Hodge-special fourfolds

This module provides support for Hodge-special fourfolds, such as cubic fourfolds and Gushel-Mukai fourfolds. For some example see the function `fourfold()`.

For more computational details, see the paper at https://www.tandfonline.com/doi/abs/10.1080/10586458.2023.2184882 and references therein.

> **Note**
>
> For some of the functions provided, you must have `Macaulay2` with the package `SpecialFanoFourfolds` (version 2.7.1 or later) installed on your computer; see https://faculty.math.illinois.edu/Macaulay2/doc/Macaulay2/share/doc/Macaulay2/SpecialFanoFourfolds/html/index.html.

AUTHORS:

- Giovanni Staglianò (2023-06-15): initial version

```
class sage.schemes.hodge_special_fourfolds.sff.Cubic_fourfold(S, X, V=None,
    check=True)
```

Bases: `Hodge_special_fourfold`

The class of Hodge-special cubic fourfolds in `PP^5`.

```
K3(verbose=None)
```

Associated K3 surfaces to rational cubic fourfolds.

This just runs the `Macaulay2` function `associatedK3surface`, documented at https://faculty.math.illinois.edu/Macaulay2/doc/Macaulay2/share/doc/Macaulay2/SpecialFanoFourfolds/html/_associated__K3surface_lp__Special__Cubic__Fourfold_rp.html. See also the paper at https://www.tandfonline.com/doi/abs/10.1080/10586458.2023.2184882 for more computational details.

OUTPUT:

`Embedded_projective_variety`, a (minimal) K3 surface associated to `self`.

EXAMPLES:

```
sage: X = fourfold(surface(3,1,1)); X
Cubic fourfold of discriminant 14 = 3*10-4^2 containing a rational surface in PP^5 of
sage: T = X.K3(verbose=False); T
surface in PP^8 of degree 14 and sectional genus 8 cut out by 15 hypersurfaces of deg
sage: building = T.building() # a tuple of 4 objects obtained in the construction of
sage: building[0] # the first of which is the Fano map
dominant rational map defined by forms of degree 2
source: PP^5
target: quadric hypersurface in PP^5
```

```
class sage.schemes.hodge_special_fourfolds.sff.Embedded_projective_variety(PP,
    polys=[])
```

Bases: `AlgebraicScheme_subscheme_projective`

The class of closed subvarieties of projective spaces.

This is a subclass of the class `AlgebraicScheme_subscheme_projective`. It is designed to provide better support for projective surfaces and fourfolds.

Constructing a closed subvariety of an ambient projective space.

> **Warning**
>
> You should not create objects of this class directly. The preferred method to construct such subvarieties is to use `projective_variety()`.

INPUT:

- `PP` – an ambient projective space
- `polys` – a list of homogeneous polynomials in the coordinate ring of `PP`

OUTPUT:

The closed subvariety of `PP` defined by `polys`

EXAMPLES:

```
sage: P, (x0,x1,x2,x3) = ProjectiveSpace(3, GF(101), 'x').objgens()
sage: X = projective_variety([x0^2-x1*x2, x1^3+x2^3+x3^3], P); X     # the input P is opt
curve of degree 6 and arithmetic genus 4 in PP^3 cut out by 2 hypersurfaces of degrees (2
sage: X.ambient_space() is P
True
sage: X.defining_polynomials()
(x0^2 - x1*x2, x1^3 + x2^3 + x3^3)
sage: X.describe()
dim:.................. 1
codim:................ 2
degree:............... 6
sectional genus:...... 4
generators:........... (2, 3)
dim sing. l.:......... -1
```

Here is another example using simple functions of the class.

```
sage: from sage.misc.randstate import set_random_seed
sage: set_random_seed(0)
sage: K = GF(65521)
sage: PP(4,K)
PP^4
sage: PP(4,K).empty()
empty subscheme of PP^4
sage: X = PP(4,K).empty().random(2,2,3)
sage: X
curve of degree 12 and arithmetic genus 13 in PP^4 cut out by 3 hypersurfaces of degrees
sage: X.describe()
dim:.................. 1
codim:................ 3
degree:............... 12
sectional genus:...... 13
generators:........... (2, 2, 3)
dim sing. l.:......... -1
sage: X.dimension()
1
sage: X.codimension()
3
sage: X.degree()
12
sage: X.sectional_genus()
13
sage: X.ambient()
PP^4
sage: p = X.point(); p
one-point scheme in PP^4 of coordinates [1, 52775, 1712, 653, 60565]
sage: p.dimension() == 0 and p.degree() == 1 and p.is_subset(X)
True
```

You can also convert such varieties into `Macaulay2` objects.

```
sage: X_ = macaulay2(X); X_                                          # optional - macaula
curve in PP^4 cut out by 3 hypersurfaces of degrees 2^2 3^1

ProjectiveVariety, curve in PP^4
sage: X_.cls()                                                       # optional - macaula
EmbeddedProjectiveVariety

Type
```

ambient()

Return the ambient projective space of the variety

This is mathematically equal to the output of the `ambient_space()` method.

EXAMPLES:

```
sage: X = Veronese(1,3); X
cubic curve of arithmetic genus 0 in PP^3 cut out by 3 hypersurfaces of degree 2
sage: P = X.ambient(); P
PP^3
sage: type(P) is type(X)
True
```

↑ Back to top

codimension()

Return the codimension of the variety

OUTPUT:

An integer.

EXAMPLES:

```
sage: X = Veronese(1,3); X
cubic curve of arithmetic genus 0 in PP^3 cut out by 3 hypersurfaces of degree 2
sage: X.codimension()
2
```

cone_of_lines(point=None)

Return the union of the lines contained in `self` and passing through the point `point`. If `point` is not given, a random point on `self` is used.

EXAMPLES:

```
sage: from sage.misc.randstate import set_random_seed
sage: set_random_seed(12345)
sage: Q = PP(3,KK=GF(333331)).empty().random(2); Q
quadric surface in PP^3
sage: p = Q.point()
sage: Q.cone_of_lines(p).irreducible_components()
[line in PP^3, line in PP^3]
```

degree()

Return the degree of the projective variety

OUTPUT:

An integer.

EXAMPLES:

```
sage: X = Veronese(1,3); X
cubic curve of arithmetic genus 0 in PP^3 cut out by 3 hypersurfaces of degree 2
sage: X.degree()
3
```

degrees_generators()

Return the degrees of a minimal set of generators for the defining ideal of the variety.

OUTPUT:

A tuple of integers.

EXAMPLES:

```
sage: X = PP(4).empty().random(1,2,3,1)
sage: X.degrees_generators()
(1, 1, 2, 3)
```

describe()

Print a brief description of the variety.

OUTPUT:

Nothing.

EXAMPLES:

```
sage: X = Veronese(2,2)
sage: X.describe()
dim:.................. 2
codim:................ 3
degree:............... 4
sectional genus:...... 0
generators:........... (2, 2, 2, 2, 2, 2)
dim sing. l.:......... -1
```

↑ Back to top

difference(other)

Return the Zariski closure of the difference of `self` by `other`.

EXAMPLES:

```
sage: X = Veronese(1,3)
sage: Y = X.ambient().point()
sage: Z = X.union(Y)
sage: Z.difference(Y) == X and Z.difference(X) == Y
True
sage: Z - Y == X and Z - X == Y
True
```

dimension()

Return the dimension of the variety

OUTPUT:

An integer.

EXAMPLES:

```
sage: X = Veronese(1,3); X
cubic curve of arithmetic genus 0 in PP^3 cut out by 3 hypersurfaces of degree 2
sage: X.dimension()
1
```

embedding_morphism(codomain=None)

Return the embedding morphism of the variety in its ambient space.

OUTPUT:

`Rational_map_between_embedded_projective_varieties`

EXAMPLES:

```
sage: X = Veronese(1,2)
sage: X.embedding_morphism()
morphism defined by forms of degree 1
source: conic curve in PP^2
target: PP^2
image: conic curve in PP^2
```

empty()

Return the empty subscheme of the variety (and of its ambient space)

EXAMPLES:

```
sage: X = PP(3)
sage: X.empty()
empty subscheme of PP^3
```

hilbert_polynomial()

Return the Hilbert polynomial of the projective variety.

OUTPUT:

A polynomial over the rationals.

EXAMPLES:

```
sage: X = Veronese(1,3); X
cubic curve of arithmetic genus 0 in PP^3 cut out by 3 hypersurfaces of degree 2
sage: X.hilbert_polynomial()
3*t + 1
```

↑ Back to top

hyperplane_section(cache=True)

Return a random hyperplane section of the variety.

OUTPUT:

`Embedded_projective_variety`, the intersection of `self` with a random hyperplane of the ambient projective space.

EXAMPLES:

```
sage: X = Veronese(2,2)
sage: H = X.hyperplane_section(); H
curve of degree 4 and arithmetic genus 0 in PP^4 cut out by 6 hypersurfaces of degree
sage: j = H.embedding_as_hyperplane_section(); j
rational map defined by forms of degree 1
source: curve of degree 4 and arithmetic genus 0 in PP^4 cut out by 6 hypersurfaces o
target: surface in PP^5 of degree 4 and sectional genus 0 cut out by 6 hypersurfaces
sage: j.source() is H and j.target() is X
True
sage: j.image()
curve of degree 4 and arithmetic genus 0 in PP^5 cut out by 7 hypersurfaces of degree
```

intersection(other)

Return the scheme-theoretic intersection of `self` and `other` in their common ambient space.

EXAMPLES:

```
sage: o = PP(5).empty()
sage: X = o.random(2,2)
sage: Y = o.random(1,3)
sage: X.intersection(Y)
curve of degree 12 and arithmetic genus 13 in PP^5 cut out by 4 hypersurfaces of degr
```

irreducible_components()

Return the irreducible components of the projective scheme `self`.

OUTPUT:

A tuple of irreducible subschemes of the same ambient space of the scheme `self`.

EXAMPLES:

```
sage: L = PP(4).empty().random(1,1,1)
sage: C = PP(4).empty().random(1,1,2)
sage: X = L + C
sage: X.irreducible_components()
[line in PP^4, conic curve in PP^4]
```

is_subset(Y)

Return `True` if `self` is contained in `Y`, `False` otherwise.

OUTPUT:

`bool`

linear_span()

Return the linear span of the variety.

OUTPUT:

`Embedded_projective_variety`

EXAMPLES:

```
sage: X = PP(5).empty().random(1,1,2)
sage: X.linear_span()
```

```
linear 3-dimensional subspace of PP^5
sage: X.is_subset(_)
True
```

↑ Back to top

parametrize(verbose=None)

Try to return a rational parameterization of the variety.

OUTPUT:

`Rational_map_between_embedded_projective_varieties`, a birational map from `PP(n)` to `self`, where `n` is the dimension of `X`.

An exception is raised if something goes wrong.

EXAMPLES:

```
sage: S = surface(5,7,0,1)
sage: h = S.parametrize()
sage: h
dominant rational map defined by forms of degree 5
source: PP^2
target: surface in PP^7 of degree 9 and sectional genus 3 cut out by 12 hypersurfaces

sage: L = PP(7).empty().random(1,1,1,1)
sage: L.parametrize()
dominant rational map defined by forms of degree 1
source: PP^3
target: linear 3-dimensional subspace of PP^7

sage: L.ambient().parametrize()
dominant rational map defined by forms of degree 1
source: PP^7
target: PP^7

sage: with seed(0): p = PP(4,KK=GF(65521)).point(); p
one-point scheme in PP^4 of coordinates [1, 37782, 10657, 17260, 21224]
sage: p.parametrize()
dominant rational map defined by forms of degree 1
source: PP^0
target: one-point scheme in PP^4 of coordinates [1, 37782, 10657, 17260, 21224]

sage: Veronese(2,2).parametrize()
dominant rational map defined by forms of degree 2
source: PP^2
target: surface in PP^5 of degree 4 and sectional genus 0 cut out by 6 hypersurfaces

sage: C = Veronese(2,2).intersection(PP(5).empty().random(1)); C
curve of degree 4 and arithmetic genus 0 in PP^5 cut out by 7 hypersurfaces of degree
sage: C.parametrize(verbose=False)            # optional - macaulay2
rational map defined by forms of degree 4
source: PP^1
target: curve of degree 4 and arithmetic genus 0 in PP^5 cut out by 7 hypersurfaces
```

point(verbose=None, algorithm='sage')

Pick a random point on the variety defined over a finite field.

EXAMPLES:

```
sage: X = Veronese(2,2)
sage: with seed(0): p = X.point(); p
one-point scheme in PP^5 of coordinates [1, 10850, 2338, 30739, 2409, 33291]
sage: type(p) is type(X) and p.dimension() == 0 and p.degree() == 1 and p.is_subset(X
True
```

With the input `algorithm='macaulay2'` the computation is transferred to `Macaulay2`

random(*args)

Return a random complete intersection containing the variety.

INPUT:

A tuple of positive integers `(a,b,c,...)`

OUTPUT:

`Embedded_projective_variety`, a random complete intersection of type `(a,b,c,...)` containing the variety.

An exception is raised if such a com ↑ Back to top on does not exist.

EXAMPLES:

```
sage: X = Veronese(1,5)
sage: X.random(2,3)
complete intersection of type (2, 3) in PP^5
sage: X.is_subset(_)
True
```

random_coordinate_change()

Apply a random coordinate change on the ambient projective space of `self`.

EXAMPLES:

```
sage: from sage.misc.randstate import set_random_seed
sage: set_random_seed(0)
sage: C = Veronese(1,3,KK=GF(13))
sage: C.defining_ideal()
Ideal (x2^2 - x1*x3, x1*x2 - x0*x3, x1^2 - x0*x2) of Multivariate Polynomial Ring in
sage: D = C.random_coordinate_change()
sage: D.defining_ideal()
Ideal (2*x0^2 + 5*x0*x1 - 4*x1^2 - 2*x0*x2 - x1*x2 + 3*x2^2 - 2*x0*x3 + x1*x3 + 2*x2
```

sectional_genus()

Return the arithmetic genus of the sectional curve of the variety.

OUTPUT:

An integer.

EXAMPLES:

```
sage: X = Veronese(2,2); X
surface in PP^5 of degree 4 and sectional genus 0 cut out by 6 hypersurfaces of degre
sage: X.sectional_genus()
0
```

singular_locus()

Return the singular locus of the variety.

OUTPUT:

`Embedded_projective_variety`

EXAMPLES:

```
sage: X = Veronese(1,3)
sage: X.singular_locus()
empty subscheme of PP^3
sage: type(_) is type(X) and _.is_subset(X)
True
sage: Y = surface(3,3,nodes=1); Y
rational 1-nodal surface in PP^5 of degree 6 and sectional genus 1 cut out by 5 hyper
sage: Y.singular_locus()
0-dimensional subscheme of degree 5 in PP^5
```

to_built_in_variety()

Return the same mathematical object but in the parent class.

OUTPUT:

`AlgebraicScheme_subscheme_projective`

EXAMPLES:

```
sage: Veronese(1,3)
cubic curve of arithmetic genus 0 in PP^3 cut out by 3 hypersurfaces of degree 2
sage: _.to_built_in_variety()
```

```
Closed subscheme of Projective Space of dimension 3 over Finite Field of size 33331 (
x2^2 - x1*x3,
x1*x2 - x0*x3,
x1^2 - x0*x2
```


topological_euler_characteristic(verbose=None, algorithm=None)

Return the topological Euler characteristic of the variety.

> **Warning**
>
> This uses a probabilistic approach which could give wrong answers (especially over finite fields of small order).

With the input `algorithm='macaulay2'` the computation is transferred to `Macaulay2`.

OUTPUT:

An integer.

EXAMPLES:

```
sage: X = Veronese(2,2); X
surface in PP^5 of degree 4 and sectional genus 0 cut out by 6 hypersurfaces of degre
sage: X.topological_euler_characteristic()
3
```

union(other)

Return the scheme-theoretic union of `self` and `other` in their common ambient space.

EXAMPLES:

```
sage: P = PP(5)
sage: X = P.point()
sage: Y = P.point()
sage: X.union(Y)
0-dimensional subscheme of degree 2 in PP^5
sage: X.union(Y) == X + Y
True
```

class sage.schemes.hodge_special_fourfolds.sff.GushelMukai_fourfold(S, X, V=None, check=True)

Bases: `Hodge_special_fourfold`

The class of Hodge-special Gushel-Mukai fourfolds in `PP^8`

K3(verbose=None)

Associated K3 surfaces to rational Gushel-Mukai fourfolds.

This just runs the `Macaulay2` function `associatedK3surface`, documented at https://faculty.math.illinois.edu/Macaulay2/doc/Macaulay2/share/doc/Macaulay2/SpecialFanoFourfolds/html/_associated__K3surface_lp__Special__Gushel__Mukai__Fourfold_rp.html. See also the paper at https://www.tandfonline.com/doi/abs/10.1080/10586458.2023.2184882 for more computational details.

OUTPUT:

`Embedded_projective_variety`, a (minimal) K3 surface associated to `self`.

EXAMPLES:

```
sage: X = fourfold('6'); X
Gushel-Mukai fourfold of discriminant 10('') containing a plane in PP^8, class of the
sage: T = X.K3(verbose=False); T
surface in PP^6 of degree 10 and sectional genus 6 cut out by 6 hypersurfaces of degr
sage: building = T.building() # a tuple of 4 objects obtained in the construction of
sage: building[0] # the first of which is the Fano map
dominant rational map defined by forms of degree 1
source: 5-dimensional variety of degree 5 in PP^8 cut out by 5 hypersurfaces of degre
target: quadric hypersurface in PP^5
```

class sage.schemes.hodge_special_fourfolds.sff.Hodge_special_fourfold(S, X, V=None,

```
    check=True)
```

Bases: `Embedded_projective_variety`

The class of Hodge-special fourfolds ↑ Back to top

This is a subclass of the class `Embedded_projective_variety`. An object of this class is just a (smooth) projective variety of dimension 4, although it is better to think of `X` as a pair `(S, X)`, where `X` is the fourfold and `S` is a particular special surface contained in `X`. Usually there is also a fixed ambient fivefold `V` where `S` and `X` live.

Constructing a Hodge-special fourfold.

> **Warning**
>
> You should not create objects of this class directly. The preferred method to construct such fourfolds is to use `fourfold()`.

INPUT:

- `S` `Embedded_projective_variety` – an irreducible surface.
- `X` `Embedded_projective_variety` – a smooth fourfold containing the surface `S`.
- `V` `Embedded_projective_variety` – a fivefold where `X` is a hypersurface (optional).

OUTPUT:

The Hodge-special fourfold corresponding to the pair `(S,X)`.

If the input fourfold `X` is missing, it will be chosen randomly. So, typically we just specify the surface `S`. For instance, if `S` is a surface in `PP^5`, then `fourfold(S)` returns a random cubic fourfold containing `S`; if `S` is a surface in `PP^7`, then `fourfold(S)` returns a random complete intersection of 3 quadrics containing `S` and contained in a complete intersection `V` of 2 quadrics.

EXAMPLES:

```
sage: S = surface(3,4)
sage: X = fourfold(S); X
Cubic fourfold of discriminant 14 = 3*13-5^2 containing a rational surface in PP^5 of deg
sage: X.surface()
rational surface in PP^5 of degree 5 and sectional genus 1 cut out by 5 hypersurfaces of
sage: X.ambient_fivefold()
PP^5
```

You can also convert such fourfolds into `Macaulay2` objects.

```
sage: X_ = macaulay2(X); X_                                                 # optional -
hypersurface in PP^5 defined by a form of degree 3

ProjectiveVariety, cubic fourfold containing a surface of degree 5 and sectional genus 1
sage: X_.surface()                                                          # optional -
surface in PP^5 cut out by 5 hypersurfaces of degree 2

ProjectiveVariety, surface in PP^5
```

`ambient_fivefold()`

Return the ambient fivefold of the fourfold.

OUTPUT:

`Embedded_projective_variety`, the ambient fivefold of `self`.

EXAMPLES:

```
sage: S = surface(3,4)
sage: X = fourfold(S)
sage: X.ambient_fivefold()
PP^5
```

`congruence(degree=None, num_checks=3, point=None, verbose=None, algorithm_for_image='sage', algorithm_for_point='sage', macaulay2_detectCongruence=False)`

Detect and return a congruence of secant curves for the surface of `self` in the ambient fivefold of `self`.

This function works similar to the `Ma` ↑ Back to top on `detectCongruence`, documented at https://faculty.math.illinois.edu/Macaulay2/doc/Macaulay2/share/doc/Macaulay2/SpecialFanoFourfolds/html/_detect__Congruence.html. See also the paper at https://www.tandfonline.com/doi/abs/10.1080/10586458.2023.2184882 for more computational details.

INPUT:

`degree` – an optional integer, the degree of the curves of the congruence.

`num_checks` – an optional integer with default value 3, check that the congruence works by testing so many random points on the ambient fivefold.

`point` – optional, a point on the ambient fivefold. This is only useful when you want to perform calculations on infinite fields, where the function `point()` might not work.

`verbose` – a boolean value, turn on or off verbose output.

`algorithm_for_image` – possible values are `sage` (by default) and `macaulay2`, with `algorithm_for_image='macaulay2'` the computation of the image of the map `map_from_fivefold()` is performed using `Macaulay2` (this is recommended if you have `Macaulay2` installed).

`algorithm_for_point` – possible values are `sage` (by default) and `macaulay2`, with `algorithm_for_point='macaulay2'` the computation of random points is performed using `Macaulay2`.

`macaulay2_detectCongruence` – a boolean value, default value false, with `macaulay2_detectCongruence=True` the whole computation is performed using the `Macaulay2` function `detectCongruence`.

OUTPUT:

A congruence of curves, which behaves like a function that sends a point `p` on the ambient fivefold to the curve of the congruence passing through `p`.

EXAMPLES:

```
sage: X = fourfold(surface(3,1,1,KK=GF(65521))); X
Cubic fourfold of discriminant 14 = 3*10-4^2 containing a rational surface in PP^5 of
sage: f = X.congruence(algorithm_for_image='macaulay2'); f
Congruence of 2-secant lines
to: rational surface in PP^5 of degree 4 and sectional genus 0 cut out by 6 hypersurf
in: PP^5
sage: p = X.ambient_fivefold().point()
sage: f(p)
line in PP^5

sage: X = fourfold("GM 4-fold of discriminant 26('')",GF(65521)); X
Gushel-Mukai fourfold of discriminant 26('') containing a surface in PP^8 of degree 9
sage: f = X.congruence(macaulay2_detectCongruence=True, num_checks=1, verbose=True);
-- running Macaulay2 function detectCongruence()... --
number lines contained in the image of the quadratic map and passing through a genera
number 1-secant lines = 5
number 3-secant conics = 1
-- function detectCongruence() has terminated. --
-- checking congruence (1 of 1)...
Congruence of 3-secant conics
to: surface in PP^8 of degree 9 and sectional genus 2 cut out by 19 hypersurfaces of
in: 5-dimensional variety of degree 5 in PP^8 cut out by 5 hypersurfaces of degree 2
sage: p = X.ambient_fivefold().point()
sage: f(p)
conic curve in PP^8

sage: X = fourfold("3-nodal septic scroll", GF(61001)); X
Cubic fourfold of discriminant 26 = 3*25-7^2 containing a surface in PP^5 of degree 7
sage: X.congruence(algorithm_for_image='macaulay2', num_checks=2)
Congruence of 5-secant conics
to: surface in PP^5 of degree 7 and sectional genus 0 cut out by 13 hypersurfaces of
in: PP^5
```

discriminant(verbose=None)

Return the discriminant of the special fourfold.

OUTPUT:

An integer, the discriminant of `self`.

In several cases, such as that of cubic fourfolds, this is the discriminant of the saturated lattice spanned by `h^2` and `S`, where `S` is the class of the surface of `X` and `h` denotes the class of a hyperplane section of `X`. For theoretical details, we refer to Hassett's papers on cubic fourfolds.

EXAMPLES:

```
sage: S = surface(3,4)
sage: X = fourfold(S)
sage: X.discriminant()
14
```

fano_map(verbose=None)

Return the Fano map from the ambient fivefold.

The surface contained in the fourfold `self` must admit a congruence of secant curves inside the ambient fivefold. The generic curve of this congruence can be realized as the generic fiber of the returned map. See also `detect_congruence()`.

EXAMPLES:

```
sage: X = fourfold(surface(3,4))
sage: X.fano_map(verbose=false)                    # optional - macaulay2
dominant rational map defined by forms of degree 2
source: PP^5
target: PP^4
```

map_from_fivefold(verbose=None, algorithm='sage')

(For internal use only) Return the map from the ambient fivefold of `self` defined by the linear system of hypersurfaces containing the surface of `self` and of degree equal to the degree of `self` as a hypersurface in its ambient fivefold.

parameter_count(verbose=None)

Count of parameters.

This just runs the `Macaulay2` function `parameterCount`, documented at https://faculty.math.illinois.edu/Macaulay2/doc/Macaulay2/share/doc/Macaulay2/SpecialFanoFourfolds/html/_parameter__Count.html. See also the paper at https://www.tandfonline.com/doi/abs/10.1080/10586458.2023.2184882 for more computational details.

OUTPUT:

An integer and a tuple of three integers.

EXAMPLES:

```
sage: X = fourfold(surface(3,1,1)); X
Cubic fourfold of discriminant 14 = 3*10-4^2 containing a rational surface in PP^5 of
sage: X.parameter_count(verbose=True)                                          # opti
-- running Macaulay2 function parameterCount()... --
S: smooth rational normal scroll surface of degree 4 in PP^5
X: smooth cubic hypersurface in PP^5
(assumption: dim Ext^1(I_{S,P^5},O_S) = 0)
h^0(N_{S,P^5}) = 29
h^1(O_S(3)) = 0, and h^0(I_{S,P^5}(3)) = 28 = h^0(O_(P^5)(3)) - \chi(O_S(3));
in particular, h^0(I_{S,P^5}(3)) is minimal
h^0(N_{S,P^5}) + 27 = 56
h^0(N_{S,X}) = 2
dim{[X] : S ⊂ X} >= 54
dim P(H^0(O_(P^5)(3))) = 55
codim{[X] : S ⊂ X} <= 1
-- function parameterCount() has terminated. --
(1, (28, 29, 2))
```

```
sage: X = fourfold('1',GF(33331)); X                                    # opti
Gushel-Mukai fourfold of discriminant 10(') containing a quadric surface in PP^8, cla
sage: X.parameter_count(verbose                                         # opti
-- running Macaulay2 function parameterCount()... --
S: smooth quadric surface in PP^8
X: GM fourfold containing S
Y: del Pezzo fivefold containing X
h^1(N_{S,Y}) = 0
h^0(N_{S,Y}) = 8
h^1(O_S(2)) = 0, and h^0(I_{S,Y}(2)) = 31 = h^0(O_Y(2)) - \chi(O_S(2));
in particular, h^0(I_{S,Y}(2)) is minimal
h^0(N_{S,Y}) + 30 = 38
h^0(N_{S,Y}) = 0
dim{[X] : S ⊂ X ⊂ Y} >= 38
dim P(H^0(O_Y(2))) = 39
codim{[X] : S ⊂ X ⊂ Y} <= 1
-- function parameterCount() has terminated. --
(1, (31, 8, 0))
```

↑ Back to top

random_coordinate_change()

Apply a random coordinate change on the ambient projective space of `self`.

EXAMPLES:

```
sage: from sage.misc.randstate import set_random_seed
sage: set_random_seed(1234567)
sage: X = fourfold(Veronese(2,2,KK=101)); X
Cubic fourfold of discriminant 20 = 3*12-4^2 containing a surface in PP^5 of degree 4
sage: X.surface().defining_polynomials()
(x4^2 - x3*x5,
x2*x4 - x1*x5,
x2*x3 - x1*x4,
x2^2 - x0*x5,
x1*x2 - x0*x4,
x1^2 - x0*x3)
sage: Y = X.random_coordinate_change(); Y
Cubic fourfold of discriminant 20 = 3*12-4^2 containing a surface in PP^5 of degree 4
sage: Y.surface().defining_polynomials()
(x2^2 - 28*x0*x3 + 31*x1*x3 + 15*x2*x3 + 32*x3^2 + 9*x0*x4 - 34*x1*x4 + 33*x2*x4 + 3
x1*x2 - 46*x0*x3 + 15*x1*x3 + 28*x2*x3 - 13*x3^2 - 32*x0*x4 - 44*x1*x4 + 27*x2*x4 + 1
x0*x2 + 6*x0*x3 - 9*x1*x3 - 47*x2*x3 + 34*x3^2 + 40*x0*x4 + 15*x1*x4 + 39*x2*x4 + 17
x1^2 - 11*x0*x3 + 44*x1*x3 + 30*x2*x3 + 26*x3^2 + 45*x0*x4 + 41*x1*x4 + 9*x2*x4 + 19
x0*x1 - 14*x0*x3 + 9*x1*x3 - 27*x2*x3 + 31*x3^2 + 48*x0*x4 + 28*x1*x4 - 9*x2*x4 - 41
x0^2 - 26*x0*x3 - x1*x3 - 6*x2*x3 + x3^2 - 4*x0*x4 + 9*x1*x4 - 9*x2*x4 + 17*x3*x4 - 2
```

surface()

Return the special surface contained in the fourfold.

OUTPUT:

`Embedded_projective_variety`, the surface of `self`.

EXAMPLES:

```
sage: S = surface(3,4)
sage: X = fourfold(S)
sage: X.surface() is S
True
```

sage.schemes.hodge_special_fourfolds.sff.PP(KK=33331, var='x')

Projective space of dimension `n` over `KK`.

EXAMPLES:

```
sage: PP(5,33331,var='t')
PP^5
sage: _.coordinate_ring()
Multivariate Polynomial Ring in t0, t1, t2, t3, t4, t5 over Finite Field of size 33331
sage: PP(5,33331,var='t') is PP(5,GF(33331),var='t')
True
```

class sage.schemes.hodge_special_fourfolds.sff.**Rational_map_between_embedded_projective_varieties**(X, Y, polys)

↑ Back to top

Bases: `SchemeMorphism_polynomial_projective_space_field`

The class of rational maps between closed subvarieties of projective spaces.

This is a subclass of the class `SchemeMorphism_polynomial_projective_space_field`. It is designed to provide better support for maps related to Hodge-special fourfolds.

Constructing a rational map between projective subvarieties.

> **Warning**
>
> You should not create objects of this class directly. The preferred method to construct such maps is to use `rational_map()`.

INPUT:

- `X` – the source variety (optional).
- `Y` – the target variety (optional).
- `polys` – a list of homogeneous polynomials of the same degree in the coordinate ring of `X`

OUTPUT:

The rational map from `X` to `Y` defined by `polys`.

If `X` and `Y` are not objects of the class `Embedded_projective_variety`, they will be replaced by `projective_variety(X)` and `projective_variety(Y)` (if any exception occurs), see `projective_variety()`.

EXAMPLES:

```
sage: X = PP(4,GF(33331))
sage: Y = PP(5,GF(33331))
sage: x0, x1, x2, x3, x4 = X.coordinate_ring().gens()
sage: f = rational_map(X, Y, [x3^2-x2*x4, x2*x3-x1*x4, x1*x3-x0*x4, x2^2-x0*x4, x1*x2-x0*
rational map defined by forms of degree 2
source: PP^4
target: PP^5
sage: g = f.make_dominant(); g
dominant rational map defined by forms of degree 2
source: PP^4
target: quadric hypersurface in PP^5
```

You can also convert such rational maps into `Macaulay2` objects.

```
sage: g_ = macaulay2(g); g_                                              # optional - macaul
multi-rational map consisting of one single rational map
source variety: PP^4
target variety: hypersurface in PP^5 defined by a form of degree 2

MultirationalMap (rational map from PP^4 to hypersurface in PP^5)
sage: g_.graph().last().inverse()                                        # optional - macaul
multi-rational map consisting of 2 rational maps
source variety: hypersurface in PP^5 defined by a form of degree 2
target variety: 4-dimensional subvariety of PP^4 x PP^5 cut out by 9 hypersurfaces of mul
dominance: true
degree: 1

MultirationalMap (birational map from hypersurface in PP^5 to 4-dimensional subvariety of
```

base_locus(verbose=None, algorithm='macaulay2')

Return the base locus of the rational map `self`.

OUTPUT:

`Embedded_projective_variety` – a subvariety of `self.source()`.

EXAMPLES:

```
sage: f = rational_map(Veronese(1,3),Veronese(1,3).point().defining_polynomials()).ma
```

```
dominant rational map defined by forms of degree 1
source: cubic curve of arithmetic genus 0 in PP^3 cut out by 3 hypersurfaces of degre
target: conic curve in PP^2
sage: f.base_locus(verbose=Fals                    # optional - macaulay2
empty subscheme of PP^3
sage: f                                            # optional - macaulay2
dominant morphism defined by forms of degree 1
source: cubic curve of arithmetic genus 0 in PP^3 cut out by 3 hypersurfaces of degre
target: conic curve in PP^2
degree sequence: (1, 1)
```

↑ Back to top

This is used internally by the method `is_morphism()`.

compose(f)

Return the composition of `self` with the rational map `f`.

INPUT:

- `f` `Rational_map_between_embedded_projective_varieties` – a rational map such that `f.source() == self.target()`.

OUTPUT:

`Rational_map_between_embedded_projective_varieties`, the composition of `self` with `f`.

EXAMPLES:

```
sage: f = rational_map(Veronese(1,4))
sage: g = rational_map(f.target().point())
sage: f.compose(g)
rational map defined by forms of degree 2
source: PP^4
target: PP^4
sage: _.projective_degrees()
[1, 2, 4, 4, 2]
```

image(algorithm=None)

Return the (closure of the) image of the rational map.

OUTPUT:

`Embedded_projective_variety`, the closure of the image of `self`, a subvariety of `self.target()`.

EXAMPLES:

```
sage: f = veronese(1,4)
sage: f.image()
curve of degree 4 and arithmetic genus 0 in PP^4 cut out by 6 hypersurfaces of degree
```

inverse(check=True, verbose=None, algorithm='macaulay2')

Return the inverse of the birational map.

OUTPUT:

`Rational_map_between_embedded_projective_varieties`, the inverse rational map of `self` if `self` is birational, otherwise an exception is raised.

EXAMPLES:

```
sage: f = veronese(1,5).make_dominant()
sage: g = f.inverse(); g                                    # optional - macaula
birational morphism defined by forms of degree 1
source: curve of degree 5 and arithmetic genus 0 in PP^5 cut out by 10 hypersurfaces
target: PP^1
degree sequence: (1, 1, 1, 1, 1)
sage: f.compose(g) == 1                                     # optional - macaula
True
```

With the input `algorithm='macaulay2'` the computation is transferred to `Macaulay2` (this is currently the only option).

`inverse_image(Z, trim=True)`

Return the (closure of the) inverse image of the variety `Z` via the rational map `self`.

INPUT:

↑ Back to top

- `Z` `Embedded_projective_variety` – a subvariety of `self.target().ambient()`.

OUTPUT:

`Embedded_projective_variety`, the closure of the inverse image of `Z` via the rational map `self`, a subvariety of `self.source()`.

EXAMPLES:

```
sage: f = rational_map(Veronese(1,4)).make_dominant(); f
dominant rational map defined by forms of degree 2
source: PP^4
target: quadric hypersurface in PP^5
sage: Z = f.target().empty().random(1,1,1,1); Z
line in PP^5
sage: f.inverse_image(Z)
0-dimensional subscheme of degree 2 in PP^4
```

`is_dominant()`

Return `True` if `self` is a dominant rational map, `False` otherwise.

OUTPUT:

`bool`, whether `self.image() == self.target()`

`is_morphism(verbose=None, algorithm='macaulay2')`

Return `True` if `self` is a morphism, `False` otherwise.

OUTPUT:

`bool`, whether `self.base_locus().dimension() == -1`

`make_dominant()`

Return a new rational map with the same source variety and same defining polynomials but with `self.target()` replaced by `self.image()`

EXAMPLES:

```
sage: f = veronese(1,4); f
rational map defined by forms of degree 4
source: PP^1
target: PP^4
sage: f.make_dominant()
dominant rational map defined by forms of degree 4
source: PP^1
target: curve of degree 4 and arithmetic genus 0 in PP^4 cut out by 6 hypersurfaces
```

`projective_degrees()`

Return the projective degrees of the rational map

> **Warning**
>
> Currently, this uses a probabilistic approach which could give wrong answers (especially over finite fields of small order).

OUTPUT:

A list of integers.

EXAMPLES:

```
sage: (t0,t1,t2,t3,t4,t5,t6) = PP(6).coordinate_ring().gens()
sage: f = rational_map(matrix([[t0,t1,t2,t3,t4],[t1,t2,t3,t4,t5],[t2,t3,t4,t5,t6]]).r
rational map defined by forms of degree 3
source: PP^6
target: PP^9
sage: f.projective_degrees()
```

```
[1, 3, 9, 17, 21, 15, 5]
```

random_coordinate_change()

↑ Back to top

Apply random coordinate changes o projective spaces of the source and target of `self`.

EXAMPLES:

```
sage: from sage.misc.randstate import set_random_seed
sage: set_random_seed(0)
sage: f = veronese(1,3,KK=GF(13)); f
rational map defined by forms of degree 3
source: PP^1
target: PP^3
sage: f.defining_polynomials()
(t1^3, t0*t1^2, t0^2*t1, t0^3)
sage: g = f.random_coordinate_change(); g
rational map defined by forms of degree 3
source: PP^1
target: PP^3
sage: g.defining_polynomials()
(-t0^3 - 6*t0^2*t1 - t0*t1^2 - 6*t1^3,
5*t0^3 - 4*t0^2*t1 - 5*t0*t1^2 - 2*t1^3,
4*t0^3 + 4*t0^2*t1 + t0*t1^2 + 4*t1^3,
-t0^3 + 3*t0^2*t1 + 4*t0*t1^2 - t1^3)
```

restriction(X)

Return the restriction of `self` to the variety `X`.

INPUT:

- `X` `Embedded_projective_variety` – a subvariety of `self.source()`.

OUTPUT:

`Rational_map_between_embedded_projective_varieties`, the restriction of `self` to `X`.

restriction_from_target(Y)

Return the restriction of `self` to the inverse image of the variety `Y`.

INPUT:

- `Y` `Embedded_projective_variety` – a subvariety of `self.target().ambient()`.

OUTPUT:

`Rational_map_between_embedded_projective_varieties`, the restriction of `self` to the inverse image of `Y`.

See also the methods: `inverse_image()` and `restriction()`.

EXAMPLES:

```
sage: f = veronese(2,2)
sage: Y = f.target().empty().random(1)
sage: f.restriction_from_target(Y)
rational map defined by forms of degree 2
source: conic curve in PP^2
target: hyperplane in PP^5
```

source()

Return the source of the rational map

OUTPUT:

`Embedded_projective_variety`, the source variety, which always coincides with `projective_variety(self.domain())`.

EXAMPLES:

```
sage: f = veronese(1,4)
sage: f.source()
PP^1
```


super()

Return the composition of `self` with the embedding of the target in the ambient space.

`Rational_map_between_embedded_projective_varieties`

EXAMPLES:

```
sage: f = veronese(1,4).make_dominant(); f
dominant rational map defined by forms of degree 4
source: PP^1
target: curve of degree 4 and arithmetic genus 0 in PP^4 cut out by 6 hypersurfaces o
sage: g = f.super(); g
rational map defined by forms of degree 4
source: PP^1
target: PP^4
image: curve of degree 4 and arithmetic genus 0 in PP^4 cut out by 6 hypersurfaces of
sage: g.super() is g
True
```

target()

Return the target of the rational map

OUTPUT:

`Embedded_projective_variety`, the target variety, which always coincides with `projective_variety(self.codomain())`.

EXAMPLES:

```
sage: f = veronese(1,4)
sage: f.target()
PP^4
```

to_built_in_map()

Return the same mathematical object but in the parent class.

OUTPUT:

`SchemeMorphism_polynomial_projective_space_field`

EXAMPLES:

```
sage: rational_map(Veronese(1,3))
rational map defined by forms of degree 2
source: PP^3
target: PP^2
sage: _.to_built_in_map()
Scheme morphism:
  From: Closed subscheme of Projective Space of dimension 3 over Finite Field of size
  0
  To:   Closed subscheme of Projective Space of dimension 2 over Finite Field of size
  0
  Defn: Defined on coordinates by sending (x0 : x1 : x2 : x3) to
        (x2^2 - x1*x3 : x1*x2 - x0*x3 : x1^2 - x0*x2)
```

sage.schemes.hodge_special_fourfolds.sff.Veronese(n, d, KK=33331, var='x')

Return the image of the Veronese embedding.

OUTPUT:

`Embedded_projective_variety`

EXAMPLES:

```
sage: Veronese(2,2)
surface in PP^5 of degree 4 and sectional genus 0 cut out by 6 hypersurfaces of degree 2
```

sage.schemes.hodge_special_fourfolds.sff.**fourfold**(S, X=None, V=None, check=True)

Construct Hodge-special fourfolds.

↑ Back to top

The typical input for this function is a trip[le] of a surface `S`, a fourfold `X` containing `S`, and a fivefold `V` containing `X`. The output is an object of the class `Hodge_special_fourfold`.

EXAMPLES:

```
sage: S = surface(4,5,1,KK=GF(65521)); S
rational surface in PP^6 of degree 7 and sectional genus 2 cut out by 8 hypersurfaces of
sage: X = S.random(2,3); X
complete intersection of type (2, 3) in PP^6
sage: V = X.random(2); V
quadric hypersurface in PP^6
sage: F = fourfold(S,X,V); F
Hodge-special fourfold of degree 6 in PP^6 containing a rational surface in PP^6 of degre
sage: F == X and F.surface() is S and F.ambient_fivefold() is V
True
sage: F.discriminant()
65
```

We can use this function to retrieve fourfolds constructed with `Macaulay2`.

```
sage: G = macaulay2('specialGushelMukaiFourfold schubertCycle({3,1},GG(1,4))')
sage: G.describe()
Special Gushel-Mukai fourfold of discriminant 10('')
containing a surface in PP^8 of degree 1 and sectional genus 0
cut out by 6 hypersurfaces of degree 1
and with class in G(1,4) given by s_(3,1)
Type: ordinary
(case 6 of Table 1 in arXiv:2002.07026)
sage: fourfold(G)
Gushel-Mukai fourfold of discriminant 10('') containing a plane in PP^8, class of the sur
sage: macaulay2(_) is G
True
```

Some constructions can be done by passing a description and an optional base field.

```
sage: fourfold('general cubic 4-fold of discriminant 38',GF(65521))
Cubic fourfold of discriminant 38 = 3*46-10^2 containing a surface in PP^5 of degree 10 a
```

Some calculations can be performed very fast using virtual fourfolds (although this may provide meaningless answers).

```
sage: S = surface(4,8,ambient=5,virtual=True); S
virtual rational surface in PP^5 of degree 8 and sectional genus 3 cut out by at least 13
sage: X = fourfold(S); X
Cubic fourfold of discriminant 14 = 3*26-8^2 containing a virtual rational surface in PP^

sage: S = surface(11,0,0,1, virtual=True, class_surfaces='K3'); S   # triple projection c
virtual K3 surface in PP^5 of degree 11 and sectional genus 8 cut out by at least 9 hyper
sage: X = fourfold(S); X
Cubic fourfold of discriminant 26 = 3*49-11^2 containing a virtual K3 surface in PP^5 of
```

sage.schemes.hodge_special_fourfolds.sff.**projective_variety**(I, PP=None)

Construct a projective variety.

INPUT:

`I` – A homogeneous ideal in a polynomial ring over a field, or the list of its generators.

`PP` – (Optional) an ambient projective space whose coordinate ring is the polynomial ring of `I`.

OUTPUT:

`Embedded_projective_variety`, the projective variety defined by `I`.

EXAMPLES:

```
sage: (x0,x1,x2,x3) = ProjectiveSpace(3, GF(101), 'x').gens()
sage: I = [x0^2-x1*x2, x1^3+x2^3+x3^3]
```

```
sage: X = projective_variety(I); X
curve of degree 6 and arithmetic genus 4 in PP^3 cut out by 2 hypersurfaces of degrees (2
```

You can also use this function to convert objects of the class `AlgebraicScheme_subscheme_projective` into objects of the class `Embedded_projective_variety`.


```
sage: Y = X.ambient_space().subscheme(I); Y
Closed subscheme of Projective Space of dimension 3 over Finite Field of size 101 defined
x0^2 - x1*x2,
x1^3 + x2^3 + x3^3
sage: projective_variety(Y)
curve of degree 6 and arithmetic genus 4 in PP^3 cut out by 2 hypersurfaces of degrees (2
sage: _ == X
True
```

sage.schemes.hodge_special_fourfolds.sff.**rational_map**(*args, **kwargs)

Construct a rational map from a projective variety to another.

INPUT:

- `X` – the source variety (optional).
- `Y` – the target variety (optional).
- `polys` – a list of homogeneous polynomials of the same degree in the coordinate ring of `X`

OUTPUT:

`Rational_map_between_embedded_projective_varieties`, the rational map from `X` to `Y` defined by `polys`.

EXAMPLES:

```
sage: x0, x1, x2, x3, x4 = PP(4,GF(33331)).coordinate_ring().gens()
sage: rational_map([x3^2-x2*x4, x2*x3-x1*x4, x1*x3-x0*x4, x2^2-x0*x4, x1*x2-x0*x3, x1^2-x
rational map defined by forms of degree 2
source: PP^4
target: PP^5
```

If we pass as input a projective variety and an integer `d`, we get the rational map defined by the hypersurfaces of degree `d` that contain the given variety.

```
sage: X = Veronese(1,4)
sage: rational_map(X,2)
rational map defined by forms of degree 2
source: PP^4
target: PP^5
```

You can also use this function to convert objects of the class `SchemeMorphism_polynomial_projective_space_field` into objects of the class `Rational_map_between_embedded_projective_varieties`.

```
sage: g = _.to_built_in_map()
sage: rational_map(g)
rational map defined by forms of degree 2
source: PP^4
target: PP^5
```

sage.schemes.hodge_special_fourfolds.sff.**surface**(KK, ambient, nodes, virtual, class_surfaces, *args)

Return a surface in a projective space of dimension `ambient` over the field `KK`.

The typical usage is as follows:

INPUT:

- a tuple (a,i,j,k,…) of integers.
- a base field `KK` or its characteristic.

OUTPUT:

`Embedded_projective_variety`, the rational surface over `KK` obtained as the image of the plane via the linear system of curves of degree `a` having `i` general base points of multiplicity 1, `j` general base points of multiplicity 2, `k` general base points of multiplicity 3, and so on.

EXAMPLES:

```
sage: surface(3,1,1)
rational surface in PP^5 of degree 4 and sectional genus 0 cut out by 6 hypersurfaces of
```

We can construct virtually the surface by passing the keyword argument `virtual=True`.

```
sage: S = surface(5,8,0,1, virtual=True); S
virtual rational surface in PP^6 of degree 8 and sectional genus 3 cut out by at least 7
sage: S.materialize()
rational surface in PP^6 of degree 8 and sectional genus 3 cut out by 7 hypersurfaces of
```

Using virtualization we can easily construct other types of surfaces besides rational ones. For instance, the code `surface(a,i,j,k,..., virtual=True, class_surfaces='K3')` returns the projection of a general K3 surface of genus `g` and degree `2g-2` from `i` general points of multiplicity 1, `j` general points of multiplicity 2, `k` general points of multiplicity 3, and so on. As a specific example, we now take four simple projections plus one 3rd projection plus one 5th projection of a K3 surface of genus 32.

```
sage: surface(32,4,0,1,0,1, virtual=True, class_surfaces='K3')
virtual K3 surface in PP^7 of degree 24 and sectional genus 19 cut out by at least 28 hyp
```

Equivalently, we can first construct a minimal K3 surface of genus 32 and then we take its projection.

```
sage: S = surface(32, virtual=True, class_surfaces='K3'); S
virtual K3 surface in PP^32 of degree 62 and sectional genus 32 cut out by at least 435 h
sage: S.projection(4,0,1,0,1)    # four simple projections + one 3rd projection + one 5th
virtual K3 surface in PP^7 of degree 24 and sectional genus 19 cut out by at least 28 hyp
```

`sage.schemes.hodge_special_fourfolds.sff.`**`update_macaulay2_packages`**`()`

Update some `Macaulay2` packages to their latest version.

Execute the command `update_macaulay2_packages()` to download in your current directory all the `Macaulay2` packages needed to the functions of this module. You don't need to do this if you use the development version of `Macaulay2`.

`sage.schemes.hodge_special_fourfolds.sff.`**`verbosity`**`(b)`

Change the default verbosity for some functions of this module.

Use `verbosity(True)` to produce output messages by default. Verbosity can be disabled in the specific function you use.

INPUT:

`bool`

`sage.schemes.hodge_special_fourfolds.sff.`**`veronese`**`(n, d, KK=33331, var='x')`

Return the Veronese embedding.

OUTPUT:

`Rational_map_between_embedded_projective_varieties`

EXAMPLES:

Made

```
sage: veronese(2,3)
rational map defined by forms of degree 3
source: PP^2
target: PP^9
```