# Hodge-special fourfolds

This module provides support for Hodge-special fourfolds, such as cubic fourfolds and Gushel-Mukai fourfolds.

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 (2023-05-14): 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 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 of degre
sage: T = X.K3(verbose=False); T
surface of degree 14 and sectional genus 8 in PP^8 cut out by 15 hypersurfaces of
sage: building = T.building() # a tuple of 4 objects obtained in the construction
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]); X
curve of degree 6 and arithmetic genus 4 in PP^3 cut out by 2 hypersurfaces of degrees
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: K = GF(65521)
sage: P4 = PP(4,K); P4
PP^4
sage: P4.empty()
empty subscheme of PP^4
sage: X = P4.empty().random(2,2,3)
sage: X
curve of degree 12 and arithmetic genus 13 in PP^4 cut out by 3 hypersurfaces of degree
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(verbose=False)
sage: p.is_subset(X)
True
```

**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
```

**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
```

**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
```

**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(Target=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
```

**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 de
```

**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
```

**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 of degree 9 and sectional genus 3 in PP^7 cut out by 12 hypersurfac

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: p = L.point()
sage: p.parametrize().source()
PP^0

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

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

sage: macaulay2(_).describe()
multi-rational map consisting of one single rational map
source variety: PP^2
target variety: surface in PP^5 cut out by 6 hypersurfaces of degree 2
base locus: empty subscheme of PP^2
dominance: true
multidegree: {1, 2, 4}
degree: 1
degree sequence (map 1/1): [2]
coefficient ring: ZZ/33331
```

**point**(verbose=None, UseMacaulay2=None)

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

EXAMPLES:

```
sage: X = Veronese(2,2)
sage: p = X.point()
sage: type(p) is type(X) and p.dimension() == 0 and p.degree() == 1 and p.is_subset
True
```

**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 complete intersection 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
```

**sectional_genus**()

Return the arithmetic genus of the sectional curve of the variety

OUTPUT:

An integer.

EXAMPLES:

```
sage: X = Veronese(2,2); X
surface of degree 4 and sectional genus 0 in PP^5 cut out by 6 hypersurfaces of deg
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 of degree 6 and sectional genus 1 in PP^5 cut out by 5 hy
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 3333
x2^2 - x1*x3,
x1*x2 - x0*x3,
x1^2 - x0*x2
```

**topological_euler_characteristic**(verbose=None, UseMacaulay2=False)

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 `UseMacaulay2=True` the computation is transferred to `Macaulay2`.

OUTPUT:

An integer.

EXAMPLES:

```
sage: X = Veronese(2,2); X
surface of degree 4 and sectional genus 0 in PP^5 cut out by 6 hypersurfaces of deg
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
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 1
sage: T = X.K3(verbose=False); T
surface of degree 10 and sectional genus 6 in PP^6 cut out by 6 hypersurfaces of de
sage: building = T.building() # a tuple of 4 objects obtained in the construction o
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 deg
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

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(5,7,0,1)
sage: X = fourfold(S); X
Complete intersection of 3 quadrics in PP^7 of discriminant 47 = 8*16-9^2 containing a
sage: X.surface()
rational surface of degree 9 and sectional genus 3 in PP^7 cut out by 12 hypersurfaces
sage: V = X.ambient_fivefold(); V
complete intersection of type (2, 2) in PP^7
```

**`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
```

**`detect_congruence`**`(Degree=None, verbose=None)`

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

In the current version, this runs the `Macaulay2` function `detectCongruence`, documented at https://faculty.math.illinois.edu/Macaulay2/doc/Macaulay2/share/doc/Macaulay2/SpecialFanoFourfolds

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, if known.

OUTPUT:

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

EXAMPLES:

```
sage: S = surface(3,1,1); S
rational surface of degree 4 and sectional genus 0 in PP^5 cut out by 6 hypersurfac
sage: X = fourfold(S)
sage: f = X.detect_congruence(1)
-- running Macaulay2 function detectCongruence()... --
warning: clearing value of symbol x0 to allow access to subscripted variables based
: debug with expression   debug 6010   or with command line option   --debug 6010
warning: clearing value of symbol x1 to allow access to subscripted variables based
: debug with expression   debug 5513   or with command line option   --debug 5513
-- function detectCongruence() has terminated. --
sage: f.check()
Congruence of 2-secant lines
to: rational surface of degree 4 and sectional genus 0 in PP^5 cut out by 6 hypersu
in: PP^5
sage: p = X.ambient_fivefold().point()
sage: f(p)
line in PP^5

sage: S = surface(5,7,0,1)
sage: X = fourfold(S)
sage: X
Complete intersection of 3 quadrics in PP^7 of discriminant 47 = 8*16-9^2 containin
sage: f = X.detect_congruence()
-- running Macaulay2 function detectCongruence()... --
number lines contained in the image of the quadratic map and passing through a gene
number 1-secant lines = 9
number 3-secant conics = 8
number 5-secant cubics = 1
-- function detectCongruence() has terminated. --

sage: f.check()
Congruence of 5-secant cubic curves
to: rational surface of degree 9 and sectional genus 3 in PP^7 cut out by 12 hypers
in: complete intersection of type (2, 2) in PP^7
sage: p = X.ambient_fivefold().point()
-- running Macaulay2 function point()... --
-- function point() has terminated. --
sage: f(p)
cubic curve of arithmetic genus 0 in PP^7 cut out by 7 hypersurfaces of degrees (1,

sage: X = fourfold("3-nodal septic scroll"); X
Cubic fourfold of discriminant 26 = 3*25-7^2 containing a surface of degree 7 and s
sage: assert(X.base_ring().characteristic() == 65521)
sage: X.detect_congruence().check()
-- running Macaulay2 function detectCongruence()... --
number lines contained in the image of the cubic map and passing through a general
number 2-secant lines = 7
number 5-secant conics = 1
-- function detectCongruence() has terminated. --
Congruence of 5-secant conics
to: surface of degree 7 and sectional genus 0 in PP^5 cut out by 13 hypersurfaces o
in: PP^5

sage: X = fourfold("prebuilt-example-in-P7",2); X
Complete intersection of 3 quadrics in PP^7 of discriminant 47 = 8*16-9^2 containin
sage: X.detect_congruence().check()
-- running Macaulay2 function detectCongruence()... --
number lines contained in the image of the quadratic map and passing through a gene
number 1-secant lines = 9
number 3-secant conics = 8
number 5-secant cubics = 1
-- function detectCongruence() has terminated. --
Congruence of 5-secant cubic curves
to: surface of degree 9 and sectional genus 3 in PP^7 cut out by 12 hypersurfaces o
in: complete intersection of type (2, 2) in PP^7
```

**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)
dominant rational map defined by forms of degree 2
source: PP^5
target: PP^4
```

**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 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))
sage: X.parameter_count()
-- 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(surface(1,ambient=7))
sage: X.parameter_count()
-- running Macaulay2 function parameterCount()... --
S: plane in PP^7
X: complete intersection of type (2,2,2) in PP^7
Y: complete intersection of type (2,2) in PP^7
X is a fourfold containing S which is a hypersurface of degree 2 in Y
h^1(N_{S,Y}) = 0
h^0(N_{S,Y}) = 3
h^1(O_S(2)) = 0, and h^0(I_{S,Y}(2)) = 28 = h^0(O_Y(2)) - \chi(O_S(2));
in particular, h^0(I_{S,Y}(2)) is minimal
h^0(N_{S,Y}) + 27 = 30
h^0(N_{S,X}) = 0
dim{[X] : S ⊂ X ⊂ Y} >= 30
dim P(H^0(O_Y(2))) = 33
codim{[X] : S ⊂ X ⊂ Y} <= 3
[parameterCount in the ambient PP^7: (3, (30, 15, 0))]
-- function parameterCount() has terminated. --
(3, (28, 3, 0))
```

**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=Finite Field of size 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)

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-
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_
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()
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 
dominance: true
degree: 1

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

**base_locus**(verbose=None, UseMacaulay2=True)

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(verbose=false).defining_po
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 deg
target: conic curve in PP^2
sage: f.base_locus(verbose=False)
empty subscheme of PP^3
sage: f
dominant morphism defined by forms of degree 1
source: cubic curve of arithmetic genus 0 in PP^3 cut out by 3 hypersurfaces of deg
target: conic curve in PP^2
degree sequence: (1, 1)
```

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**()

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 degr
```

**inverse**(check=True, verbose=None, UseMacaulay2=True)

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
-- running Macaulay2 function inverse()... --
-- function inverse() has successfully terminated. --
birational morphism defined by forms of degree 1
source: curve of degree 5 and arithmetic genus 0 in PP^5 cut out by 10 hypersurface
target: PP^1
degree sequence: (1, 1, 1, 1, 1)
sage: f.compose(g) == 1
-- running Macaulay2 operator == between rational maps... --
-- Macaulay2 computation has successfully terminated. --
True
```

With the input `UseMacaulay2=True` the computation is transferred to `Macaulay2`.

**inverse_image**(Z, trim=True)

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

INPUT:

- `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: self = rational_map(Veronese(1,4)).make_dominant()
sage: Z = self.target().empty().random(1,1,1,1)
sage: self.inverse_image(Z)
0-dimensional subscheme of degree 2 in PP^4

sage: f = self.inverse()
-- running Macaulay2 function inverse()... --
-- function inverse() has successfully terminated. --
sage: W = f.target().empty().random(1,1,1,1)
sage: f.inverse_image(W).describe()
dim:................. 0
codim:............... 5
degree:.............. 1
```

**is_dominant**()

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

OUTPUT:

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

**is_morphism**(verbose=None, UseMacaulay2=True)

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]]
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]
```

**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`.

**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
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
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 si
  0
  To:   Closed subscheme of Projective Space of dimension 2 over Finite Field of si
  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=Finite Field of size 33331, var='x')**

Return the image of the Veronese embedding.

OUTPUT:

`Embedded_projective_variety`

EXAMPLES:

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

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

Construct Hodge-special fourfolds.

`sage.schemes.hodge_special_fourfolds.sff.projective_variety(I)`

Construct a projective variety.

INPUT:

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

OUTPUT:

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

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

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
sage: Y = X.to_built_in_variety(); Y
Closed subscheme of Projective Space of dimension 3 over Finite Field of size 101 defi
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
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^
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 `Degree`, we get the rational map defined by the hypersurfaces of degree `Degree` 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, *args)`

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

INPUT:

- a tuple (a,i,j,k,…) of integers.

OUTPUT:

`Embedded_projective_variety`, the rational surface 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 of degree 4 and sectional genus 0 in PP^5 cut out by 6 hypersurfaces (
```

`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(False)` to suppress messages. Verbosity can be enabled in the specific function you use.

INPUT:

`bool`

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

Return the Veronese embedding.

OUTPUT:

`Rational_map_between_embedded_projective_varieties`

EXAMPLES:

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