]This work is supported by NSF REU Grant #1461061.

$$\frac{\exists^{\geq}(x,y) \quad \exists^{\geq}(y,z)}{\exists^{\geq}(x,z)}\ (\textsc{card-trans}) \qquad \frac{\exists^{>}(x,y)}{\exists^{\geq}(x,y)}\ (\textsc{more-at least})$$

$$\frac{\exists^{>}(x,y) \quad \exists^{\geq}(y,z)}{\exists^{>}(x,z)}\ (\textsc{more-left}) \qquad \frac{\exists^{\geq}(x,y) \quad \exists^{>}(y,z)}{\exists^{>}(x,z)}\ (\textsc{more-right})$$

$$\frac{\neg\exists^{\geq}(x,y)}{\exists^{>}(y,x)}\ (\textsc{negate-at least}) \qquad \frac{\neg\exists^{>}(x,y)}{\exists^{\geq}(y,x)}\ (\textsc{negate-more})$$

$$\frac{M(x,y)}{M(x,x)}\ (\textsc{most-left}) \qquad \frac{M(x,y)}{M(y,y)}\ (\textsc{most-right})$$

$$\frac{M(x,y) \quad \exists^{\geq}(x,y)}{M(y,x)}\ (\textsc{most-at least}) \qquad \frac{\exists^{\geq}(x,y) \quad \exists^{>}(y,x)}{\phi}\ (\textsc{x-more})$$

$$\frac{}{\exists^{\geq}(x,x)}\ (\textsc{at least-axiom}) \qquad \frac{\neg M(x,y) \quad \exists^{\geq}(y,x)}{\neg M(y,x)}\ (\textsc{not-most-at least})$$

$$\frac{\neg M(x,y) \quad M(y,x)}{\exists^{>}(x,y)}\ (\textsc{most-more}) \qquad \frac{M(x,y) \quad \neg M(x,y)}{\phi}\ (\textsc{x-most})$$

$$\frac{\neg M(x,x)}{\neg M(x,y)}\ (\textsc{negate-most-left}) \qquad \frac{\neg M(x,x)}{\neg M(y,x)}\ (\textsc{negate-most-right})$$

$$\frac{\neg M(x,x)}{\exists^{\geq}(y,x)}\ (\textsc{negate-most-card}) \qquad \frac{\neg M(x,x) \quad \exists^{\geq}(x,y)}{\neg M(y,y)}\ (\textsc{negate-most-at least})$$

$$\frac{\neg M(x,x) \quad M(y,y)}{\exists^{>}(y,x)}\ (\textsc{negate-most-more})$$

As well as the infinite scheme:

$$\frac{M(x_1,x_2) \quad \neg M(x_2,x_1) \quad M(x_2,x_3) \quad \neg M(x_3,x_2) \quad \cdots \quad M(x_{n-1},x_n) \quad \neg M(x_n,x_{n-1}) \quad M(x_n,x_1)}{M(x_1,x_n)}$$

Many of the rules come from previous work on logic systems with $\exists^{\geq}$ and $\exists^{>}$ [LEM16] [EM15]. The infinite scheme comes from previous work on M; this rule represents the fact that there can be no one-way cycles in majority digraphs. The (MOST-AT LEAST) and (MOST-MORE) rules is the most significant addition in this logic, as they combine the two kinds of statements ("most" and cardinality judgments) in the system. Additionally, the rules (NEGATE-MOST-LEFT), (NEGATE-MOST-RIGHT), (NEGATE-MOST-CARD), (NEGATE-MOST-AT LEAST), and (NEGATE-MOST-MORE) come from a few subtle points involving sentences with "most" when negation is introduced. If we have $\neg M(x,x)$, this must mean $|[\![x]\!] \cap [\![x]\!]| \leq \frac{p}{q}|[\![x]\!]|$, meaning $|[\![x]\!]| \leq \frac{p}{q}|[\![x]\!]|$. Because we have $0 < p < q$, this can only be true when $|[\![x]\!]| = 0$, meaning $[\![x]\!] = \emptyset$. Note that any time we have $M(x,y)$ where $x \neq y$, both $[\![x]\!]$ and $[\![y]\!]$ are nonempty as a consequence of the semantics of "most".

To return to our discussion of consistency, a consistent set of sentences Γ can now be defined as one from which no proofs using (X-MORE) or (X-MOST) can be derived. Therefore, we want to show that every set of sentences Γ about which it is not true that $\Gamma \vdash \phi$ for all sentences ϕ has a model.

1.2 Example Proofs

The following is an example of a proof in the system. Consider

$$\Gamma = \{ \quad \begin{array}{l} M(x,z), \\ \exists^{\geq}(x,y), \\ \neg\exists^{\geq}(z,y) \quad \} \end{array}$$

We'll now show a proof of $M(z,x)$ using proof trees, all of whose leaves must be sentences in Γ:

$$\dfrac{M(x,z) \qquad \dfrac{\dfrac{\exists^{\geq}(x,y) \qquad \dfrac{\neg\exists^{\geq}(z,y)}{\exists^{>}(y,z)}\,\text{(NEGATE-AT LEAST)}}{\exists^{>}(x,z)}\,\text{(MORE-RIGHT)}}{\exists^{\geq}(x,z)}\,\text{(MORE-AT LEAST)}}{M(z,x)}\,\text{(MOST-AT LEAST)}$$

Following is a second example of a proof in this system. This example of a proof in this system uses (X-MOST). This is an example of an inconsistent set of sentences; we'll discuss consistent and inconsistent sets of sentences further later on in the construction. We'll have Γ defined as follows:

$$\Gamma = \{ \quad \begin{array}{l} \neg M(z,z), \\ M(x,y), \\ \exists^{\geq}(z,x) \qquad \} \end{array}$$

We'll now show a proof of $\exists^{>}(x,y)$

$$\dfrac{M(x,y) \qquad \dfrac{\dfrac{\neg M(z,z) \quad \exists^{\geq}(z,x)}{\neg M(x,x)}\,\text{(NEGATE-MOST-AT LEAST)}}{\neg M(x,y)}\,\text{(NEGATE-MOST-LEFT)}}{\exists^{>}(x,y)}\,\text{(X-MOST)}$$

1.3 Soundness

We'll now prove soundness of the rules of this logical system. Proving the soundness of the system comes down to proving the soundness of each rule. We'll begin with the definition of soundness:

Definition 1.1. A proof system is *sound* if for all ϕ such that $\Gamma \vdash \phi$, we have $\Gamma \models \phi$. That is, if we can prove ϕ from Γ using the rules of the proof system, there is no possible model such that all sentences in Γ are true and ϕ is false.

Many of the above rules have been previously proven to be sound [LEM16] [EM15]. So we only need to prove the soundness of our new additions to the system: those rules that combine statements involving "most" with statements about set cardinality.

Lemma 1.2. The rule (MOST-AT LEAST) is sound.

Proof. Fix a model $\mathcal{M}$, and assume that $\mathcal{M} \models M(x,y)$ and $\mathcal{M} \models \exists^{\geq}(x,y)$. We want to show that $\mathcal{M} \models M(y,x)$, meaning that our (MORE-AT LEAST) rule is sound. From $\mathcal{M} \models \exists^{\geq}(x,y)$, we know that $|[\![x]\!]| \geq |[\![y]\!]|$. Multiplying each side of this inequality by $\frac{p}{q}$, we get $\frac{p}{q}|[\![x]\!]| \geq \frac{p}{q}|[\![y]\!]|$. Similarly, from $\mathcal{M} \models M(x,y)$, we know that $|[\![x]\!] \cap [\![y]\!]| > \frac{p}{q}|[\![x]\!]|$. Therefore, because $|[\![x]\!] \cap [\![y]\!]| > \frac{p}{q}|[\![x]\!]|$ and $\frac{p}{q}|[\![x]\!]| \geq \frac{p}{q}|[\![y]\!]|$, we have $|[\![x]\!] \cap [\![y]\!]| > \frac{p}{q}|[\![y]\!]|$, which means that $\mathcal{M} \models M(x,y)$. $\square$

Lemma 1.3. The rule (MOST-MORE) is sound.

Proof. This proof will be similar to that of Lemma 1.3. Fix a model $\mathcal{M}$, adn assume that $\Gamma \models \neg M(x,y)$ and $\Gamma \models M(y,x)$. This means we have $\frac{p}{q}|[\![x]\!]| \geq |[\![x]\!] \cap [\![y]\!]|$ and $|[\![x]\!] \cap [\![y]\!]| > \frac{p}{q}|[\![y]\!]|$. Therefore, we have $\frac{p}{q}|[\![x]\!]| > \frac{p}{q}|[\![y]\!]|$. When we multiply both sides of the inequality by $\frac{q}{p}$, we get $|[\![x]\!]| > |[\![y]\!]|$, meaning that $\mathcal{M} \models \exists^{>}(x,y)$.

$\square$

Lemma 1.4. The rule (NEGATE-MOST-CARD) is sound.

Proof. Fix a model $\mathcal{M}$, and assume we have $\mathcal{M} \models \neg M(x,x)$. This means we have $\frac{p}{q}|[\![x]\!]| \geq |[\![x]\!] \cap [\![x]\!]|$, so $\frac{p}{q}|[\![x]\!]| \geq |[\![x]\!]|$. As $p < q$, this is only possible when $|[\![x]\!]| = 0$. For any atom y we have $|[\![y]\!]| \geq 0$. Therefore, we have $|[\![y]\!]| \geq |[\![x]\!]|$, so $\mathcal{M} \models \exists^{\geq}(y,x)$. $\square$

Lemma 1.5. The rule (NEGATE-MOST-AT LEAST) is sound.

Proof. Fix a model $\mathcal{M}$, and assume we have $\mathcal{M} \models \neg M(x,x)$ and $\mathcal{M} \models \exists^{\geq}(x,y)$. Because we have $\mathcal{M} \models \neg M(x,x)$, this means $\frac{p}{q}|\llbracket x \rrbracket| \geq |\llbracket x \rrbracket \cap \llbracket x \rrbracket|$, so $\frac{p}{q}|\llbracket x \rrbracket| \geq |\llbracket x \rrbracket|$. As in Lemma 1.4, this means that $|\llbracket x \rrbracket| = 0$. Because we have $\mathcal{M} \models \exists^{\geq}(x,y)$, we know that $|\llbracket x \rrbracket| \geq |\llbracket y \rrbracket|$, meaning $0 \geq |\llbracket y \rrbracket|$. Therefore, we must have $|\llbracket y \rrbracket| = 0$. This means that $\frac{p}{q}|\llbracket y \rrbracket| = |\llbracket y \rrbracket|$. We also know that $\llbracket y \rrbracket = \llbracket y \rrbracket \cap \llbracket y \rrbracket$, so using substitution we get $\frac{p}{q}|\llbracket y \rrbracket| = |\llbracket y \rrbracket \cap \llbracket y \rrbracket|$. This means that $\frac{p}{q}|\llbracket y \rrbracket| \geq |\llbracket y \rrbracket \cap \llbracket y \rrbracket|$, so we have $\mathcal{M} \models \neg M(y,y)$. □

Lemma 1.6. The rule (NEGATE-MOST-MORE) is sound.

Proof. Fix a model $\mathcal{M}$, and assume we have $\mathcal{M} \models \neg M(x,x)$ and $\mathcal{M} \models M(y,y)$. Because we have $\mathcal{M} \models \neg M(x,x)$, this means $\frac{p}{q}|\llbracket x \rrbracket| \geq |\llbracket x \rrbracket \cap \llbracket x \rrbracket|$, so $\frac{p}{q}|\llbracket x \rrbracket| \geq |\llbracket x \rrbracket|$. As in Lemma 1.4, this means that $|\llbracket x \rrbracket| = 0$. Because we have $\mathcal{M} \models M(y,y)$, this means $\frac{p}{q}|\llbracket y \rrbracket| < |\llbracket y \rrbracket \cap \llbracket y \rrbracket|$, so $\frac{p}{q}|\llbracket y \rrbracket| < |\llbracket y \rrbracket|$, meaning that we must have $|\llbracket y \rrbracket| > 0$. Therefore, $|\llbracket y \rrbracket| > |\llbracket x \rrbracket|$, so $\mathcal{M} \models \exists^{>}(y,x)$ □

Theorem 1.7. The proof system of this logic is sound.

Proof. We already know that all rules involving only "most" statements or only set cardinality statements are sound. The rules (MOST-AT LEAST), (MOST-MORE), (NEGATE-MOST-CARD), (NEGATE-MOST-AT LEAST), and (NEGATE-MOST-MORE) are sound by Lemmas 1.2 through 1.6. Because all of the individual rules of the system are sound, proving the system itself is sound is a matter of induction on the height of a proof tree, which we omit here. Therefore, the proof system is sound. □

2 Completeness

To prove that the system is complete, we'll want to work with a consistent set of sentences. It will be useful to have the definitions of both a consistent set of sentences and completeness.

Definition 2.1. A *consistent* set of sentences is a set of sentences in the logic from which no contradictions can be proven.

As an example, the set of sentences $\{\neg M(x,y), \exists^{>}(y,z)\}$ is consistent. The set $\{\neg\exists^{>}(x,y), \neg\exists^{\geq}(y,x)\}$ is not. The empty set $\{\}$ is also consistent.

Definition 2.2. A logical system is complete if, for all ϕ such that some set of sentences Γ models ϕ, ϕ is provable using a proof tree whose leaves are all either axioms or sentences in Γ and whose non-leaf nodes all match some rule in the proof system.

With a logic system using negation, like this one, proving completeness is equivalent to proving that any consistent set sentences Γ has a model. Now, given such a Γ, we construct a model using the rules of the system to show that the system is complete.

This construction borrows heavily from the construction used in [LEM16], and we present many key facts from that paper without proof here. The construction in [LEM16] builds a model from a proportionality $\frac{p}{q}$-digraph. Here, we will construct several graphs, one to represent the "most" relation as well as multiple others to represent cardinality comparisons. The points of these graphs are the set of variables used in Γ and the edges represent the relations between them that are provable from Γ. We will then construct a model using these graphs, following much of the construction from [LEM16].

2.1 Preliminary Facts

Here we state several definitions and facts used in [LEM16] and discuss how they are used. Because they were all previously proved, we omit the proofs.

Definition 2.3. For a set Γ of sentences, the *proportionality* $\frac{p}{q}$*-digraph* corresponding to Γ is the digraph whose vertices are the atoms of Γ such that $u \longrightarrow v$ if and only if $\Gamma \vdash M(u,v)$.

Proposition 2.4. For a consistent set of sentences Γ, the proportionality $\frac{p}{q}$-digraph corresponding to Γ has no one-way cycles.

Because we are only concerned with creating models of consistent sets, we know that the majority digraph we use in our construction has this property. This corresponds to the infinite scheme.

Additionally, we need the concept of a private intersection:

Definition 2.5. For a set $\{A_1, A_2, ..., A_n\}$ of sets, the *private intersection* of two sets A_i and A_j, written $A_i \sqcap A_j$ is the set $(A_i \cap A_j) \setminus \bigcup_{k \neq i, k \neq j} A_k$.

As an example, if we have $A_1 = \{1, 4, 5, 6\}$, $A_2 = \{2, 3, 4, 5, 7, 8\}$, and $A_3 = \{1, 4, 6, 7, 8\}$, $A_1 \cap A_3 = \{1, 4, 6\}$, but $A_1 \sqcap A_3 = \{1, 6\}$.

The construction depends upon the ability to construct sets with certain properties, as detailed in the following lemma:

Lemma 2.6. For all n, there are sets B_1, B_2,..., B_n such that $|B_i| = 2^{n-1}$, and for $i \neq q$ we have $|B_i \cap B_j| = p^2 q^{n-2} = \frac{1}{2}|B_i|$ and $|B_i \sqcap B_j| = p^2(q-p)^{n-2}$.

We also need to use the following property of natural numbers from [LEM16]:

$$\frac{p + ar}{q + ar + s} > \frac{p}{q} \text{ iff } a > \frac{ps}{r(q-p)} \tag{1}$$

Finally, we will need to use the following definition when discussing the digraphs we will use in order to construct a model of Γ:

Definition 2.7. A *topological sort* of a digraph G is an ordering of the graph's vertices such that for all $u, v \in G$, if $u \longrightarrow v$, then u comes before v in the ordering. Note that not every digraph has a topological sort; only those with no directed cycles do.

2.2 Graphs Representing Provability from a Consistent Set of Sentences

For our construction, we need to create five digraphs based on the sentences in Γ. Each of these graphs will have the set of variables in Γ as points.

We begin by constructing four graphs based on provability from Γ. These graphs are

1. G_M, where $x \longrightarrow y$ is an edge in G_M if and only if $\Gamma \vdash M(x, y)$.

2. $G_{\neg M}$, where $x \longrightarrow y$ is an edge in $G_{\neg M}$ if and only if $\Gamma \vdash \neg M(x, y)$.

3. $G_>$, where $x \longrightarrow y$ is an edge in $G_>$ if and only if $\Gamma \vdash \exists^>(y, x)$.

4. $G_\geq$, where $x \longrightarrow y$ is an edge in $G_\geq$ if and only if $\Gamma \vdash \exists^\geq(y, x)$.

We can now prove several properties of these graphs using the rules of the proof system.

Lemma 2.8. $G_>$ has a topological sort.

Proof. To show that $G_>$ has a topological sort, we need to show that it has no directed cycles. We'll do this by contradiction. Suppose that $G_>$ has some cycle $x_1 \longrightarrow x_2 \longrightarrow ... \longrightarrow x_n \longrightarrow x_1$. By our construction of these graphs, this means that $\Gamma \vdash \exists^>(x_2, x_1)$, $\Gamma \vdash \exists^>(x_3, x_2)$, ..., $\Gamma \vdash \exists^>(x_n, x_{n-1})$, $\Gamma \vdash \exists^>(x_1, x_n)$. Then, we can have the following proof tree:

$$\dfrac{\dfrac{\exists^>(x_n, x_{n-1})}{\exists^\geq(x_n, x_{n-1})}\,(\textsc{more-at least}) \qquad \exists^>(x_{n-1}, x_{n-2})}{}\,(\textsc{more-right})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\exists^>(x_n, x_2)}\,(\textsc{more-right})}{\exists^\geq(x_n, x_2)}\,(\textsc{more-at least}) \qquad \exists^>(x_2, x_1)}{\exists^>(x_n, x_1)}\,(\textsc{more-right})}{\exists^\geq(x_n, x_1)}\,(\textsc{more-at least}) \qquad \exists^>(x_1, x_n)}{\phi}\,(\textsc{x-mori}$$

However, this is a derivation from Γ using X-MORE, which contradicts the fact that Γ is a consistent set of sentences. Therefore, $G_>$ has no directed cycles, and thus has a topological sort. $\square$

Lemma 2.9. The relation $\longleftrightarrow$ in $G_{\geq}$ is an equivalence relation, with the equivalence class of a vertex x being $[x] = \{y : x \longrightarrow y \text{ and } y \longrightarrow x\}$.

Proof. We do have $\exists^{\geq}(x,x)$ for all x because $|[\![x]\!]| = |[\![x]\!]|$, so $|[\![x]\!]| \geq |[\![x]\!]|$.

If we have $x \longleftrightarrow y$ as an edge in $G_{\geq}$, we trivially also have $y \longleftrightarrow x$ as an edge.

Finally, suppose we have $x \longleftrightarrow y$ and $y \longleftrightarrow z$. This means we have $\Gamma \vdash \exists^{\geq}(x,y)$ and $\Gamma \vdash \exists^{\geq}(y,z)$, so we have the proof tree

$$\frac{\exists^{\geq}(x,y) \quad \exists^{\geq}(y,z)}{\exists^{\geq}(x,z)} \ (\text{CARD-TRANS})$$

so $\Gamma \vdash \exists^{\geq}(x,z)$. Similarly, we have $\Gamma \vdash \exists^{\geq}(z,y)$ and $\Gamma \vdash \exists^{\geq}(y,x)$, so we have the proof tree

$$\frac{\exists^{\geq}(y,x) \quad \exists^{\geq}(z,x)}{\exists^{\geq}(z,y)} \ (\text{CARD-TRANS})$$

so $\Gamma \vdash \exists^{\geq}(x,z)$. This means we have $x \longleftrightarrow z$ as an edge in $G_{\geq}$. Thus, $\longleftrightarrow$ in $G_{\geq}$ is an equivalence relation. $\square$

Now that we know $\longleftrightarrow$ in $G_{\geq}$ is an equivalence relation, we'll look at the quotient graph of $G_{\geq}$ by $\longleftrightarrow$.

Definition 2.10. A *quotient graph* G of a graph G_0 is a graph whose vertices are the equivalence classes of G_0 under some equivalence relation $\equiv$. In order to define the edge relation on the set of equivalence classes, we must have the following condition: if $v \longrightarrow w$ and $v \equiv v'$ and $w \equiv w'$, then $v' \longrightarrow w'$. If this condition holds for all vertices of G_0, we can then construct G and define its edges: for every edge $v \longrightarrow w$ of G_0 such that $[v] \neq [w]$, G has an edge $[v] \longrightarrow [w]$.

In order to use the quotient graph of $G_{\geq}$ under $\longleftrightarrow$, we must first show that $G_{\geq}$ has a quotient graph.

Proposition 2.11. $G_{\geq}$ has a quotient graph.

Proof. To show that $G_{\geq}$ has a quotient graph under the equivalence relation $\longleftrightarrow$, we must show that it has the property mentioned in the definition of a quotient graph: that for all edges $v \longrightarrow w$ of $G_{\geq}$, and for all v' and w' where $v \equiv v'$ and $w \equiv w'$, there is an edge $v' \longrightarrow w'$ of $G_{\geq}$. Let v, w, v', and w' be vertices of $G_{\geq}$, and let $v \to w$, $v \equiv v'$, and $w \equiv w'$. Since we are interested in the equivalence relation $\longleftrightarrow$, this means we have $v \longleftrightarrow v'$ and $w \longleftrightarrow w'$ in $G_{\geq}$. This means we have $\Gamma \vdash \exists^{\geq}(v,w)$, $\Gamma \vdash \exists^{\geq}(v,v')$, $\Gamma \vdash \exists^{\geq}(v',v)$, $\Gamma \vdash \exists^{\geq}(w,w')$, and $\Gamma \vdash \exists^{\geq}(w',w)$. This gives us the proof tree

$$\frac{\exists^{\geq}(v',v) \qquad \dfrac{\exists^{\geq}(v,w) \quad \exists^{\geq}(w,w')}{\exists^{\geq}(v,w')} \ (\text{CARD-TRANS})}{\exists^{\geq}(v',w')} \ (\text{CARD-TRANS})$$

Therefore, we have $\Gamma \vdash \exists^{\geq}(v',w')$, meaning there is an edge $v' \longrightarrow w'$ of $G_{\geq}$. Therefore, $G_{\geq}$ has a quotient graph. $\square$

We'll refer to the quotient graph of $G_{\geq}$ as G.

Lemma 2.12. There is a topological sort of G.

Proof. We'll show that there is a topological sort of G by showing that G is acyclic. We'll prove this by contradiction, starting by assuming that G has a cycle (other than a self-loop, which would be a trivial cycle). Consider a nontrivial cycle $[x_1] \longrightarrow [x_2] \longrightarrow ... \longrightarrow [x_n] \longrightarrow [x_1]$, where $[x_1], ..., [x_n]$ are all different equivalence classes. We know that G has the property of transitivity by (CARD-TRANS). Thus, there must also be edges $[x_1] \longrightarrow [x_3]$, $[x_1] \longrightarrow [x_4]$, ..., $[x_1] \longrightarrow [x_n]$. However, we then have $[x_1] \longrightarrow [x_n]$ and $[x_n] \longrightarrow [x_1]$, meaning that $[x_1]$ and $[x_n]$ are actually the same equivalence class. This creates a contradiction. Therefore, G has no cycles and thus has a topological sort. $\square$

The next two theorems are the most important results of this section. With them, we are able to find an ordering of the atoms of Γ such that the necessary conditions for the construction in [LEM16], as well as other useful conditions, hold. When discussing "expanding" equivalence classes, we mean replacing the equivalence class $[x]$ with an enumeration of all elements in $[x]$, in any order.

Theorem 2.13. A topological sort of G, with the equivalence classes expanded, is also a topological sort of $G_>$.

Proof. We know that $G_>$ and G both have topological sorts by Lemmas 2.7 and 2.11, respectively. Now, consider a topological sort $[x_1], [x_2], ..., [x_n]$ of G. To show that it is a topological sort of $G_>$, let $x \longrightarrow y$ be an edge of $G_>$, meaning $\Gamma \vdash \exists^>(y, x)$. We will show that x comes before y in this ordering.

We cannot have $\Gamma \vdash \exists^\geq(x, y)$; if we did, we would be able to have a derivation from Γ using (X-MORE), which contradicts the consistency of Γ. Thus, we cannot have $x \longleftrightarrow y$ as an edge in $G_\geq$, meaning x and y cannot be in the same equivalence class. Now, by the rule (MORE-AT LEAST) we know that because we have $\Gamma \vdash \exists^>(y, x)$, we must have $\Gamma \vdash \exists^\geq(y, x)$. Therefore, in the topological sort of G, we know x must come before y. Therefore, this is a topological sort of $G_>$. $\square$

Theorem 2.14. A topological sort of G, with the equivalence classes expanded, has the property that if $x \longrightarrow y$ is an edge of G_M and $y \longrightarrow x$ is an edge of $G_{\neg M}$, x comes before y.

Proof. First, by the infinite scheme involving "most", we know that we can have such an ordering of vertices because G_M contains no one-way cycles. Now, suppose we have such an x and y, where $x \longrightarrow y$ is an edge of G_M and $y \longrightarrow x$ is an edge of $G_{\neg M}$. This means $\Gamma \vdash M(x, y)$ and $\Gamma \vdash \neg M(y, x)$. Now, we know that x and y cannot be in the same equivalence class of G. We'll show this by contradiction; suppose they are. Then, we have $\Gamma \vdash \exists^\geq(x, y)$ and $\Gamma \vdash \exists^\geq(y, x)$. This gives us the proof tree

$$\dfrac{\dfrac{M(x,y) \quad \exists^\geq(x,y)}{M(y,x)}\ (\text{MOST-AT LEAST}) \qquad \neg M(y,x)}{\phi}\ (\text{X-MOST})$$

This contradicts the consistency of Γ. Therefore, x and y cannot be in the same equivalence class of G.

We'll now show that the equivalence class of x must come before that of y, meaning x comes before y in the ordering. We have the following proof tree:

$$\dfrac{M(x,y) \quad \neg M(y,x)}{\exists^>(y,x)}\ (\text{MOST-MORE})$$

Therefore, x comes before y in any topological sort of $G_>$. Because we know that a topological sort of G is a topological sort of $G_>$ from Theorem 2.12, this completes the proof. $\square$

We also want one more fact about a topological sort of G for our construction.

Proposition 2.15. Let there be at least one atom x in a consistent set of sentences Γ such that $\Gamma \vdash \neg M(x, x)$. Let $[x_0], [x_1], ..., [x_k]$ be a topological sort of the equivalence classes of G corresponding to this Γ. Then, we have $\Gamma \vdash \neg M(y, y)$ if and only if $y \in [x_0]$.

Proof. We'll start in the forward direction by assuming $\Gamma \vdash \neg M(y, y)$ and showing that $y \in [x_0]$. It will be useful to first show that all atoms y such that $\Gamma \vdash \neg M(y, y)$ are in the same equivalence class, and then to show that $y \in [x_0]$.

First, we want to show that two arbitrary atoms y_1 and y_2 such that $\Gamma \vdash \neg M(y_1, y_1)$ and $\Gamma \vdash \neg M(y_2, y_2)$ are in the same equivalence class. By applying the rule (NEGATE-MOST-CARD) to $\neg M(y_1, y_1)$, we get $\exists^\geq(y_2, y_1)$. Similarly, by applying the same rule to $\neg M(y_2, y_2)$, we get $\exists^\geq(y_1, y_2)$. We then have $\Gamma \vdash \exists^\geq(y_1, y_2)$ and $\Gamma \vdash \exists^\geq(y_2, y_1)$, so y_1 and y_2 are in the same equivalence class.

Now we can show that $y \in [x_0]$. By the rule (NEGATE-MOST-CARD), we have $\Gamma \vdash \exists^\geq(z, y)$ for all atoms z. However, now consider a z such that $\Gamma \vdash M(z, z)$. By the rule (NEGATE-MOST-MORE), this gives us $\exists^{>}(z,y)$. Because this is a consistent set, this means we cannot have $\exists^{\geq}(y,z)$ (as such a sentence would allow us to use (X-MORE)), so z will be in a different equivalence class from y. This equivalence class will come after that of y, as we will have $\exists^{\geq}(z,y)$ but not $\exists^{\geq}(y,z)$. Additionally, we showed above that all y such that $\Gamma \vdash \neg M(y,y)$ are in the same equivalence class, so that means that this equivalence class must come before any other equivalence classes, meaning that this equivalence class is $[x_0]$. Therefore, if $\Gamma \vdash \neg M(y,y)$, then $y \in [x_0]$, completing this direction of the proof.

We'll now prove the other direction. Suppose we have an atom $y \in [x_0]$. We want to show that $\Gamma \vdash \neg M(y,y)$. We already know we must have at least one atom x such that $\Gamma \vdash \neg M(x,x)$, and we know that this atom must be in $[x_0]$ by the first part of this proof. Because x and y are in the same equivalence class, we know that $\Gamma \vdash \exists^{\geq}(x,y)$ and $\Gamma \vdash \exists^{\geq}(x,y)$. Then, applying the rule (NEGATE-MOST-AT LEAST) to $\neg M(x,x)$ and $\exists^{\geq}(x,y)$, we get $\neg M(y,y)$. Therefore, $\Gamma \vdash \neg M(y,y)$. □

2.3 Construction

Theorem 2.16. The above set of rules for a logic system with M, $\exists^{>}$, and $\exists^{\geq}$ with negation is complete.

Proof. Our construction closely follows that of [LEM16]. We also use the equivalence classes of G to construct our model. We begin our construction with a topological sort of G with the equivalence classes expanded. The first thing we must do is handle any sentences of the form $\neg M(x,x)$ provable from Γ (or, referring to our graphs, any vertices with edges to themselves $G_{\neg M}$). We have two cases: $G_{\neg M}$ either has no vertex with an edge to itself, or it has at least one such vertex. In the case that $G_{\neg M}$ has no such vertices, we will continue with the construction using the original the topological sort of G with the equivalence classes expanded.

In the case that $G_{\neg M}$ has b such vertices, with $b > 0$, we will use the result from Proposition 2.15. From this result, we know that all atoms x such that $\Gamma \vdash \neg M(x,x)$ are in the first equivalence class of G, which we will call $[x_{\alpha_0}]$. We also know that these are the only such atoms in that equivalence class. Therefore, for every $x \in [x_{\alpha_0}]$, we set $[\![x]\!] = \emptyset$ in our model. Now, we have constructed the part of the model concerning all such x, and we will continue the rest of the construction using the topological sort of the equivalence classes of G with $[x_{\alpha_0}]$ removed as our topological sort.

For convenience, we will let n denote the number of distinct atoms in the ordering and relabel the atoms in this ordering $x_1, x_2, ..., x_n$. We also need to choose natural numbers a and m large enough such that $(q-p)a > q$ and $mp^2(q-p)^{n-2} > apn$.

We first construct these sets $B_1, ..., B_n$ as described in Lemma 2.5, then take m copies of all points in all sets to get sets $[\![x_1]\!], [\![x_2]\!], ..., [\![x_n]\!]$. Now we have $|[\![x_i]\!]| = mpq^{n-1}$. For $i \neq j$, we also have $|[\![x_i]\!] \cap [\![x_j]\!]| = mp^2q^{n-2}$ and $|[\![x_i]\!] \sqcap [\![x_j]\!]| > apn$. Next, we simultaneously add the same apn points to all $[\![x_i]\!]$. Note that this does not change the private intersections of any sets, but it increases the size of each set and increases the size of the intersection of all sets. We now have $|[\![x_i]\!] \cap [\![x_j]\!]| = mp^2q^{n-2} + apn$ for $i \neq j$ and $|[\![x_i]\!]| = mpq^{n-1} + apn$.

2.3.1 Modeling Cardinality Relationships

Now, let $[x_{\alpha_1}], ..., [x_{\alpha_k}]$ be the equivalence classes of G, listed as in the ordering we are using for this construction. Let S_i for $1 \leq i \leq k$ be the set of all $[\![x_i]\!]$s where $x_i \in [x_{\alpha_i}]$. We now add qi fresh points to every set in S_i, for $1 \leq i \leq k$. This does not change the size of any intersections of sets, but it increases the size of a set in the ith equivalence class to $mpq^{n-1} + apn + qi$. This also means that two sets in the same equivalence class have the same number of points.

At this point, we are done adding new points to any of the sets. All of the sets' sizes are fixed, and now all statements of the form $\exists^{\geq}(x,y)$, $\neg\exists^{\geq}(x,y)$, $\exists^{>}(x,y)$, and $\neg\exists^{>}(x,y)$ that are in Γ are modeled by our construction. By the rule (NEGATE-MORE), a sentence of the form $\neg\exists^{\geq}(x,y)$ can be expressed as $\exists^{>}(y,x)$, and by (NEGATE-AT LEAST) a sentence of the form $\neg\exists^{>}(x,y)$ can be expressed as $\exists^{\geq}(y,x)$. Therefore, it is sufficient for us to show that this model now satisfies sentences of the form $\exists^{\geq}(x,y)$ or $\exists^{>}(x,y)$.

We'll first look at sentences of the form $\exists^{>}(x,y)$. Suppose $\exists^{>}(x,y) \in \Gamma$. Here, we know that x and y are in different equivalence classes, and that x comes after y in the ordering we are using. Therefore, $[\![y]\!]$ is in S_j and $[\![x]\!]$ is in S_h, where $1 \leq j < h \leq k$. This means that the size of the set representing y is $mpq^{n-1} + apn + qj$ and the size of the set representing y is $mpq^{n-1} + apn + qh$. Because $j < h$, the set representing x is larger than the set representing j, so we have $\Gamma \models \exists^{>}(x,y)$.

We'll now look at sentences of the form $\exists^{\geq}(x,y)$. If we do not also have $\Gamma \vdash \exists^{\geq}(y,x)$, then x and y are not in the same equivalence class, and we have a case similar to the one above: $[\![y]\!]$ is in S_j and $[\![x]\!]$ is in S_h, where $j < h$, so $\Gamma \models \exists^{>}(x,y)$. If we do have $\Gamma \vdash \exists^{\geq}(y,x)$, $[\![x]\!] = [\![y]\!]$ and both are in S_j; therefore, they both have size $mpq^{n-1} + apn + qj$, so $\Gamma \models \exists^{\geq}(x,y)$ and $\Gamma \models \exists^{\geq}(y,x)$. Therefore, we have constructed a model where all statements about set cardinality provable from Γ hold.

2.4 Modeling "Most" Relationships

Next, we need to adjust the private intersections of the sets so that all sentences of the form $M(x,y)$ and $\neg M(x,y)$ that are in Γ are modeled. Note that, as our construction currently stands, for any two sets $[\![x_i]\!]$ and $[\![x_j]\!]$, where $i < j$, we model $M(x_i, x_j)$ and $M(x_j, x_i)$. Therefore, we only need to adjust the sets when this is not the case. Note that, by Theorem 2.13, we will never have $\neg M(x_i, x_j) \in \Gamma$ and $M(x_j, x_i) \in \Gamma$ when $i < j$. At this point we are only concerned with modeling sentences *in* Γ; because our system is sound, a model of these sentences will also be a model of all sentences provable from Γ.

There are 3 possible cases: the case in which we have $M(x_j, x_i) \in \Gamma$, that in which we have $M(x_i, x_j) \in \Gamma$ and $\neg M(x_j, x_i) \in \Gamma$, and that in which we have $\neg M(x_i, x_j) \in \Gamma$ and $\neg M(x_j, x_i) \in \Gamma$.

We start with the case where $M(x_j, x_i) \in \Gamma$. As previously mentioned, we already model this case because our model currently has $M(x_j, x_i)$ and $M(x_i, x_j)$. Additionally, we know that we cannot have $\neg M(x_i, x_j)$ when we have $M(x_j, x_i)$. Therefore, we have no further changes to make to the private intersection of $[\![x_i]\!]$ and $[\![x_j]\!]$ in this case. We prove that this indeed models $M(x_i, x_j)$ and $M(x_j, x_i)$ using equation (1), and substituting pn for r and qn for s:

$$\frac{|[\![x_i]\!] \cap [\![x_j]\!]|}{|[\![x_i]\!]|} = \frac{mp^2q^{n-2} + apn}{mpq^{n-1} + apn + qi} \geq \frac{mp^2q^{n-2} + apn}{mpq^{n-1} + apn + qn} > \frac{p}{q} \tag{2}$$

This shows that more than half of the elements of $[\![x_i]\!]$ are in the intersection $[\![x_i]\!] \cap [\![x_j]\!]$. A similar equation shows the same for $[\![x_j]\!]$. Thus, we model $M(x_i, x_j)$ and $M(x_j, x_i)$.

We'll next make changes to the model in the case where $\neg M(x_i, x_j)$ and $\neg M(x_j, x_i)$. In this case, the private intersection of $[\![x_i]\!]$ and $[\![x_j]\!]$ has size $mp^2(q-p)^{n-2}$, which we will call z. We remove all z points from the private intersection and return them as separate copies to $[\![x_i]\!]$ and $[\![x_j]\!]$, decreasing the size of $[\![x_i]\!] \cap [\![x_j]\!]$ by z. We know that $apn - z < 0$, so we have

$$\frac{|[\![x_i]\!] \cap [\![x_j]\!]|}{|[\![x_i]\!]|} = \frac{mp^2q^{n-2} + apn - z}{mpq^{n-1} + apn + qi} < \frac{mp^2q^{n-2}}{mpq^{n-1}} = \frac{p}{q}. \tag{3}$$

We have a similar equation to show that not more than $\frac{p}{q}$ of the elements of $[\![x_j]\!]$ are in the intersection $[\![x_i]\!] \cap [\![x_j]\!]$ now as well. Therefore, we now model $\neg M(x_i, x_j)$ and $\neg M(x_j, x_i)$. Not also that these changes do not affect intersections involving any of the other sets, so it does not change any "most" relationships between sets other than the two involved.

In the third and final case, we have $M(x_i, x_j)$ and $\neg M(x_j, x_i)$. Note that x_i and x_j must be in different equivalence classes in this case; by the rule (MOST-MORE), we can prove that $\exists^{>}(x_j, x_i)$. This means we cannot have $\exists^{\geq}(x_i, x_j)$ (as we could then form a derivation using X-MORE). Therefore, we cannot have $x_i \longleftrightarrow x_j$ as an edge in $G_{\geq}$, meaning x_i and x_j are not in the same equivalence class. This means that $|[\![x_j]\!]| > |[\![x_i]\!]|$. We need to remove enough points from the private intersection of these two sets such that the remaining number of points in the intersection of the two sets is less than or equal to half of the size of $[\![x_j]\!]$, but still more than half the size of $[\![x_i]\!]$. We can do this by choosing the natural number c such that

$$c = \lceil \frac{q-p}{q} apn \rceil - ph - 1 \tag{4}$$

where $[\![x_i]\!]$ is in the hth equivalence class. This means we have

$$\frac{p}{q} apn + ph < apn - c \leq \frac{p}{q} apn + ph + 1 \tag{5}$$

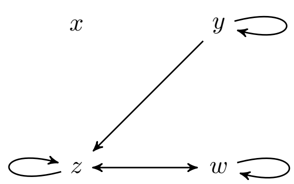
and $c < apn$. By removing c points from the private intersection of $[\![x_i]\!]$ and $[\![x_j]\!]$ and return them separately to the two sets. This means $[\![x_i]\!] \cap [\![x_j]\!]$ has size $mp^2q^{n-2} + apn - c$, and we end up with $\frac{p}{q}|[\![x_i]\!]| < |[\![x_i]\!] \cap [\![x_j]\!]| \leq \frac{p}{q}|[\![x_j]\!]|$, meaning our we have modeled $M(x_i, x_j)$ and $\neg M(x_j, x_i)$. This means we have modeled all sentences involving "most", in addition to sentences about set cardinality. Therefore, our construction is complete. □

3 Example

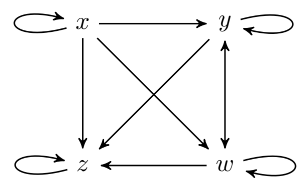
To better illustrate the implementation of this algorithm for constructing a model for any consistent set of sentences Γ, it helps to have an example. We will take

$$\Gamma = \begin{Bmatrix} \neg M(x,x) \\ M(y,x) \\ \exists^{\geq}(y,w) \\ \neg\exists^{>}(w,y) \\ \neg M(z,y) \\ M(z,w) \end{Bmatrix}$$

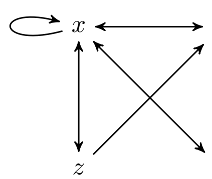
as our example Γ. Note also that we will take $p = 1$ and $q = 2$ in this example construction. We'll first construct the relevant graphs showing provability from Γ:



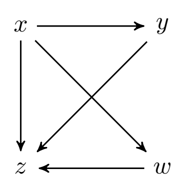
G_M



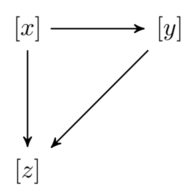
$G_{\geq}$



$G_{\neg M}$



$G_{>}$



G

From these graphs we can obtain the topological sort of G with the equivalence classes expanded x, y, w, z. We first handle the construction of sets corresponding to atoms with edges to themselves in $G_{\neg M}$. Here, that is only x, so we set $[\![x]\!] = \emptyset$. We now continue with the construction using y, w, z as our topological sort.

We take $a = 3$ and $m = 10$. First, we begin with sets B_1, B_2, and B_3 in Figure 1, then take $m = 10$ copies of each point to get $[\![y]\!]$, $[\![w]\!]$, and $[\![z]\!]$, as seen in Figure 2. We then add $apn = 9$ points simultaneously to all of the sets, shown in Figure 3.

Now, we are ready to adjust the nonempty sets to account for all statements about cardinality provable from Γ. The equivalence classes of G in order (excluding the class of empty sets) are $[x_{\alpha_1}] = \{[\![y]\!], [\![w]\!]\}$, and then $[x_{\alpha_2}] = \{[\![z]\!]\}$. Therefore, for $1 \leq i \leq 2$, we add $qi = 2i$ points to each element of the set the set $[x_{\alpha_i}]$. This means we add 2 points to $[\![y]\!]$ and $[\![w]\!]$, and 4 points to $[\![z]\!]$. This step is shown in Figure 4.

Finally, we adjust according to "most" statements. Note that our construction has already accounted for pairs of atoms x_1, x_2 where $\Gamma \vdash M(x_1, x_2)$ and $\Gamma \vdash M(x_2, x_1)$. First, we adjust for pairs of atoms x_1 and x_2 where $\Gamma \vdash \neg M(x_1, x_2)$ and $\Gamma \vdash \neg M(x_2, x_1)$. However, here, there are no such pairs among the nonempty sets. So next we adjust for pairs of atoms where $\Gamma \vdash M(x_1, x_2)$ and $\Gamma \vdash \neg M(x_2, x_1)$. Here, the only such pair of atoms is z and y. To calculate how many points we must remove from their private intersection and return to each as separate copies, we use $c = \lceil \frac{q-p}{q} apn \rceil - ph - 1$, where y is in the hth equivalence class of G. We get $c = 5 - 1 - 1 = 3$. Therefore, we remove 3 points from the private intersection and return them as separate copies to the $[\![z]\!]$ and $[\![y]\!]$. This is shown in Figure 5. Now, we have completed our construction. Figure 5 shows the semantics of each of the atoms and it can be verified that this model satisfies each sentence in Γ.

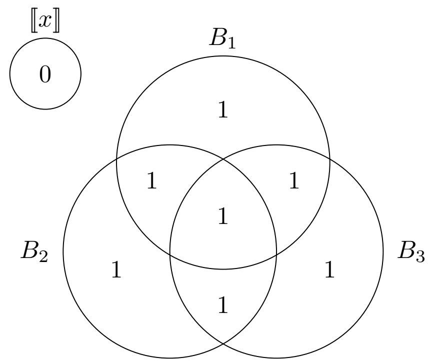


Figure 1

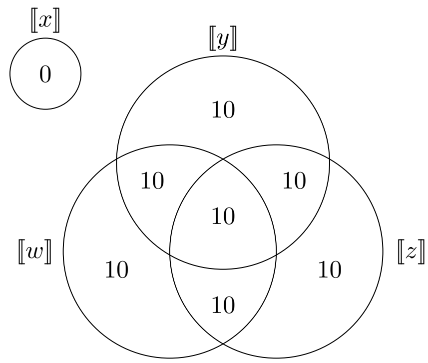


Figure 2

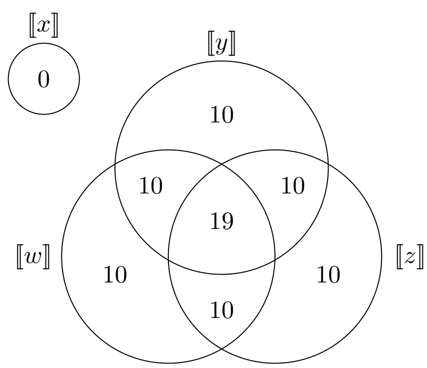


Figure 3

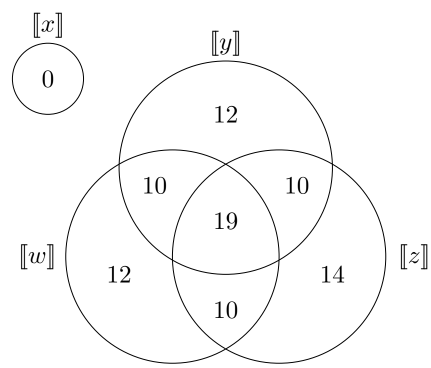


Figure 4

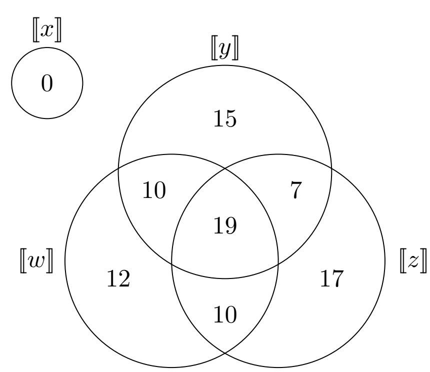


Figure 5

4 Conclusion

This paper builds off of previous work and focuses on proving the completeness of a logic system that expresses "most" as well as comparisons of set cardinality. Because this logic has negation, we prove its completeness by showing that any consistent set Γ of sentences in the logic has a model. To construct this model, we first prove that a topological sort of a one-way graph of the $\geq$ relationships specified by Γ orders the variables in Γ by cardinality of their corresponding sets and also gives a useful ordering with respect to "most". We then show that this ordering, along with some changes to the set construction algorithm provided in the previous paper on this topic by Tri Lai, Jörg Endrullis, and Lawrence Moss, [LEM16], allows us to construct a model of Γ.

References

[EM15] Jörg Endrullis and Lawrence S. Moss. Syllogistic logic with "most". In *Proceedings of WoLLIC*, pages 124–139. Springer LNCS, 2015.

[LEM16] Tri Lai, Jörg Endrullis, and Lawrence S. Moss. Proportionality digraphs. *Proceedings of the American Mathematical Society*, 144:3701–3715, 2016.

Ahlfors Regular Conformal Dimension of Two Fractals Approximated by Graphs

Jackson Morris[1] and Matthew T. Powell[2]

[1]Ohio University, jm313309@ohio.edu
[2]University of California, Irvine, mtpowell@uci.edu

July 28, 2017

Abstract

The study of fractals oftentimes involves a study of their dimension, and two such interesting dimensions are the Hausdorff dimension and the Ahlfors Regular Conformal Dimension. These two quantities are very important to the study of fractals but the Ahlfors Regular Conformal Dimension is difficult to calculate for most non-trivial fractals. In particular, there is no concrete value for the standard Sierpinski Carpet. Since finding actual values is so difficult, we studied these dimensions by looking at bounds on them. The Box-counting dimension presents an easily calculable upper bound, but better bounds are needed. By using a special type of energy, and by looking at maps between specific marked graphs, we determined upper bounds on the Ahlfors Regular Conformal Dimension that are better than the Box-counting dimension.

1 Introduction

The main object of study in this paper is the geometric idea of a fractal. Fractals appear in many different fields of study, from economics to physics, and they have many interesting properties, but in this paper we are interested in studying the idea of dimension. In particular, we are interested in the Ahlfors-regular Conformal Dimension of certain fractal carpets. This is a variant of the Hausdorff dimension–in fact the Hausdorff dimension provides an upper bound–and we know some basic information regarding the Conformal Dimension for a few well-known fractals; the Julia Rabbit has a Conformal Dimension of 1, and the Sierpinski Carpet has a Conformal Dimension between $1 + \frac{\log(2)}{\log(3)} \approx 1.630929....$ and $\frac{\log\left(\frac{9+\sqrt{41}}{2}\right)}{\log(3)} \approx 1.858183...$ where the upper bound is due to [2].

We studied these fractals by considering a sequence of graphs that converged to the fractals in question. The idea of using graphs to converge to fractals is actually quite natural; many fractals are defined iteratively, so at each iterative step we consider a graph that approximates the iteration. This allows us to construct a sequence of graphs that will converge to the fractal we want to study. We studied two specific examples (see Sections 4 and 5 for details) and determined new upper bounds on the dimension.

We begin with a brief discussion of Ahlfors-regular Conformal Dimension and then discuss elastic graphs, relating them to the p-harmonic energy of maps between elastic graphs, resulting in a definition of harmonic maps and energy. From there we will discuss how looking at ratios of energies of certain homotopy classes of maps between graphs provides bounds on the Ahlfors-regular Conformal Dimension. We will then discuss a case study of two fractals defined by elastic graphs and the methodology we used to determine relevant upper bounds on the Conformal Dimension. We conclude with a brief outline of the procedure and code used to arrive at our approximations.

Acknowledgements

We would like to extend our thanks to Indiana University for hosting the REU Program during which we conducted the research for this paper. We would like to thank Dr. Pilgrim and Dr. Thurston of Indiana University for advising us during the program and helping us grow as mathematicians. Thank you.

1.1 Ahlfors Regular Conformal Dimension

We begin our discussion with a brief overview of Hausdorff and Ahlfors-regular Conformal Dimension.

The Hausdorff dimension of a set $F \subset \mathbb{R}^n$, which we define below, is tied very closely to the Euclidean structure of $\mathbb{R}^n$. That is, it is defined with the Euclidean metric in mind. The Ahlfors-regular Conformal Dimension, on the other hand, does not rely solely on the Euclidean metric but rather on a class of metrics that are Ahlfors-regular. These metrics, for reasons that we will not go into here, allow for a more subtle study of fractals and their geometric properties.

Definition 1.1 (s-dimension Hausdorff Measure)**.** The **s-dimensional Hausdorff measure** of a set F, denoted $\mathcal{H}^s(F)$, is

$$\lim_{\delta \to 0} \inf_{\{F_n\}} \left\{ \sum_{i=1}^{\infty} |F_n|^s : |F_n| < \delta \right\} \tag{1}$$

where $\{F_n\}$ is a cover of F, $|F_n|$ is the diameter of the set F_n with respect to the Euclidean metric.

This measure generalizes the concept of length to dimensions other than 1 and so provides a useful way of studying sets. For example, $\mathcal{H}^1(B) = \infty$ where B is the open unit disc. Alternatively, $\mathcal{H}^2([0,1]) = 0$. This can be interpreted as the disc having infinite length and the unit interval have zero area. Furthermore, $\mathcal{H}^1([0,1]) = 1$ which shows that the measure can be finite and non-zero. These examples illustrate how we might define a dimension based on this measure.

Definition 1.2 (Hausdorff Dimension)**.** The Hausdorff dimension of $F \subset \mathbb{R}^n$ is

$$\mathrm{Hdim}(F) = \sup\{s : \mathcal{H}^s(F) = \infty\}. \tag{2}$$

While this is a very important aspect of fractals, and it illustrates certain geometric and analytic properties, it does not tell the whole story. A much more subtle invariant, and the quantity of prime concern to us, is the Ahlfors-regular Conformal Dimension (denoted confdim) of a set. Formally, the Ahlfors-regular Conformal Dimension of F is the infimum, over all Ahlfors-regular metric spaces quasisymmetric to F, of the Hausdorff dimension computed with respect to these metrics.

Definition 1.3 (Ahlfors-regular Metric)**.** A metric space (X, d) is **Ahlfors-regular** if for some $s \in (0, \infty)$ there exists a Borel measure μ and a constant $C > 0$ such that

$$\frac{1}{C} r^s \leq \mu(\bar{B}(z, r)) \leq C r^s \tag{3}$$

for all balls $\bar{B}(z,r)$ with $z \in X$ and $r < |X|$. If (X,d) is such a space we will say that d is an **Ahlfors-regular metric**.

Definition 1.4 (Quasisymmetrically Equivalent Metric Space)**.** Suppose two metric spaces, X, Y, are connected and Ahlfors Regular. Then a homeomorphism h is **quasisymmetric** if there exists $C > 1$ such that for all balls $B(x,r), r < |X|$, there exist $s > 0$ with

$$B\left(h(x), \frac{s}{C}\right) \subset h(B(x,r)) \subset B(h(x), Cs). \tag{4}$$

We say that two metric spaces are **quasisymmetric** if there exists a **quasisymmetric** homeomorphism between them.

Definition 1.5 (Ahlfors-regular Conformal Dimension)**.** Let F be a set in $\mathbb{R}^n$ (or $\mathbb{C}^n$) and let $\mathcal{M}$ denote the collection of Ahlfors-regular metric spaces quasisymmetrically equivalent to F. Let

$$\mathcal{A}_d^s(F) = \lim_{\delta \to 0} \inf_{\{F_n\}} \left\{ \sum_{i=1}^{\infty} |F_n|^s : |F_n| < \delta \right\} \tag{5}$$

where $\{F_n\}$ is a cover of F and the diameter is taken with respect to $d \in \mathcal{M}$. Then the **Ahlfors-regular Conformal Dimension** of F is

$$\operatorname{confdim}(F) = \inf_{d \in \mathcal{M}} \sup_{s} \{s : \mathcal{A}_d^s(F) = 0\} \tag{6}$$

As an aside, if the metric d is Ahlfors-regular then $\operatorname{Hdim}(X,d) = s$ where s is independent of which Ahlfors-regular metric we use. Furthermore, the $s-$dimensional Hausdorff measure of (X,d) is a constant multiple of the Borel measure μ from Definition 1.3.

The technical details of this definition are not utilized in this paper, this is provided for completeness and so that the reader may understand the problem.

The Hausdorff dimension provides an upper bound on the Conformal Dimension, as can be seen by the definition. It is also true that many fractals in $\mathbb{R}^n$, when using the Hausdorff measure, are Ahlfors Regular.

In order to compute this dimension, we will use new techniques to provide new bounds on the dimension of two fractal carpets (Figure 1).

Many fractals are defined iteratively, so at each iterative step we consider a graph that approximates the iteration. This allows us to construct

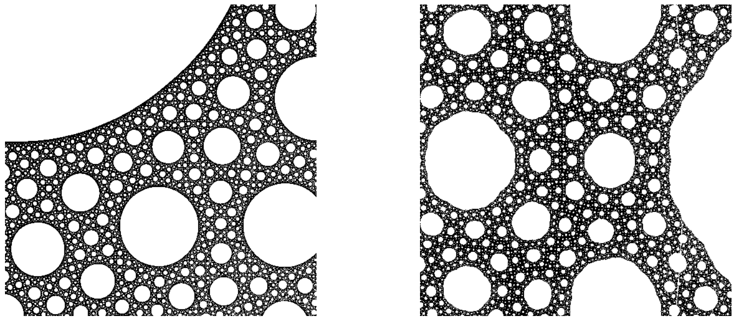


Figure 1: Julia sets of our two examples, $\frac{4(z^2-z+1)^3}{27(z(z-1))^2}$ and $\frac{-2z^3+9z-6}{z^3}$

a sequence of graphs that will converge to the fractal we want to study (see Section 1.2 for an example of this process). From here, we look at a homotopy class of maps that minimize a certain energy (Section 1.3). Finally, there is a theorem in Section 2.2 that relates ratios of these minimized energies to the Ahlfors-regular Conformal Dimension.

1.2 Graphs

As stated, we want to study certain homotopy classes of maps from graphs to other graphs that approximate our chosen fractals. In order to add some intuitive understanding to how and why we do this, we introduce some additional structure to the graphs that we are studying, which will in turn add some physical interpretations to the maps between the graphs. This will provide motivation for our later definitions of tension and energy.

We begin with some initial definitions for clarity.

Definition 1.6 (Marked Graph)**.** A graph G is marked if there is some collection of points, $a_1, ..., a_n \in G$, that we identify as **marked**.

For completeness, two marked graphs are the same if there is a bijection between them that preserves incidence, extends to a homeomorphism of the plane, and maps marked points to marked points. Marked graphs provide the initial structure that we need, but some additional properties are needed for our study.

Definition 1.7 (Elastic Graph)**.** An **elastic graph** is a marked graph G along with weights $\alpha(e)$ corresponding to each edge e in G. The edge weights

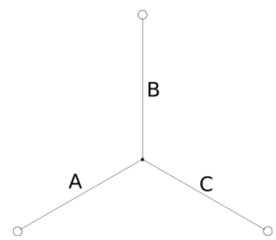


Figure 2: An elastic graph with marked endpoints

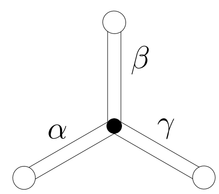


Figure 3: A length graph with marked endpoints

represent elasticities of the edges. These graphs are denoted (G, α) but throughout this paper we often use just G because the distinction between elastic graphs, weighted graphs, and later length graphs will be very clear.

We think of elastic graphs as systems of interconnected rubber bands, with each edge being represented by a rubber band and the elasticity of that band determined by the elasticity of the corresponding edge. See Figure 2 for an example.

Definition 1.8 (Length Graph)**.** A **length graph** is a marked graph G along with weights $\ell(e)$ corresponding to each edge e in G. The edge weights represent the length of the corresponding edge in G. We can denote these graphs as (G, ℓ) but when there is no ambiguity we will use merely G.

Similar to the elastic graph, we think of these length graphs as hollow tubes where the weight of an edge is the length of the corresponding tube. See Figure 3 for an example.

We are specifically interested in graphs that approximate fractals. If we consider a fractal like the Sierpinski Carpet that can be constructed iteratively and use a graph to approximate the set at each iteration, then the sequence of graphs that we define end up converging to an approximation of

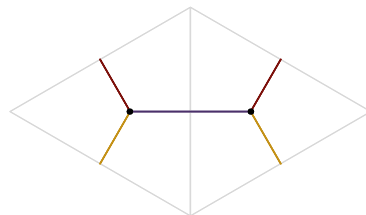


Figure 4: The tripod embedded on the Riemann sphere

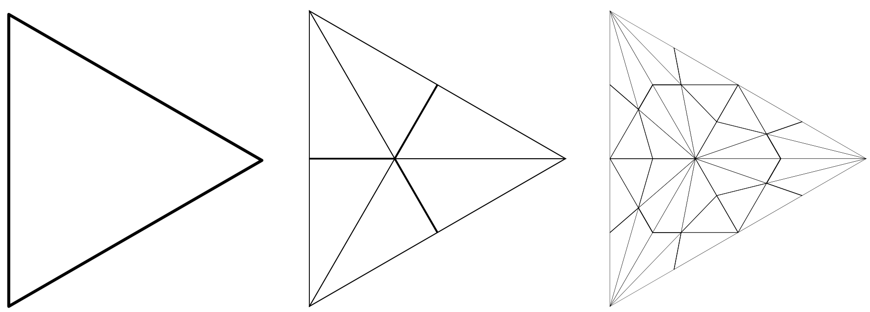


Figure 5: Triangle with two barycentric subdivisions

the fractal itself. Indeed, if we look at any computer image of a fractal, this is really just composed of a finite set of dots or pixels. Our graphs approximate the fractal in a loosely similar way that these pixels approximate the fractal.

The specific graphs that we are looking at are indexed in a sequence so that they converge to a fractal in a suitable sense. For example, consider the tripod of uniform length as an embedding on the upper-half of the Riemann sphere given by the bounding triangle of the tripod. To complete the sphere, consider another triangle-tripod adjoined to this one as in Figure 4 with edges of like color identified with each other. Consider the graphs that occur if we continually take barycentric subdivisions of an equilateral triangle as in Figure 5 and then look at the duals at each step as in Figure 6. It is these duals that we use as our sequence. If we consider these duals on the Riemann sphere as in Figure 4 and then take a limit, these duals approach a fractal that has a Julia set given by iterating the rational map $\frac{4(z^2-z+1)^3}{27(z(z-1))^2}$. We explore this example in more detail in Section 4.

When we consider maps between marked graphs in this paper, the domain is always an elastic graph and the codomain is always a length graph. One can think of this as stretching a series of rubber bands through a system of

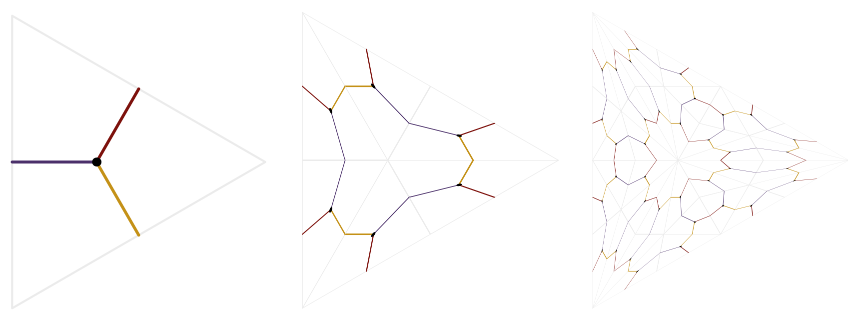


Figure 6: The dual of Figure 6

hollow tubes. Such an approach allows for a natural definition of energy of the map.

1.3 Harmonic Energy

As outlined in Section 1.2, we are particularly interested in studying fractals by using a sequence of graphs that approximate the fractal. Suppose we have such a sequence $G_0, G_1, ...$ The underlying theory of our method (Section 2.2) requires us to consider the behavior of maps between the n^{th} term in the sequence and the base graph G_0.

In particular, we are interested in studying homotopy classes of piecewise linear maps between marked graphs G_n and G_0 where G_n and G_0 are both marked graphs, the edge lengths of G_n are identified as elasticities and the edge lengths of G_0 are identified as lengths. The homotopy classes that we are interested in map marked points to marked points and since edges map linearly, the rest of the function is uniquely determined by where vertices are mapped. See Section 1.5.

Any map $f_n : G_n \to G_0$ that is piecewise linear on each edge that maps marked points to marked points has a piecewise linear derivative given by

$$f'(x) = \frac{\ell(e)}{\alpha(e)}; \quad x \in e \tag{7}$$

where $\ell(e)$ and $\alpha(e)$ are as defined in Section 1.2.

One of the properties of these maps that we are interested in studying is the concept of energy.

Definition 1.9. The p**-harmonic energy**, denoted $E_\infty^p(f_n)$ for $f_n : G_n \to G_0$, is defined to be

$$E_\infty^p = ||f_n'||_p \,.$$

This can be written more explicitly as

$$\begin{aligned} E_\infty^p(f_n) &= \left(\int_{e \in G_n} |f_n'|^p \right)^{1/p} \\ &= \left(\sum_{e \in G_n} \alpha(e) \left(\frac{\ell(e)}{\alpha(e)} \right)^p \right)^{1/p} . \end{aligned} \tag{8}$$

This definition is why we consider those maps that are piecewise linear, because in order for the L^p norm of the derivative to exist and be finite, the derivative needs to exist and be finite a.e.

We are actually interested in homotopy classes of maps, so another useful quantity is

$$E_\infty^p[f] = \inf_{\phi \in [f]} E_\infty^p(\phi)$$

where $[f]$ is the homotopy class of f. Furthermore, we do not want to study every homotopy class but instead want to focus on those that belong to a general class of maps that minimize energy and they are the maps that will allow us to approximate upper bounds for the Ahlfors-regular Conformal Dimension.

1.4 Harmonic Maps

Now that we have a working definition of the energy of our maps between graphs, it is natural to discuss when minimal values of the energy are achieved. Physically we know that the elastic network should achieve minimal energy when it is at equilibrium. That is, when there is no net force on the system. To begin this discussion, we will need to introduce what the force, or in this case tension, is.

It is reasonable to see that the tension at a point x on G due to the map f can be defined through our physical intuition of the map.

Definition 1.10. The **tension** on edge e due to a map f is

$$\begin{aligned} \tau(f, e) &= |f'(e)|^{p-1} \\ &= \left| \frac{\ell(e)}{\alpha(e)} \right|^{p-1} . \end{aligned} \tag{9}$$

This definition of tension agrees with the physical definition of tension if we allow $p = 2$, thus this definition is a generalization of the physical definition for values of p other than 2.

Alternatively, we can see that this expression for tension is precisely

$$\tau(f, e) = \frac{1}{p} \frac{\partial \left(E_{\infty}^{p}(f)\right)^{p}}{\partial \ell(e)} \tag{10}$$

where we use Equation (8) when formally taking the derivative.

In conjunction with Definition 1.10, we define an additional quantity, $\kappa(v, e)$, where v is a vertex on an elastic graph and e is an edge on an elastic graph. Then this is the set of edges e_i on the same elastic graph incident to e and v such that there exists a_i on e_i and b_i on e where $f(a_i) = f(b_i)$ and furthermore, this a_i and b_i can be chosen to be arbitrarily close to $f(v)$. That is,

$$\kappa(v, e) = \left\{ \text{edges } e_i \text{ incident to } e \text{ and } v : \begin{array}{c} \text{for all small } \epsilon, \exists a_i \in e_i, b_i \in e \text{ s.t.} \\ f(a_i) = f(b_i) \text{ and } |f(b_i) - f(v)| < \epsilon \end{array} \right\}. \tag{11}$$

Physically, this identifies those edges that pull the vertex v in the same direction as $f(e)$.

Since the forces in the elastic network are given by tensions, this means that the system is in equilibrium when the tensions at each point balance out. This motivates the concept of the triangle inequalities introduced in [5]. Given a map f and vertex v connected to edges $e_1, ..., e_n$, the map satisfies the **triangle inequalities** if:

$$\sum_{e_j \in \kappa(v, e_i)} \tau(f, e_j) \leq \sum_{e_m \notin \kappa(v, e_i)} \tau(f, e_m). \tag{12}$$

The physical interpretation of these inequalities is that the tension at a point pulling in any one direction cannot exceed the tension pulling in the opposite direction. By looking at the maps that satisfy the triangle inequalities everywhere, we see physically that the rubber band network has no net force on it, so it is in equilibrium – such maps are known as harmonic maps.

Definition 1.11 (Harmonic Map)**.** A **harmonic map** $\varphi : G_n \to G_0$ is a map that satisfies the triangle inequalities on tensions everywhere.

Alternatively, we can view the terms within the triangle inequalities as telling us the sign of the derivative of the energy function at a point. It can be seen that the triangle inequalities are always satisfied in the interior of edges, so we need only consider what happens at the vertices. Recall that every map is uniquely determined by its action on the vertices. If the triangle inequalities are satisfied at every vertex, this tells us that the derivative of the energy function for a particular map must be zero. Since the energy is a convex function (it is an L^p norm), this tells us that harmonic maps achieve minimal energy, which in turn provides an alternate and more useful definition of harmonic maps.

Alternate Definition (Harmonic Map)**.** $f : G_n \to G_0$ is a **harmonic map** if it minimizes E^p_∞.

Thus, when the triangle inequalities are satisfied the p-harmonic energy is minimized. By examining the minimal energy of these maps as p varies we can begin a process to approximate the Ahlfors-regular Conformal Dimension.

Theorem 1.4.1 (Thurston, 2016, [5])**.** *For each n and p the energy E^p_∞ has a unique minimum value.*

This theorem ensures that the minimizing energy is unique in the homotopy class, which is important for our later applications. The proof of this theorem is a direct consequence of the strict convexity of E^p_∞ from Theorem 3.1.1.

1.5 A Brief Example

Now let us consider a simple example illustrating the above definitions. Suppose we have an elastic tripod which we denote G_1 and a length tripod which we denote G_0 (Figure 7). Let f map the marked point on A to the marked point on α, the one on B to the one on β and the one on C to the one on γ. Any such map f is uniquely determined by where the center vertex v is sent, so suppose $f(v) = (a, x)$ where x indicates which edge $f(v)$ is on and y indicates how far from v' the image is. We wish to find the harmonic map for $p = 2$. First, we suppose $f(v) = v'$ and calculate the tensions. This will tell

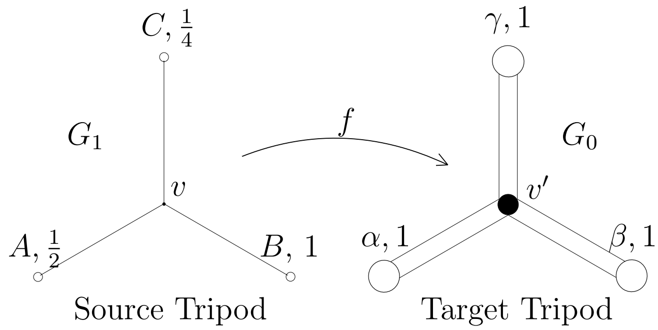


Figure 7: Source and Target Tripods for our Example

us what edge the image must be on to be optimal. We compute the tensions

$$\begin{aligned} \tau(f, A) &= 2 \\ \tau(f, B) &= 1 \\ \tau(f, C) &= 4 \end{aligned} \tag{13}$$

and apply the triangle inequalities:

$$\begin{aligned} \tau(f, A) &\leq \tau(f, B) + \tau(f, C) \\ \tau(f, B) &\leq \tau(f, A) + \tau(f, C) \\ \tau(f, C) &> \tau(f, A) + \tau(f, B). \end{aligned} \tag{14}$$

This tells us that the network of rubber bands is being pulled onto the γ edge, so we know that $x = \gamma$. Now we want to find y by examining the p-harmonic energy function. In particular, we treat this as a single variable calculus problem and locate the minimum by differentiating the energy function. Doing this tells us that the minimum value is achieved for $y = \frac{1}{7}$. We now verify the triangle inequalities are achieved for $f(v) = (\beta, \frac{1}{7})$ and conclude that $f(v)$ is a harmonic map with energy $\sqrt{\frac{48}{7}}$. This last verification is merely to ensure that we did not make any mistakes because, since the function E^p_∞ is convex, we know that there is precisely one minimum value, and thus it must correspond to the minimum we found. This is very similar to how we found harmonic maps for more complicated graphs.

Alternatively, for $p = 2$ we can solve for the harmonic map by using fact that, $f(v)$ is somewhere on the γ leg and that the harmonic map must satisfy the triangle inequalities. Thus, since there are only two directions on the γ leg for the tension to pull, we know that

$$\tau(f, C) \leq \tau(f, A) + \tau(f, B) \tag{15}$$

$$\tau(f, A) + \tau(f, B) \leq \tau(f, C) \tag{16}$$

so we have equality. Now we can use the explicit expressions for the tension τ to find how far on the γ leg $f(v)$ is mapped. The reason that we did not use this method for more complicated graphs is that the tension becomes very non-linear for $p \neq 2$. This would have added additional complexity to the problem.

Note that if we had considered a graph with n vertices then the map f would have been determined by n 2-tuples.

2 Relating Conformal Dimension and Harmonic Energy

Here we will discuss the theoretical tools that we used to find our bounds. These results will be presented without proof but the papers they appear in are cited for the reader.

2.1 p Energy

Suppose we have a sequence of elastic graphs G_n that converges to a fractal G. In fact, the G_n come from a particular construction in [6]. We ultimately want to compute confdim(F). If we were to use the definition we would run into trouble almost immediately because of how technical the definition is. Not to mention having to take an infimum over an unmanageable uncountable collection. Thus we need to use an alternate method. The method we used is due to Kevin Pilgrim and Dylan Thurston in [4]. To understand these ideas, though, we first need to introduce a useful ratio.

Definition 2.1 (p Energy)**.** Let $\{G_n\}$ be a sequence of elastic graphs that converges to a fractal G. This sequence comes with an explicit family of maps $\phi_n : G_n \to G_0$ but we really only care about the homotopy classes of these maps $[\psi_n]$. For every new metric we define on G_0–which is equivalent to changing the edge weights of the graph–we can find a $p-$harmonic map from $G_n \to K$ where K is G_0 with this new metric. Let $f : G_n \to K$ and $\varphi : G_0 \to K$ be $p-$harmonic maps. We define the p **energy** to be

$$\max_{\text{All metrics on } K} \frac{E^p_\infty[f_n]}{E^p_\infty[\varphi]} \tag{17}$$

We denote this quantity $E^p_p[G_n \to G_0]$ or $E^p_p[f_n]$.

A nice property of this new energy is that it is submultiplicative in n. That is, if $g(n) = E^p_p[G_n \to G_0]$ then $g(n+m) \leq g(n)g(m)$.

This energy gives us a way to find the Ahlfors-regular Conformal Dimension. This is explained in the next subsection.

2.2 Connection between Ahlfors-regular Conformal Dimension and Harmonic Maps

Our final goal is to obtain bounds on the Ahlfors-regular Conformal Dimension of certain fractals that can be seen as limits of elastic graphs. Supposing that we have such a fractal as a limit of graphs $G_0, G_1, ...$ then the following are theorems from [6] and [4]:

Theorem 2.2.1 (Thurston, 2016, [6])**.** *The limit $\bar{E}^p_p = \lim_{n\to\infty} E^p_p[G_n \to G_0]$ exists and is continuous and decreasing in p.*

We call this limit the **asymptotic energy**. Existence is a consequence of the submultiplicativity of E^p_p and raises the question of what the limit represents. The next theorem, due to K. Pilgrim and D. Thurston, ensures that calculating these energies will provide bounds on the Conformal Dimension.

Theorem 2.2.2 (Pilgrim-Thurston, [4])**.** *For $p = confdim(J_{G_n})$, we have $\bar{E}^p_p(J_{G_n}) = 1$. Equivalently, if $\bar{E}^p_p(J_{G_n}) > 1$ then $p < confdim(J_{G_n})$ and if $\bar{E}^p_p(J_{G_n}) < 1$ then $p > confdim(J_{G_n})$. Here J_{G_n} is the Julia set for the limiting set of G_n.*

This gives a method for calculating the Ahlfors-regular Conformal Dimension of fractals that can be expressed as limits of graphs in this manner. We will use nonlinear optimization techniques in order to arrive at good approximations of $\bar{E}^p_p$.

3 Methods

Now that we have laid the groundwork for our paper, we will discuss some of the methods we used to arrive at our results. To begin, we will discuss convexity and how we can use it to optimize functions. Then we introduce an algorithm for locating minima.

3.1 Convexity

Since we are interested in optimization, it is helpful to understand the function that we are optimizing, E^p_∞, because certain properties make optimization easier than other properties. One such property is convexity.

Definition 3.1 (Convexity)**.** Let X, Y be metric spaces, and X a convex set. A function $f : X \to Y$ is convex if $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$ for every $x_1, x_2 \in X$ and $0 < \lambda < 1$.

An immediate consequence of this definition is that any convex function has a minimum. Furthermore, if the inequality is strict, then we have what is called a strictly convex function and any local minimum is also the global minimum. We would like to say that E^p_∞ is a convex function. By defining the metric on a graph to be the infimum, over all paths connecting two points, of the length of this path, we can see that elastic and length graphs are metric spaces. Furthermore, the elastic and length graphs that we consider are convex, so the energy function is convex. To see this, observe that the L^p norm is a convex function itself when defined on a convex domain and that the energy function is uniquely determined by the map f which in turn is uniquely determined by where on the length graph vertices are mapped to. Thus our energy function is a map from a collection of points on our length graph to $\mathbb{R}$, so it is convex.

Theorem 3.1.1. *If $g : X \to Y$ is a strictly convex function and $f : Z \to X$ is a non-constant piecewise linear function, then $h = g \circ f : Z \to Y$ is a strictly convex function.*

This theorem ensures that when our domain graph is convex, as in the case of the Barycentric Subdivision where our domain is essentially $[0, 1]^n$ (see Section 4.3), then our p-harmonic energy function is also convex, which allows us to use specialized tools to minimize it.

3.2 Gradient Descent

The optimization tool that we used in our study is known as gradient descent. It is very similar to a generalization of Newton's method to $\mathbb{R}^n$. We pick a starting point, x_0, and consider the gradient of the function we want to minimize at that point, $\nabla f(x_0)$. Since the gradient indicates the direction of greatest ascent, if we move in the $-\nabla f(x_0)$ direction then we will be following the direction of greatest descent. Since we are looking for a minimum, this is what will lead us to it, hence we pick a new point $x_1 = x_0 - \nabla f(x_0)$.

In theory, if we could look at this continuously, rather than discretely, we would always be able to locate the minimum of a strictly convex function, but if we look at this method discretely we would need to 'step' along the gradient in a non-continuous way. This introduces a problem. We might step past the minimum and enter a periodic orbit, as would be the case if the norm of the gradient is too large, so we need to choose a proper step-size to stop this from happening. Thus we choose $x_n = x_{n-1} - k_{n-1}\nabla f(x_{n-1})$ with $k_0 = c$ a positive constant. Such a step-size is introduced in [1] and is given as

$$k_n = \frac{\langle \Delta x_n, \Delta y_n \rangle}{||\Delta y_n||^2} \tag{18}$$

where $\Delta x_n = x_n - x_{n-1}$ and $y_n = \nabla f(x_n) - \nabla f(x_{n-1})$. Using this algorithm, we are able to find approximate minimal values for the energy function.

This method allows us to find the minimum value of E^p_∞ for any fixed p and fixed domain and codomain graphs. In order to approximate the asymptotic energy, though, we need to be able to maximize E^p_p over the target graph K. We did this using an algorithm described by D. Thurston. We considered the harmonic map for some starting target graph K_0 and calculated the tension on each edge. We then considered the tension on the edges a_i of K_0 by looking at

$$\tau(a_i) = \sum_{x \in f^{-1}(a_i)} \tau(f, x)$$

and then defined a new K_1 by setting the new edge weights to be $\ell(a_i) = \alpha(a_i)\tau(a_i)^{\frac{1}{p-1}}$ where $\alpha(a_i)$ is the elasticity of edge a_i on G_0. This, as a limit, gives us the maximizing target graph K. This will give us E^p_p.

In practice, we are severely limited because this is an iterated process. We used the computer language Sage to implement this algorithm for the specific examples we considered and that means that any result we arrive at has a margin of error that is inherent in using a computer that can only do finitely many operations. Because of this limitation, we could not take limits and instead computed $E^p_\infty[G_n \to K]$ for small n and then observed the trend. This provided us with an upper bound on E^p_∞ which in turn provided us with an upper bound on E^p_p which resulted in an upper bound on the Ahlfors-regular Conformal Dimension of our two fractals. The details are in Sections 4 and 5 respectively.

4 Barycentric Subdivision

4.1 Preliminaries

The first example that we turned our full attention to was the barycentric subdivision of the triangle. Given a triangle, one may divide it into six by drawing lines between the vertices and the centers of the opposite edges. From this, one may continue to subdivide the triangles generated by the first division, then the ones by the second, and so on and so forth (see Figure 5). What we were interested in was the dual of these subdivisions, namely the graph generated by assigning a vertex to each triangle of the subdivision and connecting it to the vertex that represents another face when the two triangles are adjacent (see Figure 6).

Note that there are connections between the triangles at the edge and the edges of the original triangle. This is because we are viewing the triangle as being on a sphere, with the triangle taking up one hemisphere. Thus, those segments connect to the corresponding faces of the triangle on the opposite hemisphere.

The barycentric subdivision is a good first case, largely due to its clear level of self-symmetry. It is easy to show that the n^{th} level of the subdivision is six of the $(n-1)^{th}$ level linked in a hexagonal pattern, which lends itself to an easy definition in code. Another boon of symmetry is that, given threefold symmetry, we know first that the target tripod has to be symmetric, and second we know that the optimal map must be symmetric as well. The remaining twofold symmetry lets us reduce the map further, to one-sixth the size of the original.

Before we began, we knew that 1.631 was the lower bound on the conformal dimension and the upper bound was 1.84 due to the Box-counting

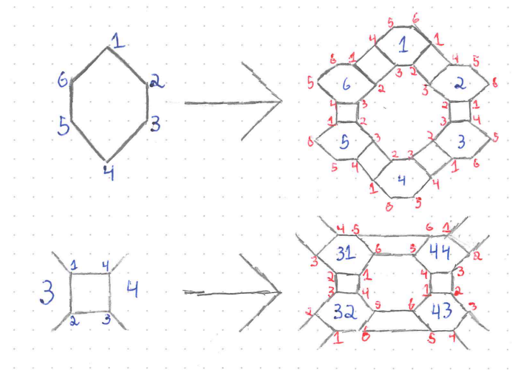


Figure 8: Construction of the Barycentric Subdivision Graphs

dimension (see Appendix A). The lower bound is due to the formula

$$\frac{1}{1 - \log_d \overline{N}}$$

where $\overline{N} = \overline{E}_1^1$ which is equal to the growth rate of the number of independent non-trivial loops in the graph and d is the degree of the map (in this case 6) [4].

4.2 Graph Generation

As alluded to earlier, the self-similarity of the barycentric subdivision was critical to efficiently creating a general method of representing the graphs. However, the way of looking at the self-similarity was used differed from the obvious self-similarity. Rather than thinking of G_n as being six copies of G_{n-1}, we instead conceptualize it as G_{n-1} with each vertex expanded into a hexagon, with all edges doubled to accommodate as in Figure 8.

The details of how the numbering worked specifically are detailed above in Figure 8. In short, a hexagon in G_{n-1} is expanded into a ring of double-connected hexagons, with 2's and 3's lining the interior of the ring's connections, 1's and 4's on the exterior connections, and 5 and 6 serving to connect this ring to other structures in G_n. In particular, it's worth noting that a hexagon's 1 and 2 will be connected to the hexagon indexed 1 greater than it (mod 6) while its 3 and 4 will be connected to the hexagon one less than it.

The second image shows the details of how the 5s and 6s connect externally. Given a connection between two hexagons in G_{n-1}, these are expanded as demonstrated in that image. The important thing to note is that due to the opposing orientations of the expanded hexagons, 5s get connected to 6s and visa versa. It's also important to note that the numbering of the original hexagons is unimportant here; 3 and 4 could be replaced by any connected pair, including 5-6s. This is very relevant for how we coded G_n.

The specific code for the graph generation are found in B.1, but for our purposes, here's pseudocode (similar to Python). In this, we treat numbers as points in G_n by their base-6 representation.

```
gn:
 if n = 1
     return g1
 for i in range (0,6^n)
     x = i[0](base 6)
     y = i[0]+1 (mod 6)
     y = i - i[0] + y
     z = i[0]-1 (mod 6)
     z = i - i[0] + z
     if (x = 1)
         d = i[1] + 1 (mod 6)
         d = i - i[1] - i[0] + d*6 + 4
     if (x = 2)
         d = i[1] + 1 (mod 6)
         d = i - i[1] - i[0] + d*6 + 3
     if (x = 3)
         d = i[1] - 1 (mod 6)
         d = i - i[1] - i[0] + d*6 + 2
     if (x = 4)
         d = i[1] - 1 (mod 6)
         d = i - i[1] - i[0] + d*6 + 1
     if (x = 5 or 6)
         while (i[j] is 5 or 6)
             i[j] = 5 - i[j]
         if j = n
             d = A, B, or C
         else
             d = i
     add [x,y,d] to graph at relevant index
```

Let's walk through exactly what this is doing. First, the recursion does not work for $n = 1$, so it simply returns a predefined $g1$ if it's asked to run 1. The next thing of note is setting up the overall loop. For this, it's important to know that we're treating numbers as points in G_n by looking at their base six representation. In the loop, i is the number of our current point and x is the last coordinate of the point, that is, where it is in the smallest hexagon that contains it. Thus, y and z are the points in that hexagon that i is connected to. The remainder of the function determines what the final connection of the point is, and this is where the earlier discussion really comes into play. If x is 1 or 2, it connects the point to the 4 or 3 of the one-higher indexed hexagon. If it is 3 or 4 is connects the point to the 2 or 1 of the one-lower indexed hexagon. The most complicated case is if x is 5 or 6.

For this, the function goes up through the point's coordinates until it finds a non-5 or 6 coordinate. If it does not find these before the last coordinate, it attaches the point to one of the endpoints depending on the last coordinate. If it does find a 5 or 6, it takes 5-the value of each coordinate up to the one it found, and attaches to the resulting point. One can see why this is done by considering the second of the demonstrative images above; 5 is connected to 6, 6 to 5, and any 1's, 2's, 3's, or 4's below the first layer are connected predictably to corresponding 4's, 3's, 2's, and 1's.

4.3 Optimization

In terms of the actual gradient descent, this was far more straightforward. In particular, since the symmetry informs us which leg of the tripod each point ends up on, we end up optimizing over $[0,1]^n$-a convex domain. The energy and gradient functions work as one would expect. The energy function goes through point by point, calculating the energy from each connection and summing it up with a one-half scalar. The scalar normalizes the double-counting that would occur on connections between two points, while at the same time providing the appropriate scaling for connections to endpoints, as those only have length 1/2 due to the aforementioned sphere visualization. (We're treating our αs as always being 1/2) The gradient function works similarly, calculating the tension in each connection and taking the sum to find the overall direction. From there, we simply used the Barzilai-Borwein step sizes and it ran smoothly.

For the energy and graph functions, the pseudocode is as follows. For this, g is our graph, encoded as a list where the i^{th} coordinate contains a list of the points the i^{th} point is connected to. The energy function calculates the $p-$harmonic energy of a map while the gradient function calculates the overall gradient at each point. The actual code can be found in B.2.

```
Core Loop:
for i in range (0,6^n)
    a = f(i)
    b = f(g[i][0])
    c = f(g[i][1])
    if (f(g[i][2]) is not an endpoint)
        d = f (g[i][2])
        x = 1
    else
        if A
            d = 0
            x = 2
        if B
            d = a
            x = 2
        if C
            d = 1
```

```
            x = 2

Energy:
 energy = energy + .5(abs(a-b))^p +.5(abs(a-c))^p + .5x(abs(a-d))^p

Gradient:
 gradient =(sgn(a-b)p(abs(a-b))^(p-1) + sgn(a-c)p(abs(c-a))^(p-1) + p*gn(a-d)(abs(x*(a-d)))^(p-1))
```

These functions, as you can see, are closely related so we will discuss them together. The first thing both functions do after setting up the standard loop is to define a as f of i, b and c as f of the second and third entries of the i^{th} entry in g. For this, keep in mind that g is a list of triples that contain the points connected to the i^{th} point and f is our map. For the purposes of the barycentric subdivision, we already know which leg of the tripod each point ends up on, so the $f-$values are just numbers between 0 and 1. Therefore, we have defined a, b, and c to be the images of the i^{th} point and its two simplest connections. Then, the function moves to the last connection. Given that, due to construction, only this last connection could potentially be an endpoint, it first checks if this connection is not an endpoint. If it is not, the function sets d to be the image of this point and a scaling factor to 1. What it does if it does get an endpoint is largely a matter of what sixth of the triangle you're looking at–for the purposes of our program, we looked at the bottom-right triangle of the subdivision, so the A side gets sent to the center, the C side gets sent to the endpoint, and going through the B side gives an image in the same location. The d values reflect this, and the 2 scalar reflects the effect of the lesser side-length on the calculation. The gradient and energy function differ in the final calculation. Energy takes the sum of the distances of these images raised to the p power and scaled by a factor of 1/2–this accounts for the double-counting from connected points and the scaling for connections to endpoints in one fell swoop. Gradient instead raises the distances to the $(p-1)$ power and has factors for the sign of the difference, p, and the inverse of the determined scalar. Rather than summing up, it puts each individual term in a list.

The final function of note to discuss is the optimization function:

```
f = startpt
k = .1
g = gradient (f)
old f = f
old g = g
for x in range (0,6^n)
    if (sgn(h[x])=1)
        if (f[x] = 0)
            u = 100000
        else
            u = (f[x]/h[x])
    elif (sgn(h[x])=-1)
```

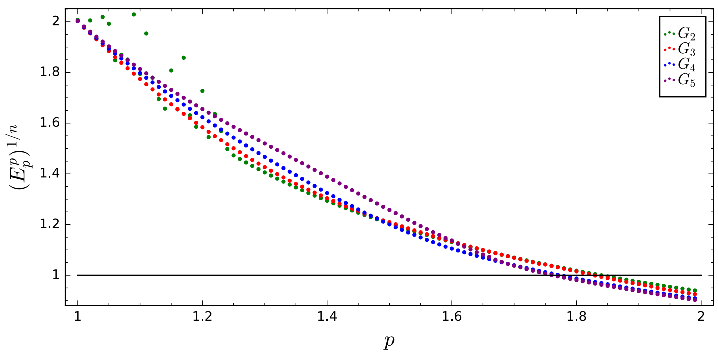


Figure 9: Plots of $E_p^p[G_n \to G_0]$ vs p for $n = 2, 3, 4, 5$.

```
    if (f[x] = 1)
        u = 100000
    else
        u = ((f[x]-1)/h[x])
elif (h[x]=0)
        u = 10000
if (u < k):
    k = u
f[x] = f[x] - k*h[x]
if (f[x]>1):
    f[x] = 1
if (f[x]<0):
    f[x] = 0
g = gradient
k = stepsize (f,n,m,h,g,p)
if k = 0
    end
```

As part of a trick to save on time, this function takes a starting point from an earlier optimization, which it initializes the map to, because we believe the harmonic maps should be close. A starting step-size of .1 is chosen, and it calculates the current gradient. Then, it moves into storing things in the storage lists, and adjusts the step size to prevent overstepping boundaries. Next, it makes the actual movements, having more overshoot insurance just in case. Finally, it recalculates gradient and step-size using the storage vectors. The last few lines are a time-saving kill-switch; if the step-size is zero, it terminates the program early.

4.4 Results

Figure 9 depicts the graph of the $(E_p^p)^{1/n}$ energies as p varies. From this, the main takeaway is noting where the energies equal 1, which given the earlier theorems each give us an upper bound on the Ahlfors-regular Conformal

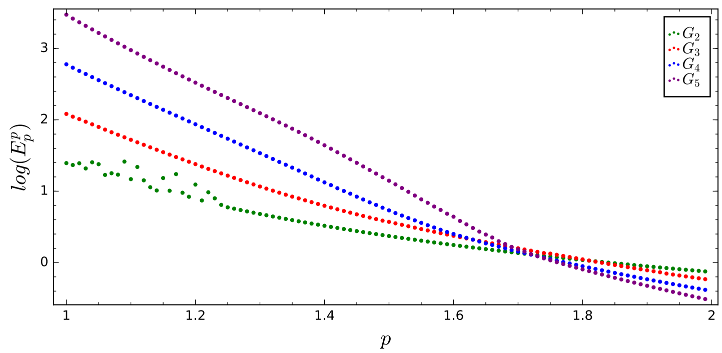


Figure 10: Logarithmic Plots for $E_p^p[G_n \to G_0]$ for $n = 2, 3, 4, 5$.

Dimension of the overall fractal. In terms of more specific things, there is some notable pathological behavior for lesser values of p, though we have discovered that this is due to computer error and furthermore, these values are less than the previous lower bound on the Conformal Dimension so they are of little interest to us. For $n \geq 6$, the runtime was too great to get solid results by the time of this paper. Acquired cross points were $G_2 : 1.83, G_3 : 1.82, G_4 : 1.77, G_5 : 1.76$. Thus we have an upper bound of approximately 1.76 on the Conformal Dimension.

Another thing we can look at to find an approximation to the Ahlfors-regular Conformal Dimension of the fractal is the relative logarithms of these E_p^p graphs. The reason for this is a weak intuitive argument.

Recall that E_p^p is submultiplicative, and so $\log(E_p^p)$ is subadditive. Suppose that $p <$ confdim(G). Then we know that $E_p^p > 1$ for every n, so $\log(E_p^p) > 0$ for every n. Furthermore, subadditivity and our data tells us that $\log(E_p^p)$ is *almost* monotonically increasing. That means that for $p <$ confdim(G), we have an almost increasing sequence. Thus if our logarithmic plot stops being decreasing in n then we should have an upper bound for the Conformal Dimension. For the Barycentric case we can see that our sequence stops being decreasing at about $p = 1.7$ so that is a potential upper bound on the Conformal Dimension.

5 Blown Up Lattes

5.1 Introduction

The Blown Up Lattes is another fractal limit of graphs. Like our first example, we start with a triangle, and similarly, we're considering it on a sphere. Additionally, the construction largely functions by taking the dual of subdivisions, however, the subdivisions in the Blown Up Lattes are more complex than that of the Barycentric Subdivision (See Figure 11).

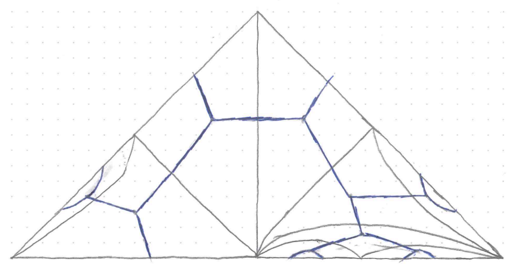


Figure 11: Second Subdivision of the Example 2 Graph

As mentioned, this is more intricate than the Barycentric Subdivision. To understand what is going on, one must first accept that all faces displayed in Figure 11 are triangles. Now, to continue subdividing we need to state two things. Firstly, that the next set of bisecting lines will come from the point labled D in Figure 12, and that this pattern will continue in each subdivision. The second is that the face labeled “1” in the image has inverted orientation- in the map to the sphere, it is being mapped to the opposite face. This means that the direction the curved triangle is drawn is reversed in further iterations.

This is a solid second example due to several reasons. First of all is the simplicity. The graph only grows at a rate of 3^n rather than the Barycentric's 6^n which allows our optimization to be run much more quickly than the previous example. Secondly, it is useful for its lack of symmetry. This forces us to use more general methods and more complex approaches to the optimization that can be generally applied, as opposed to the approach to the Barycentric Subdivision, which relied heavily on the degree of symmetry. All in all, it was an excellent second example, and the work done can serve as a basis for further work in general graphs of this variety.

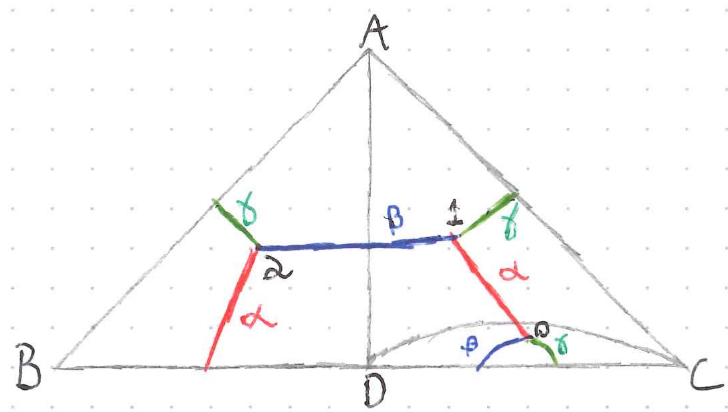


Figure 12: A more detailed look at the first level of the Blown Up Lattes

Before we began we knew that 1.461 was an established lower bound and 1.9 was the upper bound due to the Box-counting dimension (see Appendix A for upper bound discussion) and the lower bound is established using the same formula from [4] as was used previously with $d = 3$.

5.2 Graph Generation

Much like the Barycentric Subdivision, the process for generating the Blown Up Lattes graphs comes about by thinking of F_{n+1} being F_n with each point replaced by something. In this case, it is by replacing each point with an F_1 graph. In that image, the various directions from the points are labeled, and from there we can work out how this finite state automaton works. Moving in the α direction from 0 or 1 takes you to the other. Moving in the β direction from 1 or 2 takes you to the other. Moving in another direction takes one out of this tripod, with a direction depending on which side of the triangle it exits, α for the right edge, β for the left edge, and γ for the bottom edge. The code can be found in B.3.

5.3 Reused functions and changes

The first thing to generally handle are the energy, gradient, and base optimize functions. These work largely the same as in the Barycentric case with some concessions due to the lack of symmetry, so we cover them collectively. As before, the actual code can be found in B.4. In terms of notation, `tar`

refers to the the target tripod and is used for lengths. Also recall from earlier that positions are pairs; both distance along an edge and which edge the point is on.

```
Core loop:
 u = f(i)
 v = side(a)
 for j in range (0,3)
     if (g[i][j] not endpoint)
          d = f(g[i][j])
          m = side(d)
          if (d, u same side)
              x[j] = tar[side]abs(u[0]-d[0])
          else
              x[j] = tar[sidea]u[0] + tar(sided)d[0]
          (sidetotal = sidetotal + sgn(x[j])p(abs(x[j])))
      if (g[i][j] is an endpoint)
          if (u same side)
              x[j] = tar(side)(2-u[0])
          else
              x[j] = 2tar(side)+2u[0]tar(otherside)
      (sidetotal = sidetotal + sgn(x[j])p(abs(x[j]))

Energy:
 energy = energy + (1/2)(x[0])^p + (1/2)(x[1])^p + (1/2)(x[2])^p

Gradient:
(a,b, and c being sidetotals)
if (a-b-c > 0):
    k = 'A'
elif (b-a-c > 0):
    k = 'B'
elif (c-a-b > 0):
    k = 'C'
else:
    k = 'A'
return [(sgn(x[0])p(abs(x[0]))^(p-1)+sgn(x[1])p(abs(x[1]))^(p-1)+psgn(x[2])(abs(x[2]))^(p-1),k]

Optimize change:
if ((s[i][0] == 0)):
    if (s[i][1] != g[i][1]):
        s[i][1] = g[i][1][:]
```

All changes from the previous incarnations of these functions are simply concessions due to the lack of symmetry and will be highlighted here. Both energy and gradient begin by recording which side the indexed point is on, and then run through a loop to account for any position being potentially connected to an endpoint. It sets a placeholder to the image of the indexed point, calculates the distance between that and the image of each connection, accounting for side lengths, and then takes the appropriate sum at the end. The only other thing of note is how the gradient handles which direction it marks the result as. This is from simply totaling up the amount it is being pulled in each direction and if one of these totals is larger than the sum of the other two, it marks it as that, otherwise it simply defaults to A. As for the basic optimizer, essentially the only change is how it handles pulling through the center. Before updating positions, it simply checks whether or

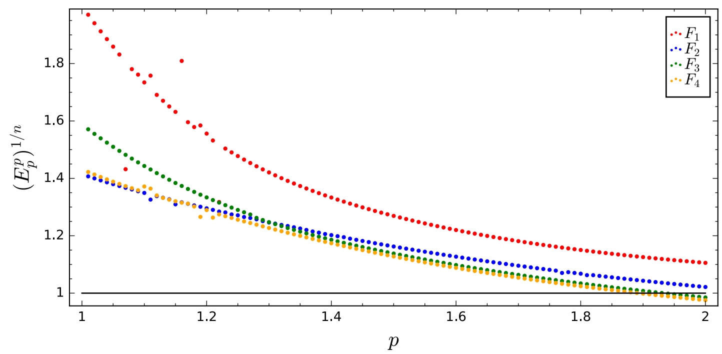


Figure 13: Standard graphs for $E_p^p[F_n \to F_0]$ for $n = 2, 3, 4, 5$.

not a point is at the center, and if it is, places it on the side that it is being pulled in and recalculates gradient.

5.4 Tripod Optimization

The final piece of discussion is an algorithm of Dr. Thurston's to find the ideal target tripod. In this specific incarnation, the algorithm works by calculating the sum of the tensions on each edge of the base tripod and then normalizing them to use as the tripod lengths for the next step.

```
Lengths:
 u = points[i]
 v = side(u[1])
 for i in range (0,3)
     x = graph[i][j]
     if (sameside)
         p = tar[side]*(u[0]-x[0])
     else
         p = tar[side]*x[0]+tar[otherside]*u[0]
     append p
TensionF:
 image/original^(p-1)

TensionK:
 if attached to endpoint
     sum(endpoint) = sum(endpoint) + tension

newlengths:
 .5*x^(p-1)
```

The actual code can be found in B.5.

5.5 Results

Much like with the Barycentric Subdivision, Figure 13 depicts the plots of the $E_p^{p\,1/n}$ energies as p varies from 1 to 2. Once again, we can note where these plots intersect 1 and thus obtain upper bounds on the Ahlfors-regular Conformal Dimension from this. Also present in this graph is the poor behavior at low values of p, but this does not influence the behavior for the values that we are interested in. For $n \geq 5$, unfortunately, we ran into runtime issues while running the optimization and as such those are not included here. The F_1 and F_2 graphs do not cross 1, but the F_3 crosses at 1.93 and the F_4 at 1.9. This gives us an overall bound of 1.9.

The logarithm graphs here depict the same thing that we discussed earlier for the Barycentric Subdivision example, so we have a potential upper bound on the dimension of 1.75.

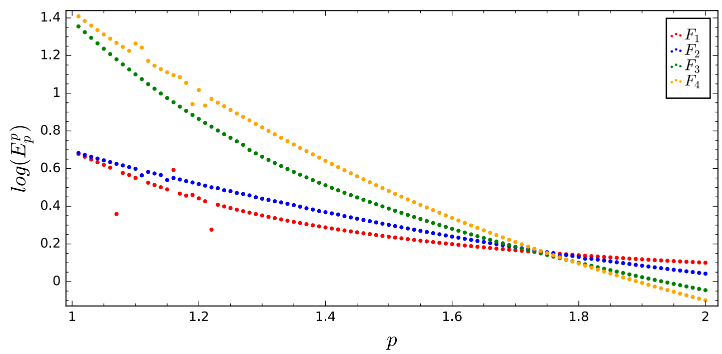


Figure 14: Logarithmic graphs for $E_p^p[F_n \to F_0]$ for $n = 1, 2, 3, 4$.

6 Conclusion

We have been able to give upper bounds on two specific fractal carpets. The Barycentric Subdivison fractal has a Conformal Dimension bounded above by 1.76 and the second cubic map has an upper bound of 1.9. For the second map, this is no worse than the bound from the crossing 1. In the future, it would be interesting to generalize our code and approach to find upper bounds on any fractal carpet that can be defined as a limit of graphs. Furthermore, we began this project with the intention of studying the Sierpinski Carpet, but did not have the time to do a proper study of it, so it would be interesting to know what bound this method gives us.

Finally, this method provides a means of studying fractals generated by graphs, but would it be possible to modify it so that it could apply to a

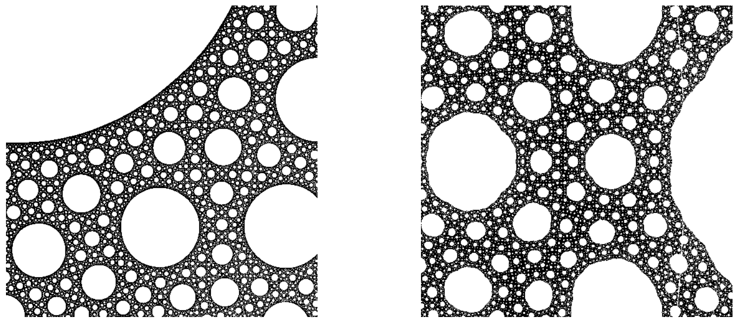


Figure 15: Julia sets of our two examples, $\frac{4(z^2-z+1)^3}{27(z(z-1))^2}$ and $\frac{-2z^3+9z-6}{z^3}$, constructed as described in Section A

larger class of fractals? If so, it would give us a useful tool for studying the Ahlfors-regular Conformal Dimension.

A Initial Bounds

As we discussed at the start of our paper, the Hausdorff dimension provides a natural upper bound on the Conformal Dimension. Similarly, the Box-counting dimension provides a natural upper bound on the Hausdorff dimension and it is easy to calculate rough approximations of the Box-counting dimension.

Definition A.1. Box-Counting Dimension The Box-counting Dimension of a set $F \subset \mathbb{R}^n$ is given by

$$\lim_{\delta \to 0} -\frac{\log(N_\delta(F))}{\log(\delta)}$$

where $N_\delta(F)$ is the infimum of the number of δ mesh boxes that intersect F for $\delta < 1$.

What follows is our use of this dimension to provide more trivial bounds on the Conformal Dimension. This will illustrate how the bounds we obtained through our energy method are an improvement on the trivial bounds obtained through the Box-counting dimension.

The Box-counting dimension is easily estimated given an image of a Julia set. We generated Julia sets of the two fractals we were interested in and colored a pixel white if it escaped to ∞ under iterations of the function and colored the pixel black if it did not escape to ∞ as in 15. This process means that the black portion of our image is an approximation of the Julia set. From here, we decreased the mesh, δ, of our pixels and counted the number of black pixels present, denoted b_δ. By considering

$$-\frac{\log(b_\delta)}{\log(\delta)} \tag{19}$$

we were able to take a linear regression to find approximate values for the box-counting dimension. The barycentric subdivision example had a bound of 1.84 for the dimension and the Blown Up Lattes had a bound of 1.9. We got a bound on the Barycentric example of 1.76, and on the cubic example of 1.9 as well. The 1.9 is not an improvement, but our soft estimate indicates our bound should be much lower, and the F_n ran into runtime issues earlier in the process.

B Code

B.1 Barycentric Graph

```
def gn(n):
    if (n==1):
        return g1
    graph = []
    for i in range(0,(6^n)):
        done = False
        x = i % (6)
        y = (x-1)%6
        z = (x+1)%6
        y = y + i - x
        z = z + i - x
        f = []
        if (x == 1):
            d = (i - (i%36)) + 6*(((((i%36)-(i%6))/6)+1)%6) + 4
            graph.append([(y),(z),(d)])
        if (x == 2):
            d = (i - (i%36)) + 6*(((((i%36)-(i%6))/6)+1)%6) + 3
            graph.append([(y),(z),(d)])
        if (x == 3):
            d = (i - (i%36)) + 6*(((((i%36)-(i%6))/6)-1)%6) + 2
            graph.append([(y),(z),(d)])
        if (x == 4):
            d = (i - (i%36)) + 6*(((((i%36)-(i%6))/6)-1)%6) + 1
            graph.append([(y),(z),(d)])
        if (x==5 or x==0):
            f.append(x)
            for j in range (1, n-1):
                f.append(((i-(i%(6^j)))/(6^j))%6)
                if (f[j]==1 or f[j]==2 or f[j]==3 or f[j]==4):
                    d = i - (i%(6^(j+2)))
                    for k in range (0, j+1):
                        m = 5 - f[k]
                        d = d + (6^k)*m
                    x = ((i-(i%6^(j+1)))/(6^(j+1))%6)
                    if (f[j]==1 or f[j]==2):
                        x = (x+1)%6
                    if (f[j]==3 or f[j]==4):
                        x = (x-1)%6
```

```
                d = d + (6^(j+1))*x
                done = True
            if (done == True):
                graph.append([(y),(z),(d)])
                break
        if (done == False):
            m = (i-(i%(6^(n-1))))/(6^(n-1))
            if (m == 1 or m == 2):
                graph.append([(y),(z),('A')])
            elif (m == 3 or m == 4):
                graph.append([(y),(z),('B')])
            elif (m == 5 or m == 0):
                graph.append([(y),(z),('C')])
        done = False
    return graph
```

B.2 Barycentric Energy and Gradient

```
def energy (g,f,p,n):
    Energy = 0
    for i in range (0,n):
        a = f[i]
        b = f[g[i][0]]
        c = f[g[i][1]]
        if (g[i][2]!= 'A'and g[i][2]!='B' and g[i][2]!='C'):
            d = f[g[i][2]]
            x = 1
        if (g[i][2]=='A'):
            d = 0
            x = 2
        if (g[i][2]=='B'):
            d = a
            x = 2
        if (g[i][2]=='C'):
            d = 1
            x = 2
        Energy = Energy + (1/2)*(abs(a-b))^p + (1/2)*(abs(a-c))^p + (1/2)*(abs(x*(a-d)))^p
    return (6*Energy)^(1/p)

def gradient (g,f,p,n):
    Gradient = []
    for i in range (0,n):
        a = f[i]
        b = f[g[i][0]]
        c = f[g[i][1]]
        if (g[i][2]!= 'A'and g[i][2]!='B' and g[i][2]!='C'):
            d = f[g[i][2]]
            x = 1
        if (g[i][2]=='A'):
            d = 0
            x = 2
        if (g[i][2]=='B'):
            d = a
            x = 2
        if (g[i][2]=='C'):
            d = 1
            x = 2
        Gradient = Gradient + [sgn(a-b)*p*(abs(a-b))^(p-1) + sgn(a-c)*p*(abs(c-a))^(p-1) + p*sgn(a-d)*(abs(x*(a-d)))^(p-1)]
    return Gradient
```

B.3 Barycentric Optimize

```
def optimizeGn(graph, p, n, number,startpt):
    f = startpt[:]
    k = .1
    g = gradient(graph, f, p, 6^n)
    m = []
    h = []
    for i in range (0,6^n):
        m.append(0)
        h.append(0)
    for j in range(1,number):
        for x in range (0,6^n):
```

```
            h[x] = g[x]
            m[x] = f[x]
            if (sgn(h[x])==1):
                if (f[x] == 0):
                    u = 100000
                else:
                    u = (f[x]/h[x])
            elif (sgn(h[x])==-1):
                if (f[x] == 1):
                    u = 100000
                else:
                    u = ((f[x]-1)/h[x])
            elif (h[x]==0):
                u = 10000
            if (u < k):
                k = u
        for i in range(0,6^n):
            f[i] = f[i] - k*h[i]
            if f[i] < 0:
                f[i] = 0
            if f[i] > 1:
                f[i] = 1
        g = gradient(graph, f, p, 6^n)
        k = stepsize (f,m,n,h,g,p)
        if (k == 0):
            return [energy(graph,f,p,6^n),f]
    return [energy(graph, f, p, 6^n),f]
```

B.4 Lattes Graph Generation

```
def connec (point, init, n):
    f = []
    state = init
    for i in range (0,n):
        f.append(((point - (point%(3^i)))/(3^i))%3)
        if (state == 1):
            if (f[i] == 0):
                f[i] = 1
                state = 0
            elif (f[i] == 1):
                f[i] = 0
                state = 0
            elif (f[i] == 2):
                f[i] = 2
                state = 3
        elif (state == 2):
            if (f[i] == 0):
                f[i]=0
                state = 3
            elif (f[i] == 1):
                f[i] = 2
                state = 0
            elif (f[i] == 2):
                f[i] = 1
                state = 0
        elif (state == 3):
            if (f[i] == 1):
                f[i] = 1
                state = 1
            elif (f[i] == 2):
                f[i] = 2
                state = 2
    if (state == 0):
        d = 0
        for i in range (0,n):
            d = d + f[i]*(3^i)
        return d
    elif (state == 1):
        return 'A'
    elif (state == 2):
        return 'B'
    elif (state == 3):
        return 'C'

def fn(n):
    graph = []
```

```
    for i in range (0,3^n):
        a = connec (i,1,n)
        b = connec (i,2,n)
        c = connec (i,3,n)
        graph.append([a,b,c])
    return graph
```

B.5 Lattes Core Functions

```
 def energy (g,f,p,n,tar):
    Energy = 0
    x = [0,0,0]
    for i in range (0,n):
        a = f[i][:]
        if (a[1] == 'A'):
            u = 0
        if (a[1] == 'B'):
            u = 1
        if (a[1] == 'C'):
            u = 2
        for j in range (0,3):
            if (g[i][j]!= 'A'and g[i][j]!='B' and g[i][j]!='C'):
                d = f[g[i][j]][:]
                if (d[1] == 'A'):
                    m = 0
                if (d[1] == 'B'):
                    m = 1
                if (d[1] == 'C'):
                    m = 2
                if (d[1]==a[1]):
                    x[j] = tar[m]*abs(a[0]-d[0])
                else:
                    x[j] = tar[u]*a[0]+tar[m]*d[0]
            if (g[i][j]=='A'):
                if (a[1]=='A'):
                    x[j] = tar[0]*(2-2*a[0])
                else:
                    x[j] = 2*tar[0]+2*a[0]*tar[u]
            if (g[i][j]=='B'):
                if (a[1]=='B'):
                    x[j] = tar[1]*(2-2*a[0])
                else:
                    x[j] = 2*tar[1]+2*a[0]*tar[u]
            if (g[i][j]=='C'):
                if (a[1]=='C'):
                    x[j] = tar[2]*(2-2*a[0])
                else:
                    x[j] = 2*tar[2]+2*a[0]*tar[u]
        Energy = Energy + (1/2)*(x[0])^p + (1/2)*(x[1])^p + (1/2)*(x[2])^p
    return (Energy)^(1/p)

def gradient (g,f,p,i,tar):
    x = [0,0,0]
    u = f[i][:]
    a = 0
    b = 0
    c = 0
    if (u[1] == 'A'):
        v = 0
    if (u[1] == 'B'):
        v = 1
    if (u[1] == 'C'):
        v = 2
    for j in range (0,3):
        if (g[i][j]!= 'A'and g[i][j]!='B'and g[i][j]!='C'):
            d = f[g[i][j]][:]
            if d[1] == 'A':
                m = 0
            if d[1] == 'B':
                m = 1
            if d[1] == 'C':
                m = 2
            if (d[1]==u[1]):
                x[j] = tar[m]*(u[0]-d[0])
            else:
                x[j] = tar[v]*u[0]+tar[m]*d[0]
```

```
            if (d[1]=='A'):
                a = a + (abs(x[j]))^(p-1)
            if (d[1]=='B'):
                b = b + (abs(x[j]))^(p-1)
            if (d[1]=='C'):
                c = c + (abs(x[j]))^(p-1)
        if (g[i][j]=='A'):
            if (u[1]=='A'):
                x[j] = tar[v]*(2*u[0]-2)
            else:
                x[j] = tar[0]*2+tar[v]*2*u[0]
            a = a + (abs(x[j]))^(p-1)
        if (g[i][j]=='B'):
            if (u[1]=='B'):
                x[j] = tar[v]*(2*u[0]-2)
            else:
                x[j] = tar[1]*2+tar[v]*2*u[0]
            b = b + (abs(x[j]))^(p-1)
        if (g[i][j]=='C'):
            if (u[1]=='C'):
                x[j] = tar[v]*(2*u[0]-2)
            else:
                x[j] = tar[2]*2+tar[v]*2*u[0]
            c = c + (abs(x[j]))^(p-1)
    if (a-b-c > 0):
        k = 'A'
    elif (b-a-c > 0):
        k = 'B'
    elif (c-a-b > 0):
        k = 'C'
    else:
        k = 'A'
    return([(sgn(x[0])*p*(abs(x[0]))^(p-1)+sgn(x[1])*p*(abs(x[1]))^(p-1)+p*sgn(x[2])*(abs(x[2]))^(p-1)),k])

def optimizeFn(graph, p, n, number, tar, startpt):
    s = startpt[:]
    g = []
    u = 100000
    for i in range (0,3^n):
        g.append([0,'A'])
    k = .1
    for l in range (0,3^n):
        g[l] = gradient(graph,s,p,l,tar)[:]
    m = [(0,0)]*(3^n)
    h = [(0,0)]*(3^n)
    for j in range(1,number):
        for x in range (0,3^n):
            h[x] = g[x][:]
            m[x] = s[x][:]
            if (sgn(g[x][0])==1):
                if (s[x][0] == 0):
                    u = 100000
                else:
                    u = (s[x][0]/g[x][0])
            elif (sgn(g[x][0])==-1):
                if (s[x][0] == 1):
                    u = 100000
                else:
                    u = ((s[x][0]-1)/g[x][0])
            if (u < k):
                k = u
        for i in range(0,3^n):
            if (s[i][0] == 0):
                if (s[i][1] != g[i][1]):
                    s[i][1] = g[i][1][:]
                    g[i] = gradient(graph,s,p,i,tar)[:]
        for r in range (0,3^n):
            s[r][0] = s[r][0] - k*g[r][0]
            if s[r][0] < 0:
                s[r][0] = 0
            if s[r][0] > 1:
                s[r][0] = 1
        for v in range (0,3^n):
            g[v] = gradient(graph,s,p,v,tar)[:]
        k = stepsize(s,m,n,h,g,p)
        if k == 0:
            return (energy(graph,s,p,3^n,tar),s)
```

```
    return (energy(graph,s,p,3^n,tar),s)
```

B.6 Lattes Tripod Optimization

```
 def lengths (graph, points, n, tar):
    length = []
    x = [0,0,0]
    for i in range (0,n):
        u = graph[i][:]
        if u[1] == 'A':
            v = 0
        if u[1] == 'B':
            v = 1
        if u[1] == 'C':
            v = 2
        for j in range (0,3):
            x[j]=graph[i][j]
            if (x[j] == 'A'):
                if (points[i][1]=='A'):
                    p = [tar[v]*(1-points[i][0]),'A']
                else:
                    p = [tar[0]+points[i][0]*tar[v],'A']
                length.append(p)
            if (x[j] == 'B'):
                if (points[i][1]=='B'):
                    p = [tar[v]*(1-points[i][0]),'B']
                else:
                    p = [tar[1]+tar[v]*points[i][0],'B']
                length.append(p)
            if (x[j] == 'C'):
                if (points[i][1]=='C'):
                    p = [tar[v]*(1-points[i][0]),'C']
                else:
                    p = [tar[2]+tar[v]*points[i][0],'C']
                length.append(p)
    return length

def tensionF (length,alpha,p):
    return (length/alpha)^(p-1)

def tensionsK (lengths):
    a = 0
    b = 0
    c = 0
    for i in range (0,len(lengths)):
        if (lengths[i][1]=='A'):
            a = a + lengths[i][0]
        if (lengths[i][1]=='B'):
            b = b + lengths[i][0]
        if (lengths[i][1]=='C'):
            c = c + lengths[i][0]
    return [a,b,c]

def newlengths (ten,p):
    x = [0,0,0]
    for i in range (0,3):
        x[i] = .5*(ten[i]^(1/(p-1)))
    return x

def optimizeFn(graph, p, n, number, tar, startpt):
    s = startpt[:]
    g = []
    u = 100000
    for i in range (0,3^n):
        g.append([0,'A'])
    k = .1
    for l in range (0,3^n):
        g[l] = gradient(graph,s,p,l,tar)[:]
    m = [(0,0)]*(3^n)
    h = [(0,0)]*(3^n)
    for j in range(1,number):
        for x in range (0,3^n):
            h[x] = g[x][:]
            m[x] = s[x][:]
            if (sgn(g[x][0])==1):
                if (s[x][0] == 0):
```

```
                u = 100000
            else:
                u = (s[x][0]/g[x][0])
        elif (sgn(g[x][0])==-1):
            if (s[x][0] == 1):
                u = 100000
            else:
                u = ((s[x][0]-1)/g[x][0])
        if (u < k):
            k = u
    for i in range(0,3^n):
        if ((s[i][0] == 0)):
            if (s[i][1] != g[i][1]):
                s[i][1] = g[i][1][:]
                g[i] = gradient(graph,s,p,i,tar)[:]
    for r in range (0,3^n):
        s[r][0] = s[r][0] - k*g[r][0]
        if s[r][0] < 0:
            s[r][0] = 0
        if s[r][0] > 1:
            s[r][0] = 1
    for v in range (0,3^n):
        g[v] = gradient(graph,s,p,v,tar)[:]
    k = stepsize(s,m,n,h,g,p)
    if k == 0:
        return (energy(graph,s,p,3^n,tar),s)
return (energy(graph,s,p,3^n,tar),s)
```

References

[1] Barzilai, J., Borwein, J. M. (January, 1988). "Two-Point Step Size Gradient Method". *IMA Journal of Numerical Analysis*. Vol 8. p 141-148.

[2] Kigami, J (2014). "Quasisymmetric modification of metrics on self-similar sets". *Geometry and Analysis of Fractals*. Hong Kong.

[3] Mackay, J. M., Tyson, J. T. (2010). "Conformal Dimension Theory and Application". *AMS University Lecture Series*. Volume 54. Providence, RI.

[4] Pilgrim, K., Thurston, D. "Graph Energies and Ahlfors-Regular Conformal Dimension". In preparation.

[5] Thurston, D (2016). "Elastic Graphs". Preprint. arXiv:1607.00340v2 [math.MG].

[6] Thurston,D (2016). "A positive characterization of rational maps". Preprint, 2016, arXiv:1612.04424.

NOTES ON CUT AND PASTE AND TOPOLOGICAL QUANTUM FIELD THEORIES

MATTHEW SCHOENBAUER

Contents

Date: Started July 4, 2017. Last revision July 28, 2017.
This work is supported by NSF REU Grant number 1461061.

Abstract. In these notes we shall be concerned with a relation between topological quantum field theories (TQFTs) and cut and paste invariants. I will give a full description of cut an paste invariants, cobordisms, and TQFTs in a categrocial context. The cut and paste invariants, or SK invariants, were studied in detail in the early 1970s by a group of four young authors: Karras, Kreck, Neuman, and Ossa. SK invariants are functions on the set of smooth manifolds that are invariant under the cutting and pasting operation, and include the signature and Euler characteristic. The four authors described these and weaker invariants, called SKK invariants, whose values on manifolds depend on both the cut and paste equivalence class and the gluing diffeomorphism. I will conclude with a presentation of my work investigating a surprisingly natural group homomorphism between the group of invertible TQFTs and the group of SKK invariants. This homomorphism has an easy to describe kernel. I will describe the image in certain dimensions by giving necessary and sufficient conditions for a diffeo-invariant to be naturally extended to an n-TQFT.

1. Assumed Knowledge

These notes assume that the reader is very familiar with general and algebraic topology. In particular, the signature and Euler characteristic are important concepts. In addition to this, the reader needs to understand the basic notions of a smooth manifold and its tangent space.

Throughout the notes, all manifolds are smooth unless stated otherwise.

2. Notation

Symbol	**Meaning**
H^n_+, H^n_-	The upper and lower closed half n-spaces
∂M	The boundary of a manifold M
$\overline{M}$	The oriented manifold M with the opposite orientation
I	The unit interval $[0,1]$
$\chi(M)$	The Euler characteristic of M
$\sigma(M)$	The signature of M
V^*	The dual space of the vector space V
$M \underset{f}{\cup} M'$	The manifold resulting from gluing M and M' via a diffeomorphism f of boundary components

3. Cut and Paste

In this section we will describe cutting and pasting operations on manifolds and a naturally related equivalence relation on manifolds called the "cut and paste" relation. We will also give examples of manifolds that are cut and paste equivalent and functions on the set of manifolds that are invariant under this relation, which we will call cut and paste invariants. We will conclude by imposing further structure on these equivalence classes to form the SK-groups.

3.1. **The Pasting Operation.**

Here we describe the process of pasting a manifold with boundary together via a diffeomorhpsim of boundary components. We will first describe topological pasting, which is simpler, and then we will handle the smooth case and the oriented smooth case, which require more care.

3.1.1. *Topological Pasting.*

Let M be a compact topological n-manifold (not necessarily connected) with boundary components Σ_1 and Σ_2. Let f be a homeomorphism $\Sigma_1 \to \Sigma_2$, and form a quotient space with quotient map q as follows: If $x \notin \Sigma_1 \cup \Sigma_2$, then $q(x) = \{x\}$. If $x \in \Sigma_1$, identify $q(x) = \{x, f(x)\}$.

We will now show that under the quotient topology, the resulting space is a compact topological manifold. Compactness follows from the compactness of M, so we need only show the each point has a neighborhood homeomorphic to $\mathbb{R}^n$.

Each point not in $q(\Sigma_1 \cup \Sigma_2)$ retains its original neighborhood homeomorphic to $\mathbb{R}^n$. Now suppose $x \in \Sigma_1$, and choose and neighborhood U of $q(x)$ so that $q^{-1}(U) = U_x \cup U_{f(x)}$, where U_x and $U_{f(x)}$ are neighborhoods of x and $f(x)$, respectively, and there exist homeomorphisms $\psi_1 : U_x \to H^n_+$ and $\psi_2 : U_{f(x)} \to H^n_-$. Now the function

$$\psi_2 \circ f \circ \psi_1^{-1} : \mathbb{R}^{n-1} \times \{0\} \to \mathbb{R}^{n-1} \times \{0\} \tag{3.1}$$

is clearly a homeomorphism. Now define a homeomorphism

$$h : H^n_+ \ (= \mathbb{R}^{n-1} \times [0, \infty)) \to H^n_+ \ (= \mathbb{R}^{n-1} \times [0, \infty)) \tag{3.2}$$

by $h(\mathbf{x}, x_n) = (\psi_2 \circ f \circ \psi_1^{-1}(\mathbf{x}), x_n)$.

The homeomorphisms

$$h \circ \psi_1 \circ q^{-1} : q(U_x) \to H^n_+ \tag{3.3}$$

and

$$\psi_2 \circ q^{-1} : q(U_{f(x)}) \to H^n_- \tag{3.4}$$

agree on the intersection of their domains, and similarly for their inverses. Thus they can be pasted together to form a homeomorphism $U \to \mathbb{R}^n$.

3.1.2. *Smooth Pasting.*

Now that we have figured out how to paste a manifold with boundary together topologically, we can describe the process of gluing together smooth structures on manifolds. We cannot repeat the above process here, because it is not guaranteed that the function resulting from the gluing of two smooth functions is smooth.

Of course our only real problem is finding a suitable smooth structure on neighborhoods of the boundary points: Interior points and their neighborhoods are essentially unchanged in the gluing process. In order to zoom in on this problem, we describe a "collar neighborhood" that each smooth manifold has near any of its boundary components. In order to do this, we state some results from Morse theory.

Theorem 3.1. *Let M be a smooth manifold. There exists a smooth function $f : M \to I$ such that $f^{-1}(\partial I) = \partial M$ and ∂M has no critical points. In addition, if M is compact, f has finitely many critical points. Such a function is called a Morse function.*

Theorem 3.2. *Let M be a compact manifold, and let f be a Morse function with no critical points. Then M is diffeomorphic to the cylinder $f^{-1}(0) \times I$.*

Corollary 3.3. *Let M be a compact smooth manifold with boundary component Σ. There exists a neighborhood U of Σ such that U is diffeomorphic to $\Sigma \times I$.*

We will refer to this neighborhood as a "collar neighborhood" of Σ.

Proof. This follows quickly from the preceding two theorems. Choose a Morse function $f : M \to I$ and choose ϵ so that $f^{-1}([0, \epsilon])$ has no criticial points. The component of $f^{-1}([0, \epsilon])$ that contains Σ is the desired neighborhood. □

By the above statements, it suffices to find a way to glue the smooth structures of two cylinders in a way that makes the resulting cylinder a smooth manifold without interfering with the smooth structures of the original cylinders. Once this is complete, we will have chart neighborhoods of each point in our glued manifold that satisfy the smooth coordinate change criterion.

Let Σ be a closed manifold. We wish to find a smooth structure on the manifold obtained by gluing $\Sigma \times [0, 1]$ and $\Sigma \times [1, 2]$ via a diffeomorphism $f : \Sigma \times \{1\} \to \Sigma \times \{1\}$. Such a cylinder is called the *mapping cylinder* of f. But this manifold is homeomorphic to $\Sigma \times [0, 2]$, which has an obvious smooth structure. We can now give the pasted cylinder the smooth structure inherited by $\Sigma \times [0, 2]$ via this homeomorphism. In addition to this, the restrictions of this homeomorphism to $\Sigma \times [0, 1]$ and $\Sigma \times [1, 2]$ are diffeomorphisms, so the smooth structure of the interior points of $\Sigma \times [0, 1]$ and $\Sigma \times [1, 2]$ are unchanged.

We now have a well-defined process of gluing M via $f : \Sigma_1 \to \Sigma_2$. We will call the resultant manifold M_f.

3.2. **Oriented Smooth Pasting.** Now that we have given a smooth structure to a pasted manifold, we must ensure that we can give it an orientation that is coherent with the orientations of the original manifolds. The necessary definitions are given below.

Definition 3.4. Let V be a finite-dimensional vector space. Form an equivalence relation on the set of all ordered bases of V, declaring $[\mathbf{v}_1 \dots \mathbf{v}_n] \sim [\mathbf{w}_1 \dots \mathbf{w}_n]$ if the determinant of the linear map $\mathbf{v}_i \to \mathbf{w}_i$ is positive. There are clearly two such equivalence classes. An *orientation* on V is a choice of one of these classes. A representative of this class is called a *positive basis* of V.

Definition 3.5. Let M be a smooth manifold. Suppose we can choose orientations of each of the tangent spaces of M so that all coordinate change maps send positive bases to positive bases. In this case, M is said to be *orientable*, and this choice of bases is said to be an *orientation* of M.

Before we can deal with the problem of pasting oriented smooth manifolds via the boundary, we must uniformly describe the way in which an oriented manifold M induces an orientation on its boundary ∂M, which is a manifold in its own right.

Since since orientations are defined locally, and locally every manifold with boundary is H^n_+, it suffices to describe the induced orientation only in the case of H^n_+, which has boundary $\mathbb{R}^{n-1} \times \{\mathbf{0}\}$. Give H^+_n an orientation, and let $\mathbf{e}_i$ be the vector in $\mathbb{R}_n$ with i^{th} coordinate 1 and all other coordinates 0. $[\mathbf{e}_1, \ldots \mathbf{e}_{n-1}]$ be a basis for $\mathbb{R}^{n-1} \times \{0\}$, which is the tangent space of $\mathbf{0} \in \mathbb{R}^{n-1} \times \{0\}$. $[\mathbf{e}_1, \ldots \mathbf{e}_{n-1}]$ is declared to be a positive basis if $[\mathbf{e}_1, \ldots \mathbf{e}_{n-1}, \mathbf{e}_n]$ is a positive basis in the orientation of H^n_+. It is not hard to check that this definition is coherent with coordinate changes. In this way, the orientation of H^n_+ determines the orientation of $\mathbb{R}^{n-1} \times \{\mathbf{0}\}$, and vice versa.

Now we can deal with the gluing problem, which is locally the case of gluing H^+_n to H^n_-. It is natural to think that we should give ∂H^n_+ and ∂H^n_- the same orientation. In this case the gluing diffemorphism (the identity) is orientation-preserving. But if this is the case, then

$$[\mathbf{e}_1, \ldots \mathbf{e}_n] \text{ and } [\mathbf{e}_1, \ldots - \mathbf{e}_n] \tag{3.5}$$

are both positive bases for the tangent space at $\mathbf{0}$, even though the determinant of the map between these bases is -1. If we give H^n_- and ∂H^n_- the opposite orientation this problem is solved.

Thus when gluing two manifolds M and N via an orientation-preserving diffeomorphism $f : \partial M \to \partial N$, the resulting manifold is

$$M \underset{f}{\cup} \overline{N}. \tag{3.6}$$

We can just as well glue two oriented manifolds via an orientation-reversing diffeomorphism of the boundaries; in this case we obtain an oriented manifold by simply not flipping the orientation of either of the original manifolds.

3.3. **The Cut and Paste Relation.**

We are now prepared to describe an equivalence relation on compact manifolds.

Definition 3.6. Let M be a compact oriented (not necessarily connected) manifold with boundary $\Sigma_1 \coprod \Sigma_2$. Let f and g be orientation-preserving diffeomorphisms $\Sigma_1 \to \overline{\Sigma_2}$. We say that M_f, M_g, and all manifolds diffeomorphic to either are *cut and paste equivalent* or *SK equivalent*, and write $M_f \sim_{SK} M_g$.

The motivation for the choice of the term "cut and paste" should be clear. We can picture obtaining M_f from M_g by cutting along a codimension-1 submanifold Σ_1 of M_f and gluing it back together in another way.

Also, "SK" stands for "schneiden" and "kleben," the German words for "to cut" and "to paste."

It is even more visually appealing to think of the cutting process as cutting the manifold into two pieces. This need not be the case; for example cutting the torus $S^1 \times S^1$ along $S^1 \times \{0\}$ results in a space that is diffeomorphic to a cylinder, hence connected. However, we can allow ourselves to think of the situation as cutting the manifold into two components, for the following reason: M_f in the above scenario has a one-sided collar "neighborhood" of Σ_1, i.e. a collar neighborhood in M. Considering this neighborhood as $\Sigma_1 \times I$, we can cut M_f along the set corresponding to $\Sigma_1 \times \{1/2\}$, and during the gluing step glue this back together via the identity. This process summarizes the proof of the following proposition, which will be important for our upcoming endeavors.

Proposition 3.7. *Two closed, connected oriented manifolds N and N' are cut and paste equivalent if and only if there exists a closed, connected, oriented n-manifold M with $(n-1)$-submanifold Σ and n-submanifolds M_1, M_2 of M each with boundary Σ such that N and N' are obtained by gluing the boundaries of M_1 and M_2 via orientation-preserving diffeomorphisms of their boundaries.*

3.4. **Examples.**

We now give two examples of cut and paste equivalent manifolds.

Example 3.8. Consider the unit circle $S^1 \subset \mathbb{R}^2$ oriented counterclockwise. Cut along the points $\mathbf{p}_1 = (1,0)$, $\mathbf{p}_2 = (0,1)$, $\mathbf{p}_3 = (-1,0)$, and $\mathbf{p}_4 = (0,-1)$. We now have four line segments with the orientations indicated in Figure 1.

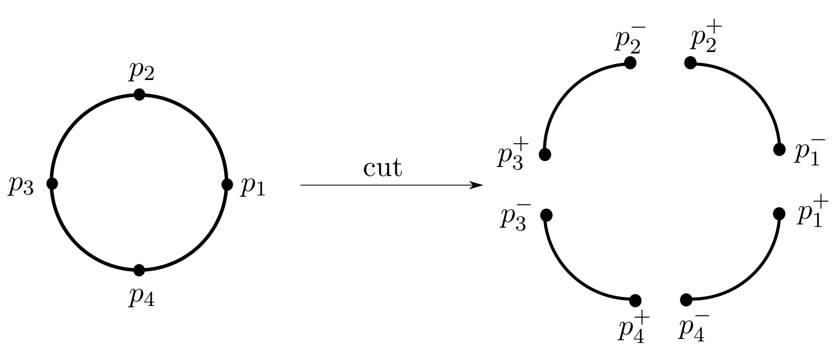


Figure 1. Cutting a circle

If we paste the boundaries via the diffeomorphism indicated in Figure 2, we obtain two circles $S^1 \coprod S^1$.

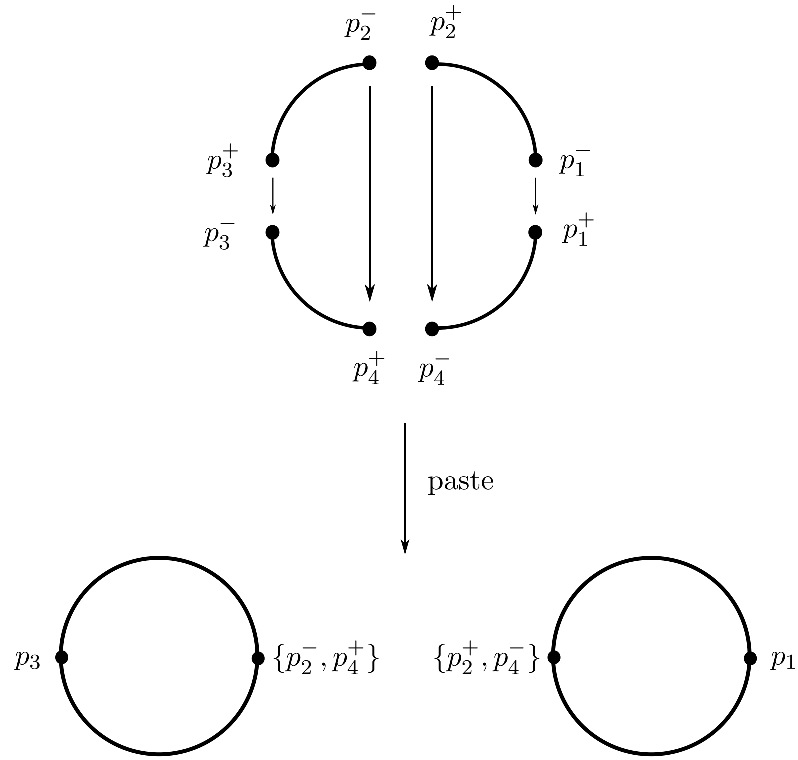


FIGURE 2. Gluing a circle

Thus $S^1 \sim_{SK} S^1 \coprod S^1$.

Example 3.9. Now we extend the ideas of both previous examples to construct a rather interesting cut and paste equivalence. Let Σ be a closed manifold, and let $f : \Sigma \to \Sigma$ be an orientation-preserving diffeomorphism. Paste together the boundary components of $\Sigma \times I$ via f. The resulting space is called the *mapping torus* of Σ and f, which we will denote $T_{\Sigma,f}$. T

Now let $T_{\Sigma,f}$ be any mapping torus. By cutting along the pasting submanifold and the submanifolds corresponding to $\Sigma \times \{1/4\}$, $\Sigma \times \{1/2\}$, and $\Sigma \times \{3/4\}$, and pasting in the a fashion similar to Example 3.8 and indicated in Figure 3, we see that $T_{\Sigma,f} \sim_{SK} T_{\Sigma,f} \coprod T_{\Sigma,f}$.

3.5. **Cut and Paste Invariants.**

We are now suitably prepared to define and give examples of cut and paste invariants. Let $\mathbb{M}_n$ denote the set of all closed oriented n-manifolds, and let $M, N \in \mathbb{M}_n$.

Definition 3.10. Let G_* be a graded abelian group with n^{th} group G_n. A function $\Theta : \mathbb{M}_n \to G_n$ for all n is said to be a *cut and paste invariant* or *SK invariant* if the following hold.

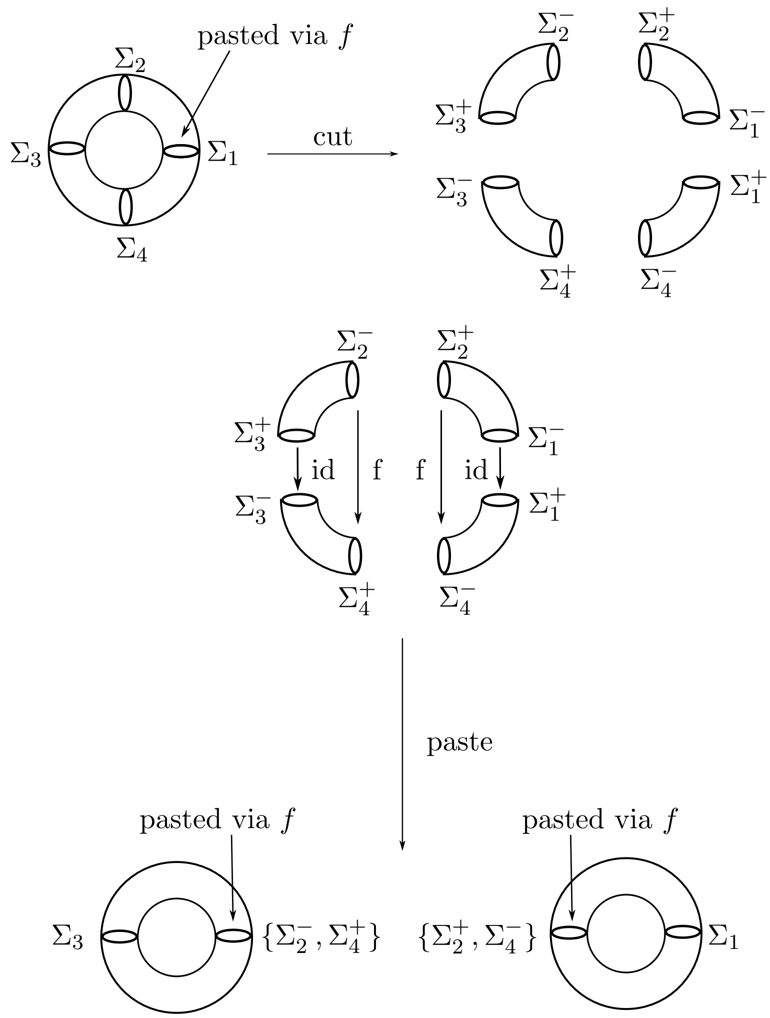


FIGURE 3. Cutting and pasting of mapping torus

- $\Theta(M) = \Theta(N)$ whenever M and N are cut and paste equivalent
- $\Theta(M \coprod N) = \Theta(M) + \Theta(N)$

The second criterion suggests that invariants that are determined by homology and cohomology will be good candidates to be cut and paste invariants. In fact this will be the case with all examples.

One important example of a cut and paste invariant is the Euler characteristic. The Euler characteristic maps each $\mathbb{M}_n$ to $\mathbb{Z}$, so the graded group we are concerned has $\mathbb{Z}$ as all its coefficient groups. To show that it is a cut and paste invariant, recall that the Euler characteristic satisfies

$$\chi(X \cup Y) = \chi(X) + \chi(Y) - \chi(X \cap Y) \tag{3.7}$$

if X and Y are open in $X \cup Y$. This shows that χ meets the second criterion above.

Also recall that since it is formulated by the homology of the space, χ is a homotopy type invariant. Now let N and N' be cut and paste equivalent n-manifolds, and choose M_1, M_2, and Σ as in Proposition 3.7. Since Σ has a collar neighborhood in M_1 and M_2, we have

$$\chi(N) = \chi(M_1) + \chi(M_2) - \chi(\Sigma) = \chi(N'). \tag{3.8}$$

Thus χ meets the first criterion above. The proof is simplified if we recall that the Euler characteristic of a closed odd-dimensional manifold is always zero. Thus the only case we really need to consider is the even-dimensional case. In this case, the intersection term vanishes because it has the homotopy type of a closed odd-dimensional manifold.

Another cut and paste invariant is the signature. That it is a cut and paste invariant follows from a similar argument as above, replacing Equation 3.7 with the Novikov additivity property (See [AS68])

$$\sigma(X \cup Y) = \sigma(X) + \sigma(Y). \tag{3.9}$$

The importance of the two preceding examples is firmly established by the following theorem.

Theorem 3.11. [KKNO73, p. 7] *Any SK invariant is a linear combination of the signature and the Euler characteristic.*

We can also define a weaker class of invariants, which will be especially interesting for later use.

Definition 3.12. Let G_* be a graded abelian group with n^{th} group G_n. A function $\xi : \mathbb{M}_n \to G_n$ for all $n \geq 0$ is said to be an *SKK invariant* if the following hold.

- $\xi(M) - \xi(N) = \xi(f, g)$
- $\xi(M \coprod N) = \xi(M) + \xi(N)$

In the first equation, M and N are cut and paste equivalent, and f and g are the gluing diffeomorphisms of M and N, respectively. $\xi(f, g)$ is a function that depends only on f and g.

Here the second "K" stands for kontrollierbar, the German word for "controllable."

If M and N are diffeomorphic, then $\xi(M) = \xi(N)$ for all SKK invariants ξ. Show this, it suffices to show that $\xi(f, f) = 0$. But this follows from $\xi(M) - \xi(M) = 0$.

It is clear from the definitions that every SK invariant is an SKK invariant. For an SK invariant, the "error" function is always zero.

We give one example of an SKK invariant. We give another example, along with a complete classification of SKK invariants in Section 5.

Definition 3.13. Let M be an n-dimensional manifold. We define the *Kervaire semicharacteristic* $\chi_{1/2}$ by

$$\chi_{1/2}(M) = \begin{cases} \frac{1}{2}\chi(M) & n \text{ even} \\ \big(\sum_{i=0} \operatorname{rank} H_{2i}(M)\big) \bmod 2 & n \text{ odd.} \end{cases} \tag{3.10}$$

See [KKNO73, p. 44] for a proof that $\chi_{1/2}$ is an SKK invariant in dimensions $4n+1$. It is an SKK invariant in all dimensions other than dimensions $4n-1$.

Ignoring dimensions $4n-1$, the Kervaire semicharacteristic is a good example of an SKK invariant for which the graded group G_* is not just copies of the same group.

3.6. The SK and SKK Groups.

We now describe a group structure on the set $\mathbb{M}_n^{\S}$ of all diffeomorphism classes of closed oriented n-manifolds. We then form quotient groups of this group that will allow us to give more concise and useful definitions of SK and SKK invariants. We first introduce an important algebraic tool.

Definition 3.14. Let $\mathcal{M}$ be a commutative monoid. Let $\mathcal{G}(\mathcal{M})$ be the abelian group whose elements are all symbols m and $-m$, where $m \in \mathcal{M}$, subject to all the relations of $\mathcal{M}$ in addition to $(-m)+(-m') = -(m+m')$ and $m + (-m) = 0$. $\mathcal{G}(\mathcal{M})$ is called the *Grothendieck group* of $\mathcal{M}$.

$\mathcal{G}(\mathcal{M})$ is clearly the minimal abelian group that contains $\mathcal{M}$ and preserves its relations. For convenience, define $\Theta(-M) = -\Theta(M)$ for all monoid homomorphisms $\Theta : \mathcal{M} \to G$, where G is a group.

We take advantage of this construction to form an abelian group that contains $\mathbb{M}_n^{\S}$. $\mathbb{M}_n^{\S}$ is obviously a commutative monoid under the operation $\coprod$ with identity element $\emptyset$. Thus $\mathcal{G}(\mathbb{M}_n^{\S})$ is an abelian group. We are now ready to form our quotient group.

Definition 3.15. Let R_n^{SK} denote the subgroup of $\mathcal{G}(\mathbb{M}_n^{\S})$ generated by all elements of the form $[M] \coprod -[M']$ where $[M] \sim_{SK} [M']$. $\mathcal{G}(\mathbb{M}_n^{\S})/R_n^{SK} = SK_n$ is called the n^{th} *SK group*.

We will let $[M]_{SK}$ denote the SK equivalence class of the manifold M. The examples of Section 3.4 show that S^1 and any other mapping torus represent the identity element in the respective SK group.

This definition give us a more concise way of defining SK invariants.

Definition 3.16. Let G_* be a graded abelian group with n^{th} group G_n. An SK invariant is an element of the graded group $\operatorname{Hom}(SK_n, G_n)_*$.

Obviously all SK invariants are homomorphisms $\mathcal{G}(\mathbb{M}_n^{\S}) \to G_n$ for all n, and by definition they are the only such homomorphisms that vanish on R_n^{SK}.

We can define SK_* to be the graded group whose n^{th} coordinate group is SK_n. $\mathrm{Hom}\,(SK_n, G_n)_*$ is then canonically isomorphic to the group $\mathrm{Hom}\,(SK_*, G_*)$ of homomorphisms of graded groups.

We would like to construct a quotient group of $\mathcal{G}(\mathbb{M}_n^{\S})$ that serves the same purpose for SKK invariants. How can this be done? The stated condition

$$\xi([M]) - \xi([N]) = \xi(f, g) \tag{3.11}$$

when $[M] \sim_{SK} [N]$ and f and g are the gluing diffeomorphisms of $[M]$ and $[N]$, respectively, is equivalent to the condition

$$\xi([M]) - \xi([N]) = \xi([M']) - \xi([N']) \tag{3.12}$$

when $[M'] \sim_{SK} [N']$ and f and g are also the gluing diffeomorphisms of $[M']$ and $[N']$, respectively. This leads to the following definition.

Definition 3.17. Let R_n^{SKK} denote the subgroup of $\mathcal{G}(\mathbb{M}_n^{\S})$ generated by all elements of the form

$$[M] \coprod -[N] \coprod -\big([M'] \coprod -[N']\big), \tag{3.13}$$

where $[M] \sim_{SK} [N]$, $[M'] \sim_{SK} [N']$ and the gluing diffeomorphisms are given by the above description. $\mathcal{G}(\mathbb{M}_n^{\S})/R_n^{SKK} = SKK_n$ is called the n^{th} *SKK group*.

Just as in the case of SK_n, we will let $[M]_{SKK}$ denote the SKK equivalence class of a closed manifold M and write $M \sim_{SKK} M'$ when $[M]_{SKK} = [M]_{SKK}$. We now get our desired definition of the SKK invariants.

Definition 3.18. Let G_* be a graded abelian group with n^{th} group G_n. An SK invariant is an element of the graded group $\mathrm{Hom}\,(SKK_n, G_n)_*$.

Defining the graded group SKK_* analogously to SK_*, we see again that $\mathrm{Hom}\,(SKK_n, G_n)_*$ is canonically isomorphic to the group $\mathrm{Hom}\,(SKK_*, G_*)$ of homomorphisms of graded groups.

We give two examples of nontrivial SKK equivalence. Both use a construction on manifolds with boundary called a "double," which is defined below.

Definition 3.19. Let M be a compact oriented manifold with boundary. We define the *double* of M to be the closed manifold

$$D(M) = M \coprod M. \tag{3.14}$$

Our first example matches nicely with Example 3.9.

Example 3.20. Let Σ be a closed $(n-1)$-manifold that bounds. We show that $[\Sigma \times S^1]_{SKK} = 0 \in SKK_n$.

Recall that

$$[M]_{SKK} - [N]_{SKK} = [M']_{SKK} - [N']_{SKK} \tag{3.15}$$

When $[M]_{SK} = [N]_{SK}$, $[M']_{SK} = [N']_{SK}$, and M and N are glued, respectively, via the same diffeomorphisms as M' and N'. We make use of this relation to prove the lemma.

Cut $\Sigma \times S^1$ along four diffeomorphic submanifolds Σ_1, Σ_2, Σ_3, Σ_4 as in Example 3.9. The resulting manifold has

$$\Sigma_1^+ \coprod \Sigma_1^- \coprod \Sigma_2^+ \coprod \Sigma_2^- \coprod \Sigma_3^+ \coprod \Sigma_3^- \coprod \Sigma_4^+ \coprod \Sigma_4^- \tag{3.16}$$

as its boundary. Paste via the diffeomorphism

$$f = \begin{cases} \Sigma_1^+ \to \Sigma_1^- \\ \Sigma_2^+ \to \Sigma_4^- \\ \Sigma_3^+ \to \Sigma_3^- \\ \Sigma_4^+ \to \Sigma_2^- \end{cases} \tag{3.17}$$

which is the identity on each component. The resulting manifold is clearly $\Sigma \times S^1 \coprod \Sigma \times S^1$.

Now paste the cut manifold via the diffeomorphism

$$g = \begin{cases} \Sigma_1^+ \to \Sigma_1^- \\ \Sigma_2^+ \to \Sigma_2^- \\ \Sigma_3^+ \to \Sigma_3^- \\ \Sigma_4^+ \to \Sigma_4^-, \end{cases} \tag{3.18}$$

The resulting manifold is $\Sigma \times S^1$.

We thus have two SK equivalent manifolds $\Sigma \times S^1 \coprod \Sigma \times S^1$ and $\Sigma \times S^1$, where the gluing diffeomorphisms are f and g, respectively. This is exactly the process we went through in Example 3.9, where all maps are the identity. Now we take a different turn, and cut and paste a different manifold in an analogous way in order to make use of Equation 3.15.

Choose a manifold M with $\partial M = \Sigma$. Take two copies of M and two copies of $\overline{M}$, and cut one of each along a submanifold diffeomorphic to Σ, taking advantage of the collar neighborhood of Σ. The resulting manifold, pictured in Figure 4, is diffeomorphic to

$$M \coprod (\Sigma \times I) \coprod \overline{M} \coprod (\overline{\Sigma} \times I), \tag{3.19}$$

and thus has the expression 3.16 as its boundary.

Now by pasting via the diffeomorphisms f and g, one sees that the glued manifolds are both

$$D(M) \coprod D(M) \tag{3.20}$$

Now we apply Equation 3.15. We have

$$\begin{aligned} [\Sigma \times S^1 \coprod \Sigma \times S^1]_{SKK} - [\Sigma \times S^1]_{SKK} &= [D(M) \coprod D(M)]_{SKK} \\ &\quad - [D(M) \coprod D(M)]_{SKK} \\ [\Sigma \times S^1]_{SKK} &= 0, \end{aligned}$$

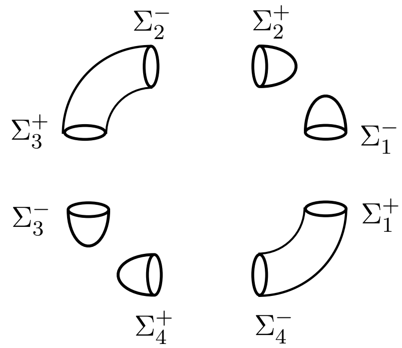


FIGURE 4. The cut manifold

as desired.

Example 3.21. Let M be a closed n-manifold. Cut M along a codimension-1 submanifold Σ with trivial normal bundle that divides M into two manifolds M' and M'', as in Figure 5.

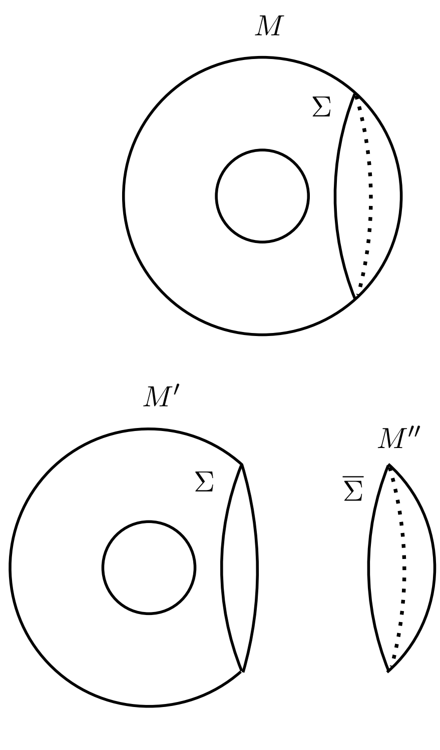


FIGURE 5. Cutting M

We will show that

$$[D(M')]_{SKK} + [D(\overline{M''})]_{SKK} = 2[M]_{SKK}. \tag{3.21}$$

This relation is pictured in Figure 6.

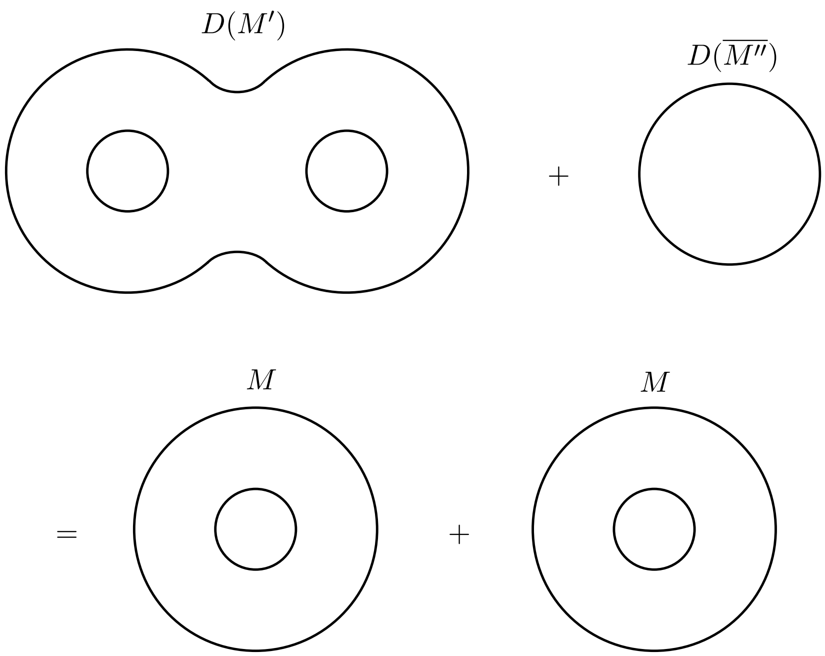


FIGURE 6. Our desired relation in SKK_n

We follow the same procedure as Example 3.20. We start with two "cut" manifolds with boundary, and glue each in two different ways. This will allow us to apply Equation 3.15 and prove the lemma. Our starting manifolds will be

$$M_1^* = M' \coprod \overline{M''} \coprod M' \coprod \overline{M''} \tag{3.22}$$

and

$$M_2^* = M' \coprod \overline{M''} \coprod \overline{M''} \coprod \overline{M''}. \tag{3.23}$$

Both have

$$\Sigma_1 \coprod \Sigma_2 \coprod \Sigma_3 \coprod \Sigma_4 \tag{3.24}$$

as their boundaries. The two gluing diffeomorphisms are

$$f = \begin{cases} \Sigma_1 \to \Sigma_3 \\ \Sigma_2 \to \Sigma_4 \end{cases} \tag{3.25}$$

$$g = \begin{cases} \Sigma_1 \to \Sigma_2 \\ \Sigma_3 \to \Sigma_4 \end{cases} \tag{3.26}$$

Gluing M_1^* via f, we obtain

$$D(M') \coprod D(\overline{M''}). \tag{3.27}$$

Gluing M_1^* via g, we obtain

$$M \coprod M. \tag{3.28}$$

Gluing M_2^* via f, we obtain

$$M \coprod D(\overline{M''}). \tag{3.29}$$

Gluing M_2^* via g, we obtain

$$M \coprod D(\overline{M''}). \tag{3.30}$$

You can visualize this process with the following sequence of figures.

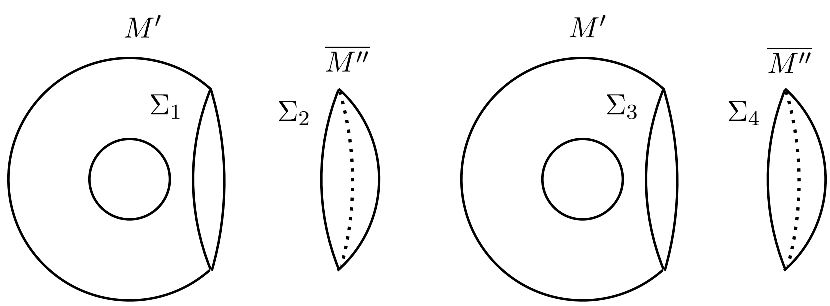


Figure 7. M_1^*

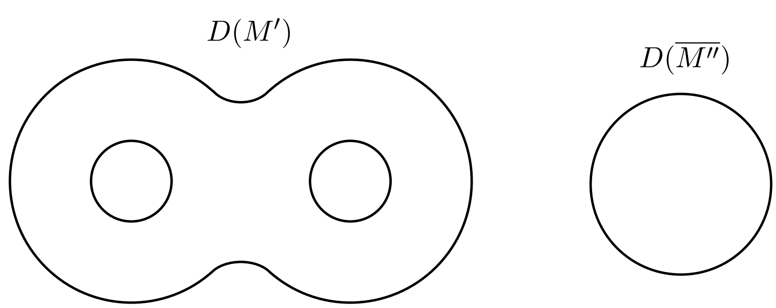


Figure 8. M_1^* glued via f

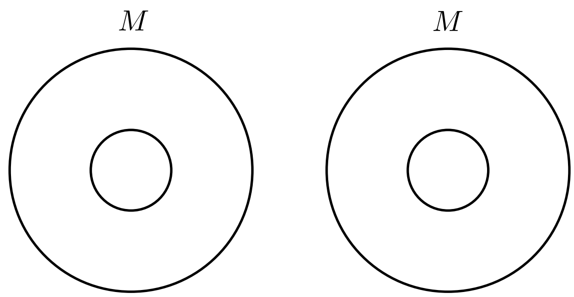


FIGURE 9. M_1^* glued via g

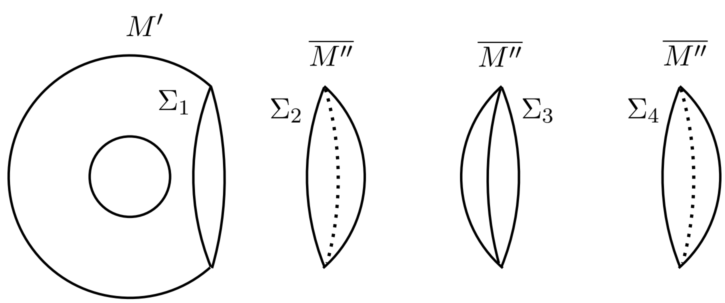


FIGURE 10. M_2^*

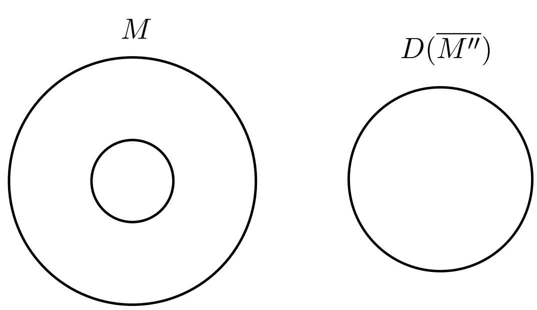


FIGURE 11. M_2^* glued via f

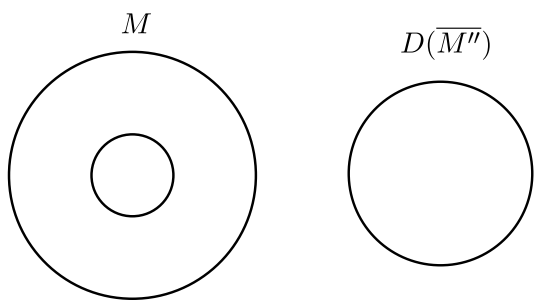


FIGURE 12. M_2^* glued via g

Now we apply Equation 3.15. We remove the $_{SKK}$ subscripts. We have

$$\begin{aligned}
[D(M')] + [D(\overline{M''})] - 2[M] &= [M] + [D(\overline{M''})] - [M] - [D(\overline{M''})] \\
[D(M')] + [D(\overline{M''})] - 2[M] &= 0 \qquad (3.31) \\
[D(M')] + [D(\overline{M''})] &= 2[M],
\end{aligned}$$

as desired.

We stated earlier that all SK invariants are SKK invariants. Let us make sense of this statement in terms of our new definition of these invariants.

In a general algebraic context, note that for any abelian groups G, H, and K with $K < H$, $\operatorname{Hom}(H/K, G)$ can be isomorphically embedded in $\operatorname{Hom}(H, G)$. The monomorphism is defined as follows: Let ϕ be a homomorphism $H/K \to G$, and let π be the quotient map $H \to H/K$. Then $\phi \circ \pi$ is a homomorphism $H \to G$. Also, if $\phi \circ \pi$ is the zero homomorphism, then $\phi = 0$ because π is surjective. Thus we have a monomorphism $\operatorname{Hom}(H/K, G) \to \operatorname{Hom}(H, G)$.

Now we note that SK_* is a quotient group of SKK_*. This follows from the fact that $R_n^{SKK} < R_n^{SK}$. Indeed every element in R_n^{SKK} of the above form

$$[M] \coprod -[N] \coprod -\big([M'] \coprod -[N']\big) \qquad (3.32)$$

can be written as

$$\big([M] \coprod -[M']\big) \coprod -\big([N] \coprod -[N']\big). \qquad (3.33)$$

The left parenthetical term has gluing diffeomorphism $f \coprod f$ and is SK equivalent to the right parenthetical term, which has gluing diffeomorphism $g \coprod g$. This is clearly an element of R_n^{SK}.

Now let $H_n = SK_n$ and $K_n = R_n^{SK}/R_n^{SKK} < SK_n$. Then $\operatorname{Hom}(H_n/K_n, G)_*$ can be embedded in $\operatorname{Hom}(H_n, G)_*$, which algebraically expresses the fact that all SK invariants are SKK invariants.

4. Category Theory

We now shift our focus to category theory, which will for a while seem unrelated to the previous section. Category theory has a level of abstraction and generality that transcends the rest of mathematics. Indeed most mathematical constructs serve as examples of ideas in category theory.

This section will include mostly definitions, all of which will be useful for the upcoming sections.

Definition 4.1. A *category* $\mathbf{C}$ consists of

- A class of objects $\mathbf{C_0}$
- For each pair (X, Y) with $X, Y \in \mathbf{C_0}$, a set $\mathbf{C}(X, Y)$ of arrows $X \to Y$
- An associative law of composition of arrows $\mathbf{C}(X, Y) \times \mathbf{C}(Y, Z) \to \mathbf{C}(X, Z)$
- For each $X \in \mathbf{C}$ an identity arrow $id_X \in \mathbf{C}(X, X)$ with $a \circ id_X = a$ and $id_X \circ b = b$ for all $a \in \mathbf{C}(X, Y)$ and $b \in \mathbf{C}(Y, X)$

Again, this probably seems rather abstract, so we clarify with a list of examples given in the following chart.

Category	**Objects**	**Arrows**	**Identity arrow**
Set	Sets	Set maps	Identity map
Man	Smooth manifolds	Smooth maps	Identity map
Top	Topological spaces	Continuous maps	Identity map
Grp	Groups	Group homomorphisms	Identity homomorphism
$\mathbf{Vect}_{\mathbb{k}}$	$\mathbb{k}$-Vector spaces	Linear maps	Identity map
FinOrd	Finite ordered sets	Order-preserving maps	Identity map

As you can see, many common mathematical objects are categories. It should be clear from this that categories are something worth studying.

Note that after reading the definition of a category it is tempting to think of categories as just collections of sets along with maps between the sets. The chart shows that this is not the case, as the objects in general have more structure than just a set. All of the above categories have a notion of isomorphism, which we define properly below.

Definition 4.2. Let $\mathbf{C}$ be a category and $X, Y \in \mathbf{C_0}$. An arrow $a \in \mathbf{C}(X, Y)$ is called an *isomorphism* is there is an arrow $a^{-1} \in \mathbf{C}(Y, X)$ with $a \circ a^{-1} = id_X$ with $a^{-1} \circ a = id_Y$.

Below chart of isomorphisms that corresponds to the above chart of examples of categories.

Category	**Isomorphism**
Set	Bijections
Man	Diffeomorphisms
Top	Homeomorphisms
Grp	Group Isomorphisms
$\mathbf{Vect}_{\mathbb{k}}$	Vector space isomorphisms
FinOrd	Order-preserving bijections

It might seems strange that all the arrows in the both charts are just certain kinds of maps. Why, then, would we give them a name other than "maps"? Well, in general there are plenty of categories whose arrows are not maps at all. In the next section we will see an example of such a category.

We now present an object that serves as a sort of "function" between categories.

Definition 4.3. A *functor* $\mathbf{F}$ between two categories $\mathbf{C}$ and $\mathbf{D}$ consists of a map $\mathbf{C_0} \to \mathbf{D_0}$ and for each $X \in \mathbf{C_0}$ a map $\mathbf{F} : \mathbf{C}(X,Y) \to \mathbf{D}(F(X), F(Y))$ satisfying

- $\mathbf{F}(a) \circ \mathbf{F}(b) = \mathbf{F}(a \circ b)$ for $a \in \mathbf{C}(X,Y)$ and $b \in \mathbf{C}(Y,Z)$
- $\mathbf{F}(id_X) = id_{\mathbf{F}(X)}$

Here are some examples of functors.

Example 4.4. "Forgetful" functors are functors that disregard structure. For example, the functor $\mathbf{Man} \to \mathbf{Top}$ that disregards smooth structure is forgetful. Other forgetful functors are $\mathbf{Top} \to \mathbf{Set}$, $\mathbf{Vect}_{\mathbb{k}} \to \mathbf{Grp}$, and $\mathbf{FinOrd} \to \mathbf{Set}$.

Example 4.5. A less trivial example is the functor $\mathbf{Set} \to \mathbf{Vect}_{\mathbb{k}}$ that assigns to each set the free $\mathbb{k}$-vector space whose basis is that set, and to each set map the linear map formed by the unique linear extension of the set map.

We now plunge further into the abstraction and define another sort of "function" on between functors themselves.

Definition 4.6. Let $\mathbf{C}$ and $\mathbf{D}$ be categories, and $\mathbf{F}$ and $\mathbf{G}$ be functors $\mathbf{C} \to \mathbf{D}$. A *natural transformation* $u : \mathbf{F} \to \mathbf{G}$ consists of an arrow $u(X) \in \mathbf{D}(\mathbf{F}(X), \mathbf{G}(X))$ for each $X \in \mathbf{C_0}$ such that the following diagram commutes for each $X, Y \in \mathbf{C_0}$ and $a \in \mathbf{C}(X,Y)$.

$$\begin{tikzcd}
\mathbf{F}(X) \arrow[r, "u(X)"] \arrow[d, "\mathbf{F}(a)"'] & \mathbf{G}(X) \arrow[d, "\mathbf{G}(a)"] \\
\mathbf{F}(Y) \arrow[r, "u(Y)"] & \mathbf{G}(Y)
\end{tikzcd}$$

Functors and their natural transformations behave similarly to categories and functors. This similarity is clarified with the following definition.

Definition 4.7. Let $\mathbf{C}$ and $\mathbf{D}$ be categories. The *functor category* of $\mathbf{C}$ and $\mathbf{D}$, denoted $\mathbf{Cat}(\mathbf{C}, \mathbf{D})$ has as its objects the functors $\mathbf{C} \to \mathbf{D}$, and natural transformations of functors as its arrows. The associative composition law is the regular composition of arrows, and the identity natural transformation consists only of identity arrows.

We will be interested in certain types of categories called symmetric monoidal categories. In order to properly define such categories, we must give a number of preliminary definitions.

Definition 4.8. Let $\mathbf{C}$ and $\mathbf{D}$ be categories. The *Cartesian product category* $\mathbf{C} \times \mathbf{D}$ is the category with objects $X_{\mathbf{C}} \times X_{\mathbf{D}}$ where $X_{\mathbf{C}} \in \mathbf{C_0}$ and $X_{\mathbf{D}} \in \mathbf{D_0}$ and arrows $a_{\mathbf{C}} \times a_{\mathbf{D}} : X_{\mathbf{C}} \times X_{\mathbf{D}} \to Y_{\mathbf{C}} \times Y_{\mathbf{D}}$ where $a_{\mathbf{C}} \in \mathbf{C}(X_{\mathbf{C}}, Y_{\mathbf{C}})$ and $a_{\mathbf{D}} \in \mathbf{D}(X_{\mathbf{D}} \times Y_{\mathbf{D}})$.

This definition may seem unnecessary, but remember that categories are not sets, and that the usual definition of Cartesian product does not apply here. It is also needed for the following definition.

Definition 4.9. A *monoidal category* is a triple $(\mathbf{C}, \odot, \mathcal{I})$, where $\mathbf{C}$ is a category, $\odot$ is a functor $\mathbf{C} \times \mathbf{C} \to \mathbf{C}$, and $\mathcal{I}$ is an object in $\mathbf{C}$ satisfying the following properties. We write $\odot(X, Y)$ as $X \odot Y$.

- $(X \odot Y) \odot Z = X \odot (Y \odot Z)$
- $X \odot \mathcal{I} = \mathcal{I} \odot X = X$

Note the similarity with the definition of a monoid ([Koc04, p. 140]).

Some examples we would like to give of monoidal categories are $(\mathbf{Vect}_{\Bbbk}, \otimes, \Bbbk)$, $(\mathbf{Man}, \coprod, \emptyset)$, and $(\mathbf{Top}, \coprod, \emptyset)$. One might believe she or he can check easily that all of these meet the above criteria. Strictly speaking, however, this is not possible. For example $(U \otimes V) \otimes W$ is not equal to $U \otimes (V \otimes W)$, but the two are canonically isomorphic. We really should require that the equalities of the above definition be suitably natural isomorphisms, but to do so properly would muddy the notation quite a bit and would not bring about any essential change in the applications. We will consider each of the above examples as monoidal categories. The dissatisfied reader is deferred to [Koc04, p. 154].

The following definition should not be terribly surprising.

Definition 4.10. An object X in the monoidal category $(\mathbf{C}, \odot, \mathcal{I})$ is said to be *invertible* if there exists an object $X^{-1} \in \mathbf{C}$ with $X \odot X^{-1} = X^{-1} \odot X = \mathcal{I}$.

Example 4.11. The invertible elements in the above monoidal categories are not very interesting. If X is invertible in $(\mathbf{Man}, \coprod, \emptyset)$ or $(\mathbf{Top}, \coprod, \emptyset)$, then there exists Y such that $X \coprod Y = \emptyset$, so we must have $X = \emptyset$. For similar reasons, the only invertible elements in $(\mathbf{Vect}_{\Bbbk}, \otimes, \Bbbk)$ are the vector spaces isomorphic to $\Bbbk$.

With this new structure on categories, it is reasonable to define functors that respect it.

Definition 4.12. Let $(\mathbf{C}, \odot, \mathcal{I})$ and $(\mathbf{C}', \odot', \mathcal{I}')$ be monoidal categories. A *monoidal functor* $F : \mathbf{C} \to \mathbf{C}'$ is a functor satisfying the following properties.

- $F(X \odot Y) = F(X) \odot' F(Y)$
- $F(\mathcal{I}) = \mathcal{I}'$
- $F(a \odot b) = F(a) \odot' F(b)$

Where $a \in \mathbf{C}(X, Y)$ and $b \in \mathbf{C}'(X', Y')$.

Example 4.13. The only monoidal functor that we have access to right now is the forgetful functor $(\mathbf{Man}, \coprod, \emptyset) \to (\mathbf{Top}, \coprod, \emptyset)$. Checking the axioms is trivial. Later we will develop more interesting examples of monoidal functors.

We can also extend the idea of natural transformation to fit this monoidal structure.

Definition 4.14. Let $(\mathbf{C}, \odot, \mathcal{I})$ and $(\mathbf{C}', \odot', \mathcal{I}')$ be monoidal categories and $\mathbf{F}$ and $\mathbf{G}$ be monoidal functors $\mathbf{C} \to \mathbf{C}'$. A natural transformation $u : \mathbf{F} \to \mathbf{G}$ is called a *monoidal natural transformation* if $u(X \odot Y) = u(X) \odot' u(Y)$ for all $X, Y \in \mathbf{C_0}$ and $u(\mathcal{I}) = id_{\mathcal{I}'}$.

Definition 4.15. Let $(\mathbf{C}, \odot, \mathcal{I})$ and $(\mathbf{C}', \odot', \mathcal{I}')$ be monoidal categories. Let $\mathbf{MonCat}(\mathbf{C}, \mathbf{C}')$ denote the category whose objects are monoidal functors and whose arrows are monoidal natural transformations. $\mathbf{MonCat}(\mathbf{C}, \mathbf{C}')$ is called a *monoidal functor category* and has the natural structure of a monoidal category. The "product" functor

$$\circledcirc : \mathbf{MonCat}(\mathbf{C}, \mathbf{C}') \times \mathbf{MonCat}(\mathbf{C}, \mathbf{C}') \to \mathbf{MonCat}(\mathbf{C}, \mathbf{C}') \tag{4.1}$$

evaluated on monoidal functors $\mathbf{F}$ and $\mathbf{G}$ is the monoidal functor that sends an element X to $\mathbf{F}(X) \odot' \mathbf{G}(X)$, and acts similarly on arrows. The identity object is the functor that sends all objects the the identity object of $\mathbf{C}'$ and all arrows to the identity arrow.

We have one last structure to add to our categories, which is a notion of symmetry with the $\odot$ functor.

Definition 4.16. A *symmetric monoidal category* is a quadruple $(\mathbf{C}, \odot, \mathcal{I}, \tau)$ where $(\mathbf{C}, \odot, \mathcal{I})$ is a monoidal category and τ is collection of arrows $\{\tau_{X,Y} \in \mathbf{C}(X, Y) \mid X, Y \in \mathbf{C_0}\}$ satisfying the following properties.

- For all arrows $a : X \to X'$ and $b : Y \to Y'$, the following diagram commutes.

$$\begin{tikzcd} X \odot Y \arrow[r, "\tau_{X,Y}"] \arrow[d, "a \odot b"'] & Y \odot X \arrow[d, "b \odot a"] \\ X' \odot Y' \arrow[r, "\tau_{X',Y'}"] & Y' \odot X' \end{tikzcd}$$

- For all $X, Y, Z \in \mathbf{C_0}$ the following diagrams commute.

$$\begin{tikzcd}
X \odot Y \odot Z \arrow[rr, "\tau_{X,Y\odot Z}"] \arrow[dr, "\tau_{X,Y}\odot id_Z"'] & & Y \odot Z \odot X \\
& Y \odot X \odot Z \arrow[ur, "id_Y\odot \tau_{X,Z}"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
X \odot Y \odot Z \arrow[rr, "\tau_{X\odot Y,Z}"] \arrow[dr, "id_X\odot \tau_{Y,Z}"'] & & Z \odot X \odot Y \\
& X \odot Z \odot Y \arrow[ur, "\tau_{X,Z}\odot id_Y"'] &
\end{tikzcd}$$

- $\tau_{X,Y} \circ \tau_{Y,X} = id_{X \odot Y}$

$\tau_{X,Y}$ is called the *twist map* of X and Y.

In the analogy between monoids and monoidal categories, a symmetric monoidal category corresponds to a commutative monoid.

To make sense of the second axiom, note its similarity to the relations in the symmetric group.

Here are some examples of symmetric monoidal categories.

Example 4.17. $(\mathbf{Vect}_{\Bbbk}, \otimes, \Bbbk, \tau)$, $(\mathbf{Man}, \coprod, \emptyset, \tau)$, and $(\mathbf{Top}, \coprod, \emptyset, \tau)$ are all symmetric monoidal categories. τ in these examples is the canonical map $V \otimes V' \to V' \otimes V$ or $M \coprod M' \to M' \coprod M$.

Note here that $M \coprod M'$ is *not* the same thing as $M' \coprod M$. The isomorphism between them is so obvious that we often forget that it is not the identity.

The next definition follows the pattern of the previous definitions.

Definition 4.18. Let $(\mathbf{C}, \odot, \mathcal{I}, \tau)$ and $(\mathbf{C}', \odot', \mathcal{I}', \tau')$ be symmetric monoidal categories. A *symmetric monoidal functor* $\mathbf{F} : \mathbf{C} \to \mathbf{C}'$ is a monoidal functor that satisfies $\mathbf{F}(\tau_{X,Y}) = \tau'_{\mathbf{F}(X),\mathbf{F}(Y)}$.

The following definition uses nearly all the ideas presented in this section and will be the most important for the later sections.

Definition 4.19. Let $(\mathbf{C}, \odot, \mathcal{I}, \tau)$ and $(\mathbf{C}', \odot', \mathcal{I}', \tau')$ be symmetric monoidal categories. Let $\mathbf{SymMonCat}(\mathbf{C}, \mathbf{C}')$ denote the category whose objects are symmetric monoidal functors $\mathbf{C} \to \mathbf{C}'$ and whose arrows are monoidal natural transformations. $\mathbf{SymMonCat}(\mathbf{C}, \mathbf{C}')$ is called a *symmetric monoidal functor category* and has the natural structure of a symmetric monoidal category. Given two symmetric monoidal functors $\mathbf{F}$ and $\mathbf{G}$, the twist map $\mathcal{T}_{\mathbf{F},\mathbf{G}}$ is the natural transformation $u : \mathbf{F} \circledcirc \mathbf{G} \to \mathbf{G} \circledcirc \mathbf{F}$ that assigns to each object X the arrow $\tau'_{\mathbf{F}(X),\mathbf{G}(X)}$.

5. Cobordisms

The goal of this section is to introduce a new category, n-cobordisms, that will begin to justify the painful amount of abstraction-present in the previous section. For this, we will be using ideas from both the previous sections. Once this is done, we will construct the cobordism groups and relate them to SKK invariants.

5.1. **The Cobordism Category.** Before the proper definition is given, it is important the the reader can at least have a picture of what a cobordism is. You can think of an oriented n-cobordism between two $(n-1)$-manifolds Σ_0 and Σ_1 is an oriented m-manifold whose boundary is $\overline{\Sigma}_0 \coprod \Sigma_1$. We will write this cobordism as $M : \Sigma_0 \rightsquigarrow \Sigma_1$. This notation indicates one need for a better definition. Although it is not a map, we want the manifold M to be, in some sense, "going from Σ_0 to Σ_1." This will allow us to make M the arrow in our category.

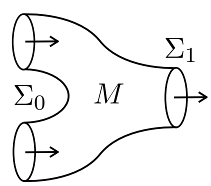


FIGURE 13. A cobordism

Another reason that we need a better definition is because there are plenty of manifolds with boundary $\overline{\Sigma}_0 \coprod \Sigma_1$, and we would like to form an equivalence relation on this set so that two cobordisms that have no differing properties that we care about are members of the same class.

Now we can state the definition of cobordism.

Definition 5.1. Let Σ_0 and Σ_1 be closed oriented $(n-1)$-manifolds. An oriented *n-cobordism* M is a manifold along with orientation-preserving diffeomorphisms

$$\phi_0 : \Sigma_0 \to \overline{\partial_0 M} \quad \text{and} \quad \phi_1 : \Sigma_1 \to \partial_1 M, \tag{5.1}$$

where

$$\partial M = \partial_0 M \cup \partial_1 M. \tag{5.2}$$

There do exist unoriented cobordisms, which resemble our description of cobordism at the beginning of this chapter. We have no need for this type of cobordism.

One benefit of this somewhat convoluted definition is that we can have a cobordism between a closed manifold and itself. This would not be possible if we defined a cobordism between Σ_0 and Σ_1 to simply be an n-manifold with in-boundary Σ_0 and out-boundary Σ_1. If we did this, the in-boundaries and out-boundaries may be diffeomorphic, but never the same manifold.

We are obviously limited in the dimension of cobordisms that can be drawn on paper. We will show what we can. Here are a few examples of oriented cobordisms. In our examples, $\partial_0 M$ consists of those boundary components with arrows facing toward the manifold, and $\partial_1 M$ consists of the components with arrows facing away from the manifold.

Example 5.2. The simplest oriented cobordism is a closed line segment with two points on the end, as in Figure 14.

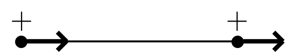


FIGURE 14. Cobordism of two points

A zero manifold is oriented simply by assigning a plus or minus to each of its points.

Figure 15 shows some examples of oriented 2-cobordisms. The first shows that M, Σ_0, and Σ_1 need not be connected. The second, which is often called a "pair of pants" (for obvious reasons), shows that the boundary manifolds need not have the same number of components. The third, quite importantly, is a cobordism between two empty 1-manifolds.

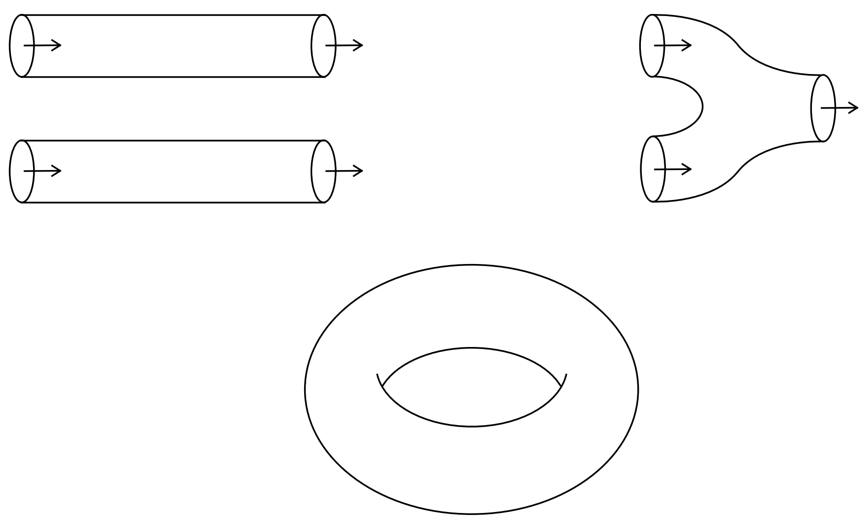


FIGURE 15. Examples of 2-cobordisms

Now we explain, as promised, when we can consider two cobordisms to be equivalent.

Definition 5.3. Two corbordisms $M : \Sigma_0 \rightsquigarrow \Sigma_1$ and $M' : \Sigma_0 \rightsquigarrow \Sigma_1$ are said to be *equivalent* if there exists an orientation-preserving diffeomorphism ψ making Diagram 5.3 commute.

$$\begin{tikzcd} & M & \\ \Sigma_0 \arrow[ur, "\phi_0"] \arrow[dr, "\phi_0'"'] & & \Sigma_1 \arrow[ul, "\phi_1"'] \arrow[dl, "\phi_1'"] \\ & M' \arrow[uu, "\psi"] & \end{tikzcd}$$

Cobordism equivalence is easily seen to be an equivalence relation. Thus we can ignore superficial differences in cobordisms and refer only to cobordism classes.

Note that two n-manifolds M and M' can be diffemorphic while the cobordisms $M : \Sigma_0 \rightsquigarrow \Sigma_1$ and $M' : \Sigma_0 \rightsquigarrow \Sigma_1$ are not equivalent. One simple counterexample, presented below, will be necessary for our categorical description of cobordism.

Definition 5.4. Let Σ and Σ' be closed $(n-1)$-manifolds. The cobordism

$$(\Sigma \coprod \Sigma') \times I : \Sigma \coprod \Sigma' \rightsquigarrow \Sigma' \coprod \Sigma \tag{5.3}$$

where ϕ_0 is the identity and ϕ_1 is the canonical twist map of the category **Man** is called the *twist cobordism* and is denoted $\tau(\Sigma, \Sigma')$.

Figure 16 gives a picture of $\tau(S^1, S^1_*)$.

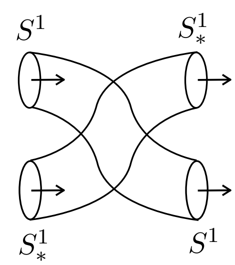


Figure 16. Twist

We note that $\tau(\Sigma, \Sigma')$ is not equivalent to the cylinder cobordism

$$(\Sigma \coprod \Sigma') \times I : \Sigma \coprod \Sigma' \rightsquigarrow \Sigma \coprod \Sigma', \tag{5.4}$$

even though their n-manifolds are the same.

Writing a cobordism as $M : \Sigma_0 \rightsquigarrow \Sigma_1$ can be somewhat cumbersome notationally, so we often abbreviate and have M denote the whole cobordism $M : \Sigma_0 \rightsquigarrow \Sigma_1$. We have just shown that this is a bad idea, since cobordisms contain important data beyond just the n-manifold M. However, it is the most concise notation, and we will only use it when it causes no problems.

We want to be able to describe oriented n-cobordism classes as arrows in a category whose objects are $(n-1)$-manifolds. What we are still missing is some way of "composing" cobordisms and an identity cobordism. Our composition process will be gluing of n-manifolds, and our identity cobordism will be the cylinder cobordism, as we now describe.

Let $M : \Sigma_0 \rightsquigarrow \Sigma_1$ and $M' : \Sigma_1 \rightsquigarrow \Sigma_2$ be cobordisms. Let MM' be the manifold constructed as follows: Reverse the orientation of M', so that the diffeomorphism

$$\phi_0' \circ \phi_1^{-1} : \partial_1 M \to \overline{\partial_0 M'} \tag{5.5}$$

is an orientation-preserving diffeomorphism of unions of boundary components. Now we set

$$MM' = M \bigcup_{\phi_0' \circ \phi_1^{-1}} \overline{\overline{M'}} = M \bigcup_{\phi_0' \circ \phi_1^{-1}} M'. \tag{5.6}$$

Then $MM' : \Sigma_0 \rightsquigarrow \Sigma_2$ is a cobordism, where the boundary diffeomorphisms are those inherited by the original cobordisms.

It is easy to check that this gluing process is invariant under cobordism equivalence. That is, if $M \sim N$ and $M' \sim N'$ (here $\sim$ denotes cobordism equivalence), then $MM' \sim NN'$. This is done using the collar neighborhood construction of Section 3.1.2. Also, this process is associative: $(MM')M'' \sim M(M'M'')$. Note that cobordism gluing would not be associative if we were dealing with single cobordisms rather than cobordism classes.

Our last step before we can describe cobordisms categorically is to give an identity element, which we have stated is the cylinder. Showing this is easy. If $M : \Sigma_0 \rightsquigarrow \Sigma_1$ is a cobordism, then Σ_1 has a collar neighborhood in M, which is diffeomorphic to the cylinder $\Sigma_1 \times I$. Of course gluing another copy of $\Sigma_1 \times I$ to the out-boundary of M gives an equivalent cobordism. The same argument applies to the in-boundary, and we have $(\Sigma_0 \times I)M \sim M \sim M(\Sigma_1 \times I)$.

Definition 5.5. Let **nCob** denote the category whose objects are closed $(n-1)$-manifolds and whose arrows are n-cobordisms.

We now give this category a symmetric monoidal structure. The monoidal operation will be $\coprod$, with identity element $\emptyset$. It is clear that $(M \coprod M') \coprod M'' \sim M \coprod (M' \coprod M'')$ and $M \coprod \emptyset \sim \emptyset \coprod M \sim M$. We also see that the twist cobordism satisfies the axioms of Definition 4.16. We give a pictorial representation of the relation $\tau(S^1, S^1_*)\tau(S^1_*, S^1)$ in Figure 17.

We have now reached our desired categorical description of cobordisms.

Proposition 5.6. $(\mathbf{nCob}, \coprod, \emptyset, \tau)$ *is a symmetric monoidal category.*

5.2. **Cobordism Groups.** We can use the idea of cobordism to form yet another equivalence relation on closed n-manifolds. We will again be able to give these equivalence classes an abelian group structure.

Definition 5.7. Let Σ_0 and Σ_1 be closed oriented $(n-1)$-manifolds. Σ_0 and Σ_1 are called *cobordant* if there exists a cobordism $M : \Sigma_0 \rightsquigarrow \Sigma_1$. In this case we denote $\Sigma_0 \sim_\Omega \Sigma_1$, or $[\Sigma_0]_\Omega = [\Sigma_1]_\Omega$.

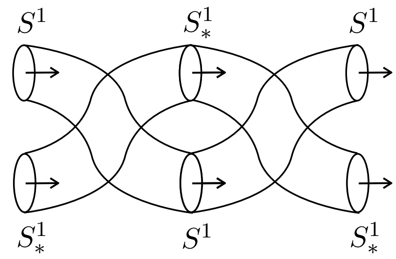


FIGURE 17. Double twist

This is clearly an equivalence relation. Two diffeomorphic manifolds are cobordant via a cylinder cobordism, so this relation is coarser than the diffeomorphism relation. This relation has the property

$$\Sigma_0 \sim_\Omega \Sigma_1,\ \Sigma_0' \sim_\Omega \Sigma_1' \implies \left(\Sigma_0 \coprod \Sigma_0'\right) \sim_\Omega \left(\Sigma_1 \coprod \Sigma_1'\right), \tag{5.7}$$

which allows us to define an associative law of composition on the set of n-dimensional cobordism classes, which we denote by Ω_n. But each element $[\Sigma]_\Omega$ in this monoid has an inverse, namely $[\overline{\Sigma}]_\Omega$ (See Figure 19). This leads to the following definition.

Definition 5.8. Ω_n is an abelian group under the operation of disjoint union, which we will call the *n-cobordism group*.

There is an equivalent way of formulating this group. In particular, we can construct Ω_n by taking a quotient of the monoid $\mathbb{M}_n^{\S}$ of closed oriented n-manifolds by the submonoid generated by all elements $[\Sigma]_\Omega$, where Σ bounds.

We now relate the cobordism groups to the SKK groups.

Theorem 5.9. [KKNO73, p. 44] *The homomorphism $SKK_n \to \Omega_n$ that assigns to each manifold its cobordism class is a surjective SKK invariant.*

Theorem 5.10. [KKNO73, p. 44] *The following sequence is exact*

$$0 \to I_n \to SKK_n \to \Omega_n \to 0, \tag{5.8}$$

where

$$I_n = \begin{cases} I_n = \mathbb{Z} & n \equiv 0 \bmod 2 \\ I_n = \mathbb{Z}_2 & n \equiv 1 \bmod 4 \\ I_n = 0 & n \equiv 3 \bmod 4 \end{cases} \tag{5.9}$$

In addition, χ splits the sequence in dimensions divisible by 4, and $\chi_{1/2}$ splits the sequence in dimensions $n \equiv 1, 2 \bmod 4$.

This reduces the problem of completely classifying the SKK groups and the SKK invariants to the problem of classifying the cobordism groups. We list the most interesting descriptions of SKK invariants in each dimension. The only nontrivial $\mathbb{Z}$-valued $2n$-dimensional SKK invariants are linear combinations of χ and elements of $\operatorname{Hom}(\Omega_{2n}, \mathbb{Z})$. In dimensions $4n+1$, the $\mathbb{Z}_2$-valued invariants include combinations of $\chi_{1/2}$ and elements of $\operatorname{Hom}(\Omega_{4n+1}, \mathbb{Z}_2)$. It follows from this that σ is a bordism invariant. Lastly, $\operatorname{Hom}(SKK_{4n+3}, G) = \operatorname{Hom}(\Omega_{4n+3}, G)$ for all G.

This gives us plenty of information about elements of the SKK group, as the following lemma illustrates.

Lemma 5.11. *Let Σ be a closed n-manifold. Then $[\Sigma \times S^1]_{SKK}$ has finite order in SKK_n.*

Proof. By Theorem 5.10, SKK_n is isomorphic to $I_n \oplus \Omega_n$. The projection map $SKK_n \to I_n$ is given by $\chi_{1/2}$ or χ, and the projection map $SKK_n \to \Omega_n$ is given by the homomorphism $[X]_{SKK} \to [X]_\Omega$. I_n is infinite if and only if the splitting homomorphism is given by a multiple of χ. But it is clear that $\chi(\Sigma \times S^1) = 0$ by the multiplicativity of the Euler characteristic. Also, $[\Sigma \times S^1]_\Omega = 0$ because

$$\Sigma \times S^1 = \partial(\Sigma \times D^2). \tag{5.10}$$

Thus $[\Sigma \times S^1]_{SKK}$ has finite order. $\square$

Much is known about the cobordism groups. One particular result, making use of the theory of characteristic classes, will be especially useful to us.

Theorem 5.12. [MS74, p. 216] *Ω_n is finite if $n \not\equiv 0$ mod 4. Ω_{4n} is free of rank $p(n)$, the number of partitions of n. The Pontryagin numbers are a complete invariant on Ω_{4n}.*

6. Topological Quantum Field Theories

In this section we define topological quantum field theories (TQFTs) and give examples of computations that can be done with them. The name might suggest that we are going to be doing some physics in this section, but that is not the case. Although TQFTs were created by physicists and certainly have physical relevance, they are also studied by mathematicians for purposes independent of physical appliations. We will be taking the route of the mathematicians.

Michael Atiyah gave an axiomatic definition of TQFTs in his 1988 paper "Topological quantum field theories" [Ati88]. His definition is equivalent to the following:

Definition 6.1. An *n-dimensional oriented TQFT* is a rule $\mathcal{T}$ that assigns to each oriented $(n-1)$-manifold Σ a vector space $\mathcal{T}(\Sigma)$ and to each oriented cobordism $M : \Sigma_0 \to \Sigma_1$ a linear map $\mathcal{T}(M) : \mathcal{T}(\Sigma_0) \to \mathcal{T}(\Sigma_1)$, satisfying the following properties.

(1) Two equivalent cobordisms have the same image.

$$M \sim M' \implies \mathcal{T}(M) = \mathcal{T}(M') \tag{6.1}$$

(2) A glued cobordism goes to a composition of linear maps.
$$\mathcal{T}(MM') = \mathcal{T}(M) \circ \mathcal{T}(M') \tag{6.2}$$
(3) A cylinder cobordism gets sent to the identity map.
$$\mathcal{T}(\Sigma \times I) = Id_{\mathcal{T}(\Sigma)} \tag{6.3}$$
(4) Disjoint unions of $(n-1)$-manifolds and cobordisms get sent to tensor products.
$$\mathcal{T}(\Sigma \coprod \Sigma') = \mathcal{T}(\Sigma) \otimes \mathcal{T}(\Sigma') \tag{6.4}$$
$$\mathcal{T}(M \coprod M') = \mathcal{T}(M) \otimes \mathcal{T}(M') \tag{6.5}$$
(5) The empty manifold $\emptyset$ gets sent to the ground field $\Bbbk$.
$$\mathcal{T}(\emptyset) = \Bbbk \tag{6.6}$$

A TQFT is represented pictorially in Figure 18

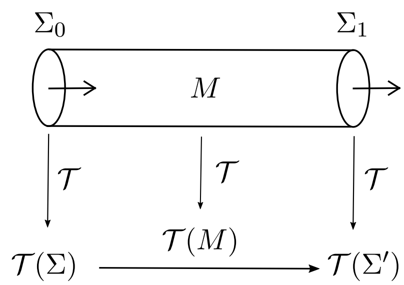


Figure 18. A TQFT

You might notice that this list of axioms seems to be expressing functoriality for $\mathcal{T}$. This leads to the following more concise definition.

Definition 6.2. An *n-dimensional oriented TQFT* is a symmetric monoidal functor $\mathbf{nCob} \to \mathbf{Vect}_{\Bbbk}$.

You can easily check that these definitions are almost equivalent. The second is a bit stronger, because the first does not require that $\mathcal{T}$ satisfy the twist axioms of Definition 4.16. This would not cause much of a problem, but in any case we will use the first definition.

Now we make use of some algebraic constructions that we can use to give some surprising restrictions on TQFTs.

Definition 6.3. Let V and W be $\Bbbk$ vector spaces. A linear map $\gamma : V \otimes W \to \Bbbk$ is called a *pairing*. γ is called *nondegenerate* if there exists a linear map $\beta : \Bbbk \to W \otimes V$, called a *copairing* such that the composites
$$V(= V \otimes \Bbbk) \xrightarrow{id_V \otimes \beta} V \otimes W \otimes V \xrightarrow{\gamma \otimes id_V} \Bbbk \otimes V(= V) \tag{6.7}$$
and
$$W(= \Bbbk \otimes W) \xrightarrow{\beta \otimes id_W} W \otimes V \otimes W \xrightarrow{id_W \otimes \gamma} W \otimes \Bbbk(= W) \tag{6.8}$$
are equal to id_V and id_W, respectively.

See [Koc04, p. 83-85] for proofs of the following.

Lemma 6.4. *If $\gamma : V \otimes W \to \Bbbk$ is a nondegenerate pairing, then both V and W are finite-dimensional.*

Lemma 6.5. *Given a nondegnerate pairing $\gamma : V \otimes W \to \Bbbk$, there are canonical isomorphisms $\lambda : V \to W^*$ and $\lambda' : W \to V^*$ defined by*

$$\lambda(v) := w \to \gamma(v \otimes w) \quad \textit{and} \quad \lambda'(w) := v \to \gamma(v \otimes w). \tag{6.9}$$

Now we show how this relates to TQFTs. Consider the cylinder $\Sigma \times I$. Instead of considering $\Sigma \times I$ a cobordism $\Sigma \rightsquigarrow \Sigma$, we can consider it a cobordism $\overline{\Sigma} \coprod \Sigma \rightsquigarrow \emptyset$. Thus cobordism is often called a "U-tube." The name is justified by Figure 19.

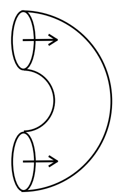


Figure 19. A "U-tube"

We will have $(\Sigma \times I)''$ denote the reversed cobordism $\emptyset \rightsquigarrow \Sigma \coprod \overline{\Sigma}$, pictured in Figure 20.

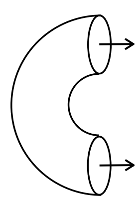


Figure 20. Reversed U-tube

Under a TQFT $\mathcal{T}$, $(\Sigma \times I)'$ gets sent to a pairing

$$\mathcal{T}((\Sigma \times I)') : \mathcal{T}(\overline{\Sigma}) \otimes \mathcal{T}(\Sigma) \to \Bbbk \tag{6.10}$$

and $(\Sigma \times I)''$ gets sent to a map

$$\mathcal{T}((\Sigma \times I)'') : \Bbbk \to \mathcal{T}(\Sigma) \otimes \mathcal{T}(\overline{\Sigma}). \tag{6.11}$$

We will show that $\mathcal{T}((\Sigma \times I)'')$ is a copairing for $\mathcal{T}((\Sigma \times I)')$. This will show that the pairing is nondegenerate, from which two interesting facts will follow. The first is that $\mathcal{T}(\Sigma)$ is necessarily of finite dimension, and the second is that there is a canonical isomorphism between $\mathcal{T}(\overline{\Sigma})$ and $\mathcal{T}(\Sigma)^*$.

Consider the composition in Figure 6.

This cobordism, sometimes called the *snake decomposition*, is a composition of the cobordisms $\Sigma \times I \coprod (\Sigma \times I)''$ and $(\Sigma \times I)' \coprod \Sigma \times I$ and is clearly

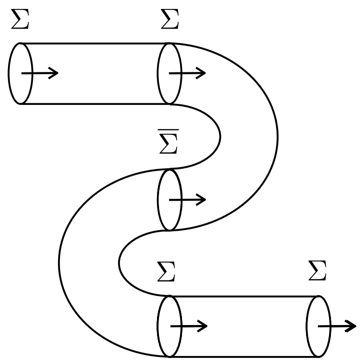


FIGURE 21. Snake decomposition of the cylinder

equivalent to to identity cobordism. Thus it gets sent under $\mathcal{T}$ to $id_{\mathcal{T}(\Sigma)}$. However, it also gets sent to the map

$$\mathcal{T}(\Sigma) \otimes \Bbbk \xrightarrow{id_{\mathcal{T}(\Sigma)} \otimes (\Sigma \times I)''} \mathcal{T}(\Sigma) \otimes \mathcal{T}(\overline{\Sigma}) \otimes \mathcal{T}(\Sigma) \xrightarrow{\mathcal{T}((\Sigma \times I)') \otimes id_{\mathcal{T}(\Sigma)}} \Bbbk \otimes \mathcal{T}(\Sigma) \tag{6.12}$$

by the axioms for TQFTs. Thus Equation 6.7 is satisfied.

Now reverse the orientation, as in Figure 22.

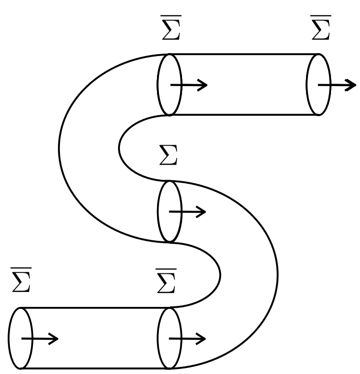


FIGURE 22. Reverse snake decomposition

Here we see that

$$\Bbbk \otimes \mathcal{T}(\overline{\Sigma}) \xrightarrow{\mathcal{T}\left(\overline{(\Sigma \times I)''}\right) \otimes id_{\mathcal{T}(\overline{\Sigma})}} \mathcal{T}(\overline{\Sigma}) \otimes \mathcal{T}(\Sigma) \otimes \mathcal{T}(\overline{\Sigma}) \xrightarrow{id_{\mathcal{T}(\overline{\Sigma})} \otimes \mathcal{T}\left(\overline{(\Sigma \times I)'}\right)} \mathcal{T}(\overline{\Sigma}) \otimes \Bbbk \tag{6.13}$$

is equal to $id_{\mathcal{T}(\overline{\Sigma})}$, and Equation 6.8 is satisfied. Thus $\mathcal{T}((\Sigma \times I)'')$ is a copairing for $\mathcal{T}((\Sigma \times I)')$.

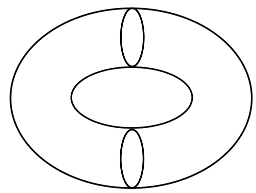


FIGURE 23. Decomposition of $\Sigma \times S^1$

Now consider the cobordism $\Sigma \times S^1$, and decompose it as in Figure 23.

This is a cobordism between two empty manifolds; thus under any TQFT it is sent to a linear map $\Bbbk \to \Bbbk$, which is essentially a scalar. What do we know about this scalar? Well, our decomposition above shows that $\Sigma \times S^1 \sim (\Sigma \times I)''(\Sigma \times I)'$. We take avantage of our knowledge of these smaller pieces to do some investigating and see which scalar is associated with the composite $\mathcal{T}(\Sigma \times I)'' \circ \mathcal{T}(\Sigma \times I)'$.

The only important fact here is that we have the composite of a copairing, a twist, and its pairing, not that we are dealing with manifolds or TQFTs. For this reason we use the simpler notation of maps $\gamma : V \otimes W \to \Bbbk$ and $\beta : \Bbbk \to V \otimes W$.

Now remember that if $\{\mathbf{v}_i\}_{1\leq i\leq n}$ is a basis for V and $\{\mathbf{w}_i\}_{1\leq i\leq n}$ is a basis for W, then $\{\mathbf{v}_i \otimes \mathbf{w}_j\}_{1\leq i,j\leq n}$ is a basis for $V \otimes W$. Now γ is completely determined by assigning a scalar γ_{ij} to each of these n^2 elements, and β is completely determined by

$$\beta(1) = \sum_{i,j}^{n} \beta_{ij} \cdot \mathbf{v}_i \otimes \mathbf{w}_j \in V \otimes W. \tag{6.14}$$

Thus we can associate to γ and β the $n \times n$ matrices $\{\gamma_{ij}\}$ and $\{\beta_{ij}\}$.

We wish to use the nondegeneracy condition to see what the relationship between these two matrices is. Let τ denote the obvious twist $W \otimes V \to V \otimes W$. The condition requires

$$(id_V \otimes (\gamma \circ \tau)) \circ (\beta \otimes id_V) = id_V. \tag{6.15}$$

This holds if and only if it holds on the basis elements of V. Thus we must have for each $\mathbf{v}_k$

$$\begin{aligned}
\mathbf{v}_k &= (id_V \otimes (\gamma \circ \tau))(\beta(1) \otimes \mathbf{v}_k) \\
&= (id_V \otimes (\gamma \circ \tau))((\sum_{i,j}^{n} \beta_{ij} \cdot \mathbf{v}_i \otimes \mathbf{w}_j) \otimes \mathbf{v}_k) \\
&= (id_V \otimes (\gamma \circ \tau))(\sum_{i,j}^{n} \beta_{ij} \cdot \mathbf{v}_i \otimes \mathbf{w}_j \otimes \mathbf{v}_k) \\
&= \sum_{i}^{n} \mathbf{v}_i (\sum_{j}^{n} \gamma_{ij} \beta_{kj}).
\end{aligned} \tag{6.16}$$

This means that

$$\sum_{j}^{n} \gamma_{ij}\beta_{kj} = 0 \quad \text{for } i \neq k \tag{6.17}$$

and

$$\sum_{j}^{n} \gamma_{ij}\beta_{kj} = 1 \quad \text{for } i = k. \tag{6.18}$$

This is equivalent to the condition that

$$\{\beta_{ij}\}^T = \{\gamma_{ij}\}^{-1}. \tag{6.19}$$

We have uncovered the restriction given by the nondegeneracy condition, and we can now find our scalar associated with $\Sigma \times S^1$. We keep the simpler notation.

$$\begin{aligned} \gamma(\beta(1)) &= \gamma(\sum_{i,j}^{n} \beta_{ij}(\mathbf{v}_i \otimes \mathbf{w}_j)) \\ &= \sum_{i,j}^{n} \gamma_{ij}\beta_{ij} \\ &= n \end{aligned} \tag{6.20}$$

The last equality comes from Equations 6.17 and 6.18. Thus the scalar associated with $\Sigma \times S^1$ is $\dim \mathcal{T}(\Sigma)$. That is, the image of $\Sigma \times S^1$ under any TQFT is completely determined by the image of Σ.

7. Invertible TQFTs

Since each n-TQFT is a symmetric monoidal functor $\mathbf{nCob} \to \mathbf{Vect}_{\Bbbk}$ we can consider the symmetric monoidal functor category

$$\mathbf{SymMonCat}(\mathbf{nCob}, \mathbf{Vect}_{\Bbbk}) \tag{7.1}$$

whose objects are oriented n-TQFTs and whose arrows are natural transformations of n-TQFTs. We will denote this category as $\mathbf{nTQFT}_{\Bbbk}$. The product of two TQFTs $\mathcal{T}_1$ and $\mathcal{T}_2$ is the TQFT

$$\mathcal{T}_1 \otimes \mathcal{T}_2 \tag{7.2}$$

that assigns to each $(n-1)$-manifold Σ the vector space

$$\mathcal{T}_1(\Sigma) \otimes \mathcal{T}_2(\Sigma) \tag{7.3}$$

and to each cobordism $M : \Sigma \rightsquigarrow \Sigma'$ the linear map

$$\mathcal{T}_1(M) \otimes \mathcal{T}_2(M). \tag{7.4}$$

The identity is the trivial TQFT, which sends each $(n-1)$-manifold to $\Bbbk$ and each n-cobordism to the identity.

The arrows are natural transformations of TQFTs. As a reminder, these natural transformations assign to each $(n-1)$-manifold Σ and each pair $(\mathcal{T}_1, \mathcal{T}_2)$ of TQFTs a linear map

$$u(\Sigma) : \mathcal{T}_1(\Sigma) \to \mathcal{T}_2(\Sigma) \tag{7.5}$$

that satisfies

$$\mathcal{T}_2(M) \circ u(\Sigma) = u(\Sigma') \circ \mathcal{T}_1(M) \tag{7.6}$$

for each cobordism $M : \Sigma \rightsquigarrow \Sigma'$. The twist map in this category is the natural transformation between TQFTs $\mathcal{T}_1$ and $\mathcal{T}_2$ that assigns to each $(n-1)$-manifold Σ the cobordism $\tau(\mathcal{T}_1(\Sigma), \mathcal{T}_2(\Sigma))$.

We wish to study the invertible objects in $\mathbf{nTQFT}_{\Bbbk}$, that is, the TQFTs $\mathcal{T}$ with an inverse $\mathcal{T}'$ such that $\mathcal{T} \otimes \mathcal{T}'$ is the trivial TQFT. Since tensoring multiplies the dimension of finite-dimensional vector spaces, each vector space that $\mathcal{T}$ assigns to an $(n-1)$-manifold must be 1-dimensional, i.e. isomorphic to $\Bbbk$.

We investigate what linear maps invertible TQFTs can assign to cobordisms. Since all linear maps $\Bbbk \to \Bbbk$ are simply scalar multiplication, each map that $\mathcal{T}$ assigns to an n-manifold is a nonzero scalar. If Λ_1 and Λ_2 are maps $\Bbbk \to \Bbbk$ where Λ_i is multiplication by the scalar λ_i, then the map

$$\Lambda_1 \otimes \Lambda_2 : \Bbbk \otimes \Bbbk \to \Bbbk \otimes \Bbbk \tag{7.7}$$

can be canonically identified with the multiplication by $\lambda_1 \cdot \lambda_2$ map $\Bbbk \to \Bbbk$. Since this tensored linear map must be the identity for the product of an invertible TQFT and its inverse, all linear maps assigned by invertible TQFTs must be invertible, or multiplication by a nonzero scalar. A pictorial representation of an invertible TQFT is given in figure 24.

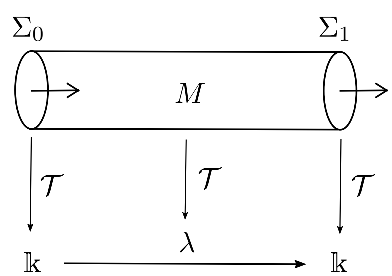


FIGURE 24. An invertible TQFT

It is clear that the set of invertible TQFTs forms a group under the composition operation in $\mathbf{nTQFT}_{\Bbbk}$. We will denote this group as $\mathbf{nTQFT}_{\Bbbk}^{\dagger}$.

What we will do next combines many of the ideas that we have presented so far. Our goal is to determine the relationship between how invertible TQFTs act on two cut and paste equivalent closed manifolds considered as cobordisms $\emptyset \rightsquigarrow \emptyset$.

Let $\mathcal{T}$ be an invertible TQFT and let M and N be cut and paste equivalent closed manifolds, with gluing diffemorphisms f and $g : \sigma \to \overline{\Sigma}$, respectively, as in figure 25.

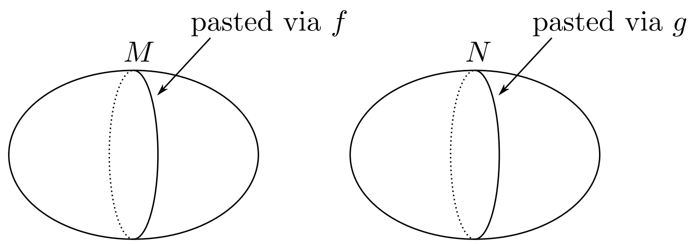


FIGURE 25. Cut and paste equivalent manifolds

Also let δ be the canonical isomorphism $\mathrm{Hom}\,(\Bbbk^\times, \Bbbk^\times) \to \Bbbk^\times$, where $\Bbbk^\times = \Bbbk - \{0\}$, and the homomorphisms are of multiplicative groups. Now using the fact that Σ has a collar neighborhood in both M and N, we can replace M and N with the equivalent cobordisms

$$M'C_fM'' \quad \text{and} \quad M'C_fM'', \tag{7.8}$$

as in Figure 7.

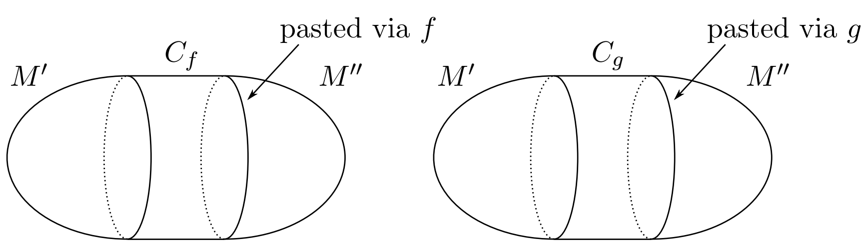


FIGURE 26. Equivalent cobordisms

Here C_f denotes the mapping cylinder[1] of f. Now evaluating $\mathcal{T}$ on both cobordisms and taking a quotient, we see an interesting relation.

$$\frac{\delta(\mathcal{T}(MC_fM'))}{\delta(\mathcal{T}(MC_gM'))} = \frac{\delta(\mathcal{T}(M)) \cdot \delta(\mathcal{T}(C_f)) \cdot \delta(\mathcal{T}(M'))}{\delta(\mathcal{T}(M)) \cdot \delta(\mathcal{T}(C_g)) \cdot \delta(\mathcal{T}(M'))} = \frac{\delta(\mathcal{T}(C_f))}{\delta(\mathcal{T}(C_g))} \tag{7.9}$$

The above equation is valid because all scalars are required to be nonzero by the invertibility of $\mathcal{T}$.

[1]See Section 3.1.2

8. Results

We now relate the ideas of invertible TQFTs and SKK invariants. Let $\mathbb{k}_*^\times$ be a graded group with n^{th} coordinate group $\mathbb{k}_n^\times$, where $\mathbb{k}_n$ is a field. The relation will be expressed by means of a homomorphism of graded groups

$$\Psi_* : *\mathbf{TQFT}^\dagger_{\mathbb{k}_*} \to \mathrm{Hom}\,(SKK_n, \mathbb{k}_n^\times)_* \tag{8.1}$$

where $*\mathbf{TQFT}^\dagger_{\mathbb{k}_*}$ is the graded group whose n^{th} coordinate group is $\mathbf{nTQFT}^\dagger_{\mathbb{k}_n}$.

Let $\mathcal{T}_n \in \mathbf{nTQFT}^\dagger_{\mathbb{k}_n}$. We define a homomorphism

$$\Psi_n(\mathcal{T}_n) : \mathcal{G}(\mathbb{M}_n^\S) \to \mathbb{k}_n^\times \tag{8.2}$$

as follows. If $[X] \in \mathcal{G}(\mathbb{M}_n^\S)$, then $\Psi_n(\mathcal{T}_n)([X]) = \delta(\mathcal{T}_n([X]))$. That this is a group homomorphism follows from the TQFT axioms.

Now let M and N be cut and paste equivalent closed oriented n-manifolds, where f and g are the gluing diffeomorphisms of M and N, respectively. We have shown in Section 7 that

$$\frac{\Psi_n(\mathcal{T}_n)(M)}{\Psi_n(\mathcal{T}_n)(N)} = \frac{\delta(\mathcal{T}_n(M))}{\delta(\mathcal{T}_n(N))} = \frac{\delta(\mathcal{T}_n(C_f))}{\delta(\mathcal{T}_n(C_g))} = \Psi_n(\mathcal{T}_n)(f, g). \tag{8.3}$$

The notation is a bit different since we are dealing with a multiplicative group, but this is precisely the first criterion of Definition 3.12. Thus $\Psi_n(\mathcal{T}_n)$ induces a homomorphism $SKK_n \to \mathbb{k}_n^\times$, which we will continue to denote $\Psi_n(\mathcal{T}_n)$. Thus $\Psi_n(\mathcal{T}_n)$ is an SKK invariant on n-manifolds, and Ψ_n is a homomorphism $\mathbf{nTQFT}^\dagger_{\mathbb{k}_n} \to \mathrm{Hom}\,(SKK_n, \mathbb{k}_n^\times)$.

We define Ψ_* to be the homomorphism of graded groups whose n^{th} coordinate homomorphism is Ψ_n.

Ψ_* gives a natural relationship between invertible TQFTs and SKK invariants. Note that this relationship only makes sense if each G_n of Definition 3.12 is the multiplication group of a field. This causes a problem for the SKK invariants we have given, since the target groups here include $\mathbb{Z}$, which is not isomorphic to the multiplication group of any field. However, the problem is solved by the exponential map $x \to e^x$, as we now show.

If there is a monomorphism $f : G_n \to \mathbb{k}_n^\times$, then there is a natural monomorphism

$$\tilde{f} : \mathrm{Hom}\,(SKK_n, G_n), \to \mathrm{Hom}\,(SKK_n, \mathbb{k}_n^\times). \tag{8.4}$$

Thus if G_n is isomorphic to a subgroup of some $\mathbb{k}_n^\times$, then we can uniquely identify each element of $\mathrm{Hom}\,(SKK_n, G_n)$ with an element of $\mathrm{Hom}\,(SKK_n, \mathbb{k}_n^\times)$, which can then be potentially identified with an invertible $*$-TQFT.

$x \to e^x$ maps $\mathbb{Z}$ monomorphically to a subgroup of $\mathbb{R}^\times$. Thus, for example, χ and σ can be thought of as elements of $\mathrm{Hom}\,(SKK_n, \mathbb{R}^\times)_*$.

We will be interested in finding the kernel and image of Ψ_*. The kernel is easy to describe.

Theorem 8.1. *The kernel of Ψ_* is the graded subgroup $\mathscr{T}_*$ of $*\mathbf{TQFT}^{\dagger}_{\Bbbk_*}$, where $\mathscr{T}_n$ consists of all invertible n-TQFTs $\mathcal{T}_n$ such that $\mathcal{T}_n(M) = id_{\Bbbk_n^\times}$ for all closed n-manifolds M.*

Proof. Let $\mathcal{T}_n$ be an oriented invertible n-TQFT, and suppose $\Psi_n(\mathcal{T}_n)$ is the trivial SKK_n-invariant, i.e. the invariant that sends all closed n-manifolds to $1 \in \Bbbk_n^\times$. Then obviously $\mathcal{T}_n \in \mathscr{T}_n$. It is also clear that $\mathcal{T}_n \in \mathscr{T}_n \implies \mathcal{T}_n \in \ker \Psi_n$. □

The proposition above expresses a degree of "forgetfulness" of Ψ_*. TQFTs assign values to all cobordisms, with or without boundary, and SKK invariants are only defined on closed manifolds. Thus by applying Ψ_* to $\mathcal{T}_*$ one loses some information about $\mathcal{T}_*$, as the following theorem shows.

Theorem 8.2. *$\mathscr{T}_*$ is trivial if and only if $\Bbbk_*^\times$ is trivial.*

Proof. First suppose $\Bbbk_*^\times$ is trivial. Then there is obviously only one possible TQFT in each dimension, which is the trivial TQFT. In this case $\mathscr{T}_*$ is clearly trivial.

Now suppose $\Bbbk_*^\times$ is nontrivial. We must define $\mathcal{T}_* \in *\mathbf{TQFT}^{\dagger}_{\Bbbk_*}$ so that $\mathcal{T}_*$ evaluates closed manifolds trivially and manifolds with boundary nontrivially. It suffices to define $\mathcal{T}_*$ in dimension n. We proceed as follows.

Let $\mathcal{T}(\Sigma) = \Bbbk$ for all $(n-1)$-manifolds Σ. Then assign to each closed connected $(n-1)$-manifold Σ a scalar λ_Σ. We require $\lambda_\Sigma \notin \{0,1\}$ if $\Sigma \neq \emptyset$ whenever this is possible, and $\Sigma_\emptyset = 1$. Now for any cobordism

$$M : \coprod_{i=0}^{n} \Sigma_i \rightsquigarrow \coprod_{j=0}^{m} \Sigma_j \tag{8.5}$$

we define

$$\mathcal{T}_n(M) = \delta^{-1}\Big(\prod_{i=0}^{n} \lambda_{\Sigma_i} \cdot \prod_{j=0}^{m} \lambda_{\Sigma_j}^{-1}\Big). \tag{8.6}$$

Now we check the axioms of Definition 6.1 in order. The value $\mathcal{T}_n$ assigns to an n-cobordism depends only on its in-boundary and out-boundary. Equivalent cobordisms must have the same in-boundary and out-boundary, so (1) holds.

Now consider axiom (2), and let

$$M' : \coprod_{j=0}^{m} \Sigma_j \rightsquigarrow \coprod_{k=0}^{l} \Sigma_k \tag{8.7}$$

be another cobordism. We have

$$\begin{aligned}
\mathcal{T}_n(MM') &= \delta^{-1}\Big(\prod_{i=0}^{n}\lambda_{\Sigma_i}\cdot\prod_{k=0}^{m}\lambda_{\Sigma_k}^{-1}\Big)\\
&= \delta^{-1}\Big(\prod_{i=0}^{n}\lambda_{\Sigma_i}\cdot\prod_{j=0}^{m}\lambda_{\Sigma_j}^{-1}\cdot\prod_{j=0}^{m}\lambda_{\Sigma_j}\cdot\prod_{k=0}^{l}\lambda_{\Sigma_k}^{-1}\Big)\\
&= \delta^{-1}\Big(\prod_{i=0}^{n}\lambda_{\Sigma_i}\cdot\prod_{j=0}^{m}\lambda_{\Sigma_j}^{-1}\Big)\circ\delta^{-1}\Big(\cdot\prod_{j=0}^{m}\lambda_{\Sigma_j}\cdot\prod_{k=0}^{l}\lambda_{\Sigma_k}^{-1}\Big)\\
&= \mathcal{T}_n(M)\circ\mathcal{T}_n(M')
\end{aligned}\tag{8.8}$$

and (2) is satisfied. A connected cylinder cobordism of Σ is a cobordism $\Sigma \rightsquigarrow \Sigma$, so we have

$$\begin{aligned}
\mathcal{T}_n(\Sigma\times I) &= \delta^{-1}(\lambda_\Sigma\cdot\lambda_\Sigma^{-1})\\
&= \delta^{-1}(1)\\
&= id_{\Bbbk_n^\times}
\end{aligned}\tag{8.9}$$

so (3) is satisfied. The disconnected case follows from (4), which clearly follows from Equation 8.6. (5) is satisfied trivially.
Thus $\mathcal{T}_n$ is an n-TQFT, and for all closed manifolds M,

$$\begin{aligned}
\mathcal{T}_n(M) &= \delta^{-1}(\lambda_\emptyset\circ\lambda_\emptyset^{-1})\\
&= \delta^{-1}(1)\\
&= id_{\Bbbk_n^\times}.
\end{aligned}\tag{8.10}$$

Since n was chosen arbitrarily, we have a $*$-TQFT $\mathcal{T}_*$, which is by the previous statement an element of $\mathscr{T}_*$. We chose $\mathcal{T}_*$ to take on nontrivial values, so $\mathcal{T}_*$ is not the trivial TQFT. □

We now want to try to figure out the image of Ψ_*. One problem, of course, comes from the "forgetfullness" of Ψ_*. Given an SKK invariant ξ, we would like to choose an invertible $*$-TQFT $\mathcal{T}_*$ such that $\Psi_*(\mathcal{T}_*)=\xi$. But ξ gives us no explicit information as to how $\mathcal{T}_*$ should evaluate cobordisms with nonempty boundary!

The SKK invariants that we've listed, however, do give us information about how to evaluate such cobordisms. The following proposition gives a necessary and sufficient condition for an invariant Θ on orientated diffeomorphism classes of n-manifolds with boundary to be "naturally identified" with an invertible n-TQFT. By this, we mean that we can define an n-TQFT that sends all boundary manifolds to $\Bbbk$ and evaluates cobordisms the same way Θ evaluates manifolds, modulo the identity $\mathrm{Hom}\,(\Bbbk^\times,\Bbbk^\times)\cong\Bbbk^\times$.

Here let $\mathbb{M}_n^\ddagger$ denote the set of orientated diffeomorphism classes of compact n-manifolds with boundary.

Theorem 8.3. *Let Θ be a function $\mathbb{M}_n^\ddagger \to \Bbbk^\times$. Θ can be naturally identified with an invertible n-TQFT if and only if*

$$\Theta([M \underset{f}{\cup} M']) = \Theta(M) \cdot \Theta(M') \tag{8.11}$$

for all $[M], [M'] \in \mathbb{M}_n^\ddagger$. Here f is any orientation-preserving diffeomorphism $\partial_1 M \to \overline{\partial_0 M'}$, where $\partial_1 M$ and $\partial_0 M'$ are unions of boundary components of M and M', respectively.

Proof. First suppose that Equation 8.11 holds. We check the TQFT axioms of Definition 6.1.

(1) is clearly satisfied, since for any equivalent n-cobordisms M and N, there is an orientation-preserving diffeomorphism $\psi : M \to N$. Θ must evaluate such cobordisms equally. (2) is satisfied by Equation 8.11. To show (3), note that

$$\Theta([\Sigma \times I]) = \Theta([\Sigma \times I]) \cdot \Theta([\Sigma \times I]) \tag{8.12}$$

for all closed $(n-1)$-manifolds Σ, so we must have $\Theta([\Sigma \times I]) = 1$. (4) is also satisfied by Equation 8.11, where f is an empty map. (5) is satisfied trivially.

Now suppose Θ can be naturally extended to an n-TQFT $\mathcal{T}$. Then $\mathcal{T}$ must evaluate n-cobordisms based only on their oriented diffeomorphism class. Now let $f : \partial_1 M \to \overline{\partial_0 M'}$ be an orientation-preserving diffeomorphism. We can easily form cobordisms $M : \overline{\partial_0 M} \rightsquigarrow \partial_1 M$ and $M' : \partial_1 M \rightsquigarrow \partial_1 M'$, where the in-boundary diffeomorphism of M' is f. The glued cobordism then has

$$M \underset{f}{\cup} M' \tag{8.13}$$

as its n-manifold. Thus by axiom (2) of Definition 6.1, Equation 8.11 must hold. ∎

We actually just proved that under the stated conditions, axiom (2) of Definition 6.1 is equivalent to all the axioms of Definition 6.1. However, Equation 8.11 is more readily accessible for our examples, as the following corollaries show.

Corollary 8.4. *If an oriented n-diffeomorphism invariant Θ on manifolds with boundary can be naturally identified with an invertible n-TQFT, then it restricts to a linear combination of the Euler characteristic and signature on closed manifolds.*

Proof. Because our choice of f was arbitrary in Theorem 8.3, Θ restricts to an SK invariant on closed manifolds. The result follows from Theorem 3.11. ∎

Corollary 8.5. *χ and $\chi_{1/2}$ can be naturally identifed with n-TQFTs if and only if n is even. A linear combination of Pontryagin numbers can be naturally identified with a TQFT if and only if it equals the signature.*

Proof. Remember that we must compose each of the additive integer and rational invariants with the map $x \to e^x$ in order for them to meet the hypotheses of Theorem 8.3. Thus what we normally think of as addition for these invariants becomes multiplication.

For n even, χ and $\chi_{1/2}$ satisfy Equation 8.11 by Equation 3.8 and the fact that the Euler characteristic of any closed odd-dimensional manifold is zero. The result then follows from Theorem 8.3.

Now let n be odd. The n-disk has Euler characteristic 1. We can glue two n-disks to form an n-sphere, which has Euler characteristic 0, contradicting Equation 8.11. Thus the Euler characteristic cannot be naturally identified with an n-TQFT for n odd.

That $\chi_{1/2}$ cannot be naturally identified with an n-TQFT for n odd follows from Corollary 8.4.

Now σ satisfies Equation 8.11 by Equation 3.9. The final statement then follows from Corollary 8.4. □

Note that the above corollary is not a proof that Ψ_* is not surjective. We have only given conditions for an *oriented diffeomorphism invariant* on manifolds with boundary to be naturally identified with a TQFT. In general an n-TQFT can pick up more information than just the oriented diffeomorphism class of the n-manifold; in particular, it notices the choice of boundary manifolds and diffeomorphisms. These choices completely determine how cobordisms are glued. This is why Θ does not notice how manifolds are glued.

So we have not ruled out the possibility that, for example, the Pontryagin numbers are in the image of Ψ_*. However, if we were to identify them with a $*$-TQFT, the TQFT would have to detect more than the oriented diffeomorphism type of a cobordism, and therefore disagree with the Pontryagin numbers on some cobordisms with boundary. The same holds for all torsion bordism invariants and the Kervaire semicharacteristic. Such identifications would be rather unnatural.

We now have a description of the image of Ψ_* in certain dimensions for torsion-free multiplicative groups.

Theorem 8.6. *Let $\Bbbk_n^\times$ be torsion-free for $n = 4$ and all $n \not\equiv 0$ mod 4 . Then Ψ_n is surjective in each of these dimensions.*

Proof. Since $\Bbbk_4^\times$ is torsion-free, any homomorphism $\xi : SKK_4 \to \Bbbk_4^\times$ is the composition of the homomorphism $\chi \oplus \sigma : SKK_4 \to \mathbb{Z} \oplus \mathbb{Z}$ and a homomorphism $\xi' : \mathbb{Z} \oplus \mathbb{Z} \to \Bbbk_4^\times$. The oriented diffeomorphism invariant

$$\xi' \circ (\chi \oplus \sigma) : \mathbb{M}_4^\ddagger \to \Bbbk_4^\times \tag{8.14}$$

obeys Equation 8.11, and thus defines a TQFT. The same argument applies for $n \equiv 2$ mod 4, replacing $\chi \oplus \sigma$ with $\chi/2$ and $\mathbb{Z} \oplus \mathbb{Z}$ with $\mathbb{Z}$. For $n \equiv 1, 3$ mod 4, SKK_n is finite and thus admits no nontrivial homomorphisms $SKK_n \to \Bbbk_n^\times$. □

References

[AS68] M. F. Atiyah and I. M. Singer. The index of elliptic operators. III. *Ann. of Math. (2)*, 87:546–604, 1968.

[Ati88] Michael Atiyah. Topological quantum field theories. *Inst. Hautes Études Sci. Publ. Math.*, (68):175–186 (1989), 1988.

[KKNO73] U. Karras, M. Kreck, W. D. Neumann, and E. Ossa. *Cutting and pasting of manifolds;* SK*-groups*. Publish or Perish, Inc., Boston, Mass., 1973. Mathematics Lecture Series, No. 1.

[Koc04] Joachim Kock. *Frobenius algebras and 2D topological quantum field theories*, volume 59 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 2004.

[Mil97] John W. Milnor. *Topology from the differentiable viewpoint*. Princeton Landmarks in Mathematics. Princeton University Press, Princeton, NJ, 1997. Based on notes by David W. Weaver, Revised reprint of the 1965 original.

[MS74] John W. Milnor and James D. Stasheff. *Characteristic classes*. Princeton University Press, Princeton, N. J.; University of Tokyo Press, Tokyo, 1974. Annals of Mathematics Studies, No. 76.

Department of Mathematics, University of Notre Dame, Notre Dame, IN 465561-4618.

E-mail address: `mschoenb@nd.edu`

Completeness of Π

Siva Somayyajula

July 2017

Abstract

Π is a reversible programming language by Sabry et al. inspired by type-theoretic isomorphisms. We give a model for Π: a univalent universe of finite types in homotopy type theory. Using properties of *univalent fibrations*, the underlying concept of this model, we give formal proofs in Agda that programs in Π are complete with respect to this model. Additionally, we discuss this model and extensions to Π through the lens of synthetic homotopy theory.

Contents

1 Introduction

1.1 Reversibility

Reversibility is a paradigm in which computations and their effects may be reversed. This is prevalent in computing applications, giving rise to ad hoc implementations in both hardware and software alike. In particular, transactional databases operate on the basic concept that operations on data may be committed to memory or rolled back [11], and version control systems like `darcs` are based on *patch theory*, an algebra for file changes[1]. At the software level, this has motivated the development of general-purpose reversible programming languages.

Instead of relying on an operational model, the Π language by Sabry et al. begins with different foundations. To elaborate, a natural type-theoretic notion of reversibility is given by type isomorphisms i.e. lossless transformations over structured data. Thus, Π is a calculus for such isomorphisms, giving rise to a feature-complete reversible functional programming language [11]. To understand Π and its model, we give a brief introduction to the type theories we use to formalize them.

1.2 Type Theory

A type theory is a formal system for *types*, *terms*, and their computational interactions. A helpful analogy to understand type theory at first is to conceptualize types as sets and terms as their elements. Like set theory, type theories have rules governing *type formation* as there are axioms about set construction e.g. the axiom of pairing, but there are important distinctions. Whereas set theory makes set membership a proposition provable within the system, terms do not exist without an a priori notion of what type they belong to—one writes $a : A$ (pronounced "a inhabits A") to introduce a term a of type A [5]. As a result, terms are also called *inhabitants*, and we will use those terms (pun intended) interchangeably throughout the rest of the paper.

Perhaps the distinguishing feature of type theories are their explicit treatment of computation: computation rules dictate how terms reduce to values. To programming language theorists, type theories formally describe programming languages and computation rules are precisely the structured operational semantics. On the other hand, set theories have no such equivalent concept.

This emphasis on computation has several applications to computer science. First, the type systems of such programming languages as Haskell are based on certain type theories (specifically, System F). Aside from their utility in programming language design, sufficiently sophisticated type theories are suitable as alternative foundations of mathematics to set theory. In fact, Martin-Löf type theory (MLTT) is the basis of many programs aiming to formalize constructive mathematics. To understand how this is possible, recall that set theories consist of rules governing the behavior of sets as well as an underlying logic to express propositions and their truth. Thus, it remains to show that type theories, under the availability of certain type formers, are languages that can express the construction of arbitrary mathematical objects as well as encode propositions as types and act as deductive systems in their own right [6].

Thus, we will first give a brief introduction to MLTT in Agda, a programming language and proof assistant based on MLTT.

1.3 Martin-Löf Type Theory

Continuing the analogy that types are sets, the following table describes the set-theoretic analogue of each type former in MLTT. The syntax of the terms inhabiting these types are in almost one-to-one correspondence with classical mathematics, with caveats explained below [12].

type	set
U or Type	universal set
$\mathbb{0}$	$\varnothing$
$\mathbb{1}$	singleton
$\mathbb{N}$	Peano numbers
$A + B$	coproduct $A \sqcup B$
$A \times B$	$A \times B$
$A \to B$	function space B^A

The function type is perhaps the most novel type to mathematicians who are used to set theory. First, functions are no longer specialized sets amenable to implicit descriptions, so we require an explicit syntax to construct them. Inspired by Alonzo Church's lambda calculus, functions of type $A \to B$ are written $\lambda x \to e$ (called a *lambda abstraction*) where x is the argument of type A and e is an expression of type B that may freely use x. In Agda, one may either use lambda abstractions or traditional mathematical notation to write functions—we will use both throughout this paper. Then, to apply a function f to argument x, one can write either $f\ x$ or $f(x)$—we will use the former in writing Agda and the latter elsewhere. As an example, consider the following definition of add for the natural numbers. First, the type of the term is declared and then the definition is given.

$$
\begin{aligned}
&\mathsf{add} : \ \ \to \ \ \to \\
&\mathsf{add}\ \mathsf{0}\ \ n = n \\
&\mathsf{add}\ (\mathsf{succ}\ m)\ n = \mathsf{succ}\ (\mathsf{add}\ m\ n)
\end{aligned}
$$

This definition makes use of *currying*—as opposed to writing this multiargument function as being of type $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, we have written a function that consumes an argument of type $\mathbb{N}$–the first argument–and then returns a function of type $\mathbb{N} \to \mathbb{N}$ that consumes the second argument and produces the sum. While the syntactic shortcuts of Agda abstract this distinction away; one could have written $\lambda m \to \lambda n \to \ldots$. Thus, in classical mathematics, add would be applied as $add(1)(2)$. This technique is common in type theory and will be preferred to traditional notation in this paper. Now, to demonstrate the promises of computational benefits by MLTT, we can request Agda to evaluate the following expression:

$$\mathsf{add}\ 1\ 2 \to 3$$

For all types A and B, we can also write a function that swaps the components of a tuple in $A \times B$ and run it on a pair of natural numbers.

$$
\begin{aligned}
&\mathsf{swap} : A \times B \to B \times A \\
&\mathsf{swap}\ (a\ ,\ b) = (b\ ,\ a)
\end{aligned}
$$

$$\mathsf{swap}\ (1, 2) \to (2, 1)$$

Furthermore, we can define types of our own, like $\mathbb{2}$: the Boolean type consisting of two *canonical inhabitants* representing truth values.

$$
\begin{aligned}
&\mathbb{2} : \mathsf{Type}_0 \\
&\mathbb{2} = \mathbb{1} + \mathbb{1}
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{pattern}\ \mathsf{true}\ \ = \mathsf{i}_1\ \mathsf{0}_1 \\
&\mathsf{pattern}\ \mathsf{false} = \mathsf{i}_2\ \mathsf{0}_1
\end{aligned}
$$

We use the term *canonical* to distinguish *values* inhabiting types, as opposed to the infinite possible expressions that evaluate to said values. Here, i_1 and i_2 are the canonical injections of A and B into $A+B$, respectively. Furthermore, 0_1 is the canonical inhabitant of $\mathbb{1}$. Agda's pattern syntax allows us to associate the names true and false with the given values.

This is also our first exposure to MLTT's universe. To avoid Russell's paradox, the universe of types does not contain itself. Instead, Agda has a hierarchy of universes where U_0 is the universe of small types inhabited by $\mathbb{0}$, $\mathbb{1}$, $\mathbb{N}$, etc. Further universes are given by $U_i : U_{i+1}$ and the various type formers like the coproduct inhabit different universes based on its component types. For brevity, we will switch between employing *typical ambiguity*, eliding which universe we are working in by simply writing U, and specifying the level explicitly in code. Now, we may write a function $P : \mathbb{2} \to U$.

```
P : 𝟚 → Type₀
P true  = 𝟙
P false = 𝟘
```

Note that functions like this whose codomains are universes are called *type families*, as they return types instead of ordinary terms.

MLTT then introduces *dependent types*, which generalize the function and Cartesian product types.

Definition 1.1 (Dependent types [6])**.** Let A be a type and $P : A \to U$ be a type family. The *dependent function* type $\prod_{a:A} P(a)$ is inhabited by functions f where if $a : A$, then $f(a) : P(a)$ i.e. functions whose codomain type varies with their input.

Similarly, the *dependent pair* type $\sum_{a:A} P(a)$ is inhabited by (a, b) where $a : A$ and $b : P(a)$ i.e. pairs where the type of the second component varies with the first component.

The utility of these two type formers is elucidated in the following explanation: while we now have a calculus to express arbitrary mathematical objects, we still lack a deductive system to perform mathematical reasoning. In order to develop this, we must first introduce the *Brouwer-Heyting-Kolmogorov (BHK) interpretation*, which not only captures the intuition for proofs in informal mathematics but also expresses them as computable objects.

Definition 1.2 (BHK interpretation [10])**.** We define a proof by induction on the structure of a logical formula.

- There is no proof of $\perp$

Now, let a be a proof of A and b be a proof of B. A proof of…

- …$A \wedge B$ is (a, b) i.e. a proof of A *and* a proof of B
- …$A \vee B$ is either $(0, a)$ or $(1, b)$ i.e. a proof of A *or* a proof of B
- …$A \implies B$ is a computable function that converts a proof of A to a proof of B
- …$\neg A$ is a proof of $A \implies \bot$

Then, fix a domain of discourse D. A proof of…

- …$\forall_{x \in D} P(x)$ is a computable function that converts $a \in D$ to a proof of $P(a)$
- …$\exists_{x \in D} P(x)$ is a pair (a, b) where $a \in D$ and b is a proof of $P(a)$

The proofs described by this interpretation are in exact one-to-one correspondence with the terms inhabiting the various type formers we have just introduced, as shown below [12].

proposition	type
$\bot$	$\mathbb{0}$
$\top$	$\mathbb{1}$
$A \vee B$	$A + B$
$A \wedge B$	$A \times B$
$A \implies B$	$A \to B$
$\neg A$	$A \to \mathbb{0}$
predicate	type family
$\forall_{a \in A} P(a)$	$\prod_{a : A} P(a)$
$\exists_{a \in A} P(a)$	$\sum_{a : A} P(a)$

We can make concrete the correspondence between propositions and types (and consequently proofs and terms) below.

Definition 1.3 (Propositions-as-types)**.** Let A be a type representing a proposition P. If $a : A$, then a is a proof of P in the sense of the BHK interpretation.

With this principle in mind, we can prove some basic propositions in constructive logic, like DeMorgan's law: $\neg A \wedge \neg B \iff \neg(A \vee B)$.

```
DeMorgans₁ : ¬ A × ¬ B → ¬ (A + B)
DeMorgans₁ (¬a , _) (i₁ a) = ¬a a
```

```
DeMorgans₁ (_ , ¬b) (i₂ b) = ¬b b

DeMorgans₂ : ¬ (A + B) → ¬ A × ¬ B
DeMorgans₂ ¬a+b = (( a → ¬a+b (i₁ a)) , ( b → ¬a+b (i₂ b)))
```

Computationally, DeMorgan's law is simply the universal property of the coproduct. Given morphisms $A \to \mathbb{0}$ and $B \to \mathbb{0}$, one can construct a morphism $A+B \to \mathbb{0}$ and vice versa. As a result, the propositions-as-types principle reduces theorem proving to a purely computational endeavor. Now, we can examine the dependent function and pair types. Let us first define the $\leq$ relation on the natural numbers—in MLTT, it is a type family indexed by two natural numbers.

```
_≤_ :  →  → Type₀
0 ≤ n   = 𝟙
(succ m) ≤ (succ n) = m ≤ n
m ≤ n  = 𝟘
```

This definition is quite straightforward: for any number n, $0 \leq n$, and $S(m) \leq S(n)$ if and only if $m \leq n$. Otherwise, the relation does not hold i.e. is defined as absurdity. This allows us to construct computable evidence that a certain number is less than or equal to another one. We can now prove a basic result like $\forall_{n \in \mathbb{N}} \neg(S(n) \leq n)$ by writing a dependent function. Note that in Agda, the dependent function type $\prod_{a:A} P(a)$ is written $(a : A) \to P(a)$.

```
- The codomain type varies on n
succ−n≰n : (n :  ) → ¬ (succ n ≤ n)
- By induction on n
succ−n≰n 0        = id
succ−n≰n (succ n) = succ−n≰n n
```

For the base case, the goal $\neg(1 \leq 0)$ evaluates to $\mathbb{0} \to \mathbb{0}$. Thus, a term of this type is the identity function. For the inductive step, realize that the goal $\neg(S(S(n)) \leq S(n))$ evaluates to $\neg(S(n) \leq n)$. By induction, succ − n ≤ n n : $\neg(S(n) \leq n)$, so the proof is complete.

As stated before, existential quantification is encoded as the dependent pair type—in Agda, $\sum_{a:A} P(a)$ is written Σ A P. Now, we can prove the analogous proposition that for any set A and predicate P on A, $\neg\exists_{a \in A} P(a) \implies \forall_{a \in A} \neg P(a)$.

```
¬Σ−is−Π¬ : ¬ (Σ A P) → (a : A) → ¬ (P a)
¬Σ−is−Π¬ ¬Σ a Pa = ¬Σ (a , Pa)
```

As a result, we could have proven the penultimate result using existential quantification.

```
succ−n≰n′ : (n :  ) → ¬ (succ n ≤ n)
succ−n≰n′ = ¬Σ−is−Π¬ lemma where
  - By induction on n
  lemma : ¬ (Σ  (  n → succ n ≤ n))
  lemma (0  , 1≰0)  = 1≰0
  lemma (succ n , succ−n≰n) = lemma (n , succ−n≰n)
```

The identification of types and propositions mean that proofs are themselves mathematical objects that may be reasoned about—that is, we are doing *proof-relevant mathematics*. Furthermore, the computational content of MLTT is directly accessible. Although these examples are quite tame, more complex proofs are of great utility in software engineering. For example, Euclid's proof of the existence of a greatest common factor (GCF) formalized in a language like Agda is an executable algorithm which computes the GCF correctly. The implications of proof relevance, amongst other things, have motivated the development of *homotopy type theory*, the type theory underlying the results of this paper.

1.4 Homotopy Type Theory

In the previous section, we gave an informal exposition of MLTT by appealing to set theory—in other words, we interpreted the various type formers as set constructors, terms as elements, and discussed their computational and logical interactions. However, we are missing a type that expresses *propositional equality* i.e. propositions that two objects a and b are equal.

Definition 1.4 (Identity type [12])**.** For all types A and $a, b : A$, the *identity type* $a = b$ is inhabited by proofs that a and b are equal, called *identifications*.

By definition, the canonical method of introducing an inhabitant of this type is by reflexivity: $refl = \prod_{a:A} a = a$.

Structural induction upon terms of this type is not as straightforward as with the other type formers. One would expect to be able to simply reduce every encounter of the identity type to reflexivity during theorem proving, but that defies the homotopy-theoretic interpretation of type theory due to *homotopy type theory* (HoTT). When types are interpreted as spaces and terms as points, we get the following correspondence [12].

type theory	homotopy theory
type	space
term	point
type family	fibration
$a = b$	path space

The last point is crucial—the identity type on points a and b is interpreted as the space of paths from a to b. As a result, being able to reduce any term inhabiting the identity type to reflexivity is tantamount to contracting any path to a constant loop, which is nonsensical in homotopy theory! In fact, only when at least one endpoint—either a or b—is free to vary, can one contract a path to a constant loop by moving the free point to the other. This intuition allows us to first define the $PathFrom$ type family, which maps a fixed point x to the space of paths emanating from it i.e. an entire subspace of free points.

Definition 1.5 (PathFrom [8])**.**

$$PathFrom(x) \triangleq \sum_{y:A} x = y$$

The following principle then allows us to reduce certain paths to constant loops under the exact conditions described.

Definition 1.6 (Paulin-Mohring's J [8])**.** Given a type family $P : PathFrom(x) \to U$, $J : P(x, refl(x)) \to \prod_{p:PathFrom(x)} P(p)$ with the following computation rule:

$$J\ r\ (x, refl(x)) \to r$$

Thus, it is impossible to prove that *all* inhabitants of the identity type are identical to reflexivity [7]. Likewise, not every path is contractible to a constant loop. In fact, one can only prove that inhabitants of $PathFrom(x)$ are propositionally equal to $(x, refl(x))$ since the second endpoint is left free.

```
PathFrom−unique : (yp : PathFrom x) → yp == (x , refl x)
PathFrom−unique = J (  yp → yp == (x , refl x)) (refl (x , refl x))
```

As a result, this allows us to add so-called nontrivial (non-reflexivity) inhabitants to the identity type via separate inference rules without rendering the system inconsistent. Motivated by the simplicial set model of type theory, HoTT adds such inhabitants expressing the extensional equality of various objects. For example, given functions $f, g : A \to B$, if one has evidence $\alpha : \prod_{x:A} f(x) = g(x)$, the axiom of function extensionality gives $\mathsf{funext}(\alpha) : f = g$. However, the crux of HoTT lies in Voevodsky's univalence axiom, which is an extensionality axiom for *types*. Before we introduce it, we must first define what it means for two types to be *equivalent*, or extensionally equal.

Definition 1.7 (Quasi-inverse [12])**.** A *quasi-inverse* of a function $f : A \to B$ is the following dependent triple:

- $g : B \to A$
- $\alpha : \prod_{x:A} g(f(x)) = x$
- $\beta : \prod_{x:B} f(g(x)) = x$

For the purposes of this paper, we will refer to functions that have quasi-inverses as equivalences, although there are other equivalent notions in type theory. In Agda, we must explicitly specify which type of equivalence we are providing i.e. qinv−is−equiv for quasi-inverses. We can now give our notion of extensionality for types.

Definition 1.8 (Type equivalence [12])**.** Given types X and Y, $X \simeq Y$ if there exists a function $f : X \to Y$ that is an equivalence.

Perhaps the most trivial equivalence is given below.

Theorem 1.1 (Identity equivalence)**.** For any type A, $A \simeq A$ by the identity function—the dependent pair of id and evidence that it has a quasi-inverse is called ide A in Agda.

An immediate result is that an equality between types can be converted to an equivalence.

Theorem 1.2 (idtoeqv)**.** For all types A and B, $A = B \to A \simeq B$.

Proof. Using J reduces the proof goal to giving a term of type $A \simeq A$ i.e. the identity equivalence.

```
idtoeqv : A == B → A ≃ B
idtoeqv p = J ( {(B , _) → A ≃ B}) (ide A) (B , p)
```

□

Axiom 1.1 (Univalence [12])**.** idtoeqv is an equivalence.

By declaring that idtoeqv has a quasi-inverse, this axiom gives us the following data:

- $ua : A \simeq B \to A = B$, a function that converts equivalences to paths
- $\prod_{f : A \simeq B} idtoeqv(ua(f)) = f$
- $\prod_{p : A = B} ua(idtoeqv(p)) = p$

The last two data are called *propositional computation rules*, as they dictate how *ua* reduces propositionally, outside of the computation rules built into type theory. However, this raises the question: how do terms evaluate to a value in the presence of univalence? This is actually still an open question—for now, homotopy type theory lacks *canonicity*, the guarantee that every term has a canonical form.

Univalence is justified when we broaden our interpretation of types to not just spaces but to *homotopy types*—spaces regarded up to homotopy equivalence. In that sense, *ua* is simply the trivial assertion that spaces that are homotopy equivalent are equal (up to homotopy equivalence).

Before moving onto Π and its model, we must establish one last concept and rethink our previous conception of propositions-as-types. Recall that we are doing proof-relevant mathematics. However, classical mathematics is decidedly proof-irrelevant since propositions are simply assigned a truth value without additional information. In terms of type theory, this would mean the terms of every type would be indistinguishable up to propositional equality. As a result, the only information we would have about a tautology encoded as a type is that it is inhabited by *some* value, and an absurdity would simply be uninhabited. We formalize this intuition below.

Definition 1.9 (Mere proposition [12])**.** A type is a *mere proposition* if all of its inhabitants are propositionally equal. That is, the following type is inhabited:

$$isProp(A) \triangleq \prod_{x,y : A} x = y$$

This allows us to formalize analogies between classical mathematics (we avoid the phrase "classical logic," which is related to mere propositions but not expounded here) and type theory.

Theorem 1.3 (Logical equivalence [12])**.** For all mere propositions A and B, if $A \to B$ and $B \to A$, then $A \simeq B$. That is, to show that two mere propositions are equivalent, it is sufficient to show that they are logically equivalent.

Proof. To show $f : A \to B$ and $g : B \to A$ are inverses, we identify $g(f(x))$, x, $f(g(y))$ and y for $x : A$, and $y : B$, respectively, by the fact that A and B are mere propositions.

$$
\begin{aligned}
&\textsf{logical-equiv} :\\
&\quad \textsf{is-prop}\ A \to \textsf{is-prop}\ B \to (A \to B) \to (B \to A) \to A \simeq B\\
&\textsf{logical-equiv}\ pA\ pB\ f\ g =\\
&\quad f\ ,\ \textsf{qinv-is-equiv}\ (g\ ,\ (\ x \to pA\ (g\ (f\ x))\ x)\ ,\ (\ y \to pB\ (f\ (g\ y))\ y))
\end{aligned}
$$

□

For types that are not mere propositions, we may construct an analogue that is.

Definition 1.10 (Propositional truncation [12])**.** For a type A, its propositional truncation $\| A \|$ is described by the following

- If $a : A$, then $| a | : \| A \|$
- $identify : \Pi_{x,y : \|A\|} x = y$

By $identify$, the propositional truncation of any type is a proposition, hence the name.

Structural induction upon inhabitants of a propositional truncation is subtle—a function can only recover the original term underneath the truncation bars if its codomain itself is a mere proposition. We will see this principle show up as recTrunc later on.

In short, mere propositions allow us to encode proof-irrelevance into type theory. This is key in defining the *univalent universe of finite types*, the model of Π, which we will do in the next section.

2 Univalent Universe of Finite Types

The underlying characterization of this subuniverse relies on a concept called *univalent fibrations*.

2.1 Univalent Fibrations

An elementary result in homotopy theory is that a path between points x and y in the base space of a fibration induces an equivalence between the fibers over x and y. By univalence, this equivalence is a path as well. We formalize this result below.

Theorem 2.1 (transporteqv)**.** For any type A and $x, y : A$, $\prod_{P:A\to U} x = y \to P(x) \simeq P(y)$.

Proof. By J, we may reduce the proof goal to giving a term of type $P(x) \simeq P(x)$ i.e. the identity equivalence.

```
transporteqv : (P : A → Type ℓ) → x == y → P x ≃ P y
transporteqv P p = J ( {(y , _) → P x ≃ P y}) (ide (P x)) (y , p)
```

□

However, converse is not always true—type families that satisfy this property are called univalent fibrations.

Definition 2.1 (Univalent Fibration [4])**.** For all types A, a type family $P : A \to U$ is a *univalent fibration* if transporteqv(P) is an equivalence.

That is, univalent fibrations come with a quasi-inverse of transporteqv that converts fiberwise equivalences to paths in the base space. Even though it is rarely the case that any given type family is a univalent fibration, the following theorem characterizes a class of families that are.

Theorem 2.2 (Rose, 2017)**.** Let $P : U \to U$ be a type family. If for all $X : U$, $P(X)$ is a mere proposition, then the first projection $p_1 : \sum_{X:U} P(X) \to U$ is a univalent fibration.

2.2 The is–finite Family

We will now examine the is–finite type family which forms the basis of the model for Π. First, we require a canonical notion of a finite type.

Definition 2.2 (El)**.** The El family sends a natural number n to a finite type with n canonical inhabitants.

$$
\begin{aligned}
&\mathsf{El} : \quad \to \mathsf{Type}_0 \\
&\mathsf{El}\ 0 \ = \mathbb{0} \\
&\mathsf{El}\ (\mathsf{succ}\ n) = \mathbb{1} + \mathsf{El}\ n
\end{aligned}
$$

To see that this definition is sufficient, we can enumerate all n canonical inhabitants of $\mathsf{El}\ n$.

$$
\begin{array}{c|l}
1 & i_1(0_1) \\
2 & i_2(i_1(0_1)) \\
3 & i_2(i_2(i_1(0_1))) \\
\vdots & \vdots \\
\text{n} & \underbrace{i_2(i_2(\ldots}_{n}(i_1(0_1))\ldots))
\end{array}
$$

Notice that we never reach $i_2(i_2(\ldots(i_2(...))\ldots))$ because that would require giving an inhabitant of $\mathbb{0}$, which is impossible. Thus, we are guaranteed n canonical inhabitants. Now, we are ready to define the $\mathsf{is-finite}$ family.

$$
\begin{aligned}
&\mathsf{is{-}finite} : \mathsf{Type}_0 \to \mathsf{Type}_1 \\
&\mathsf{is{-}finite}\ A = \Sigma \quad (\ n \to \|\ A == \mathsf{El}\ n\ \|)
\end{aligned}
$$

Viewed as a predicate, this says "a type is finite if it is equivalent to a canonical finite type." Computationally, we require a proof-irrelevant identification of A and $\mathsf{El}\ n$ for some n. Then, we define the univalent universe of finite types to be the subuniverse of types satisfying this predicate.

$$
\begin{aligned}
&\mathsf{M} : \mathsf{Type}_1 \\
&\mathsf{M} = \Sigma\ \mathsf{Type}_0\ \mathsf{is{-}finite}
\end{aligned}
$$

Terms of this type are triples consisting of (1) a type A, (2) the "size" of A, and (3) a path witnessing the given size is correct by identifying A with a canonical finite type of the same size. The reason we truncate the above instance of the identity type is to yield the following result.

Theorem 2.3 (Rose, 2017)**.** The first projection p_1 of triples in M is a univalent fibration.

Proof. For any A, $isFinite(A)$ is a mere proposition due to the truncation of its second component, amongst other things. Thus, from theorem 2.2, p_1 is a univalent fibration. □

This is the workhorse of our completeness result—intuitively, to induce a path between two triples, one simply needs to give an equivalence between their first components, which minimizes our proof obligations.

3 Pi

Now that we are acquainted with HoTT and finite types, we can examine the Π programming language by Sabry et al. Π starts with the notion that type equivalences are a natural expression of reversibility—one can write and execute a program and invert its effects via its quasi-inverse. Π then restricts its type calculus to the semiring $(\{\mathbb{0}, \mathbb{1}\}, +, \times)$ up-to type equivalence. As a result, a complete characterization of equivalences over these types is precisely the semiring axioms in figure 1. Note that Π uses $\leftrightarrow$ for $\simeq$.

$$
\begin{array}{rrcll}
id{\leftrightarrow} : & \tau & \leftrightarrow & \tau & : id{\leftrightarrow} \\
\\
unite_{+}l : & \mathbb{0} + \tau & \leftrightarrow & \tau & : uniti_{+}l \\
swap_{+} : & \tau_1 + \tau_2 & \leftrightarrow & \tau_2 + \tau_1 & : swap_{+} \\
assocl_{+} : & \tau_1 + (\tau_2 + \tau_3) & \leftrightarrow & (\tau_1 + \tau_2) + \tau_3 & : assocr_{+} \\
\\
unite_{*}l : & \mathbb{1} \times \tau & \leftrightarrow & \tau & : uniti_{*}l \\
swap_{*} : & \tau_1 \times \tau_2 & \leftrightarrow & \tau_2 \times \tau_1 & : swap_{*} \\
assocl_{*} : & \tau_1 \times (\tau_2 \times \tau_3) & \leftrightarrow & (\tau_1 \times \tau_2) \times \tau_3 & : assocr_{*} \\
\\
absorbr : & \mathbb{0} \times \tau & \leftrightarrow & \mathbb{0} & : factorzl \\
dist : & (\tau_1 + \tau_2) \times \tau_3 & \leftrightarrow & (\tau_1 \times \tau_3) + (\tau_2 \times \tau_3) & : factor
\end{array}
$$

$$
\frac{\vdash c : \tau_1 \leftrightarrow \tau_2}{\vdash !c : \tau_2 \leftrightarrow \tau_1}
\qquad
\frac{\vdash c_1 : \tau_1 \leftrightarrow \tau_2 \quad \vdash c_2 : \tau_2 \leftrightarrow \tau_3}{\vdash c_1 \odot c_2 : \tau_1 \leftrightarrow \tau_3}
$$

$$
\frac{\vdash c_1 : \tau_1 \leftrightarrow \tau_2 \quad \vdash c_2 : \tau_3 \leftrightarrow \tau_4}{\vdash c_1 \oplus c_2 : \tau_1 + \tau_3 \leftrightarrow \tau_2 + \tau_4}
\qquad
\frac{\vdash c_1 : \tau_1 \leftrightarrow \tau_2 \quad \vdash c_2 : \tau_3 \leftrightarrow \tau_4}{\vdash c_1 \otimes c_2 : \tau_1 \times \tau_3 \leftrightarrow \tau_2 \times \tau_4}
$$

Figure 1: Level 1 Programs (equivalences) in Π [3]

For example, recall that the Boolean data type can be encoded as $\mathbb{1} + \mathbb{1}$. Negation, which sends true to false and vice versa, is an equivalence. We may define it in many ways—we give two below.

$$
\begin{array}{l}
\mathsf{NOT}_1 : \mathbb{2} \longleftrightarrow \mathbb{2} \\
\mathsf{NOT}_1 = \mathsf{swap}_{+}
\end{array}
$$

$\mathsf{NOT}_2 : \mathbb{2} \longleftrightarrow \mathbb{2}$
$\mathsf{NOT}_2 = \mathsf{id}\!\longleftrightarrow \circledcirc (\mathsf{swap}_+ \circledcirc \mathsf{id}\!\longleftrightarrow)$

Furthermore, one can ask whether two equivalences are extensionally equal. Π then includes a language which encodes such proofs, called *coherences*, shown in figure 2.

As a result, we can write a proof that NOT_1 and NOT_2 are equivalent by cancelling out the instances of $\mathsf{id}\!\longleftrightarrow$.

$\mathsf{NOT}_1\Leftrightarrow\mathsf{NOT}_2 : \mathsf{NOT}_1 \Leftrightarrow \mathsf{NOT}_2$
$\mathsf{NOT}_1\Leftrightarrow\mathsf{NOT}_2 = 2!\ (\mathsf{idl}\circledcirc\mathsf{l}\ 2\circledcirc\ \mathsf{idr}\circledcirc\mathsf{l})$

Now that we have a language that describes various finite types and their equivalences as well as a model for them in HoTT, we would like to determine whether the language is complete with respect to the model—that is, for every object in the model, there exists an equivalent one in the language and vice versa.

4 Completeness of Level 0

Now, we can discuss the completeness of level 0, or types in Π with respect to the given model. First, we require translations from the syntax to the model and vice versa. Assume we have the following functions defined.

```
- Converts a type in the syntax
- to the exact same type in MLTT
#⟦_⟧₀ : S → Type₀

- Computes the number of canonical
- inhabitants of a type in the syntax
size : S →

- Converts an equivalence in the
- syntax to the same one in HoTT
#⟦_⟧₁ : {X Y : S} → X ⟷ Y → #⟦ X ⟧₀ ≃ #⟦ Y ⟧₀
```

In order to write the translation into the model, we need a way of relating any type in the semiring T to $El(n)$ where $n = size(T)$. Note that the image of $El(n)$ is a subtype of S, allowing us to write an analogous function into S.

$$\mathsf{fromSize} : \quad \to \mathsf{S}$$

We can formalize the relationship between fromSize and El n as follows.

$$\mathsf{fromSize{=}El} : \{n : \quad\} \to \#[\![\, \mathsf{fromSize}\ n \,]\!]_0 == \mathsf{El}\ n$$

Then, we define canonical, which converts a type in the semiring to its "canonical" form.

$$\mathsf{canonical} : \mathsf{S} \to \mathsf{S}$$
$$\mathsf{canonical} = \mathsf{fromSize} \circ \mathsf{size}$$

Here is an example of the action of canonical:

$$\mathsf{canonical}\ ((\mathbb{1} + \mathbb{1}) \times (\mathbb{1} + \mathbb{1})) \to \mathbb{1} + \mathbb{1} + \mathbb{1} + \mathbb{1} + \mathbb{0}$$

Intuitively, a type is equivalent to its canonical form, allowing us to write a function that constructs an equivalence in the syntax between them (due to Sabry et al).

$$\mathsf{normalize} : (T : \mathsf{S}) \to T \longleftrightarrow \mathsf{canonical}\ T$$

We can finally write the translation by using the above functions. Note that we use univalence to convert the equivalence between a type and its canonical form to a path. Then, we use $\blacksquare$ to concatenate that with the path given by fromSize=El where $n = size(T)$ to generate a path of type $T = El(n)$.

$$[\![\_]\!]_0 : \mathsf{S} \to \mathsf{M}$$
$$[\![\, T \,]\!]_0 = (\#[\![\, T \,]\!]_0\ ,\ \mathsf{size}\ T\ ,\ |\ \mathsf{ua}\ \#[\![\, \mathsf{normalize}\ T \,]\!]_1 \blacksquare\ \mathsf{fromSize{=}El}\ |)$$

This definition is quite complex, so figure 3 demonstrates its action as an injection into the model.

The translation of the model into the syntax is much simpler—since one cannot perform induction on the opaque type in the first component, we must return the next best thing: a conversion of the size in the second component to a type in the syntax.

$$[\![\_]\!]_0^{-1} : \mathsf{M} \to \mathsf{S}$$
$$[\![(T\ ,\ n\ ,\ p)]\!]_0^{-1} = \mathsf{fromSize}\ n$$

We can again view the action of this translation as an injection in figure 4, taking a triple in the model to a canonical form in the syntax.

We now have the sufficient tools to discuss the completeness of level 0. Let us formalize the statements of completeness we made two sections ago.

$$cmpl^0_1 : \prod_{T_1 : S} \sum_{T_2 : M} T_1 \leftrightarrow [\![T_2]\!]^{-1}_0 \qquad\qquad cmpl^0_2 : \prod_{T_1 : M} \sum_{T_2 : S} \| T_1 = [\![T_2]\!]_0 \|$$
$$T \mapsto ([\![T]\!]_0, lem_1) \qquad\qquad\qquad T \mapsto ([\![T]\!]^{-1}_0, lem_2)$$

By sending each input to their respective translations, we have proof obligations $lem_1 : \prod_{T:S} T \leftrightarrow [\![[\![T]\!]_0]\!]^{-1}_0$ and $lem_2 : \prod_{T:M} \| T = [\![[\![T]\!]^{-1}_0]\!]_0 \|$. Intuitively, these each say that going back and forth between the syntax and model (and vice versa) produces an equivalent object—let us prove them. To prove the first lemma, consider figure 5, which depicts the round trip of applying both translations.

It seems that we simply must construct an equivalence between a type in the syntax and its canonical form, in the same way we did for $[\![\cdot]\!]_0$.

$\mathsf{lem}_1 : (T : \mathsf{S}) \to T \longleftrightarrow [\![\, [\![\, T \,]\!]_0 \,]\!]^{-1}_0$
$\mathsf{lem}_1 = \mathsf{normalize}$

The other direction is a bit more difficult. First, by theorem 2.3 and idtoeqv, we can define a function that converts paths between the first components of a triple in the model to a path between the entire triple.

$\mathsf{induce} : \{X\, Y : \mathsf{M}\} \to \mathsf{p}_1\, X == \mathsf{p}_1\, Y \to X == Y$

Now, let us observe the this round trup in figure 6—it yields a similar triple but the first component is in canonical form. Precisely by the original path, we may induce a path across both triples by the fact that the first projection is univalent.

This allows us to prove lem_2 by induction on the truncated path in the third component of a triple, which by induce, gives us the necessary result.

$\mathsf{lem}_2 : (X : \mathsf{M}) \to \| X == [\![\, [\![\, X \,]\!]^{-1}_0 \,]\!]_0 \|$
$\mathsf{lem}_2\, (T\, ,\, n\, ,\, p) =$
  $\mathsf{recTrunc}\ \_\ (\ p' \to |\ \mathsf{induce}\ (p' \blacksquare\ !\ \mathsf{fromSize{=}El})\ |)\ \mathsf{identify}\ p$

With these lemmas, we may formally state these completeness results in Agda.

$\mathsf{cmpl}^0_1 : (T_1 : \mathsf{S}) \to \Sigma\, \mathsf{M}\, (\ T_2 \to T_1 \longleftrightarrow [\![\, T_2 \,]\!]^{-1}_0)$
$\mathsf{cmpl}^0_1\, T_1 = ([\![\, T_1 \,]\!]_0\, ,\, \mathsf{lem}_1\, T_1)$

$\mathsf{cmpl}^0_2 : (T_1 : \mathsf{M}) \to \Sigma\, \mathsf{S}\, (\ T_2 \to \| T_1 == [\![\, T_2 \,]\!]_0 \|)$
$\mathsf{cmpl}^0_2\, T_1 = ([\![\, T_1 \,]\!]^{-1}_0\, ,\, \mathsf{lem}_2\, T_1)$

5 Future Work

We are currently working on completeness results on levels 1 and 2: isomorphisms and coherences. Furthermore, we would like to develop the formal theory surrounding reversible programming. In particular, there is a deep interplay between homotopy theory and reversibility. For example, we do not have a clear perception of reversible programming with *higher inductive types*, HoTT's internalization of homotopy types. Furthermore, we have the following conjecture which gives a topological characterization of our model, in terms of Eilenberg-MacLane (EM) spaces.

Conjecture 5.1 (Rose, 2017)**.**

$$M = \bigoplus_{n \in \mathbb{N}} K(S_n, 1)$$

where S_n is a symmetric group.

An EM-space $K(G, n)$ has its n^{th} homotopy group (group of n-paths under concatenation and inversion) isomorphic to G and every other one trivial [9]. Thus, this conjecture captures all the necessary information about paths in the model (and equivalences), and therefore the inherent reversibility.

6 Acknowledgements

We thank Prof. Amr Sabry, Prof. Jacques Carette, Robert Rose, Vikraman Choudhury, and Chao-Hong Chen for the collaborative effort on this project. This work is supported by NSF REU Grant #1461061.

References

[1] Patch theory.

[2] Carette, J., Chen, C.-H., Choudhury, V., and Sabry, A. Fractional types.

[3] Carette, J., and Sabry, A. Computing with semirings and weak rig groupoids. In *Proceedings of the 25th European Symposium on Programming Languages and Systems - Volume 9632* (New York, NY, USA, 2016), Springer-Verlag New York, Inc., pp. 123–148.

[4] Christensen, D. A characterization of univalent fibrations. 2015.

[5] Coquand, T. Type theory, Feb 2006.

[6] Dybjer, P., and Palmgren, E. Intuitionistic type theory, Feb 2016.

[7] Hofmann, M., and Streicher, T. The groupoid interpretation of type theory. In *In Venice Festschrift* (1996), Oxford University Press, pp. 83–111.

[8] Licata, D. R. Just kidding: Understanding identity elimination in homotopy type theory, Nov 2015.

[9] Licata, D. R., and Finster, E. Eilenberg-maclane spaces in homotopy type theory. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)* (New York, NY, USA, 2014), CSL-LICS '14, ACM, pp. 66:1–66:9.

[10] Moschovakis, J. Intuitionistic logic, Sep 1999.

[11] Sabry, A. From reversible programming languages to univalent universes and back. 2017.

[12] Univalent Foundations Program, T. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

$$\frac{c : \tau_1 \leftrightarrow \tau_2}{\mathsf{id} : c \Leftrightarrow c} \qquad \frac{c_1, c_2, c_3 : \tau_1 \leftrightarrow \tau_2 \quad \alpha_1 : c_1 \Leftrightarrow c_2 \quad \alpha_2 : c_2 \Leftrightarrow c_3}{\alpha_1 \bullet \alpha_2 : c_1 \Leftrightarrow c_3}$$

$$\frac{c_1 : \tau_1 \leftrightarrow \tau_2 \quad c_2 : \tau_2 \leftrightarrow \tau_3 \quad c_3 : \tau_3 \leftrightarrow \tau_4}{\mathsf{assoc}_{\odot}\mathsf{l} : c_1 \odot (c_2 \odot c_3) \Leftrightarrow (c_1 \odot c_2) \odot c_3 : \mathsf{assoc}_{\odot}\mathsf{r}}$$

$$\frac{c : \tau_1 \leftrightarrow \tau_2}{\mathsf{idl}_{\odot}\mathsf{l} : \mathsf{id} \odot c \Leftrightarrow c : \mathsf{idl}_{\odot}\mathsf{r}} \qquad \frac{c : \tau_1 \leftrightarrow \tau_2}{\mathsf{idr}_{\odot}\mathsf{l} : c \odot \mathsf{id} \Leftrightarrow c : \mathsf{idr}_{\odot}\mathsf{r}}$$

$$\frac{c : \tau_1 \leftrightarrow \tau_2}{\mathsf{rinv}_{\odot}\mathsf{l} : !c \odot c \Leftrightarrow \mathsf{id} : \mathsf{rinv}_{\odot}\mathsf{r}} \qquad \frac{c : \tau_1 \leftrightarrow \tau_2}{\mathsf{linv}_{\odot}\mathsf{l} : c \odot !c \Leftrightarrow \mathsf{id} : \mathsf{linv}_{\odot}\mathsf{r}}$$

$$\frac{}{\mathsf{sumid} : \mathsf{id} \oplus \mathsf{id} \Leftrightarrow \mathsf{id} : \mathsf{splitid}}$$

$$\frac{c_1 : \tau_5 \leftrightarrow \tau_1 \quad c_2 : \tau_6 \leftrightarrow \tau_2 \quad c_3 : \tau_1 \leftrightarrow \tau_3 \quad c_4 : \tau_2 \leftrightarrow \tau_4}{\mathsf{hom}_{\oplus\odot} : (c_1 \odot c_3) \oplus (c_2 \odot c_4) \Leftrightarrow (c_1 \oplus c_2) \odot (c_3 \oplus c_4) : \mathsf{hom}_{\odot\oplus}}$$

$$\frac{c_1, c_3 : \tau_1 \leftrightarrow \tau_2 \quad c_2, c_4 : \tau_2 \leftrightarrow \tau_3 \quad \alpha_1 : c_1 \Leftrightarrow c_3 \quad \alpha_2 : c_2 \Leftrightarrow c_4}{\alpha_1 \boxdot \alpha_2 : c_1 \odot c_2 \Leftrightarrow c_3 \odot c_4}$$

$$\frac{c_1, c_3 : \tau_1 \leftrightarrow \tau_2 \quad c_2, c_4 : \tau_2 \leftrightarrow \tau_3 \quad \alpha_1 : c_1 \Leftrightarrow c_3 \quad \alpha_2 : c_2 \Leftrightarrow c_4}{\mathsf{resp}_{\oplus\Leftrightarrow}\ \alpha_1\ \alpha_2 : c_1 \oplus c_2 \Leftrightarrow c_3 \oplus c_4}$$

$$\frac{c_1, c_3 : \tau_1 \leftrightarrow \tau_2 \quad c_2, c_4 : \tau_2 \leftrightarrow \tau_3 \quad \alpha_1 : c_1 \Leftrightarrow c_3 \quad \alpha_2 : c_2 \Leftrightarrow c_4}{\mathsf{resp}_{\otimes\Leftrightarrow}\ \alpha_1\ \alpha_2 : c_1 \otimes c_2 \Leftrightarrow c_3 \otimes c_4}$$

Figure 2: Level 2 Programs (coherences) in Π [2]

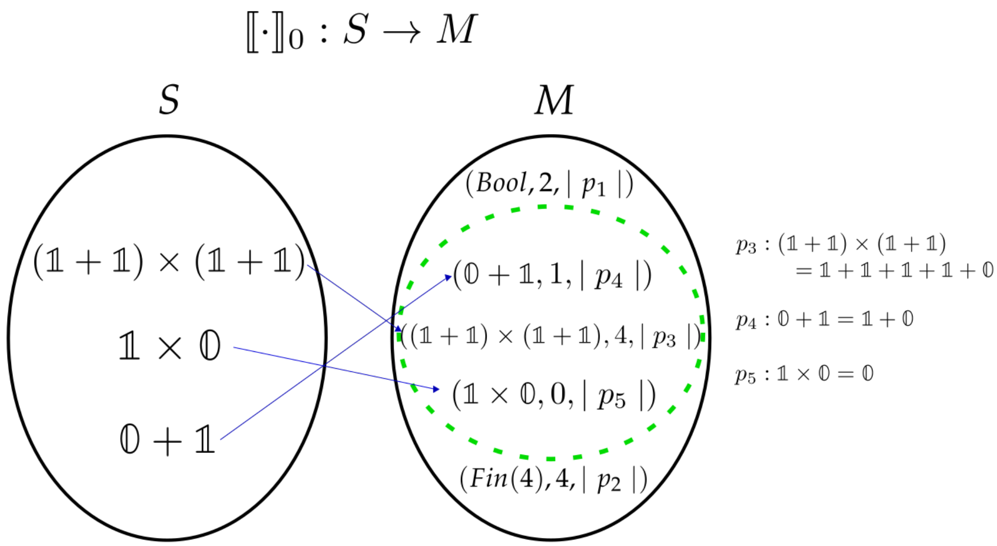


Figure 3: The action of $[\![\cdot]\!]_0$

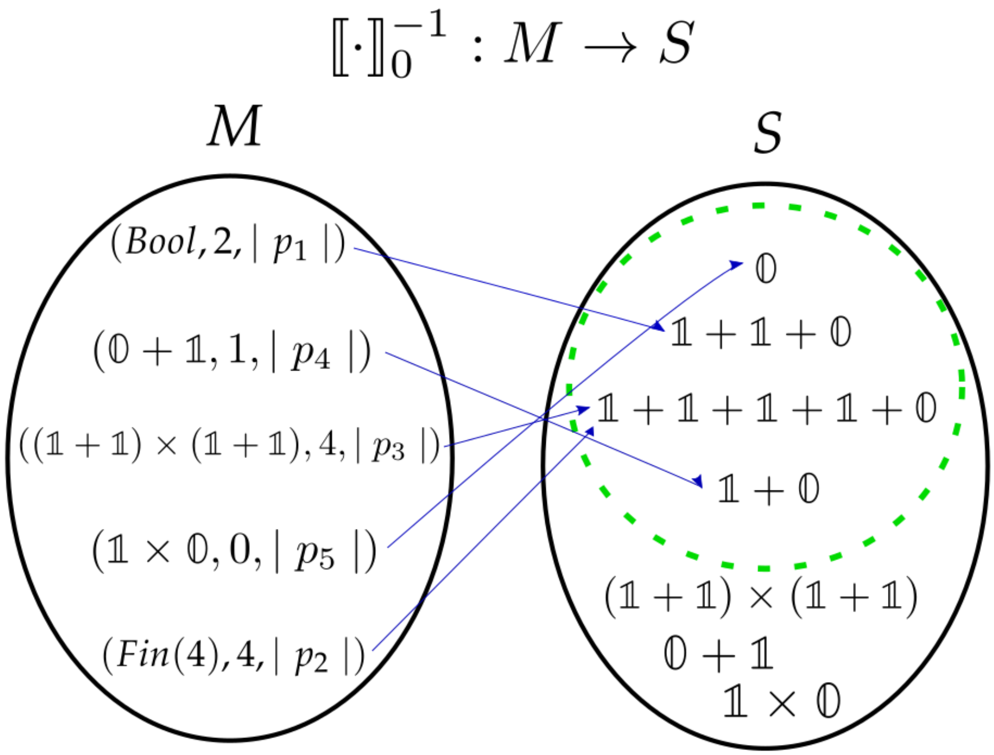


Figure 4: The action of $[\![\cdot]\!]_0^{-1}$

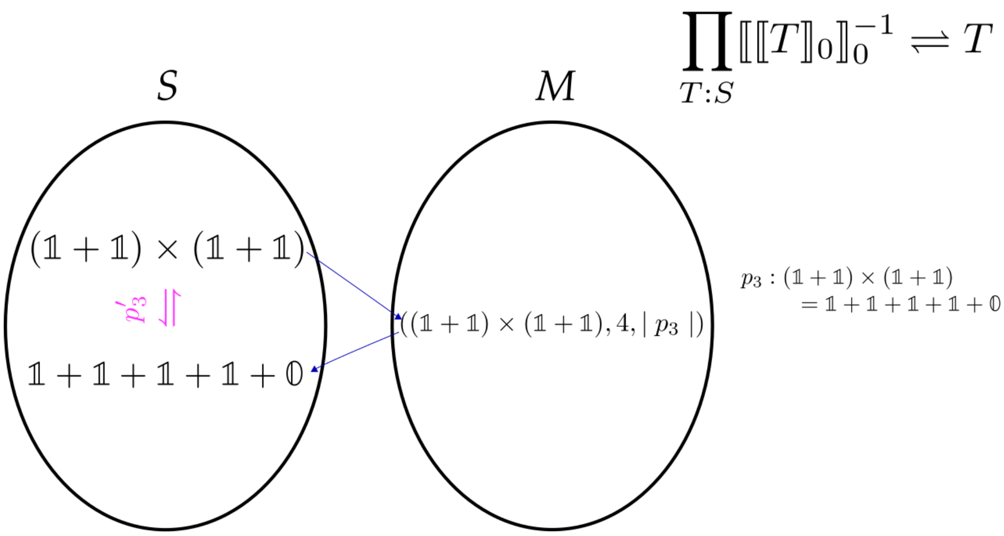


Figure 5: The action of the translation then its “inverse”

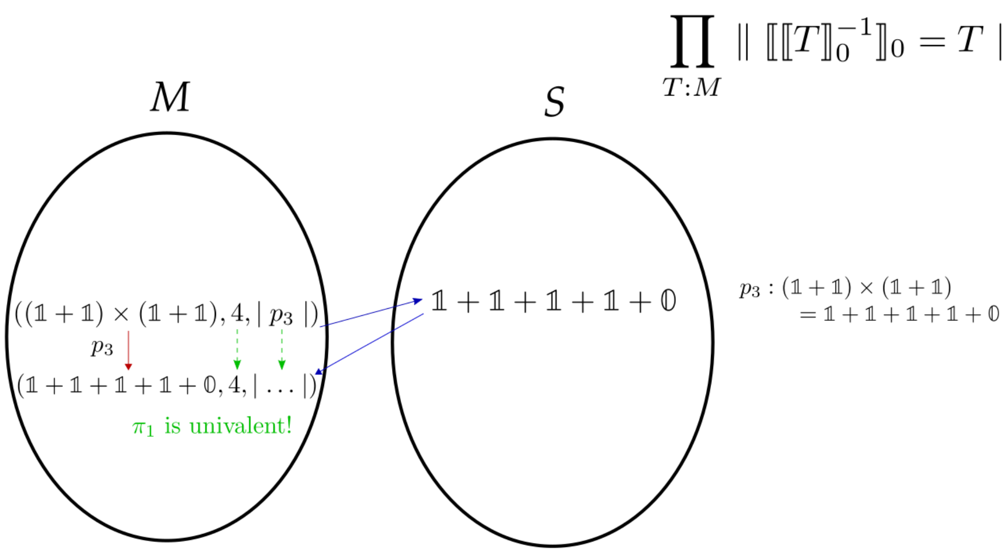


Figure 6: The action of the “inverse” then the usual translation