Hodge diamond cutter

Release v1.1

Pieter Belmans

Aug 05, 2021

CONTENTS:

A tool to work with Hodge diamonds, comes with many varieties and constructions built into it.

Hodge diamonds encode the Hodge numbers of a variety, and provide interesting information about its structure. They provide a numerical incarnation of many operations one can perform in algebraic geometry, such as blowups, projective bundles, products. They are also computed for many more specific constructions such as certain moduli spaces of sheaves, or flag varieties, ...

These Hodge numbers are defined as the dimensions of the sheaf cohomology of exterior powers of the cotangent bundle, i.e.

$$\mathrm{h}^{p,q}(X) = \dim \mathrm{H}^q(X, \Omega_X^p)$$

Here p and q range from 0 to $n=\dim X$. These numbers satisfy additional symmetry properties:

- Hodge symmetry: $\mathrm{h}^{p,q}(X)=\mathrm{h}^{q,p}(X)$
- Serre duality: $\mathrm{h}^{p,q}(X)=\mathrm{h}^{n-p,n-q}(X)$

Because of these symmetries they are usually displayed as a diamond (it's really just a square tilted 45 degrees), so that for a surface it would be:

```
                h^{2,2}
        h^{2,1}         h^{1,2}
h^{2,0}         h^{1,1}         h^{0,2}
        h^{1,0}         h^{0,1}
                h^{0,0}
```

One of their famous applications is the mirror symmetry prediction that every Calabi-Yau 3-fold has a mirror Calabi-Yau threefold, which should imply that their Hodge diamonds are transpositions. The first instance of this is the quintic 3-fold and its mirror, whose Hodge diamonds are:

```
              1
         0         0
    0         1         0
1       101       101       1
    0         1         0
         0         0
              1
```

and:

```
              1
         0         0
    0       101       0
1       1         1       1
    0       101       0
         0         0
              1
```

The following are some very basic examples of operations and constructions one can use within the Hodge diamond cutter. To get started we do:

```
sage: load("diamond.py")
```

after starting Sage.

Pretty print the Hodge diamond of a genus 2 curve:

```
sage: X = HodgeDiamond.from_matrix([[1, 2], [2, 1]])
sage: print(X)
      1
  2       2
      1
```

Compute the Euler characteristic of the product of X with itself:

```
sage: print((X*X).euler())
4
```

Pretty print the Hodge diamond of the Hilbert square of a K3 surface:

```
sage: S = HodgeDiamond.from_matrix([[1, 0, 1], [0, 20, 0], [1, 0, 1]])
sage: print(hilbn(S, 2))
                  1
               0        0
          1        21        1
     0        0        0        0
  1       21       232       21       1
     0        0        0        0
          1        21        1
               0        0
                    1
```

There are many varieties built in, e.g. the previously defined K3 surface can be compared to the built-in one:

```
sage: print(S == K3())
True
```

AUTHORS:

- Pieter Belmans (2019-01-27): initial version
- Pieter Belmans (2020-06-16): the version which got assigned a DOI
- Pieter Belmans (2021-08-04): various additions, added unit tests and proper documentation

CHAPTER

ONE

HODGE DIAMONDS

class diamond.**HodgeDiamond**(*m*)

This class implements some methods to work with Hodge diamonds.

R = Multivariate Polynomial Ring in x, y over Integer Ring

polynomial ring used internally for the Hodge-Poincaré polynomial and can be used externally to create new polynomials (and thus diamonds)

x = x

variables in the polynomial ring for Hodge-Poincaré polynomials

y = y

variables in the polynomial ring for Hodge-Poincaré polynomials

__init__(*m*)

Constructor for a Hodge diamond if you know what you are doing

This just uses the given matrix. It is probably advised to use the class methods

- *HodgeDiamond.from_matrix()*
- *HodgeDiamond.from_polynomial()*

in most cases though.

classmethod from_matrix(*m*, *from_variety=False*)

Construct a Hodge diamond from a matrix

INPUT:

- `m` – square integer matrix representing a Hodge diamond
- `from_variety` (default: False) – whether a check should be performed that it comes from a variety

EXAMPLES:

Hodge diamond of a K3 surface:

```
sage: load("diamond.py")
sage: S = HodgeDiamond.from_matrix([[1, 0, 1], [0, 20, 0], [1, 0, 1]])
sage: S == K3()
True
```

The following fails as the lack of symmetry prevents a geometric origin:

```
sage: HodgeDiamond.from_matrix([[1, 2], [0, 1]], from_variety=True)
Traceback (most recent call last):
...
AssertionError: The matrix does not
```

(continues on next page)

(continued from previous page)

```
                satisfy the conditions satisfied by the Hodge diamond of a
                smooth projective variety.
```

classmethod from_polynomial (*f*, *from_variety=False*)

Construct a Hodge diamond from a Hodge–Poincaré polynomial

INPUT:

- `f` – an integer polynomial in the ring `HodgeDiamond.R` representing the Hodge–Poincaré polynomial
- `from_variety` (default: False) – whether a check should be performed that it comes from a variety

EXAMPLES:

Hodge diamond of a K3 surface:

```
sage: load("diamond.py")
sage: (x, y) = (HodgeDiamond.x, HodgeDiamond.y)
sage: S = HodgeDiamond.from_polynomial(1 + x**2 + 20*x*y + y**2 + x**2 * y**2)
sage: S == K3()
True
```

The following fails as the lack of symmetry prevents a geometric origin:

```
sage: HodgeDiamond.from_polynomial(1 + x, from_variety=True)
Traceback (most recent call last):
...
AssertionError: The matrix does not
                satisfy the conditions satisfied by the Hodge diamond of a
                smooth projective variety.
```

property polynomial

The Hodge–Poincaré polynomial describing the Hodge diamond

Getter returns the Hodge–Poincaré polynomial

Setter sets the Hodge diamond using a Hodge–Poincaré polynomial

Type element of *HodgeDiamond.R*

EXAMPLES:

The Hodge–Poincaré polynomial of a K3 surface:

```
sage: load("diamond.py")
sage: print(K3().polynomial)
x^2*y^2 + x^2 + 20*x*y + y^2 + 1
```

Modifying the Hodge diamond of the projective plane:

```
sage: X = Pn(2)
sage: X.polynomial = X.polynomial + X.x * X.y
sage: print(X)
          1
      0       0
  0       2       0
      0       0
          1
```

property matrix
The matrix describing the Hodge diamond

Getter returns the matrix

Setter sets the Hodge diamond using a matrix

Type square matrix of integers

__eq__(*other*)
Check whether two Hodge diamonds are equal

This compares the Hodge polynomials, not the possibly oversized matrices describing the Hodge diamond.

EXAMPLES:

A quartic surface is a K3 surface:

```
sage: load("diamond.py")
sage: K3() == hypersurface(4, 2)
True
```

__ne__(*other*)
Check whether two Hodge diamonds are not equal

EXAMPLES:

The projective line is not a genus 2 curve:

```
sage: load("diamond.py")
sage: Pn(1) != curve(2)
True
```

The point is not the Lefschetz class:

```
sage: point() != lefschetz()
True
```

__add__(*other*)
Add two Hodge diamonds together

This corresponds to taking the disjoint union of varieties, or the direct sum of the Hodge structure.

EXAMPLES:

Hodge diamond of the projective line is the sum of that of a point and the Lefschetz diamond:

```
sage: load("diamond.py")
sage: Pn(1) == point() + lefschetz()
True
```

Adding zero doesn't do anything:

```
sage: K3() + zero() == K3()
True
```

__radd__(*other*)
Add two Hodge diamonds together

This is used when sum()'ing for instance.

__sub__(*other*)
Subtract two Hodge diamonds

EXAMPLES:

Hodge diamond of the projective line is the sum of that of a point and the Lefschetz diamond, but now we check it the other way around:

```
sage: load("diamond.py")
sage: Pn(1) - point() == lefschetz()
True
```

__mul__(*other*)
Multiply two Hodge diamonds

This corresponds to taking the product of two varieties.

EXAMPLES:

The quadric surface is the product of two projective lines:

```
sage: load("diamond.py")
sage: Pn(1) * Pn(1) == hypersurface(2, 2)
True
```

The product is commutative:

```
sage: K3() * curve(5) == curve(5) * K3()
True
```

The point is the unit:

```
sage: K3() * point() == point() * K3() == K3()
True
```

__rmul__(*factor*)
Multiply a Hodge diamond with a factor

This corresponds to iterated addition.

INPUT:

- `factor` – coefficient for the iterated addition

EXAMPLES:

The disjoint union of 2 K3 surfaces in two ways:

sage: load("diamond.py") sage: 2*K3() == K3() + K3() True

__pow__(*power*)
Raise a Hodge diamond to a power

This corresponds to iterated multiplication.

INPUT:

- `power` – exponent for the iterated multiplication

EXAMPLES:

The product of 2 K3 surfaces in two ways:

```
sage: load("diamond.py")
sage: K3()**2 == K3()*K3()
True
```

__call__(*i*, *y=None*)

The calling operator either does a Lefschetz twist, or an evaluation

If one parameter is present, then twist by a power of the Lefschetz Hodge diamond. If two parameters are present, then evaluate the Hodge-Poincaré polynomial

Negative values are allowed to untwist, up to the appropriate power.

INPUT:

- `i` – integer denoting the power of the Lefschetz class, or value for the first variable
- `y` – value of the second variable (default: `None`), if it is non-zero then `i` is reinterpreted as the value of the first variable

EXAMPLES:

The Lefschetz class is by definition the twist of the point:

```
sage: load("diamond.py")
sage: lefschetz() == point()(1)
True
```

We can reconstruct projective space as a sum of twists of the point:

```
sage: Pn(10) == sum([point()(i) for i in range(11)])
True
```

If we supply two parameters we are evaluation the Hodge-Poincare polynomial, e.g. to find the Euler characteristic:

```
sage: Pn(10)(1, 1) == 11
True
```

__getitem__(*index*)

Get (p, q)th entry of Hodge diamond or the ith row of the Hodge diamond

__repr__()

Output diagnostic information

This is a one-line string giving some basic information about the Hodge diamond. You'll see this when you just evaluate something which returns a Hodge diamond. To see something more useful, you'll likely want to use

- *HodgeDiamond.__str__()* via *print*
- *HodgeDiamond.pprint()*
- *HodgeDiamond.polynomial()*

EXAMPLES:

The projective line:

```
sage: load("diamond.py")
sage: Pn(1)
Hodge diamond of size 2 and dimension 1
```

__str__()
Pretty print Hodge diamond

This gets called when you specifically print the object.

EXAMPLES:

The projective line:

```
sage: load("diamond.py")
sage: print(Pn(1))
      1
  0       0
      1
```

pprint(*format='table'*)
Pretty print the Hodge diamond

INPUT:

- **format – output format (default: *"table"*), if table it pretty prints** a Hodge diamond; all else defaults to the polynomial

EXAMPLES:

The projective line:

```
sage: load("diamond.py")
sage: Pn(1).pprint()
      1
  0       0
      1
sage: Pn(1).pprint(format="polynomial")
x*y + 1
```

is_hodge_symmetric()
Check whether the Hodge diamond satisfies Hodge symmetry

This checks the equality

$$\mathrm{h}^{p,q}(X) = \mathrm{h}^{q,p}(X)$$

for $p,q=0,\ldots,\dim X$.

Almost all of the constructions provided with the library satisfy Hodge symmetry, because we (somewhat implicitly) work with things which are (or behave like) smooth projective varieties over a field of characteristic zero.

Over the complex numbers this can fail for non-Kähler manifolds, such as the Hopf surface.

In positive characteristic this can fail too, with an example given by classical and singular Enriques surfaces in characteristic 2, see [MR0491720] and Proposition 1.4.2 in [MR0986969]

- [MR0491720] Bombieri–Mumford, Enriques' classification of surfaces in char. p. III.
- [MR0986969] Cossec–Dolgachev, Enriques surfaces I, Progress in Mathematics, 1989

EXAMPLES:

Constructions satisfy this property:

```
sage: load("diamond.py")
sage: Pn(5).is_hodge_symmetric()
True
```

The Hopf surface over the complex numbers:

```
sage: S = HodgeDiamond.from_matrix([[1, 0, 0], [1, 0, 1], [0, 0, 1]])
sage: print(S)
          1
      0       1
  0       0       0
      1       0
          1
sage: S.is_hodge_symmetric()
False
```

Classical and singular Enriques surfaces in characteristic 2 (which are smooth, despite their name) also have a Hodge diamond violating Hodge symmetry:

```
sage: enriques(two="classical").is_hodge_symmetric()
False
sage: enriques(two="singular").is_hodge_symmetric()
False
sage: enriques(two="supersingular").is_hodge_symmetric()
True
```

is_serre_symmetric()

Check whether the Hodge diamond satisfies Serre symmetry

This checks the equality

$$\mathrm{h}^{p,q}(X) = \mathrm{h}^{\dim X-p,\dim X-q}(X)$$

for $p,q=0,\ldots,\dim X$.

Because Serre duality holds for all smooth projective varieties, independent of the characteristic, and also for non-Kähler varieties there are no examples where this condition fails. It can of course fail for motivic pieces, for silly reasons.

EXAMPLES:

The Hilbert scheme of 4 points on a K3 surface satisfies the symmetry:

```
sage: load("diamond.py")
sage: hilbn(K3(), 4).is_serre_symmetric()
True
```

The Lefschetz diamond fails it for silly reasons:

```
sage: lefschetz().is_serre_symmetric()
False
```

betti()

Betti numbers of the Hodge diamond

This gives an integer vector.

EXAMPLES:

Betti numbers of a K3 surface:

```
sage: load("diamond.py")
sage: K3().betti()
[1, 0, 22, 0, 1]
```

The second Betti number of the Hilbert scheme of points on a K3 surface is 23, not 22:

```
sage: [hilbn(K3(), n).betti()[2] for n in range(2, 5)]
[23, 23, 23]
```

middle()

Middle cohomology of the Hodge diamond

For smooth projective varieties the middle cohomology sits in degree equal to the dimension.

EXAMPLES:

There is an interesting link between K3 surfaces and cubic fourfolds which can be seen on the level of middle cohomology:

```
sage: load("diamond.py")
sage: (hypersurface(3, 4) - lefschetz()**2).middle()
[0, 1, 20, 1, 0]
sage: K3().middle()
[1, 20, 1]
```

euler()

The topological Euler characteristic of the Hodge diamond

This is the alternating sum of the Betti numbers, so that

$$\chi_{\text{top}} = \sum_{p,q=0}^{\dim X} (-1)^{p+q} \mathrm{h}^{p,q}$$

EXAMPLES:

The Euler characteristic of projective space grows linearly:

```
sage: load("diamond.py")
sage: [Pn(n).euler() for n in range(10)]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

For Hilbert schemes of points of K3 surfaces these are the coefficients of the series expansion of the Dedekind eta-function, see A006922 in the OEIS:

```
sage: [hilbn(K3(), n).euler() for n in range(5)]
[1, 24, 324, 3200, 25650]
```

holomorphic_euler()

Holomorphic Euler characteristic

This is the Euler characteristic of the structure sheaf, so

$$\chi(X) = \sum_{i=0}^{\dim X} (-1)^i \mathrm{h}^{0,i}(X)$$

EXAMPLES:

For projective space it is 1:

```
sage: load("diamond.py")
sage: all([Pn(n).holomorphic_euler() == 1 for n in range(10)])
True
```

For a hyperkähler variety of dimension $2n$ this number is $n+1$:

```
sage: all([K3n(n).holomorphic_euler() == n+1 for n in range(5)])
True
```

hirzebruch()
Hirzebruch's chi_y genus

For a smooth projective variety X Hirzebruch's χ_y-genus is defined as

$$\chi_y(X) = \sum_{p,q=0}^{\dim X} (-1)^{p+q} \mathrm{h}^{p,q}(X) y^p$$

which shows it is the specialisation of the Hodge-Poincaré polynomial for $x=-1$. A further specialisation to $y=-1$ gives the Euler characteristic.

EXAMPLES:

For a K3 surface we have:

```
sage: load("diamond.py")
sage: K3().hirzebruch()
2*y^2 - 20*y + 2
sage: K3().hirzebruch().subs(y=-1) == K3().euler()
True
```

For the Hilbert square of a K3 surface we get:

```
sage: hilbn(K3(), 2).hirzebruch()
3*y^4 - 42*y^3 + 234*y^2 - 42*y + 3
sage: hilbn(K3(), 2).hirzebruch().subs(y=-1) == hilbn(K3(), 2).euler()
True
```

homological_unit()
Dimensions of $\mathrm{H}^\bullet(X,O_X)$

A notion intorduced by Abuaf.

hochschild()
Dimensions of the Hochschild homology

Columns of the Hodge diamond are Hochschild homology, by the Hochschild- Kostant-Rosenberg theorem.

hh()
Shorthand for *HodgeDiamond.hochschild()*

arises_from_variety()
Check whether the Hodge diamond can arise from a smooth projective variety

The constraints are:

- satisfy Hodge symmetry
- satisfy Serre symmetry
- there is no Lefschetz twist

is_zero()
Check whether the Hodge diamond is identically zero

lefschetz_power()
Return the twist by the Lefschetz motive that is present

In other words, we see how divisible the Hodge–Poincaré polynomial is with respect to the monomial x^iy^i

dimension()

Dimension of the Hodge diamond

This takes twists by the Lefschetz class into account: we untwist by the maximal power and ony then determine how big the diamond is.

EXAMPLES:

A point is 0-dimensional:

```
sage: load("diamond.py")
sage: point().dimension()
0
```

The Lefschetz diamond is also 0-dimensional:

```
sage: print(lefschetz())
      0
  0       0
      1
sage: lefschetz().dimension()
0
```

level()

Compute the level (or complexity) of the Hodge diamond

This is a measure of the width of the non-zero part of the Hodge diamond.

EXAMPLES:

The simplest case is projective space, with level zero:

```
sage: load("diamond.py")
sage: all([Pn(n).level() == 0 for n in range(10)])
True
```

For intersections of 2 quadrics it alternates between zero and one:

```
sage: all([complete_intersection([2,2], 2*n).level() == 0 for n in range(5)])
True
sage: all([complete_intersection([2,2], 2*n+1).level() == 1 for n in␣
↪range(5)])
True
```

A Calabi-Yau variety (e.g. a hypersurface of degree $n+1$ in $\mathbb{P}^n$) has maximal level:

```
sage: all([hypersurface(n+2, n).level() == n for n in range(10)])
True
```

blowup(*other*, *codim=None*)

Compute Hodge diamond of blowup

No consistency checks are performed, this just naively applies the blowup formula from Hodge theory.

INPUT:

- `other` – Hodge diamond of the center of the blowup

- **codim – codimension of the center (optional), in case it is not** the Hodge diamond of an honest variety

EXAMPLES:

A cubic surface is the blowup of $\mathbb{P}^2$ in 6 points:

```
sage: load("diamond.py")
sage: Pn(2).blowup(6*point()) == hypersurface(3, 2)
True
```

bundle(*rank*)

Compute the Hodge diamond of a projective bundle

This applies the bundle formula from Hodge theory without any consistency checks.

INPUT:

- `rank`: rank of the vector bundle on `self`

EXAMPLES:

A projective bundle on a point is a projective space:

```
sage: load("diamond.py")
sage: point().bundle(3) == Pn(2)
True
```

A quadric surface is a $\mathbb{P}^1$-bundle on $\mathbb{P}^1$:

```
sage: Pn(1).bundle(2) == hypersurface(2, 2)
True
```

mirror()

Compute the mirror Hodge diamond

EXAMPLES:

The mirror to a quintic 3-fold is the following:

```
sage: load("diamond.py")
sage: print(hypersurface(5, 3).mirror())
                1
            0       0
        0       101     0
    1       1       1       1
        0       101     0
            0       0
                1
```

__weakref__

list of weak references to the object (if defined)

CHAPTER

TWO

HOCHSCHILD HOMOLOGY

class diamond.**HochschildHomology**(*L*)

This class implements some methods to work with (the dimensions of) Hochschild homology spaces, associated to the *HodgeDiamond* class.

The documentation is not intended to be complete, as this is mostly for my own sake.

classmethod from_list(*L*)

Constructor for Hochschild homology dimensions from a list.

INPUT:

- `L` – a list of integers representing HH_{-n} to HH_n

classmethod from_positive(*L*)

Constructor for Hochschild homology dimensions from a list when only the positive part is given.

INPUT:

- `L` – a list of integers representing HH_0 to HH_n

classmethod from_polynomial(*f*)

Constructor for Hochschild homology dimensions from Hochschild–Poincaré Laurent polynomial

INPUT

- `f` – the Hochschild–Poincaré Laurent polynomial

dimension()

Largest index `i` such that $\mathrm{HH}_i\neq 0$

euler()

Euler characteristic of Hochschild homology

symmetric_power(*k*)

Hochschild homology of the Ganter–Kapranov symmetric power of a smooth and proper dg category

This is possibly only a heuristic (I didn't check for proofs in the literature) based on the decomposition of Hochschild homology for a quotient stack, as discussed in the paper of Polishchuk–Van den Bergh

sym(*k*)

Shorthand for `` `HochschildHomology.symmetric_power` ``

CHAPTER

THREE

CONSTRUCTIONS

3.1 Elementary constructions

diamond.**zero**()
Hodge diamond for the empty space

EXAMPLES:

Zero:

```
sage: load("diamond.py")
sage: print(zero())
  0
```

diamond.**point**()
Hodge diamond for the point

EXAMPLES:

The point:

```
sage: load("diamond.py")
sage: print(point())
  1
```

diamond.**lefschetz**()
Hodge diamond for the Lefschetz motive

This is the Hodge-Poincaré polynomial of the affine line.

EXAMPLES:

The affine line:

```
sage: load("diamond.py")
sage: print(lefschetz())
      0
  0       0
      1
```

We can take powers of it, to get the Hodge-Poincaré polynomial for higher- dimensional affine spaces:

```
sage: print(lefschetz()**3)
            0
        0       0
    0       0       0
```

(continues on next page)

(continued from previous page)

```
  0       0       0       0
      0       0       0
          0       0
              1
```

diamond.**Pn**(*n*)

Hodge diamond for projective space of dimension n

INPUT:

- n: dimension, non-negative integer

EXAMPLES:

The zero-dimensional case is a point:

```
sage: load("diamond.py")
sage: Pn(0) == point()
True
```

In general projective space is the sum of powers of the Lefschetz class:

```
sage: all([Pn(n) == sum([lefschetz()**i for i in range(n + 1)]) for n in range(1,␣
↪10)])
True
```

diamond.**hypersurface**(*degree*, *dimension*)

Shorthand for a complete intersection of the given dimension where $k=1$

diamond.**weighted_hypersurface**(*degree*, *weights*)

Hodge diamond for a weighted hypersurface of degree d in P(w_0,...,w_n)

Implementation taken from https://github.com/jxxcarlson/math_research/blob/master/hodge.sage. If I'm not mistaken, the generalisation to weighted complete intersection depends on which polynomials are used, not just the degrees, even if the result is smooth?

INPUT:

- degree – degree of the hypersurface
- weights – the weights of the weighted projective space, if it is an integer we interpret it as the dimension of P^n

EXAMPLES:

Elliptic curves can be realised as hypersurfaces in 3 ways:

```
sage: load("diamond.py")
sage: weighted_hypersurface(3, 2) == weighted_hypersurface(3, [1, 1, 1])
True
sage: weighted_hypersurface(3, 2) == curve(1)
True
sage: weighted_hypersurface(4, [1, 1, 2]) == curve(1)
True
sage: weighted_hypersurface(6, [1, 2, 3]) == curve(1)
True
```

The Fano 3-fold 1.1 is a weighted hypersurface:

```
sage: fano_threefold(1, 1) == weighted_hypersurface(6, [1,1,1,1,3])
True
```

diamond.**cyclic_cover**(*ramification_degree*, *cover_degree*, *weights*)

Hodge diamond of a cyclic cover of weighted projective space

Implementation taken from https://github.com/jxxcarlson/math_research/blob/master/hodge.sage.

INPUT:

- `ramification_degree` – degree of the ramification divisor
- `cover_degree` – size of the cover
- `weights` – the weights of the weighted projective space, if it is an integer we interpret it as the dimension of `P^n`

EXAMPLES:

Some K3 surfaces are double covers of $\mathbb{P}^2$ in a sextic curve:

```
sage: load("diamond.py")
sage: cyclic_cover(6, 2, 2) == K3()
True
```

The Fano 3-fold 1.1 is a double cover of $\mathbb{P}^3$ in a sextic:

```
sage: cyclic_cover(6, 2, 3) == fano_threefold(1, 1)
True
```

diamond.**complete_intersection**(*degrees*, *dimension*)

Hodge diamond for a complete intersection of multidegree $(d_1,\ldots,d_k)$ in $\mathbb{P}^{n+k}$

For a proof, see théorème 2.3 of exposé XI in SGA7.

INPUT:

- `degrees` – the multidegree, if it is an integer we interpret it as a hypersurface
- `dimension` – the dimension of the complete intersection (not of the ambient space)

EXAMPLES:

For multidegrees $(1,\ldots,1)$ we get a lower-dimension projective space:

```
sage: load("diamond.py")
sage: complete_intersection(1, 2) == Pn(2)
True
sage: complete_intersection([1, 1], 2) == Pn(2)
True
sage: complete_intersection([1, 1], 5) == Pn(5)
True
```

3.2 Curves and moduli spaces of sheaves on them

diamond.**curve**(*genus*)

Hodge diamond for a curve of a given genus.

INPUT:

- genus: the genus of the curve, non-negative integer

EXAMPLE:

A curve of genus 0 is the 1-dimensional projective space:

```
sage: load("diamond.py")
sage: curve(0) == Pn(1)
True
```

A curve of genus 1 is an abelian variety of dimension 1:

```
sage: curve(1) == abelian(1)
True
```

A curve of genus 2:

```
sage: print(curve(2))
      1
  2       2
      1
```

diamond.**symmetric_power**(*n*, *genus*)

Hodge diamond for the nth symmetric power of a curve of given genus

For the proof, see Example 1.1(1) of [MR2777820]. An earlier reference, probably in Macdonald, should exist.

- [MR2777820] Laurentiu–Schuermann, Hirzebruch invariants of symmetric products. Topology of algebraic varieties and singularities, 163–177, Contemp. Math., 538, Amer. Math. Soc., 2011.

INPUT:

- n – exponent of the symmetric power
- genus – genus of the curve, a non-negative integer

EXAMPLES:

The symmetric square of a genus 3 curve:

```
sage: load("diamond.py")
sage: print(symmetric_power(2, 3))
          1
      3       3
  3       10      3
      3       3
          1
```

If $n=1$ we get the curve back:

```
sage: all([symmetric_power(1, g) == curve(g) for g in range(10)])
True
```

If $n=0$ we get the point:

```
sage: symmetric_power(0, 4) == point()
True
```

If $n<0$ we have the empty space:

```
sage: symmetric_power(-1, 4) == zero()
True
```

diamond.**jacobian**(*genus*)

Hodge diamond for the Jacobian of a genus g curve

This is an abelian variety of dimension *genus*, so we call *abelian()*

INPUT:

- genus – genus of the curve

EXAMPLES:

The Jacobian of a genus 3 curve:

```
sage: load("diamond.py")
sage: print(jacobian(3))
              1
          3       3
      3       9       3
  1       9       9       1
      3       9       3
          3       3
              1
```

For the projective line we get a point:

```
sage: jacobian(0) == point()
True
```

The Jacobian of an elliptic curve is isomorphic to it:

```
sage: jacobian(1) == curve(1)
True
```

diamond.**moduli_vector_bundles**(*rank*, *degree*, *genus*)

Hodge diamond for the moduli space of vector bundles of given rank and fixed determinant of given degree on a curve of a given genus.

For the proof, see Corollary 5.1 of [MR1817504].

- [MR1817504] del Baño, On the Chow motive of some moduli spaces. J. Reine Angew. Math. 532 (2001), 105–132.

If the Hodge diamond for the moduli space with non-fixed determinant of degree *d* is required, this can be obtained by:

```
jacobian(g) * moduli_vector_bundles(r, d, g)
```

INPUT:

- *rank* – rank of the bundles, at least 2
- *degree* – degree of the fixed determinant, coprime to rank

- *genus* – genus of the curve, at least 2

EXAMPLES:

The case of rank 2, degree 1 and genus 2 is famously the intersection of 2 quadrics in $\mathbb{P}^5$:

```
sage: load("diamond.py")
sage: moduli_vector_bundles(2, 1, 2) == complete_intersection([2, 2], 3)
True
```

diamond.**seshadris_desingularisation**(*genus*)

Hodge diamond for Seshadri's desingularisation of the moduli space of rank 2 bundles with trivial determinant on a curve of a given genus g.

For the statement, see Corollary 3.18 of [MR1895918].

- [MR1895918] del Baño, On the motive of moduli spaces of rank two vector bundles over a curve. Compositio Math. 131 (2002), 1-30.

INPUT:

- `genus` – the genus g, at least 2

EXAMPLES:

For $g=2$ nothing needs to be desingularised, and the answer is $\mathbb{P}^3$:

```
sage: load("diamond.py")
sage: seshadris_desingularisation(2) == Pn(3)
True
```

Already for $g=3$ the result is not a familiar variety, so we just check the Euler characteristic:

```
sage: seshadris_desingularisation(3).euler() == 112
True
```

diamond.**moduli_parabolic_vector_bundles_rank_two**(*genus*, *alpha*)

Hodge diamond for the moduli space of parabolic rank 2 bundles with fixed determinant of odd degree on a curve of genus g.

See Corollary 5.34 of [2011.14872].

- [2011.14872] Fu–Hoskins–Pepin Lehalleur, Motives of moduli spaces of bundles on curves via variation of stability and flips

This is not a proof of the formula we implemented per se, but it should be correct. Also, it could be that the choice of weights give something singular / stacky. Then it'll give bad output without warning. You have been warned.

INPUT:

- `genus` – the genus of the curve
- `alpha` – the weights of the parabolic bundles

diamond.**quot_scheme_curve**(*genus*, *length*, *rank*)

Hodge diamond for the Quot scheme of zero-dimensional quotients of given length of a vector bundle of given rank on a curve of given genus.

For the proof, see Proposition 4.5 of [1907.00826] (or rather, the reference [Bif89] in there)

- [1907.00826] Bagnarol–Fantechi–Perroni, On the motive of zero-dimensional Quot schemes on a curve

3.3 Surfaces and moduli spaces of sheaves on them

diamond.**surface**(*genus*, *irregularity*, *h11*)

Hodge diamond for a surface S with given invariants

These invariants are the geometric genus, the irregularity and the middle Hodge numbers `h11`.

INPUT:

- **genus – geometric genus of the surface, $\dim\mathrm{H}^2(S,\mathcal{O}_S)$,** a non-negative integer
- **irregularity – irregularity of the surface, $\dim\mathrm{H}^1(S,\mathcal{O}_S)$,** a non-negative integer
- **h11 – middle Hodge number, $\dim\mathrm{H}^1(S,\Omega_S^1)$,** a non-negative integer

EXAMPLES:

The projective plane:

```
sage: load("diamond.py")
sage: Pn(2) == surface(0, 0, 1)
True
```

A K3 surface:

```
sage: K3() == surface(1, 0, 20)
True
```

diamond.**ruled**(*genus*)

Hodge diamond for a ruled surface

These are $\mathbb{P}^1$-bundles over a curve of given genus.

INPUT:

- `genus` – genus of the base curve

EXAMPLES:

For genus 0 we get Hirzebruch surfaces, whose Hodge diamond is that of the quadric surface:

```
sage: load("diamond.py")
sage: ruled(0) == hypersurface(2, 2)
True
```

For higher genus the Hodge diamond looks as follows:

```
sage: print(ruled(5))
          1
      5       5
  0       2       0
      5       5
          1
```

diamond.**K3**()

Hodge diamond for a K3 surface

EXAMPLES:

The K3 surface:

```
sage: load("diamond.py")
sage: print(K3())
          1
      0       0
  1       20      1
      0       0
          1
```

diamond.**enriques**(*two=None*)

Hodge diamond for an Enriques surface

It is possible to ask for the Hodge diamond of a classical or (super)singular Enriques surface in characteristic 2.

In characteristic 2 the invariants are given in Proposition 1.4.2 of [MR0986969].

- [MR0986969] Cossec–Dolgachev, Enriques surfaces I, Progress in Mathematics, 1989

INPUT:

- **two – optional parameter to indicate the type of surface in characteristic 2** possible values are *"classical"*, *"singular"*, *"supersingular"*

EXAMPLES:

An ordinary Enriques surface:

```
sage: load("diamond.py")
sage: print(enriques())
          1
      0       0
  0       10      0
      0       0
          1
```

Enriques surfaces in characteristic 2:

```
sage: print(enriques(two="classical"))
          1
      0       1
  0       12      0
      1       0
          1
sage: print(enriques(two="singular"))
          1
      1       0
  1       10      1
      0       1
          1
sage: print(enriques(two="supersingular"))
          1
      1       1
  1       12      1
      1       1
          1
```

diamond.**hilbn**(*surface*, *n*)

Hodge diamond for Hilbert scheme of `n` points on a smooth projective surface `S`

For the proof, see Theorem 2.3.14 of [MR1312161].

- [MR1312161] Göttsche, Hilbert schemes of zero-dimensional subschemes of smooth varieties. Lecture Notes in Mathematics, 1572. Springer-Verlag, Berlin, 1994. x+196 pp.

INPUT:

- `S` – Hodge diamond for smooth projective surface
- `n` – number of points

`diamond.`**`nestedhilbn`**(*surface*, *n*)

Hodge diamond for the nested Hilbert scheme `S^{[n-1,n]}`

This is the unique nested Hilbert scheme of a smooth projective surface `S` which is itself smooth (of dimension $2n$)

3.4 Abelian varieties and related objects

`diamond.`**`abelian`**(*dimension*)

Hodge diamond for an abelian variety of a given dimension.

This description is standard, and follows from the fact that the cohomology is the exterior power of the first cohomology, as graded algebra.

See e.g. Proposition 7.27 in the Edixhoven–van der Geer–Moonen book-in-progress on abelian varieties.

`diamond.`**`kummer_resolution`**(*g*)

Hodge diamond for the standard resolution of the Kummer variety of an abelian variety of a given dimension.

There's an invariant part (Hodge numbers of even degree) and the resolution of the 2^2g singularities is added.

3.5 Fano varieties

These are Hodge diamonds of Fano varieties in the sense that their anticanonical bundle is ample.

The term "Fano variety" can also mean a variety parametrising linear subspaces on another variety. Some of these are Fano in the first sense, others are not (always). See e.g. *`diamond.fano_variety_lines_cubic()`*.

`diamond.`**`fano_threefold`**(*rho*, *ID*)

Hodge diamond of a Fano threefold

INPUT:

- `rho` - Picard rank
- `ID` - numbering from the Mori-Mukai classification

EXAMPLES:

The 17th Fano 3-fold of rank 1 is projective threespace:

```
sage: load("diamond.py")
sage: print(fano_threefold(1, 17))
              1
          0       0
      0       1       0
  0       0       0       0
      0       1       0
          0       0
              1
```

The 4th Fano 3-fold of rank 1 is an intersection of 3 quadrics:

```
sage: fano_threefold(1, 4) == complete_intersection((2, 2, 2), 3)
True
```

The 27th Fano 3-fold of rank 3 is the triple product of projective lines:

```
sage: fano_threefold(3, 27) == Pn(1)**3
True
```

diamond.**gushel_mukai**(*n*)

Hodge diamond for a smooth n-dimensional Gushel–Mukai variety

See Proposition 3.1 of [1605.05648v3].

- [1605.05648v3] Debarre–Kuznetsov, Gushel-Mukai varieties: linear spaces and periods

INPUT:

- `n` - the dimension, where $n=1,\ldots,6$

diamond.**fano_variety_intersection_quadrics_even**(*g*, *i*)

Hodge diamond for the Fano variety of $i-1$-planes on the intersection of two quadrics in $\mathbb{P}^{2g}$, using [1510.05986v3].

- [1510.05986v3] Chen–Vilonen–Xue, Springer correspondence, hyperelliptic curves, and cohomology of Fano varieties

INPUT:

- `g` – half of the dimension of the ambient projective space
- `i` – affine dimension of the linear subspaces on the intersection of quadrics

EXAMPLES:

We have that for i = g-1 we get the moduli space of parabolic bundles on $\mathbb{P}^1$ with weight $1/2$ in $2g+3$ points:

```
sage: load("diamond.py")
sage: moduli_parabolic_vector_bundles_rank_two(0, [1/2]*5) == fano_variety_
↪intersection_quadrics_even(2, 1)
True
sage: moduli_parabolic_vector_bundles_rank_two(0, [1/2]*7) == fano_variety_
↪intersection_quadrics_even(3, 2)
True
```

diamond.**fano_variety_intersection_quadrics_odd**(*g*, *i*)

Hodge diamond for the Fano variety of $g-i$-planes on the intersection of two quadrics in $\mathbb{P}^{2g+1}$, using [MR3689749].

We have that for i = 2 we get M_C(2,L) as above, for deg L odd.

- [MR3689749] Chen–Vilonen–Xue, On the cohomology of Fano varieties and the Springer correspondence, Adv. Math. 318 (2017), 515–533.

EXAMPLES:

For $i=2$ we recover the moduli space of rank 2 bundles with odd determinant on a curve of genus g:

```
sage: load("diamond.py")
sage: fano_variety_intersection_quadrics_odd(5, 2) == moduli_vector_bundles(2, 1,␣
↪5)
True
```

3.6 Homogeneous varieties and closely related constructions

These are also all Fano varieties, but they are grouped together because of their similar origin.

diamond.**partial_flag_variety**(*D*, *I*)

Hodge diamond of a partial flag variety G/P

This is computed by counting the number of Schubert cells in the appropriate dimension.

INPUT:

- `D` – Dynkin type
- `I` – indices of vertices to be omitted in defining the parabolic subgroup

EXAMPLES:

An absolute baby case is projective space:

```
sage: load("diamond.py")
sage: partial_flag_variety("A5", [2,3,4,5]) == Pn(5)
True
```

The next easiest case are quadrics:

```
sage: partial_flag_variety("B5", [2,3,4,5]) == hypersurface(2, 9)
True
sage: partial_flag_variety("D5", [2,3,4,5]) == hypersurface(2, 8)
True
```

diamond.**generalised_grassmannian**(*D*, *k*)

Hodge diamond of the generalised Grassmannian of type D modulo the maximal parabolic subgroup P_k.

This is just shorthand for `partial_flag_variety(D, I)()` where `I` is the complement of a singleton.

INPUT:

- `D` – Dynkin type
- `k` – the vertex in the Dynkin diagram defining the maximal parabolic

diamond.**grassmannian**(*k*, *n*)

Hodge diamond of the Grassmannian $\operatorname{Gr}(k,n)$ of k-dimensional subspaces in an n-dimensional vector space

INPUT:

- `k` – dimension of the subspaces
- `n` – dimension of the ambient vector space

diamond.**orthogonal_grassmannian**(*k*, *n*)

Hodge diamond of the orthogonal Grassmannian $\operatorname{OGr}(k, n)$ of k-dimensional subspaces in an n-dimensional vector space isotropic with respect to a non-degenerate symmetric bilinear form

INPUT:

- k – dimension of the subspaces
- n – dimension of the ambient vector space

diamond.**symplectic_grassmannian**(*k*, *n*)
Hodge diamond of the symplectic Grassmannian $\operatorname{SGr}(k, n)$ of k-dimensional subspaces in an n-dimensional vector space isotropic with respect to a non-degenerate skew-symmetric bilinear form

INPUT:

- k – dimension of the subspaces
- n – dimension of the ambient vector space

diamond.**lagrangian_grassmannian**(*n*)
Shorthand for the symplectic Grassmannian of Lagrangian subspaces

diamond.**horospherical**(*D*, *y=0*, *z=0*)
Horospherical varieties as discussed in [1803.05063], with labelling and notation as in op. cit.

INPUT:

- **D: either a Dynkin type from the (small) list of allowed types** in the classification _or_ a plaintext label from X1(n), X2, X3(n,m), X4, X5
- y: index for the parabolic subgroup for Y, see classification
- z: index for the parabolic subgroup for the closed orbit Z

y and z must be omitted if a plaintext description is given.

- [1803.05063] Gonzales–Pech–Perrin–Samokhin, Geometry of horospherical varieties of Picard rank one

diamond.**odd_symplectic_grassmannian**(*k*, *n*)
Hodge diamond of the odd symplectic Grassmannian $\operatorname{SGr}(k,n)$

Here n is odd. This is just shorthand for a call to `horospherical_variety()` for type C, with parameters $\lfloor n/2\rfloor$, and Y and Z determined by k and $k - 1$.

3.7 Moduli spaces attached to quivers

diamond.**quiver_moduli**(*Q*, *d*, *mu*)
Hodge diamond for the moduli space of semistable quiver representations for a quiver Q, dimension vector d, and slope-stability condition mu.

Taken from Corollary 6.9 of [MR1974891]

- [MR1974891] Reineke, The Harder-Narasimhan system in quantum groups and cohomology of quiver moduli.

INPUT:

- Q – adjacency matrix of an acyclic quiver
- d – dimension vector
- mu – stability condition, these can be produced using `mu()`

EXAMPLES:

Let's consider moduli spaces for the Kronecker quiver:

```
sage: load("diamond.py")
sage: def kronecker(d): return matrix([[0, d], [0, 0]])
```

For the 2-Kronecker quiver and dimension vector *(1,1)* a representation is given by 2 scalars, and the stability condition *(1,-1)* encodes that they are not both zero. This way we obtain the projective line:

```
sage: quiver_moduli(kronecker(2), (1, 1), mu((1, -1))) == Pn(1)
True
```

Similar to the first example, the d-Kronecker quiver gives rise to projective spaces:

```
sage: all([quiver_moduli(kronecker(d), (1, 1), mu((1, -1))) == Pn(d - 1) for d in␣
↪range(3, 10)])
True
```

We can also realise Grassmannians using the d-Kronecker quiver, for dimension vector $(1,k)$ and stability condition $(k,1)$ we get the Grassmannian $\operatorname{Gr}(k,d)$:

```
sage: quiver_moduli(kronecker(4), (1, 2), mu((2, 1))) == grassmannian(2, 4)
True
sage: quiver_moduli(kronecker(7), (1, 3), mu((3, 1))) == grassmannian(3, 7)
True
```

The flag variety $\operatorname{Fl}(n,r_1,\ldots,r_s)$ is also a quiver moduli space, for A_s quiver prefixed with an n-Kronecker quiver, dimension vector $(1,r_1,\ldots,r_s)$ with $r_1>\ldots>r_s$ and stability condition the indicator function at the first vertex:

```
sage: def flags(n, s): return matrix(ZZ, s+1, s+1, lambda i,j: 0 if i != j-1 else␣
↪(n if i == 0 else 1))
sage: quiver_moduli(flags(4, 1), (1, 1), mu((1, 0))) == Pn(3)
True
sage: quiver_moduli(flags(3, 2), (1, 2, 1), mu((1, 0, 0))) == fano_threefold(2,␣
↪32)
True
sage: quiver_moduli(flags(5, 3), (1, 4, 3, 1), mu((1, 0, 0, 0))) == partial_flag_
↪variety("A4", [2])
True
```

3.8 Hyperkähler varieties

diamond.**K3n**(*n*)

Hodge diamond of the Hilbert scheme of n points on a K3 surface

This is the first family of hyperkähler varieties, constructed by Beauville.

INPUT:

- `n` – number of points

EXAMPLES:

For $n=1$ we have a K3 surface:

```
sage: load("diamond.py")
sage: K3n(1) == K3()
True
```

For $n\geq 2$ we have second Betti number 23:

```
sage: all([K3n(n).betti()[2] == 23 for n in range(2, 5)])
True
```

diamond.**generalised_kummer**(*n*)

Hodge diamond of the nth generalised Kummer variety

For the proof, see Corollary 1 of [MR1219901].

- [MR1219901] Göttsche–Soergel, Perverse sheaves and the cohomology of Hilbert schemes of smooth algebraic surfaces. Math. Ann. 296 (1993), no. 2, 235–245.

EXAMPLES:

The first generalised Kummer is just a point:

```
sage: load("diamond.py")
sage: generalised_kummer(1) == point()
True
```

The second generalised Kummer is the Kummer K3 surface:

```
sage: generalised_kummer(2) == K3()
True
```

The higher generalised Kummers are hyperkähler varieties with second Betti number 7:

```
sage: all([generalised_kummer(n).betti()[2] == 7 for n in range(3, 10)])
True
```

diamond.**ogrady6**()

Hodge diamond for O'Grady's exceptional 6-dimensional hyperkähler variety

For the proof, see Theorem 1.1 of [MR3798592].

- [MR3798592] Mongardi–Rapagnetta–Saccà, The Hodge diamond of O'Grady's six-dimensional example. Compos. Math. 154 (2018), no. 5, 984–1013.

EXAMPLES:

The second Betti number is 8:

```
sage: load("diamond.py")
sage: ogrady6().betti()[2] == 8
True
```

diamond.**ogrady10**()

Hodge diamond for O'Grady's exceptional 10-dimensional hyperkähler variety

For the proof, see theorem A of [1905.03217]

- [1905.03217] de Cataldo–Rapagnetta–Saccà, The Hodge numbers of O'Grady 10 via Ngô strings

EXAMPLES:

The second Betti number is 24:

```
sage: load("diamond.py")
sage: ogrady10().betti()[2] == 24
True
```

3.9 Other

diamond.**Mzeronbar**(*n*)

Hodge diamond for the moduli space of n-pointed stable curves of genus 0

Taken from (0.12) in [MR1363064]. Keel's original paper has a recursion on page 550, but that seems to not work.

- [MR1363064] Manin, Generating functions in algebraic geometry and sums over trees

EXAMPLES:

The first few cases are a point, the projective line, and the blowup of $\mathbb{P}^2$ in 4 points:

```
sage: load("diamond.py")
sage: Mzeronbar(3) == point()
True
sage: Mzeronbar(4) == Pn(1)
True
sage: Mzeronbar(5) == Pn(2).blowup(4*point())
True
```

diamond.**fano_variety_lines_cubic**(*n*)

Hodge diamond for the Fano variety of lines on a smooth n-dimensional cubic hypersurface.

This follows from the "beautiful formula" or X-F(X)-relation due to Galkin–Shinder, Theorem 5.1 of [1405.5154v2].

- [1405.5154v2] Galkin–Shinder, The Fano variety of lines and rationality problem for a cubic hypersurface

INPUT:

- `n` – the dimension, where `n` is at least 2

EXAMPLES:

There are 27 lines on a cubic surface:

```
sage: load("diamond.py")
sage: fano_variety_lines_cubic(2) == 27*point()
True
```

The Fano surface of lines on a cubic threefold is a surface of general type:

```
sage: fano_variety_lines_cubic(3) == surface(10, 5, 25)
True
```

The Fano fourfold of lines on a cubic fourfold is deformation equivalent to the Hilbert square on a K3 surface:

```
sage: fano_variety_lines_cubic(4) == hilbn(K3(), 2)
True
```

PYTHON MODULE INDEX

d

INDEX

Symbols

A

B

C

D

E

F

G

H