

PLEX – A MATLAB LIBRARY FOR STUDYING SIMPLICIAL HOMOLOGY

VIN DE SILVA

ABSTRACT. This paper is a brief introduction to the software library PLEX, currently in development. This is a collection of MATLAB routines designed to handle simplicial complexes and to compute some of their topological invariants. We describe the features currently supported by PLEX, and outline future developments. There remain several open problems for future research. These mainly concern issues of computational complexity and of robustness to noisy data.

CONTENTS

0. Introduction	1
1. Building a simplicial complex	2
1.1. Building an ϵ -approximation	2
1.2. An alternative construction, preferred by PLEX	2
1.3. Remarks on ϵ	3
1.4. Implementation in PLEX	4
2. Betti numbers	5
2.1. Computing b_k	6
2.2. A note on other coefficient rings	6
3. Maps and relative homology	7
3.1. The Betti number of a map	7
3.2. Relative Betti numbers	7
4. Computational efficiency	8
4.1. Elementary Collapses	8
4.2. Sparse matrix methods	9

0. INTRODUCTION

PLEX was conceived as a tool for analysing data in a topological framework. Currently PLEX is implemented in MATLAB, a flexible environment with fast linear algebra routines built in.

Usually, we begin with a set of N data points in \mathbb{R}^d , represented as a d -by- n matrix. In general, we assume that the chosen points lie on some (topological) subspace $X \subset \mathbb{R}^d$, whose identity is unknown to us. Our goal is to find out as much as we sensibly can about X .

Date: October 24, 2003. Draft version—read at your own risk!

The PLEX software was written with the following task in mind: to recover the Betti numbers of the space X , when X is assumed to be homeomorphic to a finite simplicial complex. (The Betti numbers are among the simplest invariants associated with a topological space. The k -th Betti number is an integer which, roughly speaking, counts the number of k -dimensional holes.)

The first step is to build a simplicial complex C using the data points as vertices. The complex C is intended to be a good approximation for the unknown topological space X .

After constructing C , its Betti numbers may be calculated. This is a mathematically simple procedure, but the actual computations are not always very efficient, and indeed become slow when there are many data points. On the other hand, it seems likely that improved algorithms can be found.

1. BUILDING A SIMPLICIAL COMPLEX

The standard way of building a simplicial complex from a topological space X goes like this. Consider a covering of X by open sets. Usually this covering is taken to be locally finite. A *good* covering is one in which the open sets U_i are all contractible, as are the pairwise intersections $U_i \cap U_j$ and indeed all finite intersections $U_{i_0} \cap \dots \cap U_{i_k}$.

Given a covering by open sets $\mathcal{U} = \{U_i\}$, we construct a simplicial complex with a vertex for each non-empty open set U_i and, more generally, with a k -cell for each set of $(k+1)$ distinct indices $\{i_0, \dots, i_k\}$ for which there is a non-empty intersection $U_{i_0} \cap \dots \cap U_{i_k} \neq \emptyset$.

If X is a polyhedron or some other reasonably nice space, then the simplicial complex $C(\mathcal{U})$ derived from a good covering \mathcal{U} is homotopy equivalent to X . We can therefore work with $C(\mathcal{U})$ as a homotopy-faithful approximation to X .

1.1. Building an ϵ -approximation. Now suppose we are given a set of N points $p_i \in \mathbb{R}^d$, lying on an unknown subspace $X \subset \mathbb{R}^d$. We may construct a simplicial complex in the following way.

First, replace each point p_i by the open Euclidean ball $B_i(\epsilon/2)$ of radius $\epsilon/2$ and center p_i . This “thickened data set”, which we denote P_ϵ , is intended to be a homotopy approximation to X . Under good conditions, it can be shown that $P_\epsilon \simeq X$.

Since the set of open balls $\mathcal{B}_\epsilon = \{B_i(\epsilon/2)\}$ is a good covering of P_ϵ , we can construct the simplicial complex $C(\mathcal{B}_\epsilon)$ and regard it as an approximation to X .

The complex $C(\mathcal{B}_\epsilon)$ can be characterised very simply. It is a simplicial complex with a vertex for each data point p_i and a k -cell for each set of $k+1$ distinct points $\{p_0, \dots, p_k\}$ whose open $\epsilon/2$ neighborhoods have non-empty intersection.

Clearly, if $\epsilon < \epsilon'$ then $C(\mathcal{B}_\epsilon) \subseteq C(\mathcal{B}_{\epsilon'})$. For any given set of data points $\{p_0, \dots, p_k\}$, there is a greatest value of ϵ for which the corresponding k -cell is not in $C(\mathcal{B}_\epsilon)$. We can think of this ϵ as a kind of diameter for the set $\{p_0, \dots, p_k\}$. We refer to this quantity as the *intersection diameter* of the set, with $\epsilon/2$ being the corresponding *intersection radius*.

1.2. An alternative construction, preferred by PLEX. In practice we use a different construction. Again, we begin by specifying a diameter ϵ . We construct a complex $C = C_\epsilon(P)$ with a vertex for each data point p_i and a k -cell for each

set $\{p_0, \dots, p_k\}$ whose diameter (in the metric sense) is smaller than ϵ . This means that $\|p_i - p_j\| < \epsilon$ for every pair of points in the set.

The *metric diameter* of a set $\{p_0, \dots, p_k\}$ is just the diameter in this usual sense, namely the maximum value of $\|p_i - p_j\|$ over all pairs in the set.

It is clear that $C_\epsilon(P) \subseteq C_{\epsilon'}(P)$ whenever $\epsilon < \epsilon'$.

The construction which uses the metric diameter has the following convenient property. Consider a set of points $\{p_0, \dots, p_k\}$. Then the k -cell corresponding to this set belongs to $C_\epsilon(P)$ if and only if all its edges belong to $C_\epsilon(P)$.

We say that a simplicial complex C is *locally contractible* if it has this property.

One corollary is that the 1-skeleton C^1 (or graph) of such a complex determines the whole complex (in fact, C is the largest simplicial complex whose 1-skeleton is C^1). This is computationally very useful, because we can store C in memory simply by storing the graph of C . Of course this is a compressed form of storage, and for some computations we will need to realise the cells of the complex more fully.

The complex $C(\mathcal{B}_\epsilon)$ based on the intersection diameter does not have this property. For instance, if $\{p_0, p_1, p_2\}$ form an acute-angled triangle, then their intersection diameter is always smaller than their metric diameter. If ϵ is chosen to be somewhere in between, then the three edges of the triangle will belong to $C(\mathcal{B}_\epsilon)$ but the triangle itself will not. Thus the complex will have a small, triangle-shaped hole right there.

More generally, small simplex-shaped holes can appear anywhere in $C(\mathcal{B}_\epsilon)$, for much the same reason. If the hidden space X is locally contractible (for instance if it is a manifold or a polyhedron) then these holes are misrepresentative. By definition, such small holes do not appear in the case of a locally contractible complex.

In certain applications, where X is a fractal set with nontrivial local behavior, there may be advantages in using the intersection diameter construction to pick up subtle local information. In general it is probably better to use a locally contractible construction.

1.3. Remarks on ϵ . Both constructions mentioned above depend on the choice of a parameter ϵ which is assumed may be chosen to be “good” uniformly across the space X . In practice, especially if the data is somewhat sparse, there may be no such ϵ .

There are a number of ways of trying to deal with this situation.

One simple idea is to allow ϵ to vary over different regions of the data. Somehow one determines in advance what the “good” range of values is in a given region, and modifies ϵ accordingly. It’s not entirely clear how to do this in a principled way.

The Isomap algorithm (see <http://isomap.stanford.edu/>) tackles this problem in a number of different ways. There, as here, the idea is to construct a reasonable graph representation of X from the data. There are a number of different algorithms. For instance, one might choose ϵ in a variable way, so that the expected number of neighbours of a given vertex is some number k . More directly, one might go ahead and connect each edge to its k nearest neighbours. If X is a d -dimensional manifold, triangulated as a generic lattice, then each point is joined to $d(d+1)$ other vertices. We can use these numbers to guide the selection of k . It is usually safer to overestimate than to underestimate.

In general, all algorithms which depend on determining and fixing a good graph representation of a dataset will come up against the same set of difficulties (and find the same solutions).

However, there is an entirely different avenue open to us if we are interested in the homology of the space X . This is the idea of the *system of complexes*.

It works like this. Rather than choosing a single value of ϵ (or a single function ϵ , in the case where it varies from place to place), one considers several values of ϵ simultaneously. (Abstractly, one considers “all” possible values, but computationally some finite compromise has to be made.)

For each ϵ there are corresponding homology groups. Moreover, if $\epsilon < \epsilon'$ then there is an inclusion map between the corresponding complexes $\iota_{\epsilon'}^{\epsilon} : C_{\epsilon} \rightarrow C_{\epsilon'}$. Since homology is functorial, there is an induced map in homology corresponding to each such inclusion. The idea is that this rather enriched system of vector spaces and maps contains considerably more information about X than the individual vector spaces themselves.

This kind of approach was used in the thesis of Vanessa Robins, in studying the connectivity of fractal sets.

1.4. Implementation in PLEX. The user interface in PLEX is quite straightforward. Assume that the data set $\{p_i\}$ has been represented by a d -by- N matrix P in MATLAB. The standard procedure for converting this into a simplicial complex goes as follows.

```
>> D = euclid(P);           % this computes the N-by-N
                             % matrix of euclidean distances
                             % using a standard algorithm

>> epsilon = 0.1;          % (for example)

>> G = (D < epsilon);      % which pairs of points are less
                             % than epsilon apart?

>> C = plex(G);
```

This last command converts the symmetric 0–1 matrix G into a 1-dimensional simplicial complex, with edges as specified by the 1s in G .

The complex C belongs to an object class specially defined by the PLEX library. Among other things, the user has access to a numbered list of vertices, and a corresponding list of edges (identified by their boundary vertices). The incidence relations between edges and vertices are stored directly in a matrix with entries in $\{0, \pm 1\}$.

The standard display command gives output of the following form:

```
>> display(C)

C =

1-dimensional simplicial complex

[100  151]
```

In this example, C has 100 vertices and 151 edges.

The next step is to expand the representation of C to include cells of higher dimensions. There is an `expand` command that does just this.

```
>> expand(C)

C =

2-dimensional simplicial complex

      [100  151  78]
```

As we see, there are 78 triangles in the expanded complex. These are precisely those triangles whose edges are already present in the 1-skeleton. We can expand by several steps at a time:

```
>> expand(C,5)

C =

5-dimensional simplicial complex

      [100  151  78  15  1  0]

>>
```

There are no 5-cells in the expanded complex, and so there is no need to expand any further.

By this stage, there are various tables of information stored in fields associated with the object C . The most important are the *vertex keys* and the *boundary maps*.

The vertex keys identify the k -cells of C by their vertices. In the example above, the 3-cell vertex key is a 15-by-4 matrix of integers. Each row corresponds to a 3-cell (tetrahedron) in C and the four entries specify its vertices.

The boundary maps specify, with orientation, the incidence relations between k -cells and $(k - 1)$ -cells. In the case of the 3-cells and 2-cells of complex C above, the boundary map is a 78-by-15 matrix with entries in $\{0, \pm 1\}$.

2. BETTI NUMBERS

Once a simplicial complex C has been constructed in PLEX, one can compute its Betti numbers. Let us recall the definitions and fix our notation.

First we define *chain groups* $C_k = C_k(C)$, which are vector spaces over \mathbb{R} with a basis element for each k -cell in C . The dimension of C_k , which is the same as the number of k -cells, we will refer to as the *chain rank* of C in dimension k , or r_k for short.

Next we define *boundary maps* $d_k : C_{k+1} \rightarrow C_k$ using the formula

$$d_k([v_0 \dots v_k]) = \sum_i (-1)^i [v_0 \dots \hat{v}_i \dots v_k],$$

where $[v_0 \dots v_k]$ denotes the basis element corresponding to the k -cell with vertices v_0, \dots, v_k and where \hat{v}_i means “delete vertex v_i ”.

These are the same boundary maps referred to at the end of the previous section. The k -th boundary map is stored by PLEX as an r_k -by- r_{k+1} matrix.

Things being what they are, we find that $d_{k-1} \circ d_k = 0$. The k -th *homology group (with real coefficients)* is then defined as the quotient

$$H_k = \text{Ker } d_{k-1} / \text{Im } d_k$$

and the k th *Betti number* b_k is the dimension of H_k as a real vector space.

Remark. Since b_k is computed in terms of maps between the chain groups C_{k+1} , C_k and C_{k-1} , it is important that the complex C has been correctly computed up to dimension $k+1$. Specifically, if the `expand` command is being used to build up the complex from a low dimensional skeleton, then it is necessary to have applied `expand` up to dimension $k+1$.

2.1. Computing b_k . Having constructed the complex C up to the desired dimension, we can now compute its Betti numbers. This part of the program leads to some of the slowest computations, so we will spend some time now discussing two possible implementations.

One method uses the simple formula

$$b_k = r_k - \text{rank}(d_k) - \text{rank}(d_{k+1}),$$

which can be derived from the rank-nullity formula in linear algebra. The ranks of the matrices d_k , d_{k+1} may be computed in a standard way using the MATLAB ‘rank’ command.

Once the rank of a given boundary map has been computed, PLEX can store it for future use. Then, when the same or an adjacent Betti number is requested by the user, PLEX does not waste time repeating calculations it has done before.

Another method, which is our preferred method, uses a ‘discrete Laplacian’ operator. This is defined exactly as in smooth Hodge theory, by:

$$\Delta_k = d_k^* d_k + d_{k+1} d_{k+1}^* : C_k \longrightarrow C_k$$

The discrete Laplacian is a symmetric, nonnegative-definite matrix, and the kernel satisfies

$$\text{Ker } \Delta_k = \text{Ker } d_k \cap \text{Ker } d_{k+1}^*$$

Another way to see it is that $\text{Ker } \Delta_k$ is the orthogonal complement to $\text{Im } d_{k+1}$ inside $\text{Ker } d_k$. As such, it is isomorphic to the homology group H_k .

We refer to the kernel as the space of *harmonic k -forms*, and write $\mathcal{H}_k = \text{Ker } \Delta_k$. The dimension of \mathcal{H}_k may be computed using the MATLAB ‘rank’ command.

Computationally it is not much more expensive to compute an explicit orthonormal basis for the kernel of Δ_k . This gives us a basis for H_k ; and also a representation of the projection $\text{Ker } d_k \rightarrow H_k$ which can be computed easily in terms of dot products with the basis elements. We refer to such a basis as a *harmonic basis*.

2.2. A note on other coefficient rings. In algebraic topology it is usual to use \mathbb{Z} coefficients for homology unless a different coefficient ring proves to be easier to compute, or more suitable for a particular application. The universal coefficient theorems assert that homology groups over the ring \mathbb{Z} in fact determine (up to isomorphism) the homology groups over any other commutative ring.

The current implementation of PLEX uses \mathbb{Q} (or, effectively, \mathbb{R}) as the coefficient ring, because the commutative algebra involved reduces to standard linear algebra. Homology groups, which are modules over the coefficient ring, become vector spaces over the coefficient *field*. As such, they may be characterised by a single invariant, their dimension.

We feel that it is not an urgent concern to implement the relevant module algebra needed to support calculations with coefficients in \mathbb{Z} . There is already plenty of information in the Betti numbers themselves, and plenty of unexplored scope for using them.

On the other hand, it is an appealing prospect to implement \mathbb{Z}_2 computations in PLEX. Since \mathbb{Z}_2 is a field, we are still in the realm of linear algebra on vector spaces. The \mathbb{Z}_2 Betti numbers contain similar information to the usual Betti numbers. The practical advantage is that floating point arithmetic may be replaced by bit-wise arithmetic, resulting in a considerable scalar improvement in computational complexity.

3. MAPS AND RELATIVE HOMOLOGY

3.1. The Betti number of a map. Whenever there is a simplicial map between two simplicial complexes $f : C \rightarrow D$, there is an induced map between corresponding homology groups $H_k f$ or $f_* : H_k(C) \rightarrow H_k(D)$. This map is induced directly from a map between chain complexes $C_k f : C_k(C) \rightarrow C_k(D)$.

PLEX allows the computation of $H_k f$, in the form of a matrix representation with respect to harmonic bases of $H_k(C)$ and $H_k(D)$.

It is a straightforward procedure. The simplicial map f is represented as a two-column matrix \mathbf{f} whose first column lists the vertices of C and whose second column indicates the vertices of D they are carried to by f . The command:

```
>> ch = chmapping(C,D,f)
```

returns a series of matrices $\mathbf{ch}\{\mathbf{k}\}$ for the induced maps C_k , determined by computing the image of each k -cell under f . Once we have C_k , we can compute the matrix of H_k using dot products:

$$[H_k]_{ij} = \langle f_i, C_k(e_j) \rangle$$

where e_i and f_j are harmonic basis elements for $H_k(C)$ and $H_k(D)$ respectively. Here the various orthogonality properties of harmonic bases are important.

3.2. Relative Betti numbers. Relative homology groups are defined similarly to ordinary homology groups. If D is a subcomplex of the simplicial complex C , then relative homology is a way of describing the topology of C “modulo D ”.

Specifically, the chain groups $C_k(D)$ can be viewed naturally as subgroups of $C_k(C)$, and so we define relative chain groups:

$$C_k(C, D) = C_k(C)/C_k(D)$$

The boundary maps $d_k : C_{k+1}(C) \rightarrow C_k(C)$ pass directly to the quotient level maps $d_k : C_{k+1}(C, D) \rightarrow C_k(C, D)$ (since $d_k(C_{k+1}(D)) \subseteq C_k(D)$). The relative Betti numbers $b_k(C, D)$ are defined as $\text{Ker } d_k / \text{Im } d_{k-1}$ for these induced maps.

The easiest way to specify a subcomplex in PLEX is to provide a list of vertices `vlist`. The MATLAB commands

```
>> E = relative(C,vlist);
>> b_k = betti(E,k);
```

compute the k -th Betti number of C relative to the subcomplex of cells whose vertices do *not* meet `vlist`. Here E is stored as a special data form, a *relative complex*.

Similarly the commands

```
>> F = restrict(C,vlist);
>> b_k = betti(F,k);
```

compute the k -th Betti number of the subcomplex F induced by `vlist`. In this case, F is stored as an ordinary simplicial complex.

The `expand` command (which fills in higher-order simplices) is not enabled for relative complexes. The reason is that `relative` works by deleting all cells outside the subcomplex specified. On the other hand, to `expand` correctly one needs to know about all the neighbours of the surviving cells (and not just the surviving cells themselves). For this reason one should `expand` before using the `relative` command. No such restriction applies to the `restrict` construction, which commutes with `expand`.

4. COMPUTATIONAL EFFICIENCY

Finding the rank of an arbitrary matrix can be a very slow process. The (time) complexity of the standard Gaussian elimination algorithm for a square n -by- m matrix is $O(n^2m)$, so one can easily generate examples where the processing time is “too long”. Here we describe various ways of trying to get around the problem.

4.1. Elementary Collapses. One simple approach involves the notion of elementary collapse. Let v be a k -cell in a simplicial complex C and suppose that it belongs, as a face, to exactly one $k + 1$ -cell u .

It is easy to see that $C \setminus \{u, v\}$ is a simplicial complex which is homotopy equivalent to C . Moreover, it has fewer cells. PLEX implements this procedure iteratively, removing as many $(k, k + 1)$ -cell pairs as possible.

Here is a simple example. We begin by constructing a 5-simplex, topologically equivalent to a ball (and hence contractible).

```
>> G = ones(6);
>> C = plex(G); expand(C,5)
```

```
C =
```

```
5-dimensional simplicial complex
```

```
[6 15 20 15 6 1]
```

We should be able to remove a $(4, 5)$ -cell pair, consisting of the main 5-cell and one of its faces.

```
>> collapse(C,5)
```

```
C =
```

```
5-dimensional simplicial complex
```

```
[6 15 20 15 5 0]
```

Now we can to remove a few $(3, 4)$ -cell pairs.

```
>> collapse(C,4)
```

```
C =

5-dimensional simplicial complex

      [6  15  20  10  0  0]
```

If we prefer, we can collapse all the way:

```
>> collapse(C,inf)

C =

5-dimensional simplicial complex

      [1  0  0  0  0  0]
```

As expected, we have reduced C to a point.

This COLLAPSE operation is fairly well-behaved, but it is not quite well-defined.

In the example above, the very first cell-removal necessitated choosing a 4-cell to be removed with the central 5-cell. This choice is made arbitrarily by PLEX based on the order of listing of 4-cells in its representation of the complex C . There is no topologically meaningful “canonical choice” in this situation.

It is easy to construct examples where the choice of cell affects the subsequent progress of the iterative algorithm. PLEX makes no attempt at “intelligent” (or forward-thinking) cell selection, but it is possible that there may be some simple criteria which would improve results.

On the plus side, we have:

Proposition 4.1.1. *It is safe to COLLAPSE before EXPANDING. More precisely,*

```
>> D = expand(collapse(C,inf))
is always a collapsed form of
>> E = expand(C)
and in particular is homotopy equivalent to it. □
```

The proof consists in noting that expansion can only occur in sites which are immune to collapse.

The COLLAPSE operation will typically give a noticeable but small reduction in the size of a simplicial complex. However, it is never likely to reduce a complex significantly close to the bare bones of its homotopy structure (except in specially chosen examples). Collapses alone are not sufficient for that; a really sophisticated algorithm would need to consider elementary expansions as well. The complexity of the problem appears to grow horrifically if expansions are permitted.

4.2. Sparse matrix methods. More serious approaches to the problem should take into account the sparse nature of the boundary maps d_k and the Laplacians Δ_k . The (i, j) -th element of the symmetric matrix Δ_k is non-zero only if the i -th and j -th k -cells are neighbours in the simplicial complex. There are many algorithms specially tailored to handle sparse matrices. Eigenvalue methods work particularly well for sparse symmetric matrices. As yet, PLEX does not take advantage of any of these methods, but future releases will incorporate such ideas.