Sage Reference Manual: Differential Forms

Release 7.6

The Sage Development Team

CONTENTS

| 1 | Open subset of Euclidian space with coordinates | 1 |
|---|---|----|
| 2 | Algebra of differential forms | 3 |
| 3 | Elements of the algebra of differential forms | 5 |
| 4 | Indices and Tables | 15 |

CHAPTER

ONE

OPEN SUBSET OF EUCLIDIAN SPACE WITH COORDINATES

An open subset of Euclidian space with a specific set of coordinates. This is the background on which differential forms can be defined.

AUTHORS:

• Joris Vankerschaver (2010-07-25)

EXAMPLES:

```
sage: x, y, z = var('x, y, z')
sage: S = CoordinatePatch((x, y, z)); S
Open subset of R^3 with coordinates x, y, z
```

```
sage: u, v = var('u, v')
sage: S = CoordinatePatch((u, v)); S
Open subset of R^2 with coordinates u, v
```

TODO:

Add functionality for metric tensors

Construct a coordinate patch, i.e. an open subset of Euclidian space with a given set of coordinates.

EXAMPLES:

```
sage: x, y, z = var('x, y, z')
sage: S = CoordinatePatch((x, y, z)); S
Open subset of R^3 with coordinates x, y, z

sage: u, v = var('u, v')
sage: T = CoordinatePatch((u, v)); T
Open subset of R^2 with coordinates u, v
sage: loads(T.dumps()) == T
True
```

In a future release, it will be possible to specify a metric tensor on a coordinate patch. For now, providing any kind of metric raises an exception:

```
sage: x, y, z = var('x, y, z')
sage: m = matrix(SR, 3)
sage: S = CoordinatePatch((x, y, z), metric=m)
Traceback (most recent call last):
```

```
...
NotImplementedError: Metric geometry not supported yet.
```

coordinate (i=0)

Return the i^{th} coordinate on self

INPUT:

•i - integer (optional, default 0)

EXAMPLES:

```
sage: x, y, z = var('x, y, z')
sage: S = CoordinatePatch((x, y, z)); S
Open subset of R^3 with coordinates x, y, z
sage: S.coordinate(0)
x
sage: S.coordinate(1)
y
sage: S.coordinate(2)
z
```

coordinates ()

Return coordinates on this coordinate patch.

OUTPUT:

•list - a list of coordinates on this space.

EXAMPLES:

```
sage: x, y, z = var('x, y, z')
sage: S = CoordinatePatch((x, y, z)); S
Open subset of R^3 with coordinates x, y, z
sage: S.coordinates()
(x, y, z)
```

dim ()

Return the dimension of this coordinate patch, i.e. the dimension of the Euclidian space of which this coordinate patch is an open subset.

```
sage: a, b, c, d, e = var('a, b, c, d, e')
sage: U = CoordinatePatch((a, b, c, d, e)); U
Open subset of R^5 with coordinates a, b, c, d, e
sage: U.dim()
5
```

ALGEBRA OF DIFFERENTIAL FORMS

Algebra of differential forms defined on a CoordinatePatch (an open subset of Euclidian space, see CoordinatePatch for details).

AUTHORS:

• Joris Vankerschaver (2010-05-26)

Todo

- · Allow for forms with values in a vector space
- Incorporate Kahler differentials

REFERENCES:

- R. Abraham, J. E. Marsden, and T. S. Ratiu: Manifolds, tensor analysis, and applications. Springer-Verlag 1988, texts in Applied Mathematical Sciences, volume 75, 2nd edition.
- http://en.wikipedia.org/wiki/Differential_form

```
class sage.tensor.differential_forms. DifferentialForms ( coordinate_patch=None)
     Bases: sage.rings.ring.Algebra
```

The algebra of all differential forms on an open subset of Euclidian space of arbitrary dimension.

EXAMPLES:

To define an algebra of differential forms, first create a coordinate patch:

```
sage: p, q = var('p, q')
sage: U = CoordinatePatch((p, q)); U
Open subset of R^2 with coordinates p, q
sage: F = DifferentialForms(U); F
Algebra of differential forms in the variables p, q
```

If no coordinate patch is supplied, a default one (using the variables x, y, z) will be used:

```
sage: F = DifferentialForms(); F
Algebra of differential forms in the variables x, y, z
```

Element

```
alias of DifferentialForm
```

```
base_space ( )
```

Return the coordinate patch on which this algebra is defined.

```
sage: x, y, z = var('x, y, z')
sage: U = CoordinatePatch((x, y, z)); U
Open subset of R^3 with coordinates x, y, z
sage: F = DifferentialForms(U); F
Algebra of differential forms in the variables x, y, z
sage: F.base_space()
Open subset of R^3 with coordinates x, y, z
```

gen (i=0)

Return the i^{th} generator of self. This is a one-form, more precisely the exterior derivative of the i-th coordinate.

INPUT:

•i - integer (optional, default 0)

EXAMPLES:

```
sage: x, y, z = var('x, y, z')
sage: U = CoordinatePatch((x, y, z)); U
Open subset of R^3 with coordinates x, y, z
sage: F = DifferentialForms(U); F
Algebra of differential forms in the variables x, y, z
sage: F.gen(0)
dx
sage: F.gen(1)
dy
sage: F.gen(2)
```

gens ()

Return a list of the generators of self.

EXAMPLES:

```
sage: x, y, z = var('x, y, z')
sage: U = CoordinatePatch((x, y, z)); U
Open subset of R^3 with coordinates x, y, z
sage: F = DifferentialForms(U); F
Algebra of differential forms in the variables x, y, z
sage: F.gens()
(dx, dy, dz)
```

ngens ()

Return the number of generators of this algebra.

```
sage: x, y, z = var('x, y, z')
sage: U = CoordinatePatch((x, y, z)); U
Open subset of R^3 with coordinates x, y, z
sage: F = DifferentialForms(U); F
Algebra of differential forms in the variables x, y, z
sage: F.ngens()
3
```

CHAPTER

THREE

ELEMENTS OF THE ALGEBRA OF DIFFERENTIAL FORMS

AUTHORS:

• Joris Vankerschaver (2010-07-25)

```
class sage.tensor.differential_form_element. DifferentialForm ( parent, fun=None)
```

Bases: sage.structure.element.AlgebraElement

Differential form class.

EXAMPLES:

In order to instantiate differential forms of various degree, we begin by specifying the CoordinatePatch on which they live, as well as their parent DifferentialForms algebra.

```
sage: x, y, z = var('x, y, z')
sage: U = CoordinatePatch((x, y, z))
sage: F = DifferentialForms(U)
sage: form1 = DifferentialForm(F, 0, sin(x*y)); form1
sin(x*y)
```

In the previous example, we created a zero-form from a given function. To create forms of higher degree, we can use the subscript operator access the various components:

```
sage: form2 = DifferentialForm(F, 1); form2
0
sage: form2[0] = 1
sage: form2[1] = exp(cos(x))
sage: form2[2] = 1/ln(y)
sage: form2
1/log(y)*dz + dx + e^cos(x)*dy
```

We may calculate the exterior derivative of a form, and observe that applying the exterior derivative twice always yields zero:

```
sage: dform = form1.diff(); dform
y*cos(x*y)*dx + x*cos(x*y)*dy
sage: dform.diff()
0
```

As can be seen from the previous example, the exterior derivative increases the degree of a form by one:

```
sage: form2.degree()
1
sage: form2.diff().degree()
2
```

The d function provides a convenient shorthand for applying the diff member function. Since d appears in other areas of mathematics as well, this function is not imported in the global namespace automatically:

```
sage: from sage.tensor.differential_form_element import d
sage: form2
1/log(y)*dz + dx + e^cos(x)*dy
sage: d(form2)
-1/(y*log(y)^2)*dy/\dz + -e^cos(x)*sin(x)*dx/\dy
sage: form2.diff()
-1/(y*log(y)^2)*dy/\dz + -e^cos(x)*sin(x)*dx/\dy
sage: d(form1) == form1.diff()
True
```

The wedge product of two forms can be computed by means of the wedge member function:

```
sage: form1 = DifferentialForm(F, 2)
sage: form1[0, 1] = exp(z); form1
e^z*dx/\dy
sage: form2 = DifferentialForm(F, 1)
sage: form2[2] = exp(-z)
sage: form1.wedge(form2)
dx/\dy/\dz
```

For this member function, there exists again a procedural function which is completely equivalent:

```
sage: from sage.tensor.differential_form_element import wedge
sage: form1.wedge(form2)
dx/\dy/\dz
sage: wedge(form1, form2)
dx/\dy/\dz
sage: form1.wedge(form2) == wedge(form1, form2)
True
```

NOTES:

Differential forms are stored behind the screens as dictionaries, where the keys are the subscripts of the non-zero components, and the values are those components.

For example, on a space with coordinates x, y, z, the form

```
f = \sin(x^*y) dx \wedge dy + \exp(z) dy \wedge dz
```

would be represented as the dictionary

```
\{(0, 1): \sin(x*y), (1, 2): \exp(z)\}.
```

Most differential forms are "sparse" in the sense that most of their components are zero, so that this representation is more efficient than storing all of the components in a vector.

abs ()

Method not defined for differential forms.

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.abs()
Traceback (most recent call last):
...
NotImplementedError: Absolute value not defined for differential forms.
```

degree ()

Return the degree of self.

EXAMPLES:

```
sage: F = DifferentialForms(); F
Algebra of differential forms in the variables x, y, z
sage: f = DifferentialForm(F, 2)
sage: f[1, 2] = x; f
x*dy/\dz
sage: f.degree()
2
```

The exterior differential increases the degree of forms by one:

```
sage: g = f.diff(); g
dx/\dy/\dz
sage: g.degree()
3
```

derivative (*args, **kwargs)

Compute the exterior derivative of self. This is the same as calling the diff member function.

EXAMPLES:

```
sage: x, y = var('x, y')
sage: U = CoordinatePatch((x, y))
sage: F = DifferentialForms(U)
sage: q = DifferentialForm(F, 1)
sage: q[0] = -y/2
sage: q[1] = x/2
sage: q.diff()
dx/\dy
sage: q.derivative()
dx/\dy
```

Invoking diff on a differential form has the same effect as calling this member function:

```
sage: diff(q)
dx/\dy
sage: diff(q) == q.derivative()
True
```

When additional arguments are supplied to diff, an error is raised, since only the exterior derivative has intrinsic meaning while derivatives with respect to the coordinate variables (in whichever way) are coordinate dependent, and hence not intrinsic.

```
sage: diff(q, x)
Traceback (most recent call last):
...
ValueError: Differentiation of a form does not take any arguments.
```

diff()

Compute the exterior differential of self.

```
sage: x, y, z = var('x, y, z')
sage: F = DifferentialForms()
```

```
sage: f = DifferentialForm(F, 0, sin(x*y)); f
sin(x*y)
sage: f.diff()
y*cos(x*y)*dx + x*cos(x*y)*dy
sage: g = DifferentialForm(F, 1)
sage: g[0] = y/2
sage: g[1] = -x/2
sage: g
1/2*y*dx + -1/2*x*dy
sage: g.diff()
-1*dx/\dy
sage: h = DifferentialForm(F, 2)
sage: h[0, 1] = exp(z)
sage: h.diff()
e^z*dx/\dy/\dz
```

The square of the exterior differential operator is identically zero:

```
sage: f
sin(x*y)
sage: f.diff()
y*cos(x*y)*dx + x*cos(x*y)*dy
sage: f.diff().diff()
0

sage: g.diff().diff()
```

The exterior differential operator is a derivation of degree one on the space of differential forms. In this example we import the operator d() as a short-hand for having to call the diff() member function.

```
sage: from sage.tensor.differential_form_element import d
sage: d(f)
y*cos(x*y)*dx + x*cos(x*y)*dy

sage: d(f).wedge(g) + f.wedge(d(g))
(-x*y*cos(x*y) - sin(x*y))*dx/\dy
sage: d(f.wedge(g))
(-x*y*cos(x*y) - sin(x*y))*dx/\dy

sage: d(f.wedge(g)) == d(f).wedge(g) + f.wedge(d(g))
True
```

is_zero ()

Return True if self is the zero form.

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1); f
0
sage: f.is_zero()
True
sage: f[1] = 1
sage: f.is_zero()
False
sage: f.diff()
0
```

```
sage: f.diff().is_zero()
True
```

leading_coefficient (key=None)

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.leading_coefficient()
Traceback (most recent call last):
...
NotImplementedError: leading_coefficient not defined for differential forms.
```

leading_item (key=None)

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.leading_item()
Traceback (most recent call last):
...
NotImplementedError: leading_item not defined for differential forms.
```

leading_monomial (key=None)

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.leading_monomial()
Traceback (most recent call last):
...
NotImplementedError: leading_monomial not defined for differential forms.
```

leading_support (key=None)

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.leading_support()
Traceback (most recent call last):
...
NotImplementedError: leading_support not defined for differential forms.
```

leading_term (key=None)

Method not defined for differential forms.

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.leading_term()
```

```
Traceback (most recent call last):
...
NotImplementedError: leading_term not defined for differential forms.
```

$map_coefficients(f)$

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.map_coefficients(lambda x: x)
Traceback (most recent call last):
...
NotImplementedError: map_coefficients not defined for differential forms.
```

$map_item(f)$

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.map_item(lambda x: x)
Traceback (most recent call last):
...
NotImplementedError: map_item not defined for differential forms.
```

map_support (f)

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.map_support(lambda x: x)
Traceback (most recent call last):
...
NotImplementedError: map_support not defined for differential forms.
```

trailing_coefficient (key=None)

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.trailing_coefficient()
Traceback (most recent call last):
...
NotImplementedError: trailing_coefficient not defined for differential forms.
```

trailing_item (key=None)

Method not defined for differential forms.

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
```

```
sage: f.trailing_item()
Traceback (most recent call last):
...
NotImplementedError: leading_coefficient not defined for differential forms.
```

trailing_monomial (key=None)

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.trailing_monomial()
Traceback (most recent call last):
...
NotImplementedError: trailing_monomial not defined for differential forms.
```

trailing_support (key=None)

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.trailing_support()
Traceback (most recent call last):
...
NotImplementedError: trailing_support not defined for differential forms.
```

trailing_term (key=None)

Method not defined for differential forms.

EXAMPLES:

```
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f.trailing_term()
Traceback (most recent call last):
...
NotImplementedError: trailing_term not defined for differential forms.
```

wedge (other)

Returns the wedge product of self and other.

```
sage: x, y, z = var('x, y, z')
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f[0] = x^2
sage: f[1] = y
sage: f
x^2*dx + y*dy
sage: g = DifferentialForm(F, 1)
sage: g[2] = z^3
sage: g
z^3*dz
sage: f.wedge(g)
y*z^3*dy/\dz + x^2*z^3*dx/\dz
```

The wedge product is graded commutative:

```
sage: f.wedge(g)
y*z^3*dy/\dz + x^2*z^3*dx/\dz
sage: g.wedge(f)
-y*z^3*dy/\dz + -x^2*z^3*dx/\dz
sage: f.wedge(f)
0
```

When the wedge product of forms belonging to different algebras is computed, an error is raised:

class sage.tensor.differential_form_element. DifferentialFormFormatter (space)

This class contains all the functionality to print a differential form in a graphically pleasing way. This class is called by the latex and repr methods of the DifferentialForm class.

In a nutshell (see the documentation of DifferentialForm for more details), differential forms are represented internally as a dictionary, where the keys are tuples representing the non-zero components of the form and the values are the component functions. The methods of this class create string and latex representations out of the specification of a subscript and a component function.

EXAMPLES:

```
sage: from sage.tensor.differential_form_element import DifferentialFormFormatter
sage: x, y, z = var('x, y, z')
sage: U = CoordinatePatch((x, y, z))
sage: D = DifferentialFormFormatter(U)
sage: D.repr((0, 2), sin(x*y))
'sin(x*y)*dx/\\dz'
sage: D.latex((0, 2), sin(x*y))
'\\sin\\left(x y\\right) d x \\wedge d z'
sage: D.latex((1, 2), exp(z))
'e^{z} d y \\wedge d z'
```

latex (comp, fun)

Latex representation of a primitive differential form, i.e. a function times a wedge product of d's of the coordinate functions.

INPUT:

- •comp a subscript of a differential form.
- •fun the component function of this form.

EXAMPLES:

repr (comp, fun)

String representation of a primitive differential form, i.e. a function times a wedge product of d's of the coordinate functions.

INPUT:

- •comp a subscript of a differential form.
- •fun the component function of this form.

EXAMPLES:

```
sage: from sage.tensor.differential_form_element import_

→DifferentialFormFormatter
sage: x, y, z = var('x, y, z')
sage: U = CoordinatePatch((x, y, z))
sage: D = DifferentialFormFormatter(U)
sage: D.repr((0, 1), z^3)
'z^3*dx/\\dy'
```

sage.tensor.differential_form_element.d (form)

Returns the exterior derivative of a given form, i.e. calls the diff() member function.

EXAMPLES:

```
sage: from sage.tensor.differential_form_element import d
sage: x, y, z = var('x, y, z')
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f[2] = cos(x); f
cos(x)*dz
sage: d(f)
-sin(x)*dx/\dz
sage: f.diff()
-sin(x)*dx/\dz
sage: d(f) == f.diff()
True
```

```
sage.tensor.differential form element. sort subscript (subscript)
```

A subscript is a range of integers. This function sorts a subscript in the sense of arranging it in ascending order. The return values are the sign of the subscript and the sorted subscript, where the sign is defined as follows:

1.sign == 0 if two or more entries in the subscript were equal.

2.sign == +1, -1 if a positive (resp. negative) permutation was used to sort the subscript.

INPUT:

•subscript - a subscript, i.e. a range of not necessarily distinct integers

OUTPUT:

- •Sign of the permutation used to arrange the subscript, where 0 means that the original subscript had two or more entries that were the same
- •Sorted subscript.

EXAMPLES:

```
sage: from sage.tensor.differential_form_element import sort_subscript
sage: sort_subscript((1, 3, 2))
(-1, (1, 2, 3))
sage: sort_subscript((1, 3))
(1, (1, 3))
sage: sort_subscript((4, 2, 7, 9, 8))
(1, (2, 4, 7, 8, 9))
```

 ${\tt sage.tensor.differential_form_element.wedge}$ (${\it left, right}$)

Computes the wedge product of two forms, i.e. calls the wedge() member function.

```
sage: from sage.tensor.differential_form_element import wedge
sage: x, y, z = var('x, y, z')
sage: F = DifferentialForms()
sage: f = DifferentialForm(F, 1)
sage: f[2] = cos(x); f
cos(x)*dz
sage: g = DifferentialForm(F, 1)
sage: g[1] = sin(y); g
sin(y)*dy
sage: wedge(f, g)
-cos(x)*sin(y)*dy/\dz
sage: f.wedge(g)
-cos(x)*sin(y)*dy/\dz
sage: wedge(f, g) == f.wedge(g)
True
```

CHAPTER

FOUR

INDICES AND TABLES

- Index
- Module Index
- Search Page

PYTHON MODULE INDEX

t

```
sage.tensor.coordinate_patch, 1
sage.tensor.differential_form_element, 5
sage.tensor.differential_forms, 3
```

18 Python Module Index

```
Α
abs() (sage.tensor.differential form element.DifferentialForm method), 6
В
base_space() (sage.tensor.differential_forms.DifferentialForms method), 3
C
coordinate() (sage.tensor.coordinate_patch.CoordinatePatch method), 2
CoordinatePatch (class in sage.tensor.coordinate_patch), 1
coordinates() (sage.tensor.coordinate patch.CoordinatePatch method), 2
D
d() (in module sage.tensor.differential_form_element), 13
degree() (sage.tensor.differential form element.DifferentialForm method), 6
derivative() (sage.tensor.differential_form_element.DifferentialForm method), 7
diff() (sage.tensor.differential_form_element.DifferentialForm method), 7
DifferentialForm (class in sage.tensor.differential form element), 5
DifferentialFormFormatter (class in sage.tensor.differential_form_element), 12
DifferentialForms (class in sage.tensor.differential_forms), 3
dim() (sage.tensor.coordinate_patch.CoordinatePatch method), 2
E
Element (sage.tensor.differential_forms.DifferentialForms attribute), 3
G
gen() (sage.tensor.differential_forms.DifferentialForms method), 4
gens() (sage.tensor.differential_forms.DifferentialForms method), 4
is_zero() (sage.tensor.differential_form_element.DifferentialForm method), 8
latex() (sage.tensor.differential form element.DifferentialFormFormatter method), 12
leading coefficient() (sage.tensor.differential form element.DifferentialForm method), 9
leading_item() (sage.tensor.differential_form_element.DifferentialForm method), 9
leading_monomial() (sage.tensor.differential_form_element.DifferentialForm method), 9
leading_support() (sage.tensor.differential_form_element.DifferentialForm method), 9
```

```
leading term() (sage.tensor.differential form element.DifferentialForm method), 9
M
map coefficients() (sage.tensor.differential form element.DifferentialForm method), 10
map_item() (sage.tensor.differential_form_element.DifferentialForm method), 10
map_support() (sage.tensor.differential_form_element.DifferentialForm method), 10
Ν
ngens() (sage.tensor.differential forms.DifferentialForms method), 4
R
repr() (sage.tensor.differential_form_element.DifferentialFormFormatter method), 13
sage.tensor.coordinate_patch (module), 1
sage.tensor.differential_form_element (module), 5
sage.tensor.differential_forms (module), 3
sort_subscript() (in module sage.tensor.differential_form_element), 13
Т
trailing_coefficient() (sage.tensor.differential_form_element.DifferentialForm method), 10
trailing_item() (sage.tensor.differential_form_element.DifferentialForm method), 10
trailing_monomial() (sage.tensor.differential_form_element.DifferentialForm method), 11
trailing_support() (sage.tensor.differential_form_element.DifferentialForm method), 11
trailing_term() (sage.tensor.differential_form_element.DifferentialForm method), 11
W
wedge() (in module sage.tensor.differential_form_element), 14
wedge() (sage.tensor.differential_form_element.DifferentialForm method), 11
```

20 Index