Spring 7-12-2016

Analyzing Clustered Web Concepts with Homology

Eric Nam
San Jose State University

Analyzing Clustered Web Concepts with Homology

A project

Presented to

The Faculty of the Department of Computer Science

San Jose Staté University

In Partial Fulfilment

Of the Requirements for the Degree

Master of Science

by

Eric Nam

May 2016

The Designated Project Committee Approves the Project Titled

Analyzing Clustered Web Concepts with Homology

by

Eric Nam

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2016

Dr. Tsau Young Lin Department of Computer Science

Dr. Jon Pearce Department of Computer Science

Dr. Thomas Austin Department of Computer Science

ABSTRACT

Analyzing Clustered Web Concepts with Homology

by

Eric Nam

As data is being mined more and more from the Internet today, Data Science has become an important field of computing to make that data useful. Data Science allows people to turn all of that data into structured knowledge that is easily utilized, validated, and understandable. There are many known theories to analyze data, but this project will focus on a recently introduced method: analyzing text data with homology from mathematics to understand relationships between keyword-sets.

Using structures of algebraic topology as a starting point, keyword-sets in the text are represented by simplexes based on what they are and what their length is. These sets of simplexes come together to make up clustered simplicial complexes, all laying the groundwork for homology to come into play. By calculating homology on all of these simplicial complexes, we can then know the relations between keyword-sets better. Previous work on data analysis of text data through homology was based on establishing the relationships on the real space, but this project extends that to integer space so that the homology can reveal more detail about those relationships.

ACKNOWLEDGEMENTS

I would like first to thank my advisor Dr. Tsau Young Lin as without his mentoring and assistance over the past year, I would not be able to finish this project. I would also like to thank my committee members Dr. Jon Pearce and Dr. Thomas Austin for their advice and assistance in improve this project. Lastly, I also thank my friends and family for their candid support in my endeavors for my Masters Degree.

Contents

List of Figures

Chapter 1

Introduction

With every passing year, what we can know from the Internet and what we want to know from the Internet becomes more and more. In order to know what we want to know from the internet, software turns to data mining in order to come up with answers to both server and user queries. However, the growing amount of data gathered present newer and more specific challenges, and in order to meet these challenges, new solutions are proposed and tested.

One such field of mining is search engines. While search engines such as Google, Bing, Baidu, and more have become ubitiquous in society, at their core they do not really take into consideration how humans relate "ideas" and "concepts". As a result, what the search engine returns to the user is really not what is related to the query, but rather what is matched with the query's keywords. In this project, we use mathematics to provide a way to not only find relationships and but also

lack of relationships in our knowledgebase in order to know what is really related with a set of keywords.

To better work around the idea of human ideas and concepts, the idea of topology and keyword-sets from Lin and Chiang[10] is used. That approach, used in this project, is to turn to algebraic topology and use simplicial complexes as computational representations of human concepts, considered as sets of keywords. The complete simplicial complex of connected keywords and keyword-sets is a topological space of relations between keywords. With this topological space, the homology can be calculated from the simplicial complex to determine the relations and non-relations of human ideas, expanding the work from previous student project[1].

Homology can tell us about properties of that space. A simple example of a space can be the three keywords "wine", "cheese", and "press". Not only are all three words are separate concepts, but so are the two word combinations: "wine press", "cheese press", and "wine cheese". There are presses that press grapes for wine, there are presses that press cheese to make it, and wine and cheese often go together as a snack, so all of these keyword-sets will be represented in the space. But it is uncommon to think of the three word combination "wine cheese press" as there are no presses that serve as both a wine press and a cheese press, so that three word combination would be missing from the space. If the homology is calculated on that space, we find that our simplicial complex containing our human concepts will have a hole, created by the absence of "wine cheese press" in the space.

With the space of web concepts built from the Concept Based Semantic Search Engine[14] and similar search engines, which process documents and create simplicial complexes that contain important concepts, homology can then be used to determine topological properties about the web concepts in those processed documents. In the previous work[1], the interest was on the real space that was limited to determining the Betti numbers of the simplicial complex that tell us only about some holes. For this project, the interest is in the integer space, more specifically Abelian groups, so that homology can tell us a more complete topological description by showing its torsion groups, however the calculation described in this report only tells us the existence of the holes, not exactly what creates them.

Chapter 2

Background

Some important topics central to the calculation of the homology groups are covered here, but only covered in order to explain how homology is calculated in this project. Topics introduced are simplicial complexes, abelian groups, homology and chain complexes, and the smith normal form.

2.1 The Simplicial Complex

To represent keyword-sets, which we think of as human concepts, we represent them with simplicial complexes. To start, we define an n-simplex to be a set of independent vertices, with k-faces, the k-simplexes the subset of n-simplex's vertices.[11] Geometrically, a 0-simplex is a vertex, a 1-simplex is an open line segment that does not include its vertices, a 2-simplex is an open triangle that does not include its vertices or edges, a 3-simplex an open tetrahedron that does

not include its vertices, edges, or faces, and so and on so forth. The n-simplex and simplicial complex here is defined very similarly to Lin and Chiang[10].

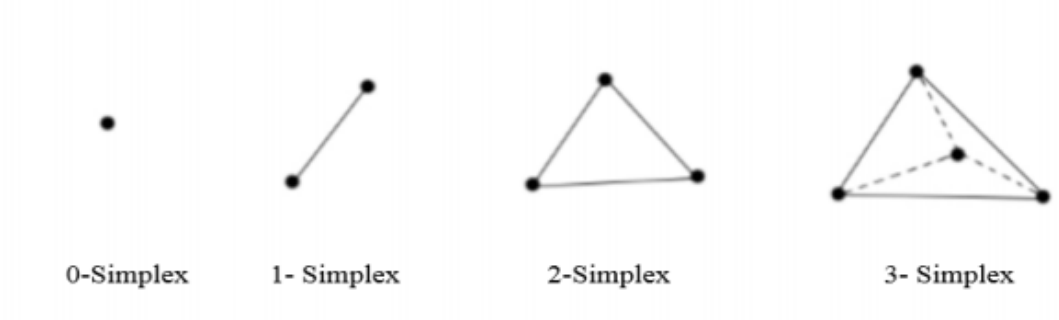
Definition 2.1.1. n-simplex: *An n-simplex is a set of independent vertices* $[v_0, ..., v_{n+1}]$. *A k-face of a n-simplex* $[v_0, ..., v_{n+1}]$ *is a k-simplex* $[v_{j_0}, ..., v_{j_{r+1}}]$ *whose vertices are a subset of* $\{v_0, ..., v_{n+1}\}$ *with cardinality k + 1.*

Restating the definition, we call any simplex with k + 1 vertices a k-simplex. Basically, a simplicial complex is a set of vertices and simplexes which are closed, meaning that for every simplex, its endpoint vertices are also in the simplicial complex.

Definition 2.1.2. simplicial complex: *A simplicial complex X is a finite set of simplexes (including sets of vertices) that must satisfy two properties: 1. Any set consisting of one vertex is a simplex. 2. The closed condition: Any face of a simplex from a complex is also in this complex.*

It follows that the vertices in a simplicial complex is the union of all of the vertices in its simplexes. We can visualize simplicial complexes as closed simplexes being connected by segments between it's vertices, as shown in Figure 2.1.

Figure 2.1: Some simplicial complexes[1]



We assign k+1 length keyword-sets, or tuples of keywords, to k-simplexes. "San" would be represented by a 0-simplex, "San Jose" would be represented by a 1-simplex, and so on. To show an more complicated example of keyword-set simplicial complex, we can split the keywords "San Jose State" into three keywords: "San", "Jose", and "State". These three keywords are represented by 0-simplexes. We know that "San Jose", "Jose State", and "San State" are represented by 1-simplexes, since they are two related keywords each.

If any these 1-simplexes are missing its endpoint vertices in a set of vertices and simplexes, then that set is not closed and it would not be a simplicial complex. Going further, if the 2-simplex that represents "San Jose State" is missing any of the closed 1-simplexes, then that 2-simplex is not closed. Finally, we can summarize that the simplicial complex for "San Jose State" would include the 0-simplexes representing "San", "Jose", "State", the 1-simplexes representing "San Jose", "Jose State", and "San State", and the 2-simplex "San Jose State".

In this project, we will only calculate for simplexes that are closed and ignore those that are not closed, because we're dealing solely with simplicial complexes, which are required to be closed. This condition is often satisfied as the concept

based search engines that we are working with abide by the Apriori principle, from association rule mining, which is a non-abstract, bottom-up restatement of the closed condition for keywords. The Apriori principle states that all subsets of processed sets, or in the case of this project, important keyword sets, have already been considered to be frequent.

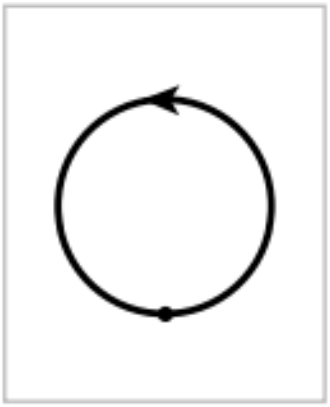
2.2 Homology

Homology is a functor that can describe the topology of a space finitely.[8] The interest of homology for web concepts is that it describes exactly the holes of the space, and this can be used to find holes or torsion in our simplicial complex of concepts. When homology finds that there are holes, it means that there are concepts missing from the complex, discovering possible additional concepts that may exist in the semantics, if we consider as the space that contains our human concepts.

An informal way of thinking about what homology groups are is looking at paths on a surface from a point to itself, and determining if those paths can be shrunk down to a single point. These smooth closed paths that start and end at a single point are called cycles, and they can be expanded to a larger cycle or shrunk down to a smaller cycle or even a single point, as long as the altered cycle remains on the surface. But a cycle cannot be shrunk down across empty space, therefore cycles that enclose holes and gaps cannot be shrunk down to a single point. Finally, two cycles are considered the same if they can be expanded or shrunk into one another without cutting or breaking them.

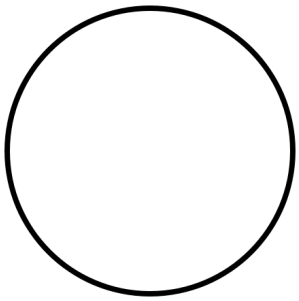
The homology group of a space is the group generated by cycles in the space. A cycle can be traversed upon. If a cycle can be shrunk down to a single point, then it is trivial because the traversal is length 0. On the other hand, if it is a cycle with a hole in it, then it can be traversed from a single point back to the very same point, shown in 2.2. This traversal can be iterated any number of times, and

Figure 2.2: Circle Cycle



this iteration can be described with a generator p that can be added together $p + p + ... + p$. This addition group is equivalent to $\mathbb{Z}$, the integer space. Traversal in the opposite direction is simply negation, so traversing it one way and then traversing it the opposite way would result in $p - p = 0$ If we have another nontrivial cycle in the same space, its traversal can also be iterated and we can describe that with a generator q that can be added together much like with p, and can be added together with p as well, resulting in $p + p + ... + p + q + q + ... + q$, resulting in the homology group $\mathbb{Z} \times \mathbb{Z}$. These generators can form basis-like structure by putting them together: $[p, q]$

Figure 2.3: The Hollow Circle



A simple example is the hollow circle in Figure 2.3. The homology group for

dimension 0, H_0, is $\mathbb{Z}$ because there is only one connected component, the circle and the dimension 0 cycle is a point that can be traversed many times. The homology group for dimension 1, H_1, is $\mathbb{Z}$ because there is a 1-dimensional hole that prevents the dimension 1 cycle, the circle itself, from being shrunk down to a point.

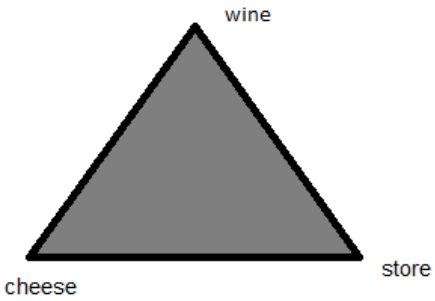
A more complex example is H_1 of the non-orientable real projective plane. While H_0 is $\mathbb{Z}$ because there is only one connected component and H_2 is 0 because no cycle goes around a 2-dimensional hole, H_1 is $\mathbb{Z}_2$. This is because traversing the cycle once around the 1-dimensional hole of the projective plane creates a cycle that cannot be shrunk down to a point, but traversing it twice does, hence $p + p = 0$ and resulting in a torsion group, equivalent to $\mathbb{Z}/2\mathbb{Z}$, also written as $\mathbb{Z}_2$

Another way of thinking of homology groups is looking at n-cycles that are bounded, or enclosed, by n+1-cycles. This is because the existence of the boundary cycle means that the hole enclosed by the boundary is filled in, and the cycle can be shrunk down to a point. Recalling the hollow circle in Figure 2.3, it can be thought of as a single vertex that cycles to itself. So there are no 0-dimensional holes because it is connected, but it has 1-dimensional hole because the hollowness means that that cycle is not bounded. Had the circle been solid, the cycle would have been a "filled" boundary and so would not have had a 1-dimensional hole.

Returning to our "wine" and "cheese" example, suppose an extended example happens in which "wine cheese" occurs because of the common pairing of the two

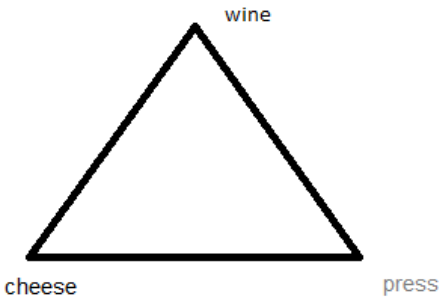
foods, and where “wine store” and “cheese store” occur because there are stores that sell wines and stores that sell cheeses. Because of the common pairing of wine and cheese, there are some stores that sell wine and cheese, so “wine cheese store” occurs. As such, we have a simplicial complex containing the concepts of the vertices (0-simplexes) “wine”, “cheese”, “store”, line segments (1-simplexes) “wine cheese”, “wine store”, “cheese store”, and triangle (2-simplex) “wine cheese store”, forming a closed 2-simplex, or in other words, a solid triangle, shown in Figure 2.4. There are no 0-dimensional holes, no 1-dimensional holes, and no 2-dimensional holes, because the cycles of “wine”, “cheese”, “store” and “wine cheese”, “wine store”, “cheese store” all have boundaries in a higher dimension. Now suppose that we have another example, one used previously: “wine cheese

Figure 2.4: Wine Cheese Store Simplicial Complex



press”. In this example, because there are no presses that can press both wine and cheese, so it does not occur in our simplicial complex of concepts. The simplicial complex contains line vertices “wine”, “cheese”, “press”, and line segments “wine cheese”, “wine press”, and “cheese press”, resulting in an open 2-simplex, or in other words, a hollow triangle as shown in Figure 2.5. Similar to the previous example, we have no 0-dimensional holes and no 1-dimensional holes. However, the cycle of “wine cheese”, “wine store”, “cheese store” has no boundary cycle in

Figure 2.5: Wine Cheese Store Simplicial Complex



the 2-dimension, showing that there is no filling to the triangle. Therefore, because there is no filling, the cycle cannot be shrunk and therefore there homology group for the 1-dimension would be $\mathbb{Z}$, showing the hollow triangle to have the same structure as the hollow circle. The idea of cycles and boundaries is

Figure 2.6: A Chain Complex[7]

$$0 \longrightarrow C_n(X) \xrightarrow{\partial_n} C_{n-1}(X) \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_2} C_1(X) \xrightarrow{\partial_1} C_0(X) \longrightarrow 0,$$

formalized through the chain complex. Shown in Figure 2.6, a chain complex of a simplicial complex X is a sequence of abelian groups $C_0, C_1, C_2, ..., C_n$, known as chain groups chained by boundary homomorphisms $\partial_n : C_n \to C_{n-1}$, and 0 is the trivial group C_n can be thought of as the set of open n-simplexes of X, each simplex denoting a generator to form a basis of generators. Putting it in this perspective likens the chain group space to linear algebra, as if the vectors of real space were instead points of integer space.

An example showing chain groups is the previous "wine cheese store" example, C_2 would by spanned by only the generator for the face "wine cheese store", C_1 would be spanned by the generator for the faces "wine cheese", "wine store", and "cheese

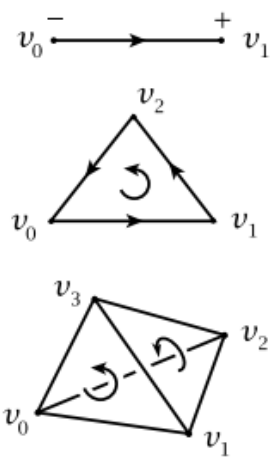
store", and C_0 would be spanned by the faces "wine", "cheese", and "store". If we consider specifically the simplicial homology groups, there is a formal definition.[8]

Definition 2.2.1. n-chains: *Elements of C_n are called n-chains and are written as finite formal sums $\Sigma_\alpha n_\alpha {e^n}_\alpha$ where $n_\alpha \in \mathbb{Z}$, ${e^n}_\alpha$ are the open n-simplexes in X, and α is an index that depends on the number of n-simplexes.*

These chain groups as linked by a boundary homeomorphism $\partial_k : C_n \to C_{n-1}$ as shown in Figure 2.2, in which each value from an n-simplex is deleted to make an oriented sum of (n-1)-simplexes. The sum is an (n-1)-chain

The boundary operator ∂_n for a vector $[x_0, x_1, ..., x_n]$ and $\hat{x}$ denotes the value removed from the vector and d is the length of the vector is defined as

Figure 2.7: Boundaries[8]



$$\partial[v_0, v_1] = [v_1] - [v_0]$$

$$\partial[v_0, v_1, v_2] = [v_1, v_2] - [v_0, v_2] + [v_0, v_1]$$

$$\partial[v_0, v_1, v_2, v_3] = [v_1, v_2, v_3] - [v_0, v_2, v_3] + [v_0, v_1, v_3] - [v_0, v_1, v_2]$$

Definition 2.2.2. Boundary homeomorphism:

$\partial[x_0, x_1, ..., x_n] := \sum_i (-1)^i [x_0, x_1, ..., \hat{x}_i, ..., x_n]$ *with the hat over x_i denoting a deletion from the sequence.[15]*

A few examples:

$\partial[A,B,C] = [B,C] - [A,C] + [A,B]$

$\partial[A,C] = [C] - [A]$

$\partial[A] = []$

Another notation can also be used:

$\partial(\Delta ABC) = \Delta BC - \Delta AC + \Delta AB$

$\partial(\Delta AC) = \Delta C - \Delta A$

$\partial(\Delta A) = 0$

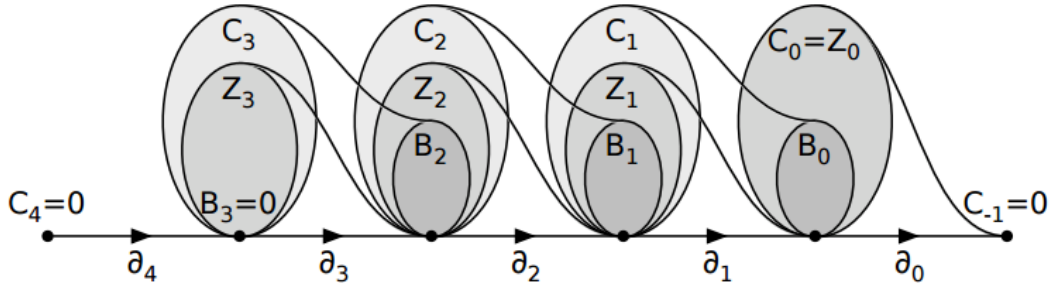
From the definition of ∂, we can determine that $\partial_{n+1} \circ \partial_n = 0$. The chain complex and its boundary operator can be thought of in linear transformation, despite the fact that the chain complexes are not real space but instead abelian groups, where ∂ is a linear transformation $C_n \to C_{n-1}$. A boundary matrix ∂_n can be created to show this transformation, if we assign each element in a basis vector to denote a respective elements in a chain complex and then use the definition ∂ to determine the matrix elements.

To compute the homology of X, we need to know two subgroups of C_n, Z_n and B_n. Z_n is the kth cycle group and B_n is the kth boundary group. In terms of the boundary operator ∂_n, they are defined as

$$Z_n = ker\partial_n \text{ and } B_n = im\partial_{n+1}$$

Though we are not dealing with real space, we can think of Z_n being the right null space of ∂_n and B_n being the row space of ∂_{n+1}. This implies that B_n is a normal subgroup of Z_n, which is a normal subgroup of C_n, which the definition of ∂ also implies. The relationship between the groups can be seen in 2.8

Figure 2.8: A Three Dimensional Chain Complex[15]



H_n can now be defined as a quotient group n-cycles mod n-boundaries, isomorphic to a direct sum of cyclic groups for some arbitrary numbers i, by the fundamental theorem of finitely generating Abelian groups

Definition 2.2.3. Homology Groups:

$$H_n := Z_n/B_n \cong \mathbb{Z}^{\beta_n} \oplus \bigoplus_i (\mathbb{Z}/d_i\mathbb{Z})$$

Definition 2.2.4. Fundamental Theorem of Finitely Generating Abelian Groups: *every finitely generated Abelian group G is isomorphic to the group direct sum of a finite number of groups, each of which is either cyclic of prime power order or isomorphic to* $\mathbb{Z}$ *[2].*

The rank of the torsion-free part of an Abelian group G, or the groups isomorphic to $\mathbb{Z}$, is the rank of G. This is also known as β_n. β_n is known as an integer value called Betti number for the kth dimension, while the other components in the direct sum are known as torsion subgroups. It is here that the previous work for human concepts on the web is extended, because homology groups of fields, in other words the real space $\mathbb{R}$, only have a Betti number component and no torsion component. By being in the integer space $\mathbb{Z}$, we can learn more about the simplicial complex, specifically what exactly is the homology group isomorphic to, but calculation of homology groups takes more work than calculating Betti

numbers. To calculate Betti numbers, only a simple and fast QR decomposition or Singular Value Decomposition of the boundary matrix is required, but to calculate homology groups the calculation of Smith Normal Form is needed.

2.3 Smith Normal Form

The Smith Normal Form is the primary method of calculating the homology groups of a simplicial complex, for reasons outlined here. It is similar to Singular Value Decomposition, but its matrices stay within the integer domain rather than be within the real of complex domain. The Smith Normal Form of an p * q matrix M is $M_i = U_i D_i V_i$ where U is an invertible integer p * p matrix, S is a diagonal p * q matrix, and V is an invertible integer q * q matrix. The entries of D are in principal ideal domain (PID), and D has diagonal entries $[b_1, b_2, ..., b_t, 0, ...0]$, with the length of the minimum of p and q and that for a diagonal entry b_i, b_{i+1} is divisible by b_i if i ¡ t.

We can see an example for the occurence of a torsion group in Appendix A for the real projective plane ($\mathbb{RP}^2$).

Morandi[12] shows that for any matrix M with entries in PID, M has a Smith Normal Form. To calculate Smith Normal Form of a matrix M, only elementary row and column operations may be used, such as swapping, multiplying the row or column by -1, and adding integer multiples to each other. Morandi also shows that if a Smith Normal Form exists for M, M is isomorphic to a direct sum of the

quotient of R-modules and each diagonal entry. This property of Smith Normal Form makes it ideal for calculating homology because of the fundamental theorem of finitely generating Abelian groups, as $\mathbb{Z}$ and its integer groups are obviously R-modules.

Figure 2.9: The Matrix D_{i+1}

$$D_{i+1} = \begin{pmatrix} b_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & b_2 & \cdots & 0 & 0 & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot & 0 & \cdots & 0 \\ 0 & 0 & \cdots & b_t & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix}$$

In Figure 2.4, we can see Matrix D_{i+1} when M is a boundary matrix ∂_i. From Dumas[6] and Zomorodian [15], we know that r is the cardinality of C_i minus $rank(\partial_i)$ and t is $rank(\partial_{i+1})$, or equivalently that t is the number of diagonal entries in ∂_{i+1}.

$$H_i \cong \mathbb{Z}^{r-t} \oplus (\mathbb{Z}/b_1\mathbb{Z}) \oplus ... \oplus (\mathbb{Z}/b_t\mathbb{Z})$$

When $r - t = 0$, $\mathbb{Z}^0$ is the identity element. For some integer i, if $b_i = 1$, then $(\mathbb{Z}/b_i\mathbb{Z})$ is trivial and can be omitted from the direct sum.

Chapter 3

Implementation

3.1 Overview

The project is implemented in python, primarily using numpy's array and matrix modules. Simplicial complexes are either read in from a file, or passed to a function in the code. For the maximal simplex with length n, n dimensions are calculated for homology. First n boundary matrices are calculated from looking at the simplexes and the Smith Normal Form is calculated for all boundary matrices. From the results of the Smith Normal Form matrices, the homology groups can be computed.

3.2 Source Data

The input data for the program is a set of simplexes, either read in from a file or passed input to a function in the code. This input set is checked to ensure that it is a valid simplicial complex by checking to see if all input simplexes are closed. If any non-closed simplex is found, then the program does not accept its input and terminates.

One of the sources for data used in this project is the Concept Based Semantic Search Engine(CBSE)[14], which creates a simplicial complex from a corpus. For this project, that is the source of the simplicial complex web concepts. The CBSE builds the simplicial complex by scanning each document for tokens and building arbitrary size n-word length tokens from the document. Stemmed tokens with stop words or tokens that are duplicates are removed, and then filtered with term frequency-inverse document frequency to determine its importance as a web concept. The n-tokens (and their subsets) that pass the tfidf threshold are accepted as n-simplices in the simplicial complex. This simplcial complex holds important web concepts in a relational database, which is then exported to a file that becomes the input for this project's program. The contents of the file is a list of simplexes that make up the CBSE's simplicial complex.

3.3 Generating Boundary Matrices

After the simplicial complex input is read and checked to see if it is closed, the boundary matrix ∂_k is calculated from its definition and for each vector the row is

added to a matrix. If the input is not a closed complex then the homology computation does not happen. Each chain group is stored in sorted order to ensure that the bases of the matrices are lined up correctly for boundary matrix calculations. Computation of homology starts after boundary matrices are generated for all of the dimensions

Informal Boundary Matrix Algorithm for ∂_k

- Iterate over all faces in C_k
- For a face, generate all copies of it, each with a single different vertex removed, sorted in the same way that the basis of C_{k-1} is in
- Look for the indexes of the newly generated faces in C_{k-1} and create a new row with alternating 1s and -1s for the corresponding indexes in the row, otherwise if not found, 0
- Add the row to an initially empty matrix, creating a m x n matrix where m is the number of faces in C_k and n is the number of faces in $C_k - 1$

3.4 Matrix Transformation

To calculate the Smith Normal Form, this project implemented python code for the Elimination Algorithm described in Dumas, combined with the MATLAB code for the Smith Normal Form. The running time of the algorithm is $(O)(n^2)$, as described in the documentation of the original MATLAB code. In general,

most Smith Normal Form algorithms have this running time starting with Kannan and Bachem as well as others,[9][4][5] but even Gaussian Elimination can work for matrices of $\mathbb{Z}_2$.[7][3] The Dumas optimization aims to eliminate calculating obvious values, using what it can quickly glance from the matrix. Obvious trivial 1 coefficients are removed so that less work has to be done by the actual Smith Normal Form algorithm. Once the Smith Normal Forms have been calculated, the torsion coefficients are found by slicing appropriately from the diagonals of the Smith Normal Form matrices, based on the next calculated dimensions' Smith Normal Form and printed out as a product representation, even though the program really means direct sum.

Informal MATLAB Smith Normal Form Algorithm

- The overview of this algorithm is to use elementary row operations and extended gcd to create a superdiagonal matrix, then eliminate non-diagonal superdiagonal entries to create a diagonal matrix.
- For all columns, zero out the column entries below the diagonal and for all rows, zero out the row entries to the right of the superdiagonal
- Then zero out the non-diagonal superdiagonal entries
- Matrix will be diagonal now, so make all negative entries positive
- Squish larger entries lower using extended gcd

Definition 3.4.1. Extended gcd: *For integers a and b, x and y are also calculated such that* $ax + by = gcd(a, b)$

Smith Normal Form Example:

First, eliminate all entries in matrix that are not part of the superdiagonal.

$$\begin{bmatrix} 2 & 12 & 8 & 20 \\ 38 & 6 & 16 & 40 \\ 22 & 4 & 10 & 14 \\ 50 & 18 & 26 & 34 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 12 & 8 & 20 \\ 0 & -222 & -136 & -340 \\ 22 & 4 & 10 & 14 \\ 50 & 18 & 26 & 34 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 12 & 8 & 20 \\ 0 & -222 & -136 & -340 \\ 0 & -128 & -78 & -206 \\ 50 & 18 & 26 & 34 \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} 2 & 4 & 0 & 20 \\ 0 & -86 & 36 & -340 \\ 0 & -50 & 22 & -206 \\ 50 & -8 & 42 & 34 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 4 & 0 & 20 \\ 0 & -86 & 36 & -340 \\ 0 & -50 & 22 & -206 \\ 0 & -108 & 42 & -466 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 4 & 0 & 0 \\ 0 & -86 & 36 & 90 \\ 0 & -50 & 22 & 44 \\ 0 & -108 & 42 & 74 \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} 2 & 4 & 0 & 0 \\ 0 & -2 & -12 & 102 \\ 0 & 0 & 46 & -358 \\ 0 & -108 & 42 & 74 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 4 & 0 & 0 \\ 0 & -2 & -12 & 102 \\ 0 & 0 & 46 & -358 \\ 0 & 0 & 690 & -5434 \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} 2 & 4 & 0 & 0 \\ 0 & -2 & 6 & 0 \\ 0 & 0 & 10 & -66 \\ 0 & 0 & 86 & -862 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 4 & 0 & 0 \\ 0 & -2 & 6 & 0 \\ 0 & 0 & 2 & -602 \\ 0 & 0 & 0 & -1472 \end{bmatrix} \rightarrow$$

The non superdiagonal entries have now been zeroed out. Now, the superdiagonal

entries will be zeroed out.

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & -2 & 6 & 0 \\ 0 & 0 & 2 & -602 \\ 0 & 0 & 0 & -1472 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & -602 \\ 0 & 0 & 0 & -1472 \end{bmatrix} \rightarrow$$

Turn diagonal entries positive.

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -1472 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1472 \end{bmatrix} \rightarrow$$

The high divisible values have already been pushed down, so the last step does not change the matrix.

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1472 \end{bmatrix} \rightarrow$$

Finally, we have reached a normal form for the matrix and can see that the entries are consistent with the properties of Smith Normal Farm.

3.5 Homology Calculation

Once, Smith Normal Form has been calculated for all of the boundary matrices, then the homology can be calculated from them. Recall our homology isomorphism and the matrix D_{i+1} for i-dimension that we will use to calculate our homology:

$$H_i \cong \mathbb{Z}^{r-t} \oplus (\mathbb{Z}/b_1\mathbb{Z}) \oplus ... \oplus (\mathbb{Z}/b_t\mathbb{Z})$$

Figure 3.1: The Matrix D_{i+1}

$$D_{i+1} = \begin{pmatrix} b_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & b_2 & \cdots & 0 & 0 & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot & 0 & \cdots & 0 \\ 0 & 0 & \cdots & b_t & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix}$$

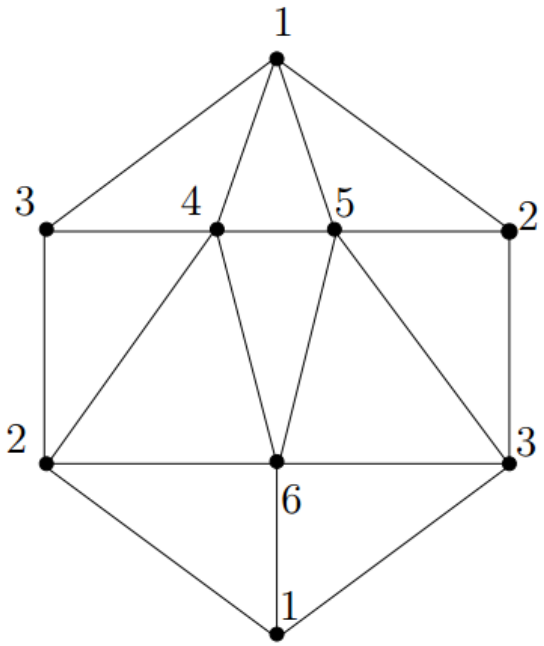
Because we have calculed Smith Normal Form already, the calculation boils down to just slicing the correct portion of the diagonal entries, and arithmetic for the Betti numbers.

Example: This example will show the homology calculation for the real projective plane, triangulated for a simplicial complex from Aanjaneya and Teillaud [13], seen in Figure 3.2.

To begin, the chain complexes and the corresponding matrices are generated:

C_0: {(1,), (2,), (3,), (4,), (5,), (6,)}

Figure 3.2: Real Projective Plane Triangulation



C_1: {(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 3), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6), (4, 5), (4, 6), (5, 6)}

C_2: {(1, 2, 5), (1, 2, 6), (1, 3, 4), (1, 3, 6), (1, 4, 5), (2, 3, 4), (2, 3, 5), (2, 4, 6), (3, 5, 6), (4, 5, 6)}

$$\partial_1 = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

$$\partial_2 = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{bmatrix}$$

We then find the Smith Normal Forms of the two matrices them through the algorithm:

$$D_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

At this point, we can see that the diagonal elements for D_1 are $[1, 1, 1, 1, 1, 0]$ and D_2 are $[1, 1, 1, 1, 1, 1, 1, 1, 1, 2]$, and $rank(D_1) = 5$ and $rank(D_2) = 10$. Finally, the homology can be calculated.

$$H_0 \cong \mathbb{Z}^{6-0-5} \cong \mathbb{Z}$$

$$H_1 \cong \mathbb{Z}^{15-5-10} \oplus (\mathbb{Z}/\mathbb{Z}) \oplus ... \oplus (\mathbb{Z}/\mathbb{Z}) \oplus (\mathbb{Z}/2\mathbb{Z}) \cong \mathbb{Z}_2$$

$$H_1 \cong \mathbb{Z}^{10-10} \cong 0$$

Chapter 4

Results

In this chapter specific examples were calculated and compared to the known homology groups of the objects, if they are known. The python code was run very simply without any special arguments or configurations, and is to just demonstrate the functionality of the code, rather than speed or special running options. In particular, more than a few of the examples have torsion groups to demonstrate that the homology group calculation calculates for abelian groups and not for fields or in other words, real space. Examples include the real projective plane and the Klein bottle. Other examples include very simple ones such as a closed 2-simplex and a torus to show that the code works as designed for the simple cases that even just Betti numbers can show.

The string output of the program is easily understood: the multiplication character 'x' refers to a direct sum while a number that follows the $\mathbb{Z}$ is meant to be read as a subscript. For example, the output string "Z x Z x Z4" is $\mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}_4$,

or also equivalently $\mathbb{Z}^2 \oplus \mathbb{Z}_4$. The output is stored as a list, so to fetch dimension k from the list is simply accessing the list at location k.

4.1 Simple Solid 2-Simplex and Hollow 2-Simplex

As an example, the homology groups are calculated for a simple, closed, solid 2-simplex, represented by the set of strings {"A B C", "A B, "B C", "A C", "A", "B", "C"}, and a simple, closed, hollow 2-simplex, represented by the set of strings {"A B, "B C", "A C", "A", "B", "C"}. The key difference between these two shapes is that the face "A B C" is in the solid 2-simplex while it is not in the hollow 2-simplex.

The homology groups are already known, the result for the solid 2-simplex should be

$$H_0 \cong \mathbb{Z};\ H_1 \cong 0;\ H_2 \cong 0$$

while the result for the hollow 2-simplex should be

$$H_0 \cong \mathbb{Z};\ H_1 \cong \mathbb{Z};$$

The homology groups result output, shown in Figure 4.1 and Figure 4.2 for our shapes is the expected output.

Figure 4.1: Simple Solid 2-Simplex Results

```
In [13]: print_group(sc.homology[0])
Out[13]: 'Z'

In [14]: print_group(sc.homology[1])
Out[14]: '0'

In [15]: print_group(sc.homology[2])
Out[15]: '0'
```

Figure 4.2: Simple Hollow 2-Simplex Results

```
In [8]: print_group(sc.homology[0])
Out[8]: 'Z'

In [9]: print_group(sc.homology[1])
Out[9]: 'Z'
```

4.2 Simple 3 Vertices

As an example, the homology groups are calculated for 3 unconnected vertices, represented by the strings "A", "B, "C". The homology groups are already known, the result should be

$$H_0 \cong \mathbb{Z}^3$$

Figure 4.3: Simple 3 Vertices Results

```
In [9]: print_group(sc3.homology[0])
Out[9]: 'Z x Z x Z'
```

The homology groups result output is the expected output.

4.3 2-Torus

As an example, the homology groups are calculated for a 2-torus, a minimal triangulation devised by Császár. The homology groups are already known, the

result should be

$$H_0 \cong \mathbb{Z};\ H_1 \cong \mathbb{Z}^2;\ H_2 \cong \mathbb{Z}$$

Figure 4.4: Simple 2-Torus Results

```
In [12]: print_group(torus2.homology[0])
Out[12]: 'Z'

In [13]: print_group(torus2.homology[1])
Out[13]: 'Z x Z'

In [14]: print_group(torus2.homology[2])
Out[14]: 'Z'
```

The homology groups result output is the expected output.

4.4 Real Projective Plane

As an example, the homology groups are calculated for the real projective plane, represented by a triangulation of 6 vertices and specific segments [13]. The homology groups are already known, the result should be

$$H_0 \cong \mathbb{Z};\ H_1 \cong \mathbb{Z}_2;\ H_2 \cong 0$$

Note that H_1 has torsion group $\mathbb{Z}_2$, a result that would not have been seen by only calculating the real space to get the Betti numbers.

The homology groups result output is the expected output.

Figure 4.5: Real Projective Plane Results

```
In [4]: print_group(rp2.homology[0])
Out[4]: 'Z'

In [5]: print_group(rp2.homology[1])
Out[5]: 'Z2'

In [6]: print_group(rp2.homology[2])
Out[6]: '0'

In [7]:
```

4.5 Klein Bottle

As an example, the homology groups are calculated for the klein bottle, represented by a triangulation of 7 vertices and specific segments. The homology groups are already known, the result should be

$$H_0 \cong \mathbb{Z};\ H_1 \cong \mathbb{Z} \oplus \mathbb{Z}_2;\ H_2 \cong 0$$

Note that H_1 has torsion group $\mathbb{Z}_2$, a result that would not have been seen by only calculating the real space to get the Betti numbers.

Figure 4.6: Klein Bottle Results

```
In [10]: print_group(klein.homology[0])
Out[10]: 'Z'

In [11]: print_group(klein.homology[1])
Out[11]: 'Z x Z2'

In [12]: print_group(klein.homology[2])
Out[12]: '0'
```

The homology groups result output is the expected output.

4.6 Frequent Keywords from Medical Data

As an example, the homology groups for frequent keywords from a database of medical data were calculated with 10750 simplexes of 273 unique keywords, and maximal keyword-set length 4. The maximum length of a simplex was 4 keywords, and the keywords in each simplex came in the order of their name.

The simplicial complex for the medical data was obtained from a Concept Based Semantic Search Engine. The search engine builds a simplicial complex by filtering stemmed n-tokens from a corpus of documents with a combination of stop word removal and term frequency-inverse document frequency filtering. The n-tokens that pass through the filtering are then considered important concepts and stored as simplexes as part of a simplicial complex, called the KnowledgeBase. The simplexes of the KnowledgeBase are precisely the input for this example. An observation that could be made on the input data is that the keywords for the data are always ordered in the same way depending on their alphabetical order, meaning that a simplex {A, B, C} will never occur as {B, C, A}, {C, B, A}, or other permutations of the order.

The Betti numbers are known to respectively be 111, 3284, 2072, 1285 for H_0, H_1, H_2, H_3, using the previous work[1]. Torsion groups may be in the data, but if they were not, we should expect a result of

$$H_0 \cong \mathbb{Z}^{111};\ H_1 \cong \mathbb{Z}^{3284};\ H_2 \cong \mathbb{Z}^{2072};\ H_3 \cong \mathbb{Z}^{1285};$$

Figure 4.7: Medical Data Results

```
In [9]: print_group(medicaldata.homology[0])
Out[9]: 'Z^111'

In [10]: print_group(medicaldata.homology[1])
Out[10]: 'Z^3284'

In [11]: print_group(medicaldata.homology[2])
Out[11]: 'Z^2072'

In [12]: print_group(medicaldata.homology[3])
Out[12]: 'Z^1285'
```

No torsion was found by the program, so we only have the Betti numbers, or the free part of the homology group showing up. The result output from the medical data is the same its previous calculation for Betti numbers.

Chapter 5

Future Work and Conclusion

This project has implemented computation of homology for human concepts found on the web, in reasonable running time with standard Smith Normal Form algorithms. The underlying structure of Smith Normal Form allows simple but essential calculation finding isomorphic groups to the homology groups. By finding torsion subgroups in the homology groups, topological holes that would have gone undetected with simply real space (field) homology analysis to find the Betti numbers are now visibly existing, shown in examples. However, the current work does not deal much with generators nor bases, which would greatly help to recognize exactly what keyword-sets or simplexes exactly consistute the topological holes. This work does not state anything on what causes the homological structure to occur, but shows whether or not they exist and how many of them exist.

Appendices

Appendix A

Example for Torsion in $\mathbb{RP}^2$

By paying attention to the formulation, we can interpret coordinate transformations and linear maps by matrix multiplications and sending row vectors to row vectors in other bases. Given the geometric bases of our chain groups:

C_2 natural basis:

$\Delta_{1,2,5}, \Delta_{1,2,6}, \Delta_{1,3,4}, \Delta_{1,3,6}, \Delta_{1,4,5}, \Delta_{2,3,4}, \Delta_{2,3,5}, \Delta_{2,4,6}, \Delta_{3,5,6}, \Delta_{4,5,6}$

C_1 natural basis:

$\Delta_{1,2}\ \Delta_{1,3}\ \Delta_{1,4}\ \Delta_{1,5}\ \Delta_{1,6}\ \Delta_{2,3}\ \Delta_{2,4}\ \Delta_{2,5}\ \Delta_{2,6}\ \Delta_{3,4}\ \Delta_{3,5}\ \Delta_{3,6}\ \Delta_{4,5}\ \Delta_{4,6}\ \Delta_{5,6}$

C_0 natural basis:

$\Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5, \Delta_6$

For a single vector with first element $\Delta_{1,2,5}$, second element $\Delta_{1,2,6}$, third element $\Delta_{1,3,4}$, they are called the natural basis. Each geometric vector corresponds to one of the row vectors with respect to natural basis.

Matrix of C_2: C_2 is the set of row vectors that with respect to natural basis.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

U_2 is the matrix whose row vectors form a new basis called normal basis.

$$U_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 5 & 15 & 37 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 0 \\ 1 & 2 & 4 & 6 & 7 & 6 & 2 & -6 & -19 & 1 \\ 0 & 0 & 0 & -1 & -3 & -5 & -5 & 0 & 14 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & -1 & -3 & -6 & -10 & -15 & -21 & -28 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 8 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 9 & 14 & 14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -7 & -28 & 0 \end{bmatrix}$$

$C_2 * U_2$: Set of row vectors is the new basis of C2 in terms of geometric basis (natural basis)

$$C_2 * U_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 5 & 15 & 37 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 0 \\ 1 & 2 & 4 & 6 & 7 & 6 & 2 & -6 & -19 & 1 \\ 0 & 0 & 0 & -1 & -3 & -5 & -5 & 0 & 14 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & -1 & -3 & -6 & -10 & -15 & -21 & -28 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 8 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 9 & 14 & 14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -7 & -28 & 0 \end{bmatrix}$$

D_2 is the linear map ∂_2 that is expressed in terms of normal basis of C_2 and an unknown basis of C_1, the unknown base is the row vector of the Inverse of V_2.

$$D_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We can see that the rank of $D_2 = 10$

$C_2 * U_2 * D_2$: The row vectors in this matrix($C_2 * U_2 * D_2$) given below is the Image of the map (expressed by multiplying D_2), which is expressed in terms of the normal basis(called N_2) of C_2 and the unknown (new) normal basis of C_1 (this unknown basis is the row vectors of Inverse(V_2)

$$C_2 * U_2 * D_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 5 & 15 & 37 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 4 & 6 & 7 & 6 & 2 & -6 & -19 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -3 & -5 & -5 & 0 & 14 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -3 & -6 & -10 & -15 & -21 & -28 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 9 & 14 & 14 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -7 & -28 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

V_2 is the set row vectors of Inverse(V_2) is the unknown basis of chain C_1, in which ∂_2 is reduced

to D_2

$$V_2 = \begin{bmatrix}
1 & -1 & 2 & 0 & -2 & 0 & -1 & 0 & 2 & 0 & -1 & 0 & 1 & 0 & 0 \\
-8 & 1 & -4 & -1 & 12 & 2 & 3 & 2 & -15 & 1 & 1 & 1 & -1 & 1 & 1 \\
28 & 0 & 3 & 4 & -35 & -1 & -14 & -6 & 49 & 0 & 2 & -3 & -5 & -6 & -5 \\
-56 & 0 & -1 & -6 & 63 & 0 & 28 & 7 & -91 & 0 & -3 & 3 & 12 & 15 & 10 \\
70 & 0 & 0 & 4 & -74 & 0 & -31 & -4 & 105 & 0 & 1 & -1 & -11 & -20 & -10 \\
-56 & 0 & 0 & -1 & 57 & 0 & 20 & 1 & -77 & 0 & 0 & 0 & 5 & 15 & 5 \\
28 & 0 & 0 & 0 & -28 & 0 & -7 & 0 & 35 & 0 & 0 & 0 & -1 & -6 & -1 \\
-8 & 0 & 0 & 0 & 8 & 0 & 1 & 0 & -9 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0
\end{bmatrix}$$

C_1 natural basis restated:

$\Delta_{1,2}$ $\Delta_{1,3}$ $\Delta_{1,4}$ $\Delta_{1,5}$ $\Delta_{1,6}$ $\Delta_{2,3}$ $\Delta_{2,4}$ $\Delta_{2,5}$ $\Delta_{2,6}$ $\Delta_{3,4}$ $\Delta_{3,5}$ $\Delta_{3,6}$ $\Delta_{4,5}$ $\Delta_{4,6}$ $\Delta_{5,6}$

$C_2 * U_2 * D_2 * V_2$: The row vectors in terms of unknown new basis is changed (by multiplying

V_2)into the row vectors in terms of natural basis at the chain C_1

$$C_2U_2D_2V_2 = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{bmatrix}$$

Which is equal to ∂_2:

$$\partial_2 = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{bmatrix}$$

U_1 (the matrix of changing natural basis of C_1 into Normal basis N_1) is the row vectors are the

new basis that are expressed in terms of geometrical basis.

$$U_1 = \begin{bmatrix}
1 & 2 & 2 & 2 & 2 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & -2 & -3 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & -1 & 1 & 0 \\
0 & 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & -1 & -1 & -1 & 0 & -1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
-1 & -2 & -3 & -4 & -5 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & -1 & 1 & 0 \\
-1 & -2 & -2 & -1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 1 & -1 \\
-1 & -2 & -2 & -2 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & -2 & -3 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & -1 & -1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & -1 & -1 & -1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 3 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 1 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$W = C_2U_2D_2V_2U_1$(the image of the linear map D_2) in terms of Normal Basis N_1(which is

defined by U_1)

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$V_2 * U_1$ The row vector of Inverse(V_2) is the unknown basis (N_2) of C_1that is transformed again by V_2 into natural basis by U_1 again into a New matrix (expressed below) whose row vectors is a

the normal basis of C_1

$$V_2 * U_1 = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 2 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 8 & -10 & -10 & -13 & 3 & 2 & 3 & 0 & 2 & -1 \\
0 & 0 & 0 & 0 & 0 & -28 & 29 & 42 & 37 & -1 & -12 & -14 & 11 & -11 & 5 \\
0 & 0 & 0 & 0 & 0 & 56 & -56 & -84 & -64 & 0 & 27 & 28 & -27 & 25 & -10 \\
0 & 0 & 0 & 0 & 0 & -70 & 70 & 101 & 74 & 0 & -31 & -31 & 31 & -30 & 10 \\
0 & 0 & 0 & 0 & 0 & 56 & -56 & -76 & -57 & 0 & 20 & 20 & -20 & 20 & -5 \\
0 & 0 & 0 & 0 & 0 & -28 & 28 & 35 & 28 & 0 & -7 & -7 & 7 & -7 & 1 \\
0 & 0 & 0 & 0 & 0 & 8 & -8 & -9 & -8 & 0 & 1 & 1 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 2 & 2 & 2 & 2 & 0 & 0 & 2 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & -1 & -2 & -3 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
1 & 2 & 3 & 4 & 5 & 0 & 1 & 0 & 0 & -1 & -1 & -1 & 1 & -1 & 0 \\
1 & 2 & 3 & 5 & 8 & 0 & 0 & 2 & 1 & 1 & 0 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 1 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1
\end{bmatrix}$$

$$d_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Rank of $D_1 = 5$

$W * D_1 = C_2 * U_2 * d_2 * V_2 * U_1 * D_1\ W$, which is the image of (the mapping by multiplying

D_2)in terms of new basis, is mapped into zero vectors, that is, W is lying in $kernel(\partial_1)$.

$$C_2U_2D_2V_2U_1D_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

From the New matrix (expressed below) whose row vectors is the normal basis N_1 of C_1

$$C_2U_2D_2V_2U_1D_1 = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 2 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 8 & -10 & -10 & -13 & 3 & 2 & 3 & 0 & 2 & -1 \\
0 & 0 & 0 & 0 & 0 & -28 & 29 & 42 & 37 & -1 & -12 & -14 & 11 & -11 & 5 \\
0 & 0 & 0 & 0 & 0 & 56 & -56 & -84 & -64 & 0 & 27 & 28 & -27 & 25 & -10 \\
0 & 0 & 0 & 0 & 0 & -70 & 70 & 101 & 74 & 0 & -31 & -31 & 31 & -30 & 10 \\
0 & 0 & 0 & 0 & 0 & 56 & -56 & -76 & -57 & 0 & 20 & 20 & -20 & 20 & -5 \\
0 & 0 & 0 & 0 & 0 & -28 & 28 & 35 & 28 & 0 & -7 & -7 & 7 & -7 & 1 \\
0 & 0 & 0 & 0 & 0 & 8 & -8 & -9 & -8 & 0 & 1 & 1 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 2 & 2 & 2 & 2 & 0 & 0 & 2 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & -1 & -2 & -3 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
1 & 2 & 3 & 4 & 5 & 0 & 1 & 0 & 0 & -1 & -1 & -1 & 1 & -1 & 0 \\
1 & 2 & 3 & 5 & 8 & 0 & 0 & 2 & 1 & 1 & 0 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 1 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1
\end{bmatrix}$$

Based on the normal basis N_1, a set of generators for $image(\partial_2)$ and a base of $kernel(\partial_1)$ is needed to compute the homology $kernel(\partial_1)/image(\partial_2)$.

To find the $kernel(\partial_1)$, we let A be defined as W with its first 5 columns dropped.

$$A = \begin{bmatrix} -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

We can write the Smith Normal Form of A as $S = U * A * V$

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Where

$$U = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 & 1 & -1 & 1 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \end{bmatrix}$$

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 1 & 1 & -3 & -3 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 1 & 1 \\ 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

With this definition, we will try to find a transformation of W that is the $kernel(\partial_1)$, or more

specifically a transformation of W that is the following matrix:

$$S_W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

To get this matrix, recall few conclusions from our previous results: $rank(image(\partial_2)) = 10$

$dim(C_2) = 15$

$rank(image(\partial_1)) = 5$

And with the rank-nullity theorem:

$nullity(C_1) = rank(kernel(\partial_1)) = dim(C_2)$–$rank(image(\partial_1)) = 15 - 5 = 10$

Recall that $W = C_2U_2D_2V_2U_1$(the image of the linear map D_2) in terms of Normal Basis

N_1(which is defined by U_1)

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Let

$$U_W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 & 1 & -1 & 1 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \end{bmatrix}$$

$$V_W = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 1 & -3 & -3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

We know that $U_W * W * V_W = U_W * C_2 * U_2 * D_2 * V_2 * U_1 * V_W$, so we have a Smith Normal

Form transformation now.

$$S_W = U_W * W * V_W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Consider Z_W, that substitutes the single 2 value in S_W with 1:

$$Z_W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Given that $S_W * Inverse(V_W) * D_1$

$= U_W * C_2 * U_2 * D_2 * V_2 * U_1 * V_W * Inverse(V_W) * D_1$

$= U_W * (C_2 * U_2 * D_2 * V_2 * U_1 * D_1)$

$= U_W * 0 = 0$ where 0 is a 0-matrix

and $Z_W * Inverse(V_W) * D_1$

where

$$Inverse(V_W) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

then

$$Z_W * Inverse(V_W) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$Z_W * Inverse(V_W) * D_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Since $Z_W * Inverse(V_W) * D_1 = 0$, the row vectors of Z_W, with respect to the appropriate basis, are in $kernel(\partial_1)$. In other words Z_W is the basis of $kernel(\partial_1)$, and S_W whose row vectors is a subspace of Z_W. If we consider Z_W/S_W, it is easy to see Z_2.

Bibliography

[1] Harleen Kaur Ahuja. Clustering web concepts using algebraic topology. *SJSU Master's Projects. Paper 448*, 2015.

[2] Margherita Barile and Eric W Weisstein. Kronecker decomposition theorem. *http://mathworld.wolfram.com/KroneckerDecompositionTheorem.html*, 2015.

[3] Otto Bretscher. *Linear Algebra with Applications.* Prentice Hall, 2009.

[4] Jennifer Seberry C. Koukouvinos, M. Mitrouli. Numerical algorithms for the computation of the smith normal form of integral matrices. *Congressus Numerantium 133*, pages 127–162, 1998.

[5] B. R. Donald and D. R. Cheng. On the complexity of computing the homology type of a triangulation. *Proc. 32nd Ann. IEEE Symp. Foundations Comput. Sci.*, page 650–661, 1991.

[6] J.-G. Dumas, F. Heckenbach, B. D. Saunders, and V. Welker. Computing simplicial homology based on efficient smith normal form algorithms. *Algebra, Geometry, and Software Systems*, pages 177—-206, 2003.

[7] Jeff Erickson. Cs 598: Computational topology (fall 2009) notes. *http://jeffe.cs.illinois.edu/teaching/comptop/2009/notes/*, 2009.

[8] Allen Hatcher. *Algebraic Topology.* Cambridge University Press, 2002.

[9] R. Kannan and A. Bachem. Polynomial algorithms for computing the smith and hermite normal forms of an integer matrix. *SIAM J. Comput*, page 8(4):499–507, 1979.

[10] T. Y. Lin and I-Jen Chiang. Granulate and conquer: Clustering web pages semantically using combinatorial topology. *Taiwanese Association for Artificial Intelligence Conference*, 2005.

[11] T. Y. Lin and J. Hsu. Knowledge based search engine: Granular computing on the web. *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2008.

[12] P. J Morandi. The smith normal form of a matrix. 2005.

[13] Monique Teillaud Mridul Aanjaneya. Triangulating the real projective plane. *Unité de recherche INRIA Sophia Antipolis Projets Geometrica*, 2007.

[14] Pradeep Roy. Concept based semantic search engine. *SJSU Master's Projects. Paper 351*, 2014.

[15] Afra Zomorodian. *Topology for Computing.* Cambridge University Press, 2005.