Resolutions

Release 9.7

The Sage Development Team

Jul 21, 2024

CONTENTS

Free and graded resolutions are tools for commutative algebra and algebraic geometry.

CHAPTER

ONE

FREE RESOLUTIONS

Let R be a commutative ring. A finite free resolution of an R-module M is a chain complex of free R-modules

$$0 \xleftarrow{d_0} R^{n_1} \xleftarrow{d_1} R^{n_1} \xleftarrow{d_2} \cdots \xleftarrow{d_k} R^{n_k} \xleftarrow{d_{k+1}} 0$$

terminating with a zero module at the end that is exact (all homology groups are zero) such that the image of d_1 is M.

EXAMPLES:

```
sage: from sage.homology.free_resolution import FreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: m = matrix(S, 1, [z^2 - y*w, y*z - x*w, y^2 - x*z]).transpose()
sage: r = FreeResolution(m, name='S')
sage: r
S^1 <-- S^3 <-- S^2 <-- 0

sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = FreeResolution(I)
sage: r
S^1 <-- S^3 <-- S^2 <-- 0
```

```
sage: from sage.homology.graded_resolution import GradedFreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = GradedFreeResolution(I)
sage: r
S(0) <-- S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
```

An example of a minimal free resolution from [CLO2005]:

```
sage: R.<x,y,z,w> = QQ[]
sage: I = R.ideal([y*z - x*w, y^3 - x^2*z, x*z^2 - y^2*w, z^3 - y*w^2])
sage: r = FreeResolution(I)
sage: r
S^1 <-- S^4 <-- S^4 <-- S^1 <-- 0
sage: len(r)
3
sage: r.matrix(2)
[-z^2 -x*z  y*w -y^2]
[   y    0   -x    0]
[  -w    y    z    x]
[   0    w    0    z]
```

AUTHORS:

- Kwankyu Lee (2022-05-13): initial version

class `sage.homology.free_resolution.`**`FreeResolution`**(*module*, *name='S'*, *algorithm='heuristic'*)

Bases: *`sage.homology.free_resolution.FreeResolution_generic`*

Minimal free resolutions of ideals or submodules of free modules of multivariate polynomial rings.

INPUT:

- `module` – a submodule of a free module M of rank n over S or an ideal of a multi-variate polynomial ring
- `name` – a string; name of the base ring
- `algorithm` – Singular algorithm to compute a resolution of `ideal`

OUTPUT: a minimal free resolution of the ideal

If `module` is an ideal of S, it is considered as a submodule of a free module of rank 1 over S.

The available algorithms and the corresponding Singular commands are shown below:

algorithm	Singular commands
`minimal`	`mres(ideal)`
`shreyer`	`minres(sres(std(ideal)))`
`standard`	`minres(nres(std(ideal)))`
`heuristic`	`minres(res(std(ideal)))`

EXAMPLES:

```
sage: from sage.homology.free_resolution import FreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = FreeResolution(I)
sage: r
S^1 <-- S^3 <-- S^2 <-- 0
sage: len(r)
2
```

```
sage: FreeResolution(I, algorithm='minimal')
S^1 <-- S^3 <-- S^2 <-- 0
sage: FreeResolution(I, algorithm='shreyer')
S^1 <-- S^3 <-- S^2 <-- 0
sage: FreeResolution(I, algorithm='standard')
S^1 <-- S^3 <-- S^2 <-- 0
sage: FreeResolution(I, algorithm='heuristic')
S^1 <-- S^3 <-- S^2 <-- 0
```

We can also construct a resolution by passing in a matrix defining the initial differential:

```
sage: m = matrix(S, 1, [z^2 - y*w, y*z - x*w, y^2 - x*z]).transpose()
sage: r = FreeResolution(m, name='S')
sage: r
S^1 <-- S^3 <-- S^2 <-- 0
sage: r.matrix(1)
[z^2 - y*w y*z - x*w y^2 - x*z]
```

An additional construction is using a submodule of a free module:

```
sage: M = m.image()
sage: r = FreeResolution(M, name='S')
sage: r
S^1 <-- S^3 <-- S^2 <-- 0
```

A nonhomogeneous ideal:

```
sage: I = S.ideal([z^2 - y*w, y*z - x*w, y^2 - x])
sage: R = FreeResolution(I)
sage: R
S^1 <-- S^3 <-- S^3 <-- S^1 <-- 0
sage: R.matrix(2)
[ y*z - x*w    y^2 - x          0]
[-z^2 + y*w          0    y^2 - x]
[         0 -z^2 + y*w -y*z + x*w]
sage: R.matrix(3)
[   y^2 - x]
[-y*z + x*w]
[ z^2 - y*w]
```

class sage.homology.free_resolution.**FreeResolution_generic**(*base_ring*, *name='F'*)

Bases: sage.structure.sage_object.SageObject

Generic base class of finite free resolutions.

A subclass must provide a _maps attribute that contains a list of the maps defining the resolution.

The matrix at index i in the list defines the differential map from $(i+1)$-th free module to the i-th free module over the base ring by multiplication on the left. The number of matrices in the list is the length of the resolution. The number of rows and columns of the matrices define the ranks of the free modules in the resolution.

Note that the first matrix in the list defines the differential map at homological index 1.

A subclass can define _initial_differential attribute that contains the 0-th differential map whose codomain is the target of the free resolution.

EXAMPLES:

```
sage: from sage.homology.free_resolution import FreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = FreeResolution(I)
sage: r.differential(0)
Coercion map:
  From: Ambient free module of rank 1 over the integral domain
Multivariate Polynomial Ring in x, y, z, w over Rational Field
  To:   Quotient module by Submodule of Ambient free module of rank 1
over the integral domain Multivariate Polynomial Ring in x, y, z, w over Rational␣
↪Field
Generated by the rows of the matrix:
[-z^2 + y*w]
[ y*z - x*w]
[-y^2 + x*z]
```

chain_complex()

Return this resolution as a chain complex.

A chain complex in Sage has its own useful methods.

EXAMPLES:

```
sage: from sage.homology.graded_resolution import GradedFreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = GradedFreeResolution(I)
sage: unicode_art(r.chain_complex())
                                                    ⎛-y   x⎞
                                                    ⎜      ⎟
           (z^2 - y*w y*z - x*w y^2 - x*z)          ⎜ z  -y⎟
                                                    ⎝-w   z⎠
 0 <- C_0 <---------- C_1 <-- C_2 <- 0
```

differential(*i*)

Return the matrix representing the `i`-th differential map.

INPUT:

- `i` – a positive integer

EXAMPLES:

```
sage: from sage.homology.graded_resolution import GradedFreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = GradedFreeResolution(I)
sage: r
S(0) <-- S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
sage: r.differential(3)
Free module morphism defined by the matrix
[]
Domain: Ambient free module of rank 0 over the integral domain
Multivariate Polynomial Ring in x, y, z, w over Rational Field
Codomain: Ambient free module of rank 2 over the integral domain
Multivariate Polynomial Ring in x, y, z, w over Rational Field
sage: r.differential(2)
Free module morphism defined as left-multiplication by the matrix
[-y  x]
[ z -y]
[-w  z]
Domain: Ambient free module of rank 2 over the integral domain
Multivariate Polynomial Ring in x, y, z, w over Rational Field
Codomain: Ambient free module of rank 3 over the integral domain
Multivariate Polynomial Ring in x, y, z, w over Rational Field
sage: r.differential(1)
Free module morphism defined as left-multiplication by the matrix
[z^2 - y*w y*z - x*w y^2 - x*z]
Domain: Ambient free module of rank 3 over the integral domain
Multivariate Polynomial Ring in x, y, z, w over Rational Field
Codomain: Ambient free module of rank 1 over the integral domain
Multivariate Polynomial Ring in x, y, z, w over Rational Field
sage: r.differential(0)
Coercion map:
```

(continues on next page)

(continued from previous page)

```
  From: Ambient free module of rank 1 over the integral domain
Multivariate Polynomial Ring in x, y, z, w over Rational Field
  To:   Quotient module by Submodule of Ambient free module of rank 1 over the␣
↪integral domain
Multivariate Polynomial Ring in x, y, z, w over Rational Field
Generated by the rows of the matrix:
[-z^2 + y*w]
[ y*z - x*w]
[-y^2 + x*z]
```

matrix(*i*)

Return the matrix representing the `i`-th differential map.

INPUT:

- `i` – a positive integer

EXAMPLES:

```
sage: from sage.homology.graded_resolution import GradedFreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = GradedFreeResolution(I)
sage: r
S(0) <-- S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
sage: r.matrix(3)
[]
sage: r.matrix(2)
[-y  x]
[ z -y]
[-w  z]
sage: r.matrix(1)
[z^2 - y*w y*z - x*w y^2 - x*z]
```

target()

Return the codomain of the `0`-th differential map.

EXAMPLES:

```
sage: from sage.homology.graded_resolution import GradedFreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = GradedFreeResolution(I)
sage: r
S(0) <-- S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
sage: r.target()
Quotient module by Submodule of Ambient free module of rank 1 over the integral␣
↪domain
Multivariate Polynomial Ring in x, y, z, w over Rational Field
Generated by the rows of the matrix:
[-z^2 + y*w]
[ y*z - x*w]
[-y^2 + x*z]
```

CHAPTER

TWO

GRADED FREE RESOLUTIONS

Let R be a commutative ring. A graded free resolution of a graded R-module M is a `free resolution` such that all maps are homogeneous module homomorphisms.

EXAMPLES:

```
sage: from sage.homology.graded_resolution import GradedFreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = GradedFreeResolution(I, algorithm='minimal')
sage: r
S(0) <-- S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
sage: GradedFreeResolution(I, algorithm='shreyer')
S(0) <-- S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
sage: GradedFreeResolution(I, algorithm='standard')
S(0) <-- S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
sage: GradedFreeResolution(I, algorithm='heuristic')
S(0) <-- S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
```

```
sage: d = r.differential(2)
sage: d
Free module morphism defined as left-multiplication by the matrix
[ y  x]
[-z -y]
[ w  z]
Domain: Ambient free module of rank 2 over the integral domain Multivariate Polynomial␣
↪Ring
in x, y, z, w over Rational Field
Codomain: Ambient free module of rank 3 over the integral domain Multivariate Polynomial␣
↪Ring
in x, y, z, w over Rational Field
sage: d.image()
Submodule of Ambient free module of rank 3 over the integral domain Multivariate␣
↪Polynomial Ring
in x, y, z, w over Rational Field
Generated by the rows of the matrix:
[ y -z  w]
[ x -y  z]
sage: m = d.image()
sage: GradedFreeResolution(m, shifts=(2,2,2))
S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
```

An example of multigraded resolution from Example 9.1 of [MilStu2005]:

```
sage: R.<s,t> = QQ[]
sage: S.<a,b,c,d> = QQ[]
sage: phi = S.hom([s, s*t, s*t^2, s*t^3])
sage: I = phi.kernel(); I
Ideal (c^2 - b*d, b*c - a*d, b^2 - a*c) of Multivariate Polynomial Ring in a, b, c, d␣
↪over Rational Field
sage: P3 = ProjectiveSpace(S)
sage: C = P3.subscheme(I)  # twisted cubic curve
sage: r = GradedFreeResolution(I, degrees=[(1,0), (1,1), (1,2), (1,3)])
sage: r
S(0) <-- S(-(2, 4))⊕S(-(2, 3))⊕S(-(2, 2)) <-- S(-(3, 5))⊕S(-(3, 4)) <-- 0
sage: r.K_polynomial(names='s,t')
s^3*t^5 + s^3*t^4 - s^2*t^4 - s^2*t^3 - s^2*t^2 + 1
```

AUTHORS:

- Kwankyu Lee (2022-05): initial version

class sage.homology.graded_resolution.**GradedFreeResolution**(*module*, *degrees=None*, *shifts=None*, *name='S'*, *algorithm='heuristic'*)

Bases: *sage.homology.free_resolution.FreeResolution*

Graded free resolutions of ideals of multivariate polynomial rings.

INPUT:

- `module` – a homogeneous submodule of a free module M of rank n over S or a homogeneous ideal of a multivariate polynomial ring S
- `degrees` – (default: a list with all entries 1) a list of integers or integer vectors giving degrees of variables of S
- `shifts` – a list of integers or integer vectors giving shifts of degrees of n summands of the free module M; this is a list of zero degrees of length n by default
- `name` – a string; name of the base ring
- `algorithm` – Singular algorithm to compute a resolution of `ideal`

If `module` is an ideal of S, it is considered as a submodule of a free module of rank 1 over S.

The degrees given to the variables of S are integers or integer vectors of the same length. In the latter case, S is said to be multigraded, and the resolution is a multigraded free resolution. The standard grading where all variables have degree 1 is used if the degrees are not specified.

A summand of the graded free module M is a shifted (or twisted) module of rank one over S, denoted $S(-d)$ with shift d.

The computation of the resolution is done by using `libSingular`. Different Singular algorithms can be chosen for best performance.

OUTPUT: a graded minimal free resolution of `ideal`

The available algorithms and the corresponding Singular commands are shown below:

algorithm	Singular commands
minimal	mres(ideal)
shreyer	minres(sres(std(ideal)))
standard	minres(nres(std(ideal)))
heuristic	minres(res(std(ideal)))

Warning: This does not check that the module is homogeneous.

EXAMPLES:

```
sage: from sage.homology.graded_resolution import GradedFreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = GradedFreeResolution(I)
sage: r
S(0) <-- S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
sage: len(r)
2

sage: I = S.ideal([z^2 - y*w, y*z - x*w, y - x])
sage: I.is_homogeneous()
True
sage: R = GradedFreeResolution(I)
sage: R
S(0) <-- S(-1)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3)⊕S(-4) <-- S(-5) <-- 0
```

K_polynomial(*names=None*)

Return the K-polynomial of this resolution.

INPUT:

- names – (optional) a string of names of the variables of the K-polynomial

EXAMPLES:

```
sage: from sage.homology.graded_resolution import GradedFreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = GradedFreeResolution(I)
sage: r.K_polynomial()
2*t^3 - 3*t^2 + 1
```

betti(*i*, *a=None*)

Return the i-th Betti number in degree a.

INPUT:

- i – nonnegative integer
- a – a degree; if None, return Betti numbers in all degrees

EXAMPLES:

```
sage: from sage.homology.graded_resolution import GradedFreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = GradedFreeResolution(I)
sage: r.betti(0)
{0: 1}
sage: r.betti(1)
{2: 3}
sage: r.betti(2)
{3: 2}
sage: r.betti(1, 0)
0
sage: r.betti(1, 1)
0
sage: r.betti(1, 2)
3
```

shifts(*i*)

Return the shifts of `self`.

EXAMPLES:

```
sage: from sage.homology.graded_resolution import GradedFreeResolution
sage: S.<x,y,z,w> = PolynomialRing(QQ)
sage: I = S.ideal([y*w - z^2, -x*w + y*z, x*z - y^2])
sage: r = GradedFreeResolution(I)
sage: r.shifts(0)
[0]
sage: r.shifts(1)
[2, 2, 2]
sage: r.shifts(2)
[3, 3]
sage: r.shifts(3)
[]
```

CHAPTER

THREE

INDICES AND TABLES

- Index
- Module Index
- Search Page

PYTHON MODULE INDEX

INDEX