SCHEDAE INFORMATICAE
VOLUME 16 2007

Acyclic Subspace Homology Algorithm for Inclusions

Natalia Żelazna
Jagiellonian University, Institute of Computer Science
Nawojki 11, 30-072 Kraków, Poland
e-mail: zelazna@ii.uj.edu.pl

Abstract. We present a new approach to computing the map induced in homology by the inclusion map, based on an acyclic subset algorithm for homology of cubical sets. This approach is based on constructing a possibly large acyclic subset in the domain of the map and computing the map induced in relative instead of plain homology.

1. Introduction

In this paper we describe an application of a homology algorithm based on an acyclic subset [5] to the computation of a map induced in homology by the inclusion map of cubical sets. The need for such a map appears, for example, in [1]. The classical approach to computing a map induced in homology by an arbitrary chain map relies on the computations "by definition" and requires solving several large systems of linear equations on the level of homology groups and has $O(n^4)$ complexity, where n depends on the size of the domain and codomain of the chain map. Some acceleration may be obtained by geometrical reductions applied to the domain and codomain of the map, as described in [4]. In this note we show that for inclusions we are able to speed up the process even more, thanks to the concept of the acyclic subspace.

2. Preliminaries

2.1. Cubical homology

Throughout this paper the sets of natural, integer, rational and real numbers will be denoted respectively by $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$. We assume that the reader is familiar with the homology theory, in particular, the homology theory of cubical sets, as introduced in [2] , however, we recall some basic concepts for the convenience of the reader.
By an elementary interval we understand a closed interval $I \subset \mathbb{R}$ of the form $I = [l, l+1]$ (nondegenerate elementary interval) or $I = [l, l]$ (degenerate elementary interval), where $l \in \mathbb{Z}$. A finite Cartesian product $Q = I_1 \times \ldots \times I_d$ of elementary intervals is called an elementary cube in $\mathbb{R}^d$. The dimension of Q is defined as the number of nondegenerate factors in the Cartesian product. The family of all elementary cubes in $\mathbb{R}^d$ is denoted by $\mathcal{K}^d$ and the family of all k-dimensional cubes in $\mathbb{R}^d$ by $\mathcal{K}^d_k$. Cubes in $\mathcal{K}^d_d$ are called full cubes.
Elementary cubes are building blocks for cubical sets. A set $X \subset \mathbb{R}^d$ is a cubical set if it is a finite union of elementary cubes. We say that a cubical set is full, if it is a union of full cubes.
Given any elementary cube $Q \in \mathcal{K}^k_d$ we define an elementary cubical k-chain as the function $\widehat{Q}\colon \mathcal{K}^k_d \to \mathbb{Z}$ defined by

$$\widehat{Q}(P) = \begin{cases} 1 & \text{if } P = Q \\ 0 & \text{otherwise,} \end{cases}$$

for $P \in \mathcal{K}^d_k$. A cubical k-chain is a function $c\colon \mathcal{K}^d_k \to \mathbb{Z}$ which is zero on all but a finite number of arguments. Every k-chain is a finite linear combination of elementary k-chains. All k-chains form a free abelian group, denoted by C_k. Note that all elementary k-chains constitute a basis for C_k.
Given two elementary chains $\widehat{P}$ and $\widehat{Q}$ we define the cubical product of these chains by

$$\widehat{P} \diamond \widehat{Q} := \widehat{P \times Q}.$$

This definition can be extended linearly to arbitrary chains.
The boundary operator is the homomorphism $\partial_k\colon C_k \to C_{k-1}$ defined recursively by

$$\partial\widehat{Q} := \begin{array}{ll} 0 & \text{if } Q = [l], \\ \widehat{[l+1]} - \widehat{[l]} & \text{if } Q = [l, l+1]. \\ \partial\widehat{I} \diamond \widehat{P} + (-1)^{\dim I} \widehat{I} \diamond \partial\widehat{P} & \text{if } Q = I \times P, \\ & \text{where } I \in \mathcal{K}^1, P \in \mathcal{K}^{d-1}. \end{array}$$

The support of a cubical chain c is the cubical set

$$|c| := \bigcup \{Q \in \mathcal{K}_k^d, c(Q) \neq 0\}.$$

For any cubical set X we define the group of k-chains of X as

$$C_k(X) := \{c \in C_k, |c| \subset X\}$$

and the boundary operator $\partial_k^X : C_k(X) \to C_{k-1}(X)$ as the restriction of the operator ∂_k to $C_k(X)$. $C_k(X)$ is a free abelian group with the basis consisting of all elementary chains with support in X. The collection $\{(C_k(X), \partial_k^X)\}_{k\in\mathbb{Z}}$ is the chain complex generated by the set X and is denoted by $C(X)$.

One can check that $\partial_k^X \partial_{k+1}^X = 0$ and so the group of k-boundaries $B_k(X) := im\partial_{k+1}^X$ is the subgroup of k-cycles $Z_k(X) := \ker \partial_k^X$. Thus we can define the k-th homology group of X as the quotient group

$$H_k(X) := Z_k(X)/B_k(X).$$

By the homology of X we mean the collection $H(X) := \{H_k(X)\}_{k\in\mathbb{Z}}$.

2.2. Acyclic subset

The classical way of computing homology consists in finding the Smith Normal Form of the boundary maps. The complexity of this algorithm is supercubical. Among the methods intended to speed up the computations are the reduction algorithms which aim at finding a smaller set with the same homology as the original set X. The acyclic subspace algorithm (see [5]) is a reduction algorithm based on the general observation that if A is an acyclic subset of X, then

$$H_n(X) \cong \begin{array}{ll} H_n(X, A) & \text{for } n \geq 1 \\ \mathbb{Z} \oplus H_n(X, A) & \text{for } n = 0 \end{array}.$$

We recall that A is acyclic if A has the same homology as the set consisting of a single point. This reduction method relies on a fast recursive algorithm constructing a possibly large acyclic subset A of X and then computing the relative homology of the pair (X, A). Relative homology requires knowledge only of elementary cubes in the neighborhood of $X \setminus A$ thanks to the excision property.

3. Homology of maps

3.1. Chain maps

Let $\mathcal{C} = \{C_k(X), \partial_k\}$ and $\mathcal{C}' = \{C_k(Y), \partial'_k\}$ be chain complexes generated by cubical sets X and Y. We recall that the chain map $\varphi\colon \mathcal{C} \to \mathcal{C}'$ is a collection of group homomorphisms $\varphi_k\colon C_k(X) \to C_k(Y)$ such that for every $k \in \mathbb{Z}$ the condition

$$\partial'_k \varphi_k = \varphi_{k-1} \partial_k.$$

is satisfied.
Given a chain map φ we define the associated homology map as the sequence of homomorphisms $\varphi_{k*}\colon H_k(X) \to H_k(Y)$ by

$$\varphi_{k*}([z]) := [\varphi_k(z)],$$

where $z \in Z_k(X)$. It is straightforward to verify that the map φ_{k*} is well defined.

4. Inclusion map

Given cubical sets X, Y such that $X \subset Y$, there is an associated chain map $i_k\colon C_k(X) \to C_k(Y)$ given by

$$\iota_k(\widehat{Q}) = \widehat{Q}$$

for every $Q \in \mathcal{K}_k(X)$.

4.1. Use of acyclic subspace

Our algorithm is based on the following theorem.

THEOREM 1. Assume $X \subset Y$ and $A \subset B$ are both acyclic and such that $A \subset X$ and $B \subset Y$. Let $\iota\colon X \to Y$ and $\kappa\colon (X, A) \to (Y, B)$ denote inclusion maps. Then $H_*(\iota)$ and $H_*(\kappa)$ are conjugated.

Proof. We have the following diagram of long exact sequences (see [2]):

$$\begin{array}{ccccccccc} \ldots \to H_n(A) & \longrightarrow & H_n(X) & \longrightarrow & H_n(X,A) & \longrightarrow & H_{n-1}(A) \to \ldots \\ \downarrow & & \downarrow{\scriptstyle \iota_*} & & \downarrow{\scriptstyle \kappa_*} & & \downarrow & . \\ \ldots \to H_n(B) & \longrightarrow & H_n(Y) & \longrightarrow & H_n(Y,B) & \longrightarrow & H_{n-1}(B) \to \ldots \end{array}$$

Since A and B are acyclic we have for all $n \geq 1$ $H_n(A) = H_n(B) = 0$, and $H_0(A) = H_0(B) = \mathbb{Z}$, therefore the conclusion follows from the exactness of the diagram.

5. Algorithm

The algorithm starts with two cubical sets, X and Y and it returns the map $\kappa_*: H_*(X, A) \to H_*(Y, B)$ conjugated with the map induced in homology by the inclusion map $\iota: X \to Y$ as in Theorem 1. Acyclic subset A of the set X mentioned in this theorem is constructed as follows: starting from an arbitrarily selected cube from X we expand A by adding such cubes $P \in X \setminus A$ that $A \cup P$ is still acyclic. This algorithm uses queue to store the neighbors of cubes added to A. These cubes are candidates for addition to A in the next steps. For every processed cube P from the queue the acyclicity test is performed in order to verify whether adding this cube will not change the homology of A. In the following formal description of the algorithm we will assume that we have the function `AcyclicSubspace(cubicalSet X)` which returns the acyclic subset of a given cubical set. Several algorithms which may be used to implement such a function are discussed in detail in [5] and all of them are available from the CAPD library [8]. After having constructed the acyclic set $A \subset X \subset Y$ we proceed with the construction of the set B such that $A \subset B \subset Y$ and B is also acyclic. This is done similarly to the construction of A but instead of an arbitrary cube and its neighbors we use the set A and its neighbors. This construction is preceded by the procedure described in [5] as shaving, consisting in removing cubes, whose intersection with the rest of set is acyclic. We also assume that the following three functions are given:

- `Shave(cubicalSet X)`
- `ExtendAcyclicSubspace(cubicalSet A, Y)`
- `HomologyMap(cubicalSet X, A, Y, B)`

The first function is implemented in the CAPD library [8] on the basis of the algorithm discussed in [5]. The second function is a straightforward adaptation of the function `AcyclicSubspace`. The third function is also implemented in [8] on the basis of the algorithm described in [4].

ALOGRITHM 0.1. HomologyOfTheInclusionMap

```
function HomologyOfInclusionMap (cubicalSet X,Y)
begin
   Shave(X);
   Shave(Y);
   A := AcyclicSubspace(X);
   B := ExtendAcyclicSubspace(cubicalSet A, Y);
   result := HomologyMap(cubicalSet X, A, Y, B);
   return result;
end;
```

Theorem 1 implies that for $n \geq 1$ computed maps $\kappa_n \colon H_n(X, A) \to H_n(Y, B)$ and $\iota_n \colon H_n(X) \to H_n(Y)$ are conjugate.

6. Implementation

The implementation is relatively simple, because – as we already mentioned – the main building blocks:

- the algorithm for the construction of acyclic subspace,
- the algorithm for the homology of a map of pairs of topological spaces,

are already contained in the CAPD library [8] and CHomP library [9]. The programs are written in C++ with the extensive use of generic programming techniques, and, in particular, the STL library.

Cubical sets are stored as bitmaps with the coordinates of a pixel corresponding to the lower left corner of a cube. Such representation consumes less memory than the mere storing pairs or triples of one byte coordinates and allows fast access to the data, by means of iterators.

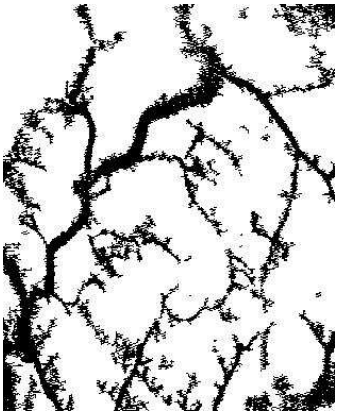

Fig. 1.

7. Numerical experiments

So far no specialized algorithm nor software is available for computing the homology of the inclusion maps. The only way to achieve this task is to apply a general algorithm for homology maps. Therefore we will compare an implementation of the algorithm described in Section 5 with the general purpose algorithm for maps presented in [4] and implemented by P. Pilarczyk in [9]. In the sequel we use the following shorthand notation:

PP Pilarczyk's implementation [7] of the homology of maps algorithm [4].

AS an implementation of the algorithm described in Section 5 based on the implementation presented in [6].

All these implementations as well as the benchmark programs were compiled together using gcc compiler version 3.4.2 ported for MS Windows XP. Presented timings were obtained on a 2GHz Pentium PC with 1.5 GB RAM running Windows XP. In all cases a size parameter represents the number of full cubical sets in the sum of the domain and codomain of the inclusion map before the extraction of an acyclic subset. Where it appears, α denotes the approximate complexity exponent in the formula $T = cn^{\alpha}$, where n denotes the size of the input and T stands for the recorded computation time.

7.1. Blood vessels

In this case the domain of the map is the bitmap presented in Fig. 1. It has 42 connected components. This is the preprocessed endoscopic picture of colon mucosa. Homology of the inclusion map is used both during the binarization of the original picture and during its analysis (see [1] for details).

The codomain of the inclusion map was constructed here by adding to the domain vertical lines as is done in the application presented in [1]. Such an operation glues separate components and the resulting set has a smaller zero homology group, but, on the other hand, such lines contribute to new holes in the codomain. Here we present results for the bitmap with extra vertical lines inserted every 40 pixels.

Size	**AS**	**PP**
250030	9.421	23.235
909626	40.157	240.015
1977600	207.782	1225.8
3453950	467.421	5739.95
α	1.51	2.06

The bitmaps were here rescaled and we can see that as the amount of data grows, the difference between times needed by the two algorithms increases.

7.2. Cubical torus

In this case the domain of the inclusion map is a cubical torus mapped to a set that consists of this torus with an additional cube added in such a way, that it changes the first homology group from $\mathbb{Z}^2$ to $\mathbb{Z}$. The test is performed by rescaling both sets in all the dimensions without changing their homology groups. The constructed acyclic subset contains here about 90% of cubes from the domain.

Size	**AS**	**PP**
20525	1.461	16.422
147897	3.516	394.703
480805	6.235	2171.42
1117940	10.547	Out of memory
2157980	13.172	Out of memory
3699630	18.546	Out of memory
α	0.63	0.90

In this case we see that **PP** is not only much slower but also demands more memory.

7.3. Cahn-Hilliard equation

The Cahn-Hilliard equation is the model which describes the process of phase separation, by which the two components of a binary alloy spontaneously separate and form domains pure in each component. The solution of this equation is the function f depending on time t and location x. In order to identify components of the alloy one may consider sets, where the function is positive or negative. In the following example the domain of the inclusion map is the cubical approximation of one of such sets, constructed on a cubical grid $128 \times 128 \times 128$. This set is connected, has many tunnels (the large first Betti number) and only one void. In the codomain there is the same set with masks, which close some of the tunnels, forming voids, and therefore change the second Betti number.

Size	**AS**	**PP**
2202735	189.03	1226.24
19924184	3025.88	Out of memory

8. Conclusions

In the paper we introduced a new approach to computing homology of the inclusion map. Presented results of numerical experiments clearly indicate that the removal of the acyclic subset significantly improves performance of algorithms computing homology of the inclusion map, diminishing both computation time and the memory requirements. We expect that a similar procedure for a projection map is possible. If so, then one can expect a significant speed up of algorithms for homology of continuous maps by applying the Górniewicz-Granas technique described in [4]. The details are left for future investigation.

9. References

[1] Mrozek M., Żelawski M., Gryglewski A., Han S.; *Image analysis by homological methods*, in preparation.

[2] Kaczynski T., Mischaikow K., Mrozek M.; *Computational homology*, The International Journal of Applied Mathematical Sciences, 157, 2004.

[3] Kaczynski T., Mrozek M., Ślusarek M., *Homology computation by reduction of chain complexes*, Computers and Mathematics with Applications, 35, 1998, pp. 59–70.

[4] Mischaikow K., Mrozek M., Pilarczyk P.; *Graph approach to the computation of the homology of continuous maps*, Foundations of Computational Mathematics, 5, 2005, pp. 199–229.

[5] Mrozek M., Pilarczyk P., Żelazna N.; *Homology algorithm based on acyclic subspace*, Computers and Mathematics with Applications, accepted.

[6] Mrozek M.; *Homology Software*, 2006,
http://www.ii.uj.edu.pl/∼mrozek/software/homology.html

[7] Pilarczyk P.; *Homology Software*,
http://www.pawelpilarczyk.com/homology.htm

[8] *Computer Assisted Proofs in Dynamics*,
http://capd.wsb-nlu.edu.pl/

[9] *CHomP, Computational Homology Project, Homology Software*,
http://chomp.rutgers.edu/

Received July 10, 2007