Algebraic Persistence
the algebra of persistence modules

Mikael Vejdemo-Johansson
Primoz Skraba

School of Computer Science
University of St Andrews
Scotland

Jozef Stefan Institute
Ljubljana, Slovenia

6 July 2012

What are we doing?

Computational Linear Algebra

- Fast matrix algorithms
- Simple rings: $\mathbb{Z}, \mathbb{R}, \mathbb{C}, \mathbb{Z}/p\mathbb{Z}$

Persistent homology

- This talk

Computational Algebraic Geometry

- Complicated rings and modules
- $\Bbbk[x_1, \ldots, x_r]$
- Gröbner bases and $O(n!)$ algorithms

Outline

Persistence modules

Introduced and identified by Zomorodian and Carlsson (2005).

Definition

A persistence module M is a graded module over the graded ring $\Bbbk[t]$.

Connection to persistent homology

Filtered chain complexes and their persistent homology both are persistence modules.

A filtered chain complex has a generator in degree n for each simplex appearing at filtration step n.

A persistent homology module has a generator in degree n killed by t^m for each barcode entry $(n, n+m)$.

Category of persistence modules

Thus, to study persistent homology, we will benefit from studying the category of persistence modules – which by the results by Zomorodian and Carlsson means studying the category of graded modules over $\mathbb{k}[t]$.

Very nice ring. Very nice category. Here are some things that are true:

- Euclidean domain. Division algorithm works. Also, therefore PID.
- Submodules of free modules (i.e. Projective modules) are free. All modules have a presentation by a short exact sequence $0 \to R \to G \to M \to 0$ where R, G are both *free* modules.

Outline

Nested modules

Since persistence modules have canonically free presentation, we can represent a persistence module by tracking the generators and relations. There are two ways to do this with a global module C of chains:

Represent chains

We maintain matrices representing

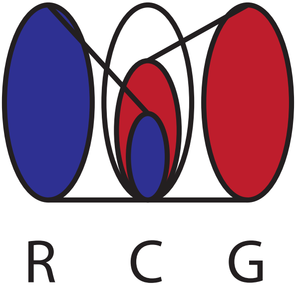
$$G \to C \text{ and } R \to C$$

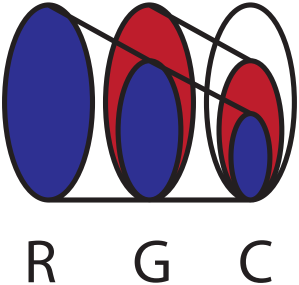
Represent relations embedding

We maintain matrices representing

$$G \to C \text{ and } R \to G$$

Nested module representations





- \+ We can work with each matrix separately
- – Larger matrices
- javaPlex output

- \+ We have swifter access to the barcode
- – We have to modify both matrices simultaneously
- Zig-Zag internal state

Homomorphisms as matrices with conditions

A homomorphism between two modules can be represented by images of the generators such that boundaries all map to boundaries.

$$\begin{array}{ccccccccc}
0 & \longrightarrow & R & \longrightarrow & G & \longrightarrow & M & \longrightarrow & 0 \\
 & & \downarrow & & \big\downarrow & & \downarrow & & \\
0 & \longrightarrow & R' & \longrightarrow & G' & \longrightarrow & N & \longrightarrow & 0
\end{array}$$

To represent a homomorphism $M \to N$, it is enough to work with a homomorphism $G \to G'$ known to map relations to relations.

This corresponds to the well-formed map requirement in
Cohen-Steiner, Edelsbrunner, Harer, Morozov:
Persistent Homology for Kernels, Images, and Cokernels.

Outline

Normal forms, equality, and membership

Question

How can we determine equality for two elements of $M = G/R$?

Question

How can we determine whether $z \in C$ represents an element of $M = G/R$?

Question

How can we determine whether $z \in G$ represents an element of R?

Question

How do we produce bases for G and R that make computation easy?

Normal forms, equality, and membership

Answer

A **Gröbner basis** comes with extensive computational guarantees.

While Gröbner bases have extensive applications in algebraic geometry, we are interested in a special case with vast simplifications.

Reduction modulo a Gröbner basis, in any order, until no more pivots (leading elements) apply is guaranteed to provide a normal form. Normal form equal to 0 implies membership. Equal normal form implies equality (modulo the Gröbner basis).

Persistence module Gröbner bases are Echelon forms

For modules over a field $\Bbbk$, a Gröbner basis is equivalent to a reduced echelon form (REF).

Helpful fact

The ring $\Bbbk[t]$ is sufficiently much like a field – a Gröbner basis of graded modules is also equivalent to a reduced echelon form.

Normal forms, equality, and membership

We shall want to maintain G and R with bases and normal form such that R is always represented by a REF, and G is always reduced with respect to R.

We can avoid redundancy by keeping a basis for G reduced to a REF as well.

This is in particular important since the persistence algorithm itself works with a membership test in the relations module as the fundamental step:

Persistence algorithm, summarized

For each σ:

1. Compute $d\sigma$.
2. Check if $d\sigma \in R$.
3. React to 2 using normal form of $d\sigma$.

Graded Smith normal form

There is a way to compute a Smith Normal Form in a graded sense.

Properties of a Graded Smith Normal Form

- Rows are ordered by increasing degree
- Columns are ordered by increasing degree
- Each row has at most one non-zero entry
- Each column has at most one non-zero entry
- Lower degree entries divide all higher degree entries

Strictly speaking, this is a permutation of the classical Smith Normal Form.

Graded Smith normal form

Core feature:

Computability

We can compute a Graded Smith Normal Form by reducing rows and columns in increasing order of degree. Thus we can compute it compatibly with the gradings present.

Conditions

To do this, we require the coefficients to come from a graded principal ideal domain. $\mathbb{k}[t]$ fulfills this requirement.

SNF and barcodes

Why should we care about Smith normal forms?

Persistence modules and barcodes

A graded Smith normal form of the inclusion map $R \to G$ is the same thing as a barcode of $M = G/R$.

Proof sketch

A graded Smith normal form is a simultaneous basis choice of R and G such that each basis element of R maps onto a $\Bbbk[t]$-multiple of a basis element of G.
This is exactly what produces a barcode: bases for cycles and boundaries such that each boundary basis element kills exactly one cycle basis element.

SNF and barcodes

Why should we care about Smith normal forms?

Persistence modules and barcodes

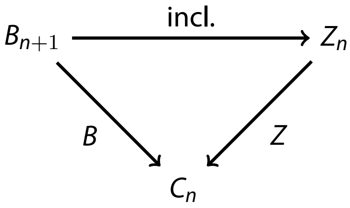
A graded Smith normal form of the inclusion map $R \to G$ is the same thing as a barcode of $M = G/R$.

Proof sketch

A graded Smith normal form is a simultaneous basis choice of R and G such that each basis element of R maps onto a $\Bbbk[t]$-multiple of a basis element of G.
This is exactly what produces a barcode: bases for cycles and boundaries such that each boundary basis element kills exactly one cycle basis element.

Example

$$C_{n+1} \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{n-1}$$



The barcode is the Smith normal form of the map incl.

Algebraic constructions

All classical algebraic constructions are available for persistence modules:

- Image
- Cokernel
- Kernel
- Pullback
- Pushout
- Tensor products
- Symmetric & Exterior powers

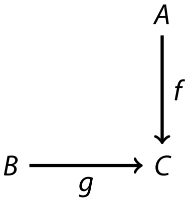
Free pullbacks

One technique that will show up a lot in the subsequent constructions is to compute a kernel of a map between free modules. This is done using a REF computation:

- Reduce the matrix of the map to a REF, tracking the operations performed.
- Operation combinations corresponding to 0-columns are generators of the kernel.

This can compute any pullback of $C \xrightarrow{f} A \xleftarrow{g} B$ where all modules are free as the kernel of $B \oplus C \xrightarrow{(f,-g)} A$.

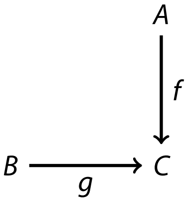
Example



We start out with two maps f, g represented by matrices F, G.
To compute the pullback of f and g, we construct the matrix corresponding to $(f, -g)$:

F	$-G$

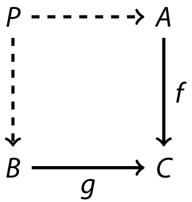
Example



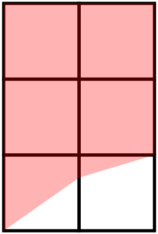
We start out with two maps f, g represented by matrices F, G.
To compute the pullback of f and g, we construct the matrix corresponding to $(f, -g)$:

I	
	I
F	$-G$

Example



We start out with two maps f, g represented by matrices F, G.
To compute the pullback of f and g, we construct the matrix corresponding to $(f, -g)$:



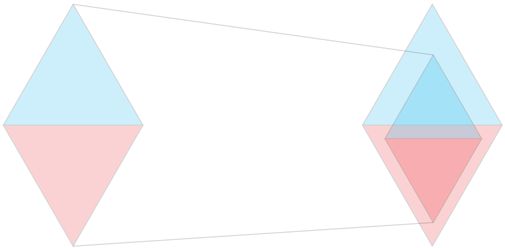
Gaussian column reduction computes the kernel with explicit generators in the top of this matrix.

Illustration

$M = G/R$ and $N = G'/R'$.

Illustration

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.



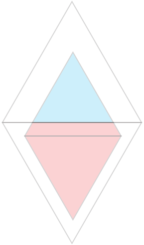
We shall be illustrating the various constructions based on this figure.

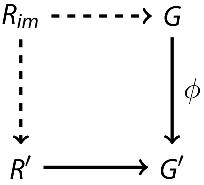
Image

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.

- Compute $\phi(g)$ for each basis element $g \in G$.
- Reduce images modulo the REF for R'.
    - These are the generators for im f.

For the relations, we need to compute a basis for $\phi(G) \cap R'$. This is the pullback of ϕ and the inclusion of R.



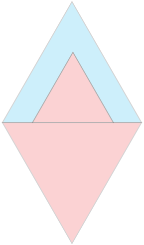


Cokernel

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.

- Compute $\phi(g)$ for each basis element $g \in G$.
- Reduce the basis of G' by the images $\phi(g)$.

This gives you generators for coker f. The relations are those in R together with all the images $\phi(g)$.



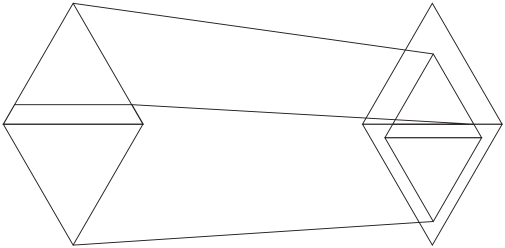
Presentation:

$$G \oplus R' \xrightarrow{\phi \oplus \iota} G'$$

Kernel

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.

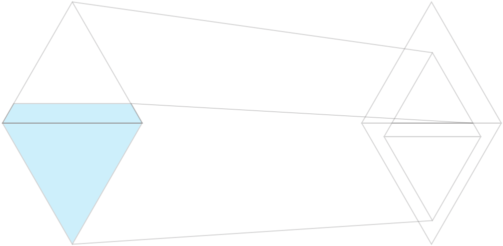
Computing the kernel is a two-step process. One step computes the generators, and the next step computes the relations.

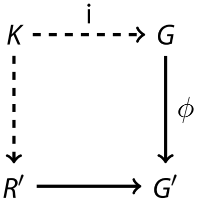


Kernel (Step 1: Generators)

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.

Generators are given as a pullback of ϕ and the inclusion map $R' \to G'$. Projecting onto the first factor, we get an embedding of the kernel generators into G. We will call this module K and the projection map $i : K \to G$.

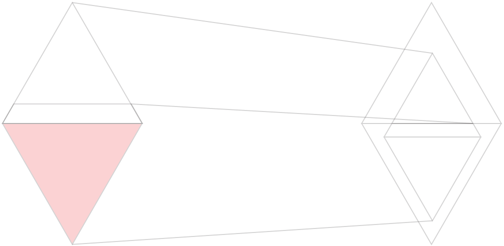


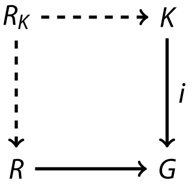


Kernel (Step 2: Relations)

$f : M \to N$ represented by $\phi : G \to G'$ such that $\phi(R) \subset R'$.

Relations of the kernel is given by a pullback of i and the inclusion map $R \to G$. The projection onto K gives the inclusion map of relations into generators for the kernel.





Pushouts and pullbacks

With the tools we developed above, we are able to compute pullbacks and pushouts for persistence modules in general.

Pullback Given $f : A \to C$ and $g : B \to C$, the pullback is $\ker((a, b) \mapsto f(a) - g(b))$.

Pushout Given $f : A \to B$ and $g : A \to C$, the pushout is $\operatorname{coker}(a \mapsto (f(a), g(a))$.

Tensor products

Modules $M = G/R$ and $N = G'/R'$.
Chosen bases B, C for G, R and B', C' for G', R'.

Generators $B \times B'$.

Relations $B \times C' \sqcup C \times B'$.

Redundant relations enumerated by $C \times C'$.

$$\begin{array}{ccccc} & & R \otimes R' & & \\ & \swarrow & & \searrow & \\ R \otimes G' & & & & G \otimes R' \\ & \searrow & & \swarrow & \\ & & G \otimes G' & & \end{array}$$

Tensor products for persistence modules

Generators and relations for $M \otimes N$

The tensor product $M \otimes N$ of $M = G/R$ and $N = G'/R'$, with presentation maps $i : R \to G$ and $j : R' \to G'$:

- Pick bases B for G, B' for G'.
- Tensor product generators have as basis: $B \times B'$.
  We write $b \otimes b'$ for the basis element from (b, b').
- Tensor product relations are generated by $ir \otimes g'$ and $g' \otimes jr'$ for all basis elements $r \in R, r' \in R', g \in G, g' \in G'$.
- We have to pick a minimal representative for basis elements on the shape $ir \otimes jr'$.

Tensor products for barcodes

Everything is simpler if we have presentation maps on Smith normal form (barcodes):

All relations are already $t^k b$ for a basis element b; so picking a representative means checking the terms $t^k b \otimes t^\ell b'$ and picking the smaller of k, ℓ for the new relation.

Symmetric and Exterior Powers

Definition

The symmetric power S^2M is $M \otimes M / \langle a \otimes b \sim b \otimes a \rangle$;
S^nM is repeated application.
The exterior power $\Lambda^2 M$ is $M \otimes M / \langle a \otimes b \sim -b \otimes a \rangle)$;
$\Lambda^n M$ is repeated application.

Generators

S^nM has n-weighted multisets from B_M as basis elements.
$\Lambda^n M$ has cardinality n sets from B_M as basis elements.

Relations

If M was presented with a Smith normal form, a basis element $\{m_1, m_2, \ldots, m_k\}$ is part of a relation for the common ideal generator of all relations killing either of the m_j.

Outline

Torsion chain complexes

One perennial problem in persistence is how to handle *torsion* on a chain level; what if simplices can disappear again?

First approach

Zig-zag homology has provided one solution: vanishing simplices are modelled with inclusions going *the other way*. (de Silva, Morozov, Carlsson)

Our approach

Torsion in the chain complex can be modelled by allowing non-trivial relations in the chain complex.

We note that these approaches lead to different results. In particular, our approach models *relative homology*.

Relative (co)homology

Classically

$$H_*(X, A) = H_*(C_*X/C_*A)$$

Our approach to modeling non-free persistence modules gives us all the tools necessary to work with a chain complex like C_*X/C_*A.

In particular, since ∂ is a map of persistence modules $C_*X/C_*A \to C_*X/C_*A$, we can compute $H_*(X, A)$.

Spectral Sequences

Computation sequence

Spectral sequences of persistence modules take the shape of a sequence of approximations to the final homology:

$$B_0 \subseteq B_1 \subseteq B_2 \subseteq \cdots \subseteq B_\infty \subseteq Z_\infty \subseteq \cdots \subseteq Z_2 \subseteq Z_1 \subseteq Z_0$$

where each Z_k and B_k are kernel/image of a differential

$$d_{k-1} : Z_{k-1}/B_{k-1} \to Z_{k-1}/B_{k-1}$$

To compute the next stage, we need to be able to compute homology when the chain complex has relations.

Original motivation for this research.

Unordered input

Inspired by this view-point, we can adapt the classical persistence algorithm to one that will not require ordered input.

Algorithm: out of order persistence

For each simplex σ:

1. Compute $d\sigma$.
2. Reduce $d\sigma$ modulo all earlier boundaries
3. If $d\sigma$ reduces to 0, then σ starts a new cycle. Loop.
4. Otherwise, $d\sigma$ is a new boundary.
   Find latest simplex τ in reduced $d\sigma$ and construct the pair (τ, σ).
   If τ was already in a pair (τ, ψ), reduce $d\psi$ modulo σ and continue the algorithm for ψ, reducing later boundary chains with this new $d\psi$.

hom-complexes

Tensor products and hom

Recall:

$$\mathrm{hom}(X, Y) = X^* \otimes Y$$

The algorithms here provide formal underpinnings for computing with $\mathrm{hom}(X, Y)$, generating the boundary map on $\mathrm{hom}(X, Y)$, and computing $H_0 \, \mathrm{hom}(X, Y)$.

Work with the hom complex for topological data analysis was investigated by Yi Ding, and later extended by Andrew Tausz.

Work by Morozov – de Silva – V-J has investigated the effect of two different dualizing functors: $\mathrm{hom}(X, \Bbbk)$, $\mathrm{hom}(X, \Bbbk[t])$.

Thank you!

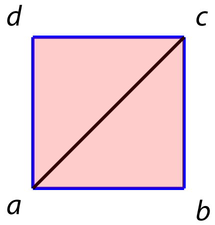
Any questions?

Thank you...

- ICMS and the ATMCS programme committee
- EPSRC and funding from grant EP/G055181/1
- FP7: 288342 (XLike)

Full example: relative homology

Consider the space:



We compute the persistent homology of the space itself, relative the blue edges as they exist at filtration value 1.

Full example: relative homology

The chain complex is:

$$abc \oplus acd \to \frac{ab \oplus ac \oplus ad \oplus bc \oplus cd}{t \cdot ab, t \cdot bc, t \cdot ad, t \cdot cd} \to \frac{a \oplus b \oplus c \oplus d}{t \cdot a, t \cdot b, t \cdot c, t \cdot d}$$

The generators module is free of rank 11.

The relations module is free of rank 8, with each generator in degree 1, and maps the generators to the elements $t \cdot ab, \ldots, t \cdot d$.

Degree here means *filtration* degree, not *topological* dimension.

Full example: relative homology

The boundary map is a morphism of persistence modules; we can compute its kernel. For the generators, we reduce:

	abc	acd	ab	ac	ad	bc	cd	a	b	c	d	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
abc	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
acd	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
ab	1	·	·	·	·	·	·	·	·	·	·	t	·	·	·	·	·	·	·
ac	-1	1	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
ad	·	-1	·	·	·	·	·	·	·	·	·	·	t	·	·	·	·	·	·
bc	1	·	·	·	·	·	·	·	·	·	·	·	·	t	·	·	·	·	·
cd	·	1	·	·	·	·	·	·	·	·	·	·	·	·	t	·	·	·	·
a	·	·	1	1	1	·	·	·	·	·	·	·	·	·	·	t	·	·	·
b	·	·	-1	·	·	1	·	·	·	·	·	·	·	·	·	·	t	·	·
c	·	·	·	-1	·	-1	1	·	·	·	·	·	·	·	·	·	·	t	·
d	·	·	·	·	-1	·	-1	·	·	·	·	·	·	·	·	·	·	·	t

$a, b, c, d, ab - ac + bc, ac - ad + cd, r_6 + t \cdot ab + r_5, r_7 + t \cdot ac + r_5,$
$r_8 + t \cdot ad + r_5, r_4 - t \cdot acd - r2 + r3 - t \cdot abc$ are the resulting generators.

Full example: relative homology

Projecting onto the chain complex gives us the cycle representatives from these computed kernel generators:

$$a, b, c, d, ab - ac + bc, ac - ad + cd, t \cdot ab, t \cdot ac, t \cdot ad, t \cdot abc + t \cdot abc$$

Remains to compute relations for the kernel – the relations for the relative cycles.

Full example: relative homology

To compute the relations module for the kernel, we need to reduce the matrix:

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
abc	·	·	·	·	·	·	·	·	·	t	·	·	·	·	·	·	·	·
acd	·	·	·	·	·	·	·	·	·	t	·	·	·	·	·	·	·	·
ab	·	·	·	·	1	·	t	·	·	·	t	·	·	·	·	·	·	·
ac	·	·	·	·	-1	1	·	t	·	·	·	·	·	·	·	·	·	·
ad	·	·	·	·	·	-1	·	·	t	·	·	t	·	·	·	·	·	·
bc	·	·	·	·	1	·	·	·	·	·	·	·	t	·	·	·	·	·
cd	·	·	·	·	·	1	·	·	·	·	·	·	·	t	·	·	·	·
a	1	·	·	·	·	·	·	·	·	·	·	·	·	·	t	·	·	·
b	·	1	·	·	·	·	·	·	·	·	·	·	·	·	·	t	·	·
c	·	·	1	·	·	·	·	·	·	·	·	·	·	·	·	·	t	·
d	·	·	·	1	·	·	·	·	·	·	·	·	·	·	·	·	·	t

The kernel of this matrix is generated by $r_1 - g_7, r_2 - g_9,$
$r_3 - t \cdot g_5 - g_8 + g_7, r_4 - t \cdot g_6 - g_9 + g_8, r_5 - t \cdot g_1,$
$r_6 - t \cdot g_2, r_7 - t \cdot g_3, r_8 - t \cdot g_4.$

Full example: relative homology

We get the presentation of the relative cycles by combining these two results; projection onto the generators $g_1, \ldots, g_{10}$ gives us the presentation map from relations to generators:

$$\ker \partial = \frac{g_1, g_2, \ldots, g_{10}}{t \cdot g_1, t \cdot g_2, t \cdot g_3, t \cdot g_4, t \cdot g_5 + g_7 - g_8, t \cdot g_6 - g_8 + g_9, g_7, g_9}$$

Full example: relative homology

For the boundary part, we need to compute the free pullback of the map from the cycles to the chains with the actual boundary map. This is, again, a matrix reduction problem:

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	abc	acd	ab	ac	ad	bc	cd	a	b	c	d
abc	·	·	·	·	·	·	·	·	·	t	·	·	·	·	·	·	·	·	·	·	·
acd	·	·	·	·	·	·	·	·	·	t	·	·	·	·	·	·	·	·	·	·	·
ab	·	·	·	·	1	·	t	·	·	·	1	·	·	·	·	·	·	·	·	·	·
ac	·	·	·	·	−1	1	·	t	·	·	−1	1	·	·	·	·	·	·	·	·	·
ad	·	·	·	·	·	−1	·	·	t	·	·	−1	·	·	·	·	·	·	·	·	·
bc	·	·	·	·	1	·	·	·	·	·	1	·	·	·	·	·	·	·	·	·	·
cd	·	·	·	·	·	1	·	·	·	·	·	1	·	·	·	·	·	·	·	·	·
a	1	·	·	·	·	·	·	·	·	·	·	·	1	1	1	·	·	·	·	·	·
b	·	1	·	·	·	·	·	·	·	·	·	·	−1	·	·	1	·	·	·	·	·
c	·	·	1	·	·	·	·	·	·	·	·	·	·	−1	·	−1	1	·	·	·	·
d	·	·	·	1	·	·	·	·	·	·	·	·	·	·	−1	·	−1	·	·	·	·

This matrix has kernel $g_5 - abc, g_6 - acd, ab - g_1 + g_2, ac - g_1 + g_3,$ $ad - g_1 + g_4, bc - g_2 + g_3, cd - g_3 + g_4$.

Full example: relative homology

Projecting the resulting kernel back into the kernel module, we get the relations induced from taking the cokernel of the boundary map as $g_5, g_6, g_1 - g_2, g_1 - g_3, g_1 - g_4, g_2 - g_3, g_3 - g_4$. Adding these to our known relations, we get a matrix for the presentation map:

$$
\begin{array}{c} \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \\ g_8 \\ g_9 \\ g_{10} \end{array}
\begin{array}{c}
\begin{array}{ccccccccccccccc} \rho_1 & \rho_2 & \rho_3 & \rho_4 & \rho_5 & \rho_6 & \rho_7 & \rho_8 & \rho_9 & \rho_{10} & \rho_{11} & \rho_{12} & \rho_{13} & \rho_{14} & \rho_{15} \end{array} \\
\begin{pmatrix}
t & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot \\
\cdot & t & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & \cdot & \cdot & 1 & \cdot \\
\cdot & \cdot & t & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & \cdot & -1 & 1 \\
\cdot & \cdot & \cdot & t & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & \cdot & -1 \\
\cdot & \cdot & \cdot & \cdot & t & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & t & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{pmatrix}
\end{array}
$$

Full example: relative homology

Computing a Smith normal form of this presentation map, we get:

$$\begin{array}{l} g1 \\ g2+g1 \\ g3+g2+g1 \\ g4+g3+g2+g1 \\ g5 \\ g6 \\ g7 \\ g8 \\ g9 \\ g10 \end{array} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & t & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

where we have chosen to ignore the basis change in the relations module for clarity. This gives us the non-trivial intervals $(0,1) : a+b+c+d$, $(1,\infty) : abc+acd$, corresponding to the space chosen.