# Simplicial Homology

Yifan Ruan
Brown University

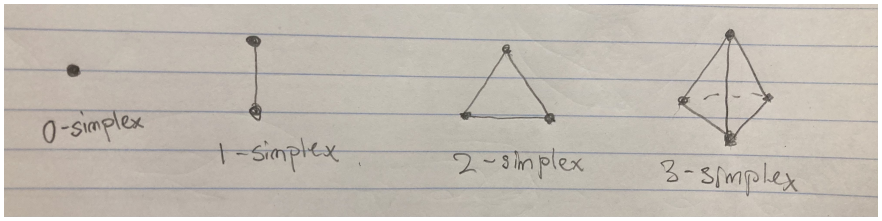
December 2021

## 1 Introduction

In this paper, we will study the concept of homology groups for simplicial complexes. Like the fundamental group of a topological space, the homology of a topological space is homotopy invariant which means that two topological spaces that are homotopy-equivalent will share the same homology. The intuitive idea behind homology is to count the number of holes of each dimension in the space. In order to more easily calculate the homology of a space, we will represent a topological space of interest by a *simplicial complex*, a homotopy-equivalent space consisting of *simplexes*, the most basic geometric shape of each dimension. With triangulation, most of our familiar topological spaces can be realized as simplicial complexes. I will begin by defining simplexes and providing some familiar examples.

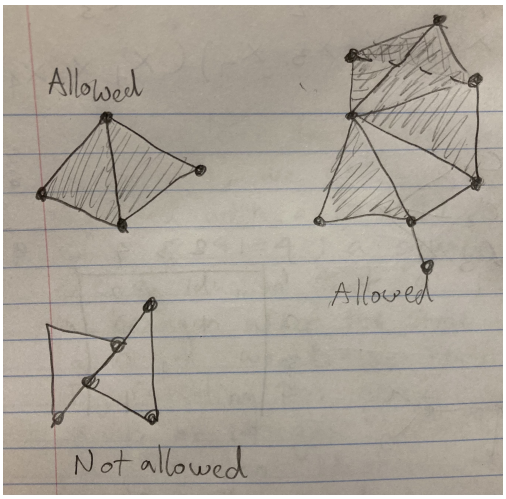
## 2 Simplexes and Simplicial Complexes

The easiest way for me to understand simplexes was through a few examples. The first 4 simplexes can be visualized in 3-dimensional space: a point, a line segment, a triangle (including its interior), and a (solid) tetrahedron respectively:



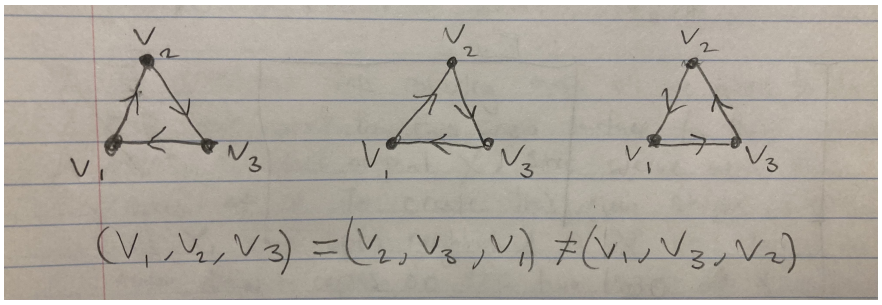

One can somewhat formalize this idea using the notion of cones. A $k$-simplex is simply a cone constructed over a $k-1$-simplex. Indeed, a triangle is a cone over a line segment, and a tetrahedron is a cone over a triangle. Notice also that the *faces* of a $k$-simplex consist of lower dimension simplexes: a line segment just has two points as its faces, a triangle has three line segments and three points as its faces, and a tetrahedron has four triangles, six line segments, and four points as its faces.

A simplicial complex, then, is a collection of simplexes that are connected like Lego pieces, where two simplexes can only be connected along a face. To be more rigorous, the nonempty intersection of any two simplexes, $\sigma_1, \sigma_2$ in a simplicial complex must be a face of both $\sigma_1$ and $\sigma_2$:




## 3 Orientations

An *orientation* of a simplicial complex is defined as an ordering of all of the 0-simplexes (points/vertices) that are contained in the simplicial complex. We say that two orientations are equivalent if they differ by an even permutation which means that if we can go from one orientation to another by swapping pairs of vertices an even number of times, they are equivalent. This gives us exactly two equivalence classes of orientations for any simplicial complex.




We say that a simplicial complex is orientable if we can assign orientations to each face, where orientation of intersections between simplexes have opposite orientations. For an orientable simplicial comples with a chosen orientation, we say it is oriented.

## 4 The abelian group of k-chains

Given an oriented simplicial complex, $S$, let $S_k$ be the set of all $k$-simplexes contained in $S$. We assign a positive orientation to each $k$ simplex by using the ordering of its vertices as they appear in the orientation we have for $S$. The same $k$ simplex with opposite orientation is considered the inverse element of the original $k$ simplex. For example,

$$(v_0, v_1) = -(v_1, v_0).$$

We then define a *simplicial k-chain* as

$$\sum_{i=1}^{N} c_i \sigma_i.$$

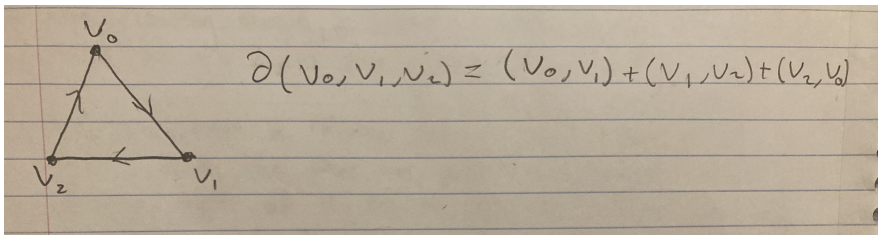
where $c_i \in \mathbb{Z}, \sigma_i \in S_k$. We can also take $c_i \in \mathbb{R}$ Then we have constructed an abelian group that is generated by the set of $k$-simplexes in $S$, where the addition operation between two simplicial $k$-chains just adds the coefficients for each $\sigma_i$. We denote the group of $k$-chains as $C_k$, which is also called the $k$th chain group of $S$.

## 5 Boundaries and Cycles

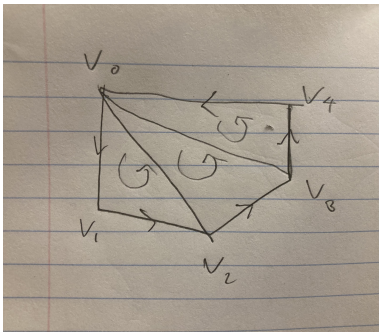
Given a $k$-simplex,$\sigma$ with an orientation, $(v_0, v_1, ..., v_n)$, we may *induce* an orientation for any $k-1$-simplex that is contained in $\sigma$ by noticing first that such a $k-1$ simplex includes every vertex from $\sigma$ except one, $v_i$. The *induced orientation* of the $k-1$ simplex is just $(v_0, ..., v_{i-1}, v_{i+1}, ..., v_n)$ if $i$ is even. If $i$ is odd, we take the opposite orientation.

Using this concept of induced orientation, we can formalize the notion of taking the boundary of a simplex. We define the *boundary* of an oriented $k$-simplex, $\sigma$, to be $\partial_k(\sigma) \in C_{k-1}$ the $(k-1)$-chain determined by the sum of its $k-1$-dimensional faces, *each taken with their induced orientation*:

$$\partial(v_0, ..., v_k) = \sum_{i=0}^{k} (-1)^i (v_0, ..., v_{i-1}, v_{i+1}, ..., v_k) \in C_{k-1}$$




Additionally, we say that the boundary of the sum of several oriented $k$-simplexes is the sum of the boundaries of each $k$-simplex.




In the above figure, we have that

$$\partial(v_0v_1v_2 + v_0v_2v_3 + v_0v_3v_4) = \partial(v_0v_1v_2) + \partial(v_0v_2v_3) + \partial(v_0v_3v_4)$$

$$= (v_0v_1 + v_1v_2 + v_2v_0) + (v_0v_2 + v_2v_3 + v_3v_0) + (v_0v_3 + v_3v_4 + v_4v_0)$$

$$= v_0v_1 + v_1v_2 + v_2v_3 + v_3v_4 + v_4v_0$$

And so we have created a map, $\partial_k : C_k \to C_{k-1}$.

An element of $C_k$, $c$, is called a $k$-cycle if $\partial_k(c) = 0$. Clearly, the sum of two cycles is still a cycle, since the boundary operator respects addition. Thus, the set of $k$ cycles in $C_k$ is a subgroup

which we will denote $Z_k$. Since the sum of the boundaries of two chains is equal to the boundary of the sum of the two chains, we also have that the set of boundaries in $C_k$ is a subgroup which we will denote as $B_k$.

One can already see that the $n$-sphere, $S^n$, is homologous to the boundary of the $(n+1)$-simplex. For instance, $S^2$ is homologous to the boundary of the 3-simplex, a tetrahedron.

## 6 Using matrices to represent boundary maps

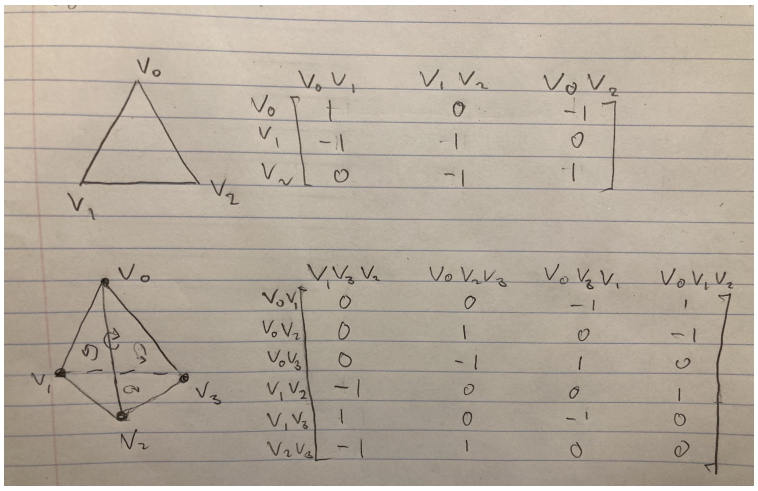
Notice that the boundary of any $k$-simplex consists of $k+1$ $(k-1)$-simplexes. Each edge of each $(k-1)$ simplex is shared by one other $(k-1)$ simplex. Thus, when we take $\partial_{k-1}(\partial_k(\sigma))$, we get exactly two copies of each $k-1$ simplex. It is not hard to verify that these two copies have opposite signs, meaning that taking $\partial^2$ over any $k$-chain always results in 0. In other words, the boundary of any element of $C_k$ is a $(k-1)$-cycle and so $B_k \in Z_k$. But is every cycle also a boundary? The answer is it depends on the dimension we are looking at and the simplicial complex in question. We define the $k$-homology group as the quotient group

$$H_k(S) = Z_k(S)/B_k(S)$$

The dimension of the $k$th homology group can intuitively be understood to be the number of $k$-dimensional "holes" in the simplicial complex. This is because we can take a closed simple curve around such a hole without separating the rest of the complex, which means we have a cycle that isn't a boundary.

A special case: the dimension of the 0th homology group represents the number of path-connected components in the given simplicial complex. This is because every single element of $C_0$ is in the kernel of $\partial_0$. However, for any two vertices, $a, b$, in the simplicial complex, $a-b$ can be a boundary only if we can connect them with a path using 1-simplexes. Thus, if $a$ and $b$ are in the same connected component, we have that $a$ and $b$ belong to the same element of the quotient space, $Z_0/B_0$: $a-b \in B_0$, and $a-(a-b)=b$. Therefore, the number of path-connected components in the simplicial complex corresponds to the number of linearly independent elements in the 0th homology group.

This is where the magic of our construction comes into play. The reader may have already noticed that each $C_k$ can be seen as a vector space with basis vectors corresponding to oriented elements of $S_k$ if we took the coefficients for the basis in $\mathbb{R}$. In case the coefficients were from $\mathbb{Z}$, we would have a free $\mathbb{Z}$-module. Indeed, we can add any two $k$-chains and we can multiply them by a scalar. Then the boundary map from $C_k$ to $C_{k-1}$ can be expressed as a matrix since it is essentially a linear transformation. As an example, I demonstrate in the attached image the matrix corresponding to $\partial_1$ for a triangle and $\partial_2$ for a tetrahedron:



Using matrix notation, then, the subspace of cycles in $C_k$ is simply the kernel of $\partial_k$, and the subspace of boundaries in $C_k$ can be expressed as the image of $\partial_{k+1}$. Since we know that the subspace of boundaries is contained in the subspace of cycles, we can just look at the rank of these two subspaces and subtract them to find the rank of the quotient group. Using triangulation, we have managed to provide a discrete way of calculating the homology group of an orientable surface.

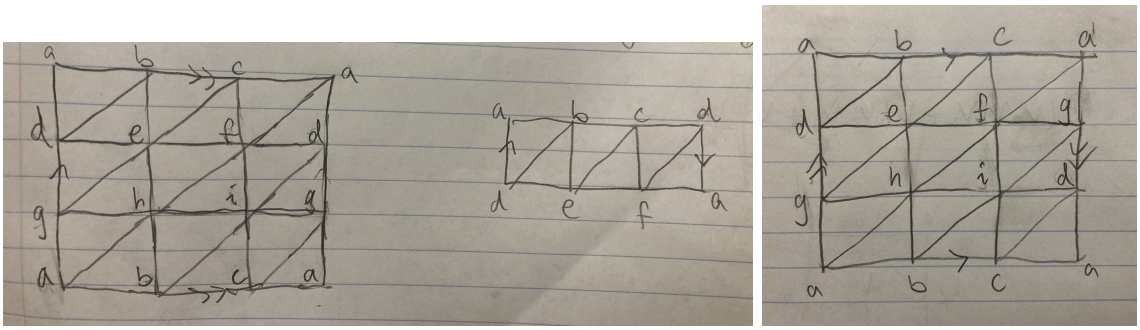
## 7 Using python to calculate homology groups

Using the principles outlined above, we can calculate the rank of a homology group for an orientable topological space, but it requires finding the boundary matrices of a simplicial complex representing the topological space. This can be quite tedious. For instance, the triangulation of a torus has 18 2-simplexes and 27 1-simplexes, which would mean that we need to manually fill in a 18*27 sized matrix to find the $\partial_2$ matrix.

I decided that I would do this using a computer program. This then creates the problem of expressing a simplicial complex in code. I decided that I could represent each vertex with a character, so each simplex would be a string of characters. Then the alphabetical ordering of all of the characters provides a canonical orientation for the overall simplicial complex. The problem is, I would need to input every single simplex in the simplicial complex in order to calculate boundary matrices. This amounts to 4 triangles, 6 edges, and 4 points for just a sphere.

I managed to solve this problem in a python program for calculating the dimensions of the homology groups for any given simplicial complex. The user just needs to input a list of top level simplexes (expressed in string form) in the simplicial complex like in the following image, and the program will calculate the rest of the simplexes along with the boundary matrices and the ranks of homology groups automatically:

```
twosphere = ["abc", "abd", "acd", "dcb"]
threesphere = ["bcde", "acde", "abde", "abce", "abcd"]
torus = ["abd", "dbe", "ebc", "efc", "cfa", "fad", "gde", "egh", "hfe", "hfi", "ifd", "idg", "agh", "ahb", "bhi", "bic", "cig", "cag"]
mobius = ["abd", "dbe", "bce", "ecf", "cdf", "fda"]
klein = ["abd", "ebd", "ebc", "efc", "cfa", "afg", "gde", "ghe", "hef", "hif", "fig", "dig", "hag", "bah", "hib", "bic", "cid", "cad"]
```



(From left to right, torus, mobius strip, klein bottle)
I obtained these simplicial complex representations for the more complex structures by using a triangulation of the square as seen in the attached images.

The first step of my program is to construct the list of all simplexes of each dimension for the inputted simplicial complex. To do this, I start by reordering each of the inputted simplexes alphabetically so that we have a standardized orientation for each simplex in the complex. Then I sort all the given simplexes into bins containing only simplexes of a certain dimension. Starting from the highest dimension bins, I take all the simplexes of dimension $k$, and add the $k-1$-simplexes that are its faces to the $k-1$ bin below. After I loop through all of the bins, I will have filled up the $k$th bin with a complete list of $k$-simplexes in the given simplicial complex. Here is an example output of my first step with the 2-sphere input as seen above $twosphere = [$"$abc$","$abd$","$acd$","$dcb$"]:

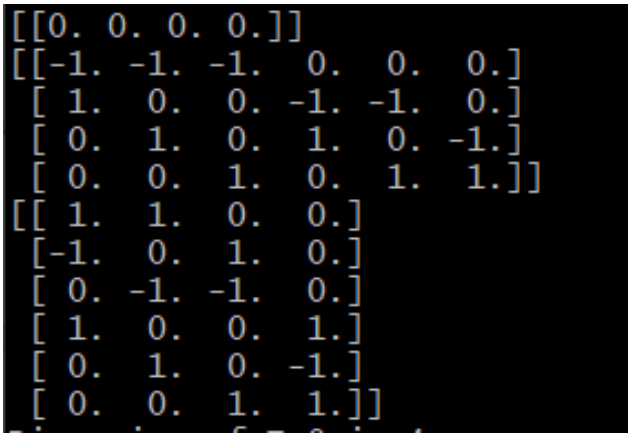
```
['a', 'b', 'c', 'd']
['ab', 'ac', 'ad', 'bc', 'bd', 'cd']
['abc', 'abd', 'acd', 'bcd']
```

As you can see, the first bin has all of the 0-simplexes, and the last bin has the 2-simplexes that I passed in.

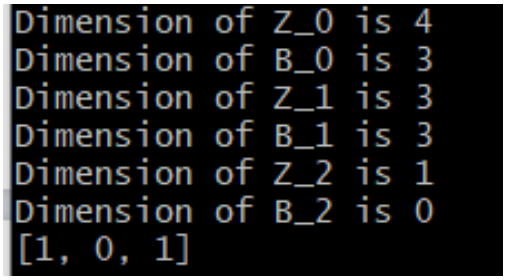
Now we essentially have the list of basis vectors for each $C_k$. The next step is to calculate the boundary maps as a matrices. I will spare you the detail of how I do this in python. Essentially, to calculate $\partial_k$, I use the formula from before :

$$\partial(v_0, ..., v_k) = \sum_{i=0}^{k} (-1)^i (v_0, ..., v_{i-1}, v_{i+1}, ..., v_k).$$

Since each basis vector is alphabetically ordered, for each column, I just need to fill in either 1 or -1 for each $(k-1)$-simplex that is contained in the respective $k$-simplex depending on which place the missing vertex is in the string representation. Here are the boundary matrices for the 2-sphere:



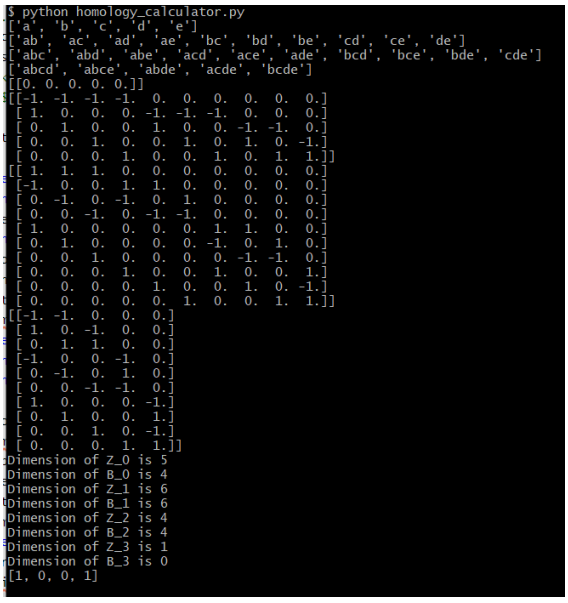

From here, I just use python packages to calculate the rank and dimension of null-space of each boundary matrix, and subtract to find the dimensions of the homology groups. The final output for the two sphere:

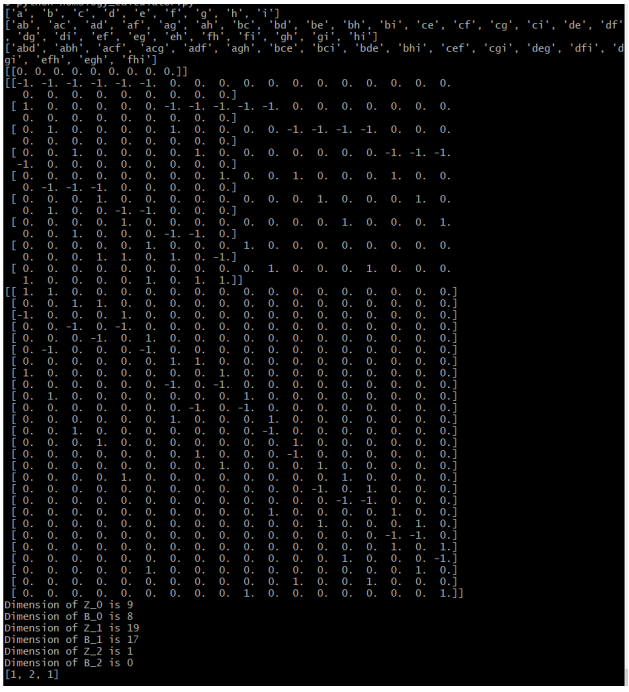



My program calculated that the 0th and 2nd homology groups of the 2-sphere have dimension 1, and the 1st homology group of the 2-sphere has dimension 0, as expected.

## 8 More results

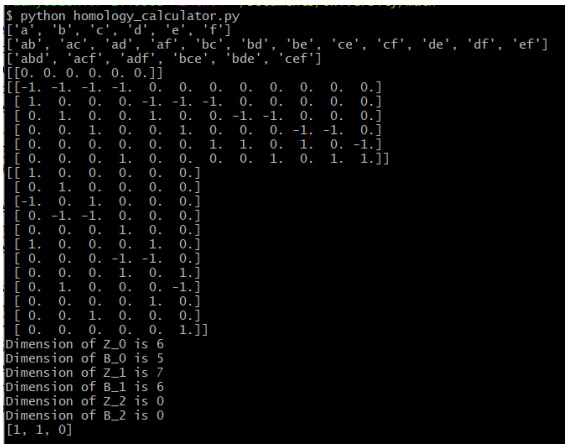
Here is the output for the 3-sphere and torus triangulations (left to right):



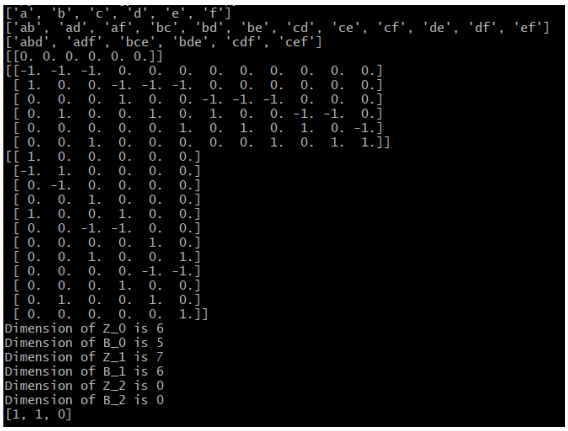



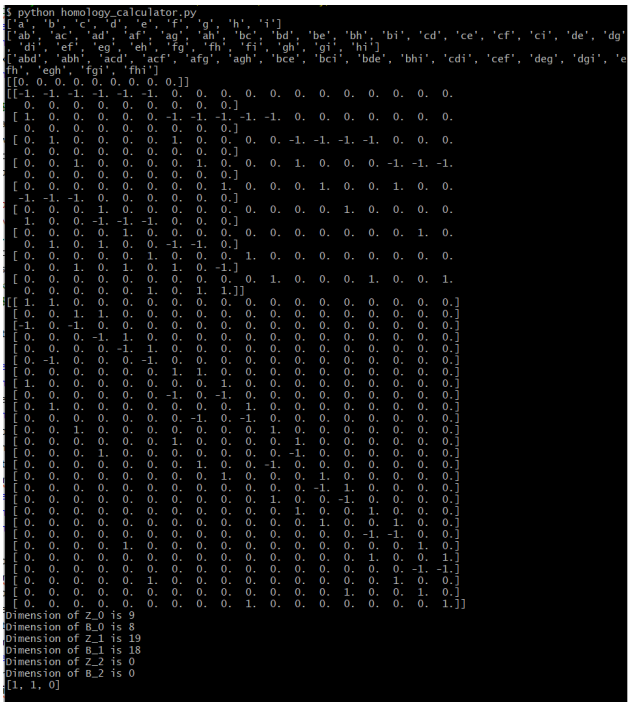

As we hoped, my program calculated that $H_3(S^3)$ and $H_0(S^3)$ have dimension 1 and $H_1(S^3) = H_2(S^3)$, and that the torus has 1 dimension in its 0th and 2nd homology groups and 2 for its 1st homology group.

The following are the results for a cylinder, which match the homology groups of the circle or $S^1$. This demonstrates how homotopically equivalent topological spaces have the same homology groups.



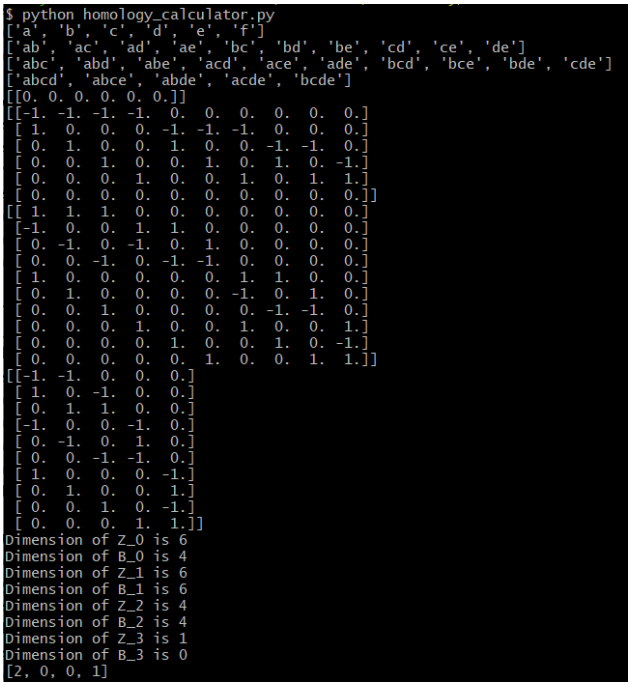

And here is the output for the Mobius-strip and Klein bottle (left to right):




Since these are nonorientable surfaces, my program's output doesn't tell the full picture. Perhaps one day I will understand the homology of nonorientable surfaces enough to write a program to solve for their homology groups as well. I googled it, and it seems that there could be a way to do this using the Smith-Normal form of these matrices: https://www.matem.unam.mx/ omar/mathX27/smith-form.html

I got curious, and decided to try adding an unconnected 0-simplex to the 3-sphere triangulation to see if my program would handle that gracefully. I just add the unused vertex, $f$, to the input: $threesphere = [$"$bcde$","$acde$","$abde$","$abce$","$abcd$","$f$"$]$.



As we might expect, the only thing that changes is the 0th homology group, since there are now 2 unconnected components in the simplicial complex.

## 9 Conclusion

We've introduced the concept of simplexes and simplicial complexes, which has enabled us to discretely express the set of $k$ dimensional closed simple curves in any simplicial complex, $S$, using elements of $C_k(S)$. We then defined the important topologically invariant concept of homology for the simplicial complex representations of topological spaces. This allowed me to calculate the homology groups of any orientable surface as long as we provide a valid triangulation of it. With this, we are now able to differentiate between certain topological spaces that have the same fundamental group. For instance, both $S^2$ and $S^3$ have trivial fundamental groups, but by examining their second homology groups, we reveal that $S^2$ has a 2 dimensional "hole" with a nontrivial $H_2$ while $S^3$ has a trivial $H_2$. It also exposes why $S^n$ is non-contractible by intuitively understanding that the nontrivialness of $H_k$ for $S^k$ represents a '$k$-dimensional hole'.

It was truly a pleasure getting to use my skills in abstract algebra, linear algebra, topology, and computer science in one final project.

## 10 Appendix: Code

Code for simplex calculation (the first step):

```
def simplex_calculator(simplexes):
    #Start by reordering each given simplex alphabetically so "acb" would become "abc"
    for i in range(len(simplexes)):
        sorted_characters = sorted(simplexes[i])
        simplexes[i] = "".join(sorted_characters)
    #Find the highest dimension simplex in this simplicial complex
    maxLen = 0
    for simplex in simplexes:
        if (len(simplex) > maxLen):
            maxLen = len(simplex)
    allSimplexes = []
    #Create an empty bin for each dimension from 0 to the maximum dimension
    for i in range(maxLen):
        allSimplexes.append(set())
    #Add each given simplex to the respective bin
    for simplex in simplexes:
        allSimplexes[len(simplex)-1].add(simplex)
    #Starting from the highest bin, add faces of each simplex to the bin below
    allSimplexes[maxLen-1] = sorted(allSimplexes[maxLen-1])
    for k in range(maxLen-1, 0, -1):
        kSimplexes = allSimplexes[k]
        for kSimplex in kSimplexes:
            for j in range(k+1):
                allSimplexes[k-1].add(kSimplex[:j] + kSimplex[j+1:])
        allSimplexes[k-1] = sorted(allSimplexes[k-1])
    #return the list of bins
    return allSimplexes
```

Code for boundary calculation:

```
def boundaries_calculator(simplexes):
    #initialize the list of boundary matrices
    boundaries = []
    #The 0th boundary matrix is just all 0's since the boundary of a point is 0.
    boundaries.append(np.zeros((1,len(simplexes[0]))))
    #loop through the bins
    for k in range(1,len(simplexes)):
        #initialize a matrix filled with 0s of the correct dimension
        matrix = np.zeros((len(simplexes[k-1]),len(simplexes[k])))
        for i in range(len(simplexes[k])):
            #for each k-simplex
            kSimplex = simplexes[k][i]
            #loop through its k-1 dimensional faces
            for j in range(k+1):
                #add either -1 or 1 to the relevant spot in our matrix
                matrix[simplexes[k-1].index(kSimplex[:j] + kSimplex[j+1:]), i] = -2*(j%2)+1
        #add this new matrix to the list of boundary matrices
        boundaries.append(matrix)
    return boundaries
```

## References


[1] M. A. Armstrong, *Basic Topology*, Springer

[2] Omar Antolín Camarena, Using the Smith normal form to compute homology, https://www.matem.unam.mx/ omar/mathX27/smith-form.html

[3] Thomas Goodwillie, Lecture notes