scc2020: A File Format for Sparse Chain Complexes in TDA

Michael Lesnick, Michael Kerber

October 30, 2020

Abstract

Several software projects concerning efficient computations for multiparameter persistence are now underway. With this, the need for a standard file format for chain complexes and presentations of multiparameter persistence modules has become apparent: Having the various codes adhere to such a standard would allow us to more easily compare the performance of software and leverage the strengths of multiple software packages in our data analyses. This document proposes such a standard.

1 Setup

Our goal is a succinct representation of chain complexes of the form

$$M_1 \xrightarrow{f_1} M_2 \xrightarrow{f_2} M_3 \xrightarrow{f_3} \ldots \xrightarrow{f_{n-1}} M_n \xrightarrow{f_n} M_{n+1} \tag{1}$$

where $M_1, \ldots, M_{n+1}$ are finitely generated free d-parameter persistence modules with base field K as defined in [**?**, p.18], and $f_1, \ldots, f_n$ are homomorphisms between these modules. Every module M_i is determined by an (ordered) set of homogeneuous generators; write m_i for the number of generators, $G_{i,1}, \ldots, G_{i,m_i}$ for the generators and $g_{i,1}, \ldots, g_{i,m_i}$ for their grades in $\mathbb{R}^d$. Then, every f_i can be represented as a $m_{i+1} \times m_i$-matrix with entries in K, where the j-th column represents $f_i(G_{i,j})$ as a linear combination (with shifts) of the generators $G_{i,1}, \ldots, G_{i,m_i}$

Remark 1. *An important special case is where $n = 2$, in which case the chain complex is simply a presentation of the cokernel of f_1.*

Remark 2. *Chain complexes of non-free persistence modules do arise in TDA, from* multi-critical *multi-filtrations. In the future it might be worthwhile to extend this specification to handle these.*

2 Format

The format allows to include an arbitrary number of empty lines and lines starting with #, which are just ignored.

- We require that the first (non-empty, non-commented) line of the file contains only the word ”scc2020”.
- The second line contains an integer $d \geq 0$ which denotes the number of persistence parameters.
- The third line contains a sequence of $n + 1$ integers $m_1, .., m_{d+1}$, separated by spaces, denoting the size of the generating sets of $M_1, .., M_{d+1}$.
- The next line, which is optional, is of the form “--reverse [list]”, where [list] is a sequence of numbers between 1 and d separated by spaces. If the number j appears in this list, then the multiparameter persistence modules will be taken to have decreasing coordinate direction j. For example, if $d = 2$, then “--reverse 1” indicates that we are working with $\mathbb{R}^{\mathrm{op}} \times \mathbb{R}$-indexed persistence modules, whereas if this line is omitted, then this indicates that we are working with $\mathbb{R}^2$-indexed persistence modules. To avoid confusion and bugs, the line “--reverse” with no arguments should return an error.

- The file continues with d blocks consisting of $m_1, m_2, ..., m_d$ lines, each block specifying the grades of the generators of M_i and the columns of the matrix for f_i. Each line in a block has the form

  $$v_1 \; v_2 \; \ldots \; v_d \; [;] \; k_1[\mathrm{C}x_1] \; k_2[\mathrm{C}x_2] \; \ldots \; k_m[\mathrm{C}x_m]$$

  For the j-th line in block i, $v_1, \ldots, v_d$ determines the grade of the j-th basis element of M_i. After the (optional) semicolon, the image of the basis element under f_i is specified by a sequence of words of the form $k\mathrm{C}x$ with k an integer and x specifying a field element. This fixes the coefficient of the (k, j)-entry in the matrix to be x. The values $k_1, \ldots, k_m$ do not have to be in increasing order, but no repetitions are allowed. Unspecified row entries are interpreted to be 0. Moreover, one can also just specify k (without Cx) which is interpreted as kC1.

- Finally, an optional block of m_{d+1} lines can be added, specifying the grades of the generating set for M_{d+1}. If used, the lines are of the form

  $$v_1 \; v_2 \; \ldots \; v_d$$

  as above. The reason this block is optional is that if one is only interested in the homology modules

  $$\ker f_{i+1} / \mathrm{im} f_i$$

  for $i \in \{2, \ldots, n\}$, then this block is not needed.

Field elements and grades This proposal does not specify the format of the grades (v_j above) or of the field elements (x_j above), though a future revision may specify this. For the grade coordinates, it is recommended to use a string with an obvious representation as a real number, for instance a decimal number or perhaps fraction. For the field elements, it is recommended to intepret integers in the obvious way (i.e., 1 is the one in the field, $3 = 1 + 1 + 1$, and so on). If chain complexes with more exotic field coefficients are specified, we recommend to describe them as a comment in the file.

3 Examples

We demonstrate which data can be represented in the defined format.

Simplicial complexes. A filtered simplicial complex defines a chain complex in a natural way, where the morphisms are defined by the boundary operator. As an example, we consider the Vietoris-Rips filtration for the three points $A = (0, 0)$, $B = (3, 0)$, and $C = (0, 4)$. The full complex consists of the 3 vertices, 3 edges and one triangle. We obtain the sequence

$$M_1 \xrightarrow{f_1} M_2 \xrightarrow{f_2} M_3$$

where M_1 is generated by ABC, M_2 by AB, AC, BC and M_3 by A, B, C. By definition of the Vietoris-Rips filtration, the three vertices are graded by 0. Since the length of the three edges are 3, 5, and 4, the edges are graded by 1.5, 2.5, and 2, and the triangle by 2.5. Hence, we can represent the simplicial complex in ssc2020 format as

```
#Lines starting with # are comments
ssc2020
#Number of parameters
1
#Sizes of generating sets
1 3 3
#First block
2.5 ; 1C1 2C-1 3C1
#Second block
```

```
1.5 ; 1C-1 2C1
2.5 ; 1C-1 3C1
2   ; 2C-1 3C1
#Third block (optional)
0
0
0
```

If the complex is only used for computations over $\mathbb{Z}_2$ coefficients (which is an important case in many applications), we can skip orientations and simplify further:

```
ssc2020
1
1 3 3

2.5 ; 1 2 3

1.5 ; 1 2
2.5 ; 1 3
2   ; 2 3

0
0
0
```

Free implicit representations This data format is inspired by the firep data structure in Rivet [**?**, Sec 5.1]; see `https://rivet.readthedocs.io/en/latest/inputdata.html#firep-algebraic-input`. There are in fact two slightly different firep formats introduced in RIVET. The latter is a in a soon-to-be-merged pull request and is intended to replace the old one, though RIVET will be backwards compatible.

A file in the new firep format can be converted directly into the ssc2020 format in the following way:

1. Include the line “ssc2020” at the top (and remove the line “--datatype firep”, if present),

- Insert a line containing just the character 2 after it. (This specifies that one considers a 2-parameter persistence module.)
- If the firep file contains the lines “–xreverse” or “–yreverse”, these should be deleted, and the line “--reverse [list]”, with list taken to be be the appropriate thing, should be given on the fourth line, as explained above.
- Other optional command-line flags, beginning with “--”, may need to be removed from the file, if present.

Converting a file in the old firep format to our format is similar.