
Resolutions

Release 10.3

The Sage Development Team

Jul 03, 2024

CONTENTS

1 Free resolutions	3
2 Graded free resolutions	9
3 Indices and Tables	15
Python Module Index	17
Index	19

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

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

$$R^{n_0} \xleftarrow{d_1} R^{n_1} \xleftarrow{d_2} \dots \xleftarrow{d_k} R^{n_k} \xleftarrow{d_{k+1}} 0$$

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'); 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 = I.free_resolution(); r
S^1 <-- S^3 <-- S^2 <-- 0
```

```
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 = I.graded_free_resolution(); 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 = I.free_resolution(); 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
- Travis Scrimshaw (2022-08-23): refactored for free module inputs


```

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 = I.graded_free_resolution()
sage: r
S(0) <-- S(-2)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3) <-- 0
sage: r.differential(3)
Free module morphism defined as left-multiplication 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:
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: 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 = I.graded_free_resolution(); 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]

```

(continues on next page)

(continued from previous page)

```
[-w z]
sage: r.matrix(1)
[z^2 - y*w y*z - x*w y^2 - x*z]
```

```
class sage.homology.free_resolution.FiniteFreeResolution_free_module(module,
                                                                    name='S',
                                                                    **kws)
```

Bases: *FiniteFreeResolution*

Free resolutions of a free module.

INPUT:

- *module* – a free module or ideal over a PID
- *name* – the name of the base ring

EXAMPLES:

```
sage: R.<x> = QQ[]
sage: M = R^3
sage: v = M([x^2, 2*x^2, 3*x^2])
sage: w = M([0, x, 2*x])
sage: S = M.submodule([v, w]); S
Free module of degree 3 and rank 2 over
Univariate Polynomial Ring in x over Rational Field
Echelon basis matrix:
[ x^2 2*x^2 3*x^2]
[ 0 x 2*x]
sage: res = S.free_resolution(); res
S^3 <-- S^2 <-- 0
sage: ascii_art(res.chain_complex())
      [ x^2 0]
      [2*x^2 x]
      [3*x^2 2*x]
0 <-- C_0 <----- C_1 <-- 0

sage: R.<x> = PolynomialRing(QQ)
sage: I = R.ideal([x^4 + 3*x^2 + 2])
sage: res = I.free_resolution(); res
S^1 <-- S^1 <-- 0
```

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

Bases: *FiniteFreeResolution*

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

INPUT:

- *module* – a submodule of a free module M of rank n over S or an ideal of a multi-variate polynomial ring
- *name* – string (optional); name of the base ring
- *algorithm* – (default: 'heuristic') 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	<code>mres(ideal)</code>
shreyer	<code>minres(sres(std(ideal)))</code>
standard	<code>minres(nres(std(ideal)))</code>
heuristic	<code>minres(res(std(ideal)))</code>

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); 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'); 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'); 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); 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]
```

(continues on next page)

(continued from previous page)

```
[-y*z + x*w]
[ z^2 - y*w]
```

class sage.homology.free_resolution.**FreeResolution** (*module*, *name='S'*, ***kws*)

Bases: SageObject

A free resolution.

Let R be a commutative ring. A *free resolution* of an R -module M is a (possibly infinite) chain complex of free R -modules

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

that is exact (all homology groups are zero) such that the image of d_1 is M .

differential (*i*)

Return the i -th differential map.

INPUT:

- i – a positive integer

target ()

Return the codomain of the 0-th differential map.

The codomain of the 0-th differential map is the cokernel of the first differential map.

EXAMPLES:

```
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 = I.graded_free_resolution()
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]
```

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: 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 = I.graded_free_resolution(algorithm='minimal')
sage: r
S(0) <-- S(-2)⊗S(-2)⊗S(-2) <-- S(-3)⊗S(-3) <-- 0
sage: I.graded_free_resolution(algorithm='shreyer')
S(0) <-- S(-2)⊗S(-2)⊗S(-2) <-- S(-3)⊗S(-3) <-- 0
sage: I.graded_free_resolution(algorithm='standard')
S(0) <-- S(-2)⊗S(-2)⊗S(-2) <-- S(-3)⊗S(-3) <-- 0
sage: I.graded_free_resolution(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: m.graded_free_resolution(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
↳needs sage.rings.function_field
```

(continues on next page)

(continued from previous page)

```

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 = I.graded_free_resolution(degrees=[(1,0), (1,1), (1,2), (1,3)])
sage: r
S((0, 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
- Travis Scrimshaw (2022-08-23): refactored for free module inputs

```

class sage.homology.graded_resolution.GradedFiniteFreeResolution(module,
                                                                    degrees=None,
                                                                    shifts=None,
                                                                    name='S', **kwds)

```

Bases: *FiniteFreeResolution*

Graded finite free resolutions.

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

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

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: 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 = I.graded_free_resolution()
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: 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 = I.graded_free_resolution()
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: 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 = I.graded_free_resolution()
sage: r.shifts(0)
[0]
sage: r.shifts(1)
[2, 2, 2]
sage: r.shifts(2)
[3, 3]
sage: r.shifts(3)
[]
```

```
class sage.homology.graded_resolution.GradedFiniteFreeResolution_free_module (mod-
                                                                    ular,
                                                                    de-
                                                                    grees=None,
                                                                    *args,
                                                                    **kws)
```

Bases: *GradedFiniteFreeResolution*, *FiniteFreeResolution_free_module*

Graded free resolution of free modules.

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

EXAMPLES:

```
sage: from sage.homology.free_resolution import FreeResolution
sage: R.<x> = QQ[]
sage: M = matrix([[x^3, 3*x^3, 5*x^3],
```

(continues on next page)

(continued from previous page)

```

.....:          [0, x, 2*x]])
sage: res = FreeResolution(M, graded=True); res
S(0)⊕S(0)⊕S(0) <-- S(-3)⊕S(-1) <-- 0

```

```

class sage.homology.graded_resolution.GradedFiniteFreeResolution_singular (mod-
                                                                    ule,
                                                                    de-
                                                                    grees=None,
                                                                    shifts=None,
                                                                    name='S',
                                                                    algo-
                                                                    rithm='heuris-
                                                                    tic',
                                                                    **kws)

```

Bases: *GradedFiniteFreeResolution, FiniteFreeResolution_singular*

Graded free resolutions of submodules and ideals of multivariate polynomial rings implemented using Singular.

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	<code>mres(ideal)</code>
shreyer	<code>minres(sres(std(ideal)))</code>
standard	<code>minres(nres(std(ideal)))</code>
heuristic	<code>minres(res(std(ideal)))</code>

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

EXAMPLES:

```
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 = I.graded_free_resolution(); 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 = I.graded_free_resolution(); r
S(0) <-- S(-1)⊕S(-2)⊕S(-2) <-- S(-3)⊕S(-3)⊕S(-4) <-- S(-5) <-- 0
```


INDICES AND TABLES

- [Index](#)
- [Module Index](#)
- [Search Page](#)

PYTHON MODULE INDEX

h

`sage.homology.free_resolution`, 3
`sage.homology.graded_resolution`, 9

B

`betti()` (*sage.homology.graded_resolution.GradedFiniteFreeResolution method*), 10

C

`chain_complex()` (*sage.homology.free_resolution.FiniteFreeResolution method*), 4

D

`differential()` (*sage.homology.free_resolution.FiniteFreeResolution method*), 4

`differential()` (*sage.homology.free_resolution.FreeResolution method*), 8

F

`FiniteFreeResolution` (*class in sage.homology.free_resolution*), 3

`FiniteFreeResolution_free_module` (*class in sage.homology.free_resolution*), 6

`FiniteFreeResolution_singular` (*class in sage.homology.free_resolution*), 6

`FreeResolution` (*class in sage.homology.free_resolution*), 8

G

`GradedFiniteFreeResolution` (*class in sage.homology.graded_resolution*), 10

`GradedFiniteFreeResolution_free_module` (*class in sage.homology.graded_resolution*), 11

`GradedFiniteFreeResolution_singular` (*class in sage.homology.graded_resolution*), 12

K

`K_polynomial()` (*sage.homology.graded_resolution.GradedFiniteFreeResolution method*), 10

M

`matrix()` (*sage.homology.free_resolution.FiniteFreeResolution method*), 5

module

`sage.homology.free_resolution`, 3

`sage.homology.graded_resolution`, 9

S

`sage.homology.free_resolution`
module, 3

`sage.homology.graded_resolution`
module, 9

`shifts()` (*sage.homology.graded_resolution.GradedFiniteFreeResolution method*), 11

T

`target()` (*sage.homology.free_resolution.FreeResolution method*), 8