Manifold tutorial

This notebook provides a short introduction to differentiable manifolds in SageMath. The tools described below have been implemented through the SageManifolds project.

If you are new to SageMath, you may first take a look at this first contact tutorial.

This notebook is valid for version 9.2 or higher of SageMath:

In [1]:
```
version()
```

Out[1]:
```
'SageMath version 10.5, Release Date: 2024-12-04'
```

First we set up the notebook to display mathematical objects using LaTeX rendering:

In [2]:
```
%display latex
```

Defining a manifold

As an example let us define a differentiable manifold of dimension 3 over $\mathbb{R}$:

In [3]:
```
M = Manifold(3, 'M', latex_name=r'\mathcal{M}', start_index=1)
```

- The first argument, `3`, is the manifold dimension; it can be any positive integer.
- The second argument, `'M'`, is a string defining the manifold's name; it may be different from the symbol set on the left-hand side of the = sign (here `M`): the latter stands for the Python variable that refers to the manifold object in the computer memory, while the string `'M'` is the mathematical symbol chosen for the manifold.
- The optional argument `latex_name=r'\mathcal{M}'` sets the LaTeX symbol to display the manifold. Note the letter 'r' in front on the first quote: it indicates that the string is a *raw* one, so that the backslash character in `\mathcal` is considered as an ordinary character (otherwise, the backslash is used to escape some special characters). If the argument `latex_name` is not provided by the user, it is set to the string used as the second argument (here `'M'`)
- The optional argument `start_index=1` defines the range of indices to be used for tensor components on the manifold: setting it to 1 means that indices will range in $\{1, 2, 3\}$. The default value is `start_index=0`.

Note that the default base field is $\mathbb{R}$. If we would have used the optional argument `field='complex'`, we would have defined a manifold over $\mathbb{C}$. See the list of all options for more details.

If we ask for `M`, it is displayed via its LaTeX symbol:

In [4]:
```
M
```

Out[4]: $\mathcal{M}$

If we use the function `print()` instead, we get a short description of the object:

In [5]:
```
print(M)
```

```
3-dimensional differentiable manifold M
```

Via the function `type()`, we get the type of the Python object corresponding to M (here the Python class `DifferentiableManifold_with_category`):

In [6]:
```
print(type(M))
```

```
<class 'sage.manifolds.differentiable.manifold.DifferentiableManifold_with_categor
y'>
```

We may also ask for the category of `M` and see that it is the category of smooth manifolds over $\mathbb{R}$:

In [7]:
```
category(M)
```

Out[7]: $\mathbf{Smooth}_{\mathbf{R}}$

The indices on the manifold are generated by the method `irange()`, to be used in loops:

In [8]:
```
[i for i in M.irange()]
```

Out[8]: $[1, 2, 3]$

If the parameter `start_index` had not been specified, the default range of the indices would have been $\{0, 1, 2\}$ instead:

In [9]:
```
M0 = Manifold(3, 'M', latex_name=r'\mathcal{M}')
[i for i in M0.irange()]
```

Out[9]: $[0, 1, 2]$

Defining a chart on the manifold

Let us assume that the manifold $\mathcal{M}$ can be covered by a single chart (other cases are discussed below); the chart is declared as follows:

In [10]:
```
X.<x,y,z> = M.chart()
```

The writing `.<x,y,z>` in the left-hand side means that the Python variables `x`, `y` and `z` are set to the three coordinates of the chart. This allows one to refer subsequently to the coordinates by their names.

In this example, the function `chart()` has no arguments, which implies that the coordinate symbols will be `x`, `y` and `z` (i.e. exactly the characters set in the `<...>` operator) and that each coordinate range is $(-\infty, +\infty)$. For other cases, an argument must be passed to `chart()` to specify the coordinate symbols and range, as well as the LaTeX symbol of a coordinate if the latter is different from the coordinate name (an example will be provided below).

The chart is displayed as a pair formed by the open set covered by it (here the whole manifold) and the coordinates:

In [11]:
```
print(X)
```

```
Chart (M, (x, y, z))
```

In [12]:
```
X
```

Out[12]: $(\mathcal{M}, (x, y, z))$

The coordinates can be accessed individually, by means of their indices, following the convention defined by `start_index=1` in the manifold's definition:

In [13]:
```
X[1]
```
Out[13]: x

In [14]:
```
X[2]
```
Out[14]: y

In [15]:
```
X[3]
```
Out[15]: z

The full set of coordinates is obtained by means of the operator `[:]` :

In [16]:
```
X[:]
```
Out[16]: (x, y, z)

Thanks to the operator `<x,y,z>` used in the chart declaration, each coordinate can be accessed directly via its name:

In [17]:
```
z is X[3]
```
Out[17]: True

Coordinates are SageMath symbolic expressions:

In [18]:
```
type(z)
```
Out[18]: `<class 'sage.symbolic.expression.Expression'>`

Functions of the chart coordinates

Real-valued functions of the chart coordinates (mathematically speaking, *functions defined on the chart codomain*) are generated via the method `function()` acting on the chart:

In [19]:
```
f = X.function(x+y^2+z^3)
f
```
Out[19]: $z^3 + y^2 + x$

In [20]:
```
f.display()
```
Out[20]: $(x, y, z) \mapsto z^3 + y^2 + x$

In [21]:
```
f(1,2,3)
```
Out[21]: 32

They belong to the class `ChartFunction` (actually the subclass `ChartFunctionRing_with_category.element_class`):

In [22]:
```
print(type(f))
```
```
<class 'sage.manifolds.chart_func.ChartFunctionRing_with_category.element_class'>
```

and differ from SageMath standard symbolic functions by automatic simplifications in all operations. For instance, adding the two symbolic functions

In [23]:
```
f0(x,y,z) = cos(x)^2; g0(x,y,z) = sin(x)^2
```

results in

In [24]:
```
f0 + g0
```

Out[24]: $(x, y, z) \mapsto \cos(x)^2 + \sin(x)^2$

while the sum of the corresponding functions in the class `ChartFunction` is automatically simplified:

In [25]:
```
f1 = X.function(cos(x)^2); g1 = X.function(sin(x)^2)
f1 + g1
```

Out[25]: 1

To get the same output with symbolic functions, one has to invoke the method `simplify_trig()`:

In [26]:
```
(f0 + g0).simplify_trig()
```

Out[26]: $(x, y, z) \mapsto 1$

Another difference regards the display; if we ask for the symbolic function `f0`, we get

In [27]:
```
f0
```

Out[27]: $(x, y, z) \mapsto \cos(x)^2$

while if we ask for the chart function `f1`, we get only the coordinate expression:

In [28]:
```
f1
```

Out[28]: $\cos(x)^2$

To get an output similar to that of `f0`, one should call the method `display()`:

In [29]:
```
f1.display()
```

Out[29]: $(x, y, z) \mapsto \cos(x)^2$

Note that the method `expr()` returns the underlying symbolic expression:

In [30]:
```
f1.expr()
```

Out[30]: $\cos(x)^2$

In [31]:
```
print(type(f1.expr()))
```

```
<class 'sage.symbolic.expression.Expression'>
```

Introducing a second chart on the manifold

Let us first consider an open subset of $\mathcal{M}$, for instance the complement U of the region defined by $\{y = 0, x \geq 0\}$ (note that `(y!=0, x<0)` stands for $y \neq 0$ OR $x < 0$; the condition $y \neq 0$ AND $x < 0$ would have been written `[y!=0, x<0]` instead):

In [32]:
```
U = M.open_subset('U', coord_def={X: (y!=0, x<0)})
```

Let us call `X_U` the restriction of the chart `X` to the open subset U:

In [33]:
```
X_U = X.restrict(U)
```

```
X_U
```

Out[33]: $(U, (x, y, z))$

We introduce another chart on U, with spherical-type coordinates (r, θ, ϕ):

In [34]:
```
Y.<r,th,ph> = U.chart(r'r:(0,+oo) th:(0,pi):\theta ph:(0,2*pi):\phi')
Y
```

Out[34]: $(U, (r, \theta, \phi))$

The method `chart()` is now used with an argument; it is a string, which contains specific LaTeX symbols, hence the prefix 'r' to it (for *raw* string). It also contains the coordinate ranges, since they are different from the default value, which is $(-\infty, +\infty)$. For a given coordinate, the various fields are separated by the character ':' and a space character separates the coordinates. Note that for the coordinate r, there are only two fields, since the LaTeX symbol has not to be specified. The LaTeX symbols are used for the outputs:

In [35]:
```
th, ph
```

Out[35]: (θ, ϕ)

In [36]:
```
Y[2], Y[3]
```

Out[36]: (θ, ϕ)

The declared coordinate ranges are now known to Sage, as we may check by means of the command `assumptions()`:

In [37]:
```
assumptions()
```

Out[37]: $[\mathtt{x\ is\ real}, \mathtt{y\ is\ real}, \mathtt{z\ is\ real}, \mathtt{r\ is\ real}, r > 0, \mathtt{th\ is\ real}, \theta > 0, \theta < \pi, \mathtt{ph\ is\ real}$

They are used in simplifications:

In [38]:
```
simplify(abs(r))
```

Out[38]: r

In [39]:
```
simplify(abs(x)) # no simplification occurs since x can take any value in R
```

Out[39]: $|x|$

After having been declared, the chart `Y` can be fully specified by its relation to the chart `X_U`, via a transition map:

In [40]:
```
transit_Y_to_X = Y.transition_map(X_U, [r*sin(th)*cos(ph), r*sin(th)*sin(ph),
                                        r*cos(th)])
transit_Y_to_X
```

Out[40]: $(U, (r, \theta, \phi)) \to (U, (x, y, z))$

In [41]:
```
transit_Y_to_X.display()
```

Out[41]:
$$\left\{\begin{array}{lcl} x & = & r\cos(\phi)\sin(\theta) \\ y & = & r\sin(\phi)\sin(\theta) \\ z & = & r\cos(\theta) \end{array}\right.$$

The inverse of the transition map can be specified by means of the method `set_inverse()`:

In [42]:
```
transit_Y_to_X.set_inverse(sqrt(x^2+y^2+z^2), atan2(sqrt(x^2+y^2),z),
                           atan2(y, x))
```

```
Check of the inverse coordinate transformation:
  r == r  *passed*
  th == arctan2(r*sin(th), r*cos(th))  **failed**
  ph == arctan2(r*sin(ph)*sin(th), r*cos(ph)*sin(th))  **failed**
  x == x  *passed*
  y == y  *passed*
  z == z  *passed*
NB: a failed report can reflect a mere lack of simplification.
```

A check of the provided inverse is performed by composing it with the original transition map, on the left and on the right respectively. As indicated, the reported failure for `th` and `ph` is actually due to a lack of simplification of expressions involving `arctan2`.

We have then

In [43]:
```
transit_Y_to_X.inverse().display()
```

Out[43]:
$$\begin{cases} r & = & \sqrt{x^2+y^2+z^2} \\ \theta & = & \arctan\left(\sqrt{x^2+y^2}, z\right) \\ \phi & = & \arctan(y, x) \end{cases}$$

At this stage, the manifold's **atlas** (the "user atlas", not the maximal atlas!) contains three charts:

In [44]:
```
M.atlas()
```

Out[44]: $[(\mathcal{M},(x,y,z)),(U,(x,y,z)),(U,(r,\theta,\phi))]$

The first chart defined on the manifold is considered as the manifold's default chart (this can be changed by the method `set_default_chart()`):

In [45]:
```
M.default_chart()
```

Out[45]: $(\mathcal{M},(x,y,z))$

Each open subset has its own atlas (since an open subset of a manifold is a manifold by itself):

In [46]:
```
U.atlas()
```

Out[46]: $[(U,(x,y,z)),(U,(r,\theta,\phi))]$

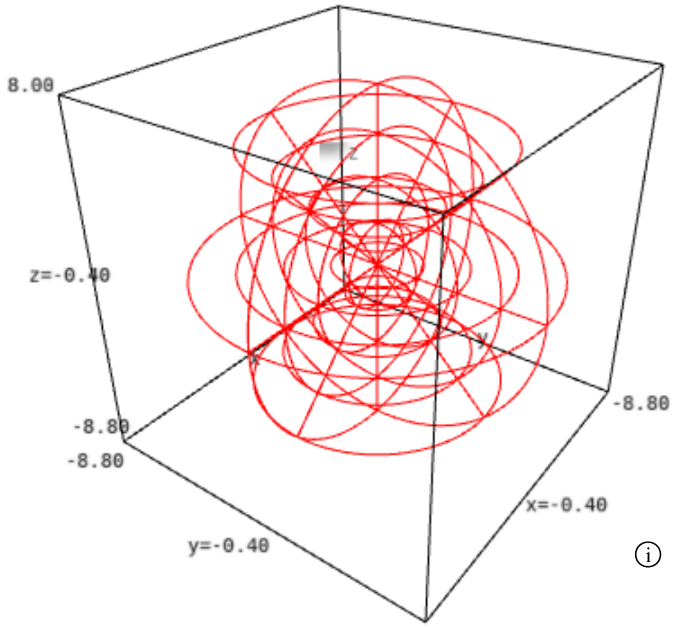
In [47]:
```
U.default_chart()
```

Out[47]: $(U,(x,y,z))$

We can draw the chart Y in terms of the chart X via the command `Y.plot(X)`, which shows the lines of constant coordinates from the Y chart in a "Cartesian frame" based on the X coordinates:

In [48]:
```
Y.plot(X)
```

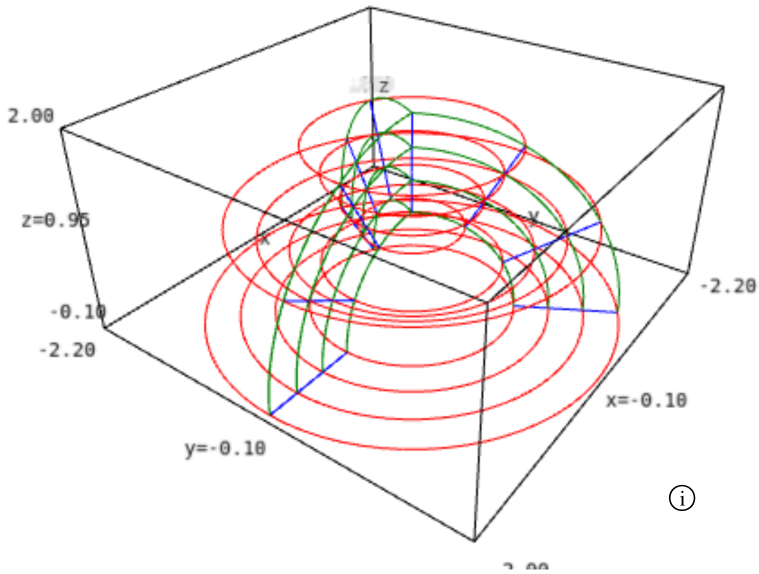
Out[48]:



The method `plot()` allows for many options, to control the number of coordinate lines to be drawn, their style and color, as well as the coordinate ranges (see the list of all options):

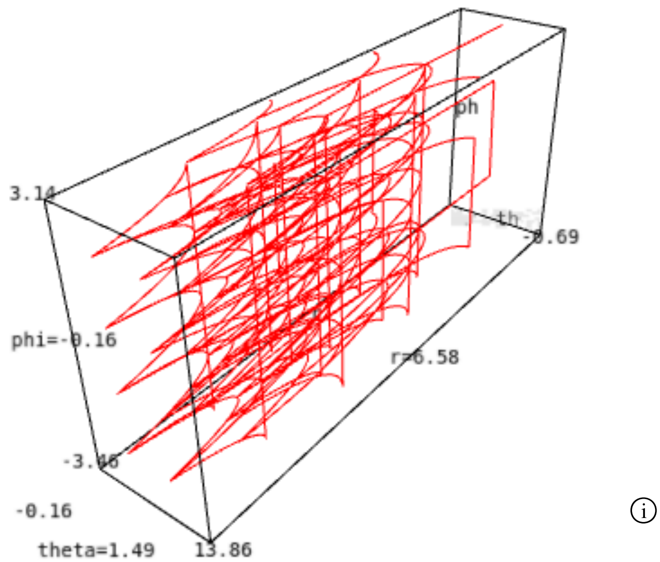
In [49]:
```
Y.plot(X, ranges={r:(1,2), th:(0,pi/2)}, number_values=4,
       color={r:'blue', th:'green', ph:'red'}, aspect_ratio=1)
```

Out[49]:



Conversly, the chart $X|_U$ can be plotted in terms of the chart Y (this is not possible for the whole chart X since its domain is larger than that of chart Y):

In [50]:
```
graph = X_U.plot(Y)
show(graph, axes_labels=['r','theta','phi'])
```



Points on the manifold

A point on $\mathcal{M}$ is defined by its coordinates in a given chart:

In [51]:
```
p = M.point((1,2,-1), chart=X, name='p')
print(p)
p
```

```
Point p on the 3-dimensional differentiable manifold M
```

Out[51]: p

Since $X = (\mathcal{M}, (x, y, z))$ is the manifold's default chart, its name can be omitted:

In [52]:
```
p = M.point((1,2,-1), name='p')
print(p)
p
```

```
Point p on the 3-dimensional differentiable manifold M
```

Out[52]: p

Of course, p belongs to $\mathcal{M}$:

In [53]:
```
p in M
```

Out[53]: True

It is also in U:

In [54]:
```
p in U
```

Out[54]: True

Indeed the coordinates of p have $y \neq 0$:

In [55]:
```
p.coord(X)
```

Out[55]: $(1, 2, -1)$

Note in passing that since X is the default chart on $\mathcal{M}$, its name can be omitted in the arguments of coord():

In [56]:
```
p.coord()
```

Out[56]: $(1, 2, -1)$

The coordinates of p can also be obtained by letting the chart act on the point (from the very definition of a chart!):

In [57]:
```
X(p)
```

Out[57]: $(1, 2, -1)$

Let q be a point with $y = 0$ and $x \geq 0$:

In [58]:
```
q = M.point((1,0,2), name='q')
```

This time, the point does not belong to U:

In [59]:
```
q in U
```

Out[59]: False

Accordingly, we cannot ask for the coordinates of q in the chart $Y = (U, (r, \theta, \phi))$:

In [60]:
```
try:
    q.coord(Y)
except ValueError as exc:
    print("Error: " + str(exc))
```

```
Error: the point does not belong to the domain of Chart (U, (r, th, ph))
```

but we can for point p:

In [61]:
```
p.coord(Y)
```

Out[61]: $\left(\sqrt{3}\sqrt{2}, \pi - \arctan\left(\sqrt{5}\right), \arctan(2)\right)$

In [62]:
```
Y(p)
```

Out[62]: $\left(\sqrt{3}\sqrt{2}, \pi - \arctan\left(\sqrt{5}\right), \arctan(2)\right)$

Points can be compared:

In [63]:
```
q == p
```

Out[63]: False

In [64]:
```
p1 = U.point((sqrt(3)*sqrt(2), pi-atan(sqrt(5)), atan(2)), chart=Y)
p1 == p
```

Out[64]: True

In SageMath's terminology, points are **elements**, whose **parents** are the manifold on which they have been defined:

In [65]:
```
p.parent()
```

Out[65]: $\mathcal{M}$

In [66]:
```
q.parent()
```

Out[66]: $\mathcal{M}$

In [67]:
```
p1.parent()
```

Out[67]: U

Scalar fields

A **scalar field** is a differentiable map $U \to \mathbb{R}$, where U is an open subset of $\mathcal{M}$.

A scalar field is defined by its expressions in terms of charts covering its domain (in general more than one chart is necessary to cover all the domain):

In [68]:
```
f = U.scalar_field({X_U: x+y^2+z^3}, name='f')
print(f)
```

```
Scalar field f on the Open subset U of the 3-dimensional differentiable manifold M
```

The coordinate expressions of the scalar field are passed as a Python dictionary, with the charts as keys, hence the writing `{X_U: x+y^2+z^3}` . Since in the present case, there is only one chart in the dictionary, an alternative writing is

In [69]:
```
f = U.scalar_field(x+y^2+z^3, chart=X_U, name='f')
print(f)
```

```
Scalar field f on the Open subset U of the 3-dimensional differentiable manifold M
```

Since `X_U` is the domain's default chart, it can be omitted in the above declaration:

In [70]:
```
f = U.scalar_field(x+y^2+z^3, name='f')
print(f)
```

```
Scalar field f on the Open subset U of the 3-dimensional differentiable manifold M
```

As a map $U \subset \mathcal{M} \longrightarrow \mathbb{R}$, a scalar field acts on points, not on coordinates:

In [71]:
```
f(p)
```

Out[71]: 4

The method `display()` provides the expression of the scalar field in terms of a given chart:

In [72]:
```
f.display(X_U)
```

Out[72]:
$$\begin{array}{llcl} f: & U & \longrightarrow & \mathbb{R} \\ & (x,y,z) & \longmapsto & z^3 + y^2 + x \end{array}$$

If no argument is provided, the method `display()` shows the coordinate expression of the scalar field in all the charts defined on the domain (except for *subcharts*, i.e. the restrictions of some chart to a subdomain):

In [73]:
```
f.display()
```

Out[73]:
$$\begin{array}{llcl} f: & U & \longrightarrow & \mathbb{R} \\ & (x,y,z) & \longmapsto & z^3 + y^2 + x \\ & (r,\theta,\phi) & \longmapsto & r^3 \cos\left(\theta\right)^3 + r^2 \sin\left(\phi\right)^2 \sin\left(\theta\right)^2 + r \cos\left(\phi\right) \sin\left(\theta\right) \end{array}$$

Note that the expression of f in terms of the coordinates (r,θ,ϕ) has not been provided by the user but has been automatically computed by means of the change-of-coordinate formula declared above in the transition map.

In [74]:
```
f.display(Y)
```

Out[74]:
$$\begin{array}{llll} f: & U & \longrightarrow & \mathbb{R} \\ & (r,\theta,\phi) & \longmapsto & r^3\cos(\theta)^3 + r^2\sin(\phi)^2\sin(\theta)^2 + r\cos(\phi)\sin(\theta) \end{array}$$

In each chart, the scalar field is represented by a function of the chart coordinates (an object of the type `ChartFunction` described above), which is accessible via the method `coord_function()` :

In [75]:
```
f.coord_function(X_U)
```

Out[75]: $z^3 + y^2 + x$

In [76]:
```
f.coord_function(X_U).display()
```

Out[76]: $(x,y,z) \mapsto z^3 + y^2 + x$

In [77]:
```
f.coord_function(Y)
```

Out[77]: $r^3\cos(\theta)^3 + r^2\sin(\phi)^2\sin(\theta)^2 + r\cos(\phi)\sin(\theta)$

In [78]:
```
f.coord_function(Y).display()
```

Out[78]: $(r,\theta,\phi) \mapsto r^3\cos(\theta)^3 + r^2\sin(\phi)^2\sin(\theta)^2 + r\cos(\phi)\sin(\theta)$

The "raw" symbolic expression is returned by the method `expr()`:

In [79]:
```
f.expr(X_U)
```

Out[79]: $z^3 + y^2 + x$

In [80]:
```
f.expr(Y)
```

Out[80]: $r^3\cos(\theta)^3 + r^2\sin(\phi)^2\sin(\theta)^2 + r\cos(\phi)\sin(\theta)$

In [81]:
```
f.expr(Y) is f.coord_function(Y).expr()
```

Out[81]: True

A scalar field can also be defined by some unspecified function of the coordinates:

In [82]:
```
h = U.scalar_field(function('H')(x, y, z), name='h')
print(h)
```

```
Scalar field h on the Open subset U of the 3-dimensional differentiable manifold M
```

In [83]:
```
h.display()
```

Out[83]:
$$\begin{array}{llll} h: & U & \longrightarrow & \mathbb{R} \\ & (x,y,z) & \longmapsto & H(x,y,z) \\ & (r,\theta,\phi) & \longmapsto & H(r\cos(\phi)\sin(\theta), r\sin(\phi)\sin(\theta), r\cos(\theta)) \end{array}$$

In [84]:
```
h.display(Y)
```

Out[84]:
$$\begin{array}{llll} h: & U & \longrightarrow & \mathbb{R} \\ & (r,\theta,\phi) & \longmapsto & H(r\cos(\phi)\sin(\theta), r\sin(\phi)\sin(\theta), r\cos(\theta)) \end{array}$$

In [85]:
```
h(p) # remember that p is the point of coordinates (1,2,-1) in the chart X_U
```

Out[85]: $H(1,2,-1)$

The parent of f is the set $C^\infty(U)$ of all smooth scalar fields on U, which is a commutative algebra over $\mathbb{R}$:

In [86]:
```
CU = f.parent()
CU
```

Out[86]: $C^\infty(U)$

In [87]:
```
print(CU)
```

```
Algebra of differentiable scalar fields on the Open subset U of the 3-dimensional di
fferentiable manifold M
```

In [88]:
```
CU.category()
```

Out[88]: **JoinCategory**

The base ring of the algebra is the field $\mathbb{R}$, which is represented here by SageMath's Symbolic Ring (SR):

In [89]:
```
CU.base_ring()
```

Out[89]: SR

Arithmetic operations on scalar fields are defined through the algebra structure:

In [90]:
```
s = f + 2*h
print(s)
```

```
Scalar field on the Open subset U of the 3-dimensional differentiable manifold M
```

In [91]:
```
s.display()
```

Out[91]:

$$\begin{array}{llcl} & U & \longrightarrow & \mathbb{R} \\ & (x, y, z) & \longmapsto & z^3 + y^2 + x + 2\,H\left(x, y, z\right) \\ & (r, \theta, \phi) & \longmapsto & r^3 \cos\left(\theta\right)^3 + r^2 \sin\left(\phi\right)^2 \sin\left(\theta\right)^2 + r\cos(\phi)\sin(\theta) + 2\,H\left(r\cos(\phi)\sin(\theta), r\sin(\right. \end{array}$$

Tangent spaces

The tangent vector space to the manifold at point p is obtained as follows:

In [92]:
```
Tp = M.tangent_space(p)
Tp
```

Out[92]: $T_p\,\mathcal{M}$

In [93]:
```
print(Tp)
```

```
Tangent space at Point p on the 3-dimensional differentiable manifold M
```

$T_p\,\mathcal{M}$ is a 2-dimensional vector space over $\mathbb{R}$ (represented here by SageMath's Symbolic Ring (SR)):

In [94]:
```
print(Tp.category())
```

```
Category of finite dimensional vector spaces over Symbolic Ring
```

In [95]:
```
Tp.dim()
```

Out[95]: 3

$T_p\,\mathcal{M}$ is automatically endowed with vector bases deduced from the vector frames defined around the point:

In [96]:
```
Tp.bases()
```

Out[96]: $\left[\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right), \left(\frac{\partial}{\partial r}, \frac{\partial}{\partial \theta}, \frac{\partial}{\partial \phi}\right)\right]$

For the tangent space at the point q, on the contrary, there is only one pre-defined basis, since q is not in the domain U of the frame associated with coordinates (r, θ, ϕ):

In [97]:
```
Tq = M.tangent_space(q)
Tq.bases()
```

Out[97]: $\left[\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)\right]$

A random element:

In [98]:
```
v = Tp.an_element()
print(v)
```

```
Tangent vector at Point p on the 3-dimensional differentiable manifold M
```

In [99]:
```
v.display()
```

Out[99]: $\frac{\partial}{\partial x} + 2\frac{\partial}{\partial y} + 3\frac{\partial}{\partial z}$

In [100…
```
u = Tq.an_element()
print(u)
```

```
Tangent vector at Point q on the 3-dimensional differentiable manifold M
```

In [101…
```
u.display()
```

Out[101… $\frac{\partial}{\partial x} + 2\frac{\partial}{\partial y} + 3\frac{\partial}{\partial z}$

Note that, despite what the above simplified writing may suggest (the mention of the point p or q is omitted in the basis vectors), u and v are different vectors, for they belong to different vector spaces:

In [102…
```
v.parent()
```

Out[102… $T_p\,\mathcal{M}$

In [103…
```
u.parent()
```

Out[103… $T_q\,\mathcal{M}$

In particular, it is not possible to add u and v:

In [104…
```
try:
    s = u + v
except TypeError as exc:
    print("Error: " + str(exc))
```

```
Error: unsupported operand parent(s) for +: 'Tangent space at Point q on the 3-dimen
sional differentiable manifold M' and 'Tangent space at Point p on the 3-dimensional
differentiable manifold M'
```

Vector Fields

Each chart defines a vector frame on the chart domain: the so-called **coordinate basis**:

In [105…
```
X.frame()
```
Out[105…
$$\left(\mathcal{M}, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)\right)$$

In [106…
```
X.frame().domain()  # this frame is defined on the whole manifold
```
Out[106… $\mathcal{M}$

In [107…
```
Y.frame()
```
Out[107…
$$\left(U, \left(\frac{\partial}{\partial r}, \frac{\partial}{\partial \theta}, \frac{\partial}{\partial \phi}\right)\right)$$

In [108…
```
Y.frame().domain() # this frame is defined only on U
```
Out[108… U

The list of frames defined on a given open subset is returned by the method `frames()` :

In [109…
```
M.frames()
```
Out[109…
$$\left[\left(\mathcal{M}, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)\right), \left(U, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)\right), \left(U, \left(\frac{\partial}{\partial r}, \frac{\partial}{\partial \theta}, \frac{\partial}{\partial \phi}\right)\right)\right]$$

In [110…
```
U.frames()
```
Out[110…
$$\left[\left(U, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)\right), \left(U, \left(\frac{\partial}{\partial r}, \frac{\partial}{\partial \theta}, \frac{\partial}{\partial \phi}\right)\right)\right]$$

In [111…
```
M.default_frame()
```
Out[111…
$$\left(\mathcal{M}, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)\right)$$

Unless otherwise specified (via the command `set_default_frame()`), the default frame is that associated with the default chart:

In [112…
```
M.default_frame() is M.default_chart().frame()
```
Out[112… True

In [113…
```
U.default_frame() is U.default_chart().frame()
```
Out[113… True

Individual elements of a frame can be accessed by means of their indices:

In [114…
```
e = U.default_frame()
e2 = e[2]
e2
```
Out[114…
$$\frac{\partial}{\partial y}$$

In [115…
```
print(e2)
```
```
Vector field ∂/∂y on the Open subset U of the 3-dimensional differentiable manifold
M
```

We may define a new vector field as follows:

In [116…
```
v = e[2] + 2*x*e[3]
print(v)
```

```
Vector field on the Open subset U of the 3-dimensional differentiable manifold M
```

In [117…
```
v.display()
```

Out[117…

$$\frac{\partial}{\partial y} + 2\,x\frac{\partial}{\partial z}$$

A vector field can be defined by its components with respect to a given vector frame. When the latter is not specified, the open set's default frame is of course assumed:

In [118…
```
v = U.vector_field(name='v') # vector field defined on the open set U
v[1] = 1+y
v[2] = -x
v[3] = x*y*z
v.display()
```

Out[118…

$$v = \left(y + 1 \right) \frac{\partial}{\partial x} - x\frac{\partial}{\partial y} + xyz\frac{\partial}{\partial z}$$

It is possible to initialize the components of the vector field while declaring it, so that the above is equivalent to

In [119…
```
v = U.vector_field(1+y, -x, x*y*z, name='v')
v.display()
```

Out[119…

$$v = \left(y + 1 \right) \frac{\partial}{\partial x} - x\frac{\partial}{\partial y} + xyz\frac{\partial}{\partial z}$$

Vector fields on U are Sage *element* objects, whose *parent* is the set $\mathfrak{X}(U)$ of vector fields defined on U:

In [120…
```
v.parent()
```

Out[120…

$$\mathfrak{X}\left(U\right)$$

The set $\mathfrak{X}(U)$ is a module over the commutative algebra $C^\infty(U)$ of scalar fields on U:

In [121…
```
print(v.parent())
```

```
Free module X(U) of vector fields on the Open subset U of the 3-dimensional differen
tiable manifold M
```

In [122…
```
print(v.parent().category())
```

```
Category of finite dimensional modules over Algebra of differentiable scalar fields
on the Open subset U of the 3-dimensional differentiable manifold M
```

In [123…
```
v.parent().base_ring()
```

Out[123…

$$C^{\infty}\left(U\right)$$

A vector field acts on scalar fields:

In [124…
```
f.display()
```

Out[124…

$$\begin{array}{llcl} f: & U & \longrightarrow & \mathbb{R} \\ & \left(x, y, z\right) & \longmapsto & z^{3} + y^{2} + x \\ & \left(r, {\theta}, {\phi}\right) & \longmapsto & r^{3} \cos\left({\theta}\right)^{3} + r^{2} \sin\left({\phi}\right)^{2} \sin\left({\theta}\right)^{2} + r \cos\left({\phi}\right) \sin\left({\theta}\right) \end{array}$$

In [125…
```
s = v(f)
print(s)
```

```
Scalar field v(f) on the Open subset U of the 3-dimensional differentiable manifold
M
```

In [126…
```
s.display()
```

Out[126…

$$\begin{array}{llcl} v\left(f\right): & U & \longrightarrow & \mathbb{R} \\ & (x, y, z) & \longmapsto & 3\,xyz^3 - (2\,x - 1)y + 1 \\ & (r, \theta, \phi) & \longmapsto & r\sin(\phi)\sin(\theta) + \left(3\,r^5\cos(\phi)\cos\left(\theta\right)^3\sin(\phi) - 2\,r^2\cos(\phi)\sin(\phi)\right)\sin\left(\theta\right. \end{array}$$

In [127…
```
e[3].display()
```

Out[127…

$$\frac{\partial}{\partial z} = \frac{\partial}{\partial z}$$

In [128…
```
e[3](f).display()
```

Out[128…

$$\begin{array}{llcl} \frac{\partial}{\partial z}\left(f\right): & U & \longrightarrow & \mathbb{R} \\ & (x, y, z) & \longmapsto & 3\,z^2 \\ & (r, \theta, \phi) & \longmapsto & 3\,r^2\cos\left(\theta\right)^2 \end{array}$$

Unset components are assumed to be zero:

In [129…
```
w = U.vector_field(name='w')
w[2] = 3
w.display()
```

Out[129…

$$w = 3\frac{\partial}{\partial y}$$

A vector field on U can be expanded in the vector frame associated with the chart (r, θ, ϕ):

In [130…
```
v.display(Y.frame())
```

Out[130…

$$v = \left(\frac{xyz^2 + x}{\sqrt{x^2 + y^2 + z^2}}\right)\frac{\partial}{\partial r} + \left(-\frac{\left(x^3y + xy^3 - x\right)\sqrt{x^2 + y^2}z}{x^4 + 2\,x^2y^2 + y^4 + (x^2 + y^2)z^2}\right)\frac{\partial}{\partial \theta} + \left(-\frac{x^2 + y^2 + y}{x^2 + y^2}\right)\frac{\partial}{\partial}$$

By default, the components are expressed in terms of the default coordinates (x, y, z). To express them in terms of the coordinates (r, θ, ϕ), one should add the corresponding chart as the second argument of the method `display()` :

In [131…
```
v.display(Y.frame(), Y)
```

Out[131…

$$v = \left(r^3\cos(\phi)\cos\left(\theta\right)^2\sin(\phi)\sin\left(\theta\right)^2 + \cos(\phi)\sin(\theta)\right)\frac{\partial}{\partial r} + \left(-\frac{r^3\cos(\phi)\cos(\theta)\sin(\phi)\sin\left(\theta\right)^3}{r}\right.$$

As a shortcut, for a coordinate frame, one may provide the name of the chart only:

In [132…
```
v.display(Y)
```

Out[132…

$$v = \left(r^3\cos(\phi)\cos\left(\theta\right)^2\sin(\phi)\sin\left(\theta\right)^2 + \cos(\phi)\sin(\theta)\right)\frac{\partial}{\partial r} + \left(-\frac{r^3\cos(\phi)\cos(\theta)\sin(\phi)\sin\left(\theta\right)^3}{r}\right.$$

In [133…
```
for i in M.irange():
    show(e[i].display(Y))
```

$$\frac{\partial}{\partial x} = \cos(\phi)\sin(\theta)\frac{\partial}{\partial r} + \frac{\cos(\phi)\cos(\theta)}{r}\frac{\partial}{\partial \theta} - \frac{\sin(\phi)}{r\sin(\theta)}\frac{\partial}{\partial \phi}$$

$$\frac{\partial}{\partial y} = \sin(\phi)\sin(\theta)\frac{\partial}{\partial r} + \frac{\cos(\theta)\sin(\phi)}{r}\frac{\partial}{\partial \theta} + \frac{\cos(\phi)}{r\sin(\theta)}\frac{\partial}{\partial \phi}$$

$$\frac{\partial}{\partial z} = \cos(\theta)\frac{\partial}{\partial r} - \frac{\sin(\theta)}{r}\frac{\partial}{\partial \theta}$$

The components of a tensor field w.r.t. the default frame can also be obtained as a list, thanks to the operator `[:]` :

In [134…
```
v[:]
```

Out[134… $[y + 1, -x, xyz]$

An alternative is to use the method `display_comp()` :

In [135…
```
v.display_comp()
```

Out[135…

$$\begin{array}{lcl} v^{x} & = & y + 1 \\ v^{y} & = & -x \\ v^{z} & = & xyz \end{array}$$

To obtain the components w.r.t. another frame, one may go through the method `comp()` and specify the frame:

In [136…
```
v.comp(Y.frame())[:]
```

Out[136…

$$\left[\frac{xyz^2 + x}{\sqrt{x^2 + y^2 + z^2}}, -\frac{\left(x^3 y + xy^3 - x\right)\sqrt{x^2 + y^2} z}{x^4 + 2\,x^2 y^2 + y^4 + (x^2 + y^2) z^2}, -\frac{x^2 + y^2 + y}{x^2 + y^2}\right]$$

However a shortcut is to provide the frame as the first argument of the square brackets:

In [137…
```
v[Y.frame(), :]
```

Out[137…

$$\left[\frac{xyz^2 + x}{\sqrt{x^2 + y^2 + z^2}}, -\frac{\left(x^3 y + xy^3 - x\right)\sqrt{x^2 + y^2} z}{x^4 + 2\,x^2 y^2 + y^4 + (x^2 + y^2) z^2}, -\frac{x^2 + y^2 + y}{x^2 + y^2}\right]$$

In [138…
```
v.display_comp(Y.frame())
```

Out[138…

$$\begin{array}{lcl} v^{r} & = & \frac{xyz^2 + x}{\sqrt{x^2 + y^2 + z^2}} \\ v^{\theta} & = & -\frac{\left(x^3 y + xy^3 - x\right)\sqrt{x^2 + y^2} z}{x^4 + 2\,x^2 y^2 + y^4 + (x^2 + y^2) z^2} \\ v^{\phi} & = & -\frac{x^2 + y^2 + y}{x^2 + y^2} \end{array}$$

Components are shown expressed in terms of the default's coordinates; to get them in terms of the coordinates (r, θ, ϕ) instead, add the chart name as the last argument in the square brackets:

In [139…
```
v[Y.frame(), :, Y]
```

Out[139…

$$\left[r^3 \cos(\phi) \cos(\theta)^2 \sin(\phi) \sin(\theta)^2 + \cos(\phi) \sin(\theta), -\frac{r^3 \cos(\phi) \cos(\theta) \sin(\phi) \sin(\theta)^3 - \cos(\phi) \cos(\right.$$

or specify the chart in `display_comp()` :

In [140…
```
v.display_comp(Y.frame(), chart=Y)
```

Out[140…

$$\begin{array}{lcl} v^{r} & = & r^3 \cos(\phi) \cos(\theta)^2 \sin(\phi) \sin(\theta)^2 + \cos(\phi) \sin(\theta) \\ v^{\theta} & = & -\frac{r^3 \cos(\phi) \cos(\theta) \sin(\phi) \sin(\theta)^3 - \cos(\phi) \cos(\theta)}{r} \\ v^{\phi} & = & -\frac{r \sin(\theta) + \sin(\phi)}{r \sin(\theta)} \end{array}$$

To get a vector component as a scalar field instead of a coordinate expression, use double square brackets:

In [141…
```
print(v[[1]])
```

```
Scalar field on the Open subset U of the 3-dimensional differentiable manifold M
```

In [142…
```
v[[1]].display()
```

Out[142…
$$\begin{array}{llcl} & U & \longrightarrow & \mathbb{R} \\ & (x,y,z) & \longmapsto & y+1 \\ & (r,\theta,\phi) & \longmapsto & r\sin(\phi)\sin(\theta)+1 \end{array}$$

In [143…
```
v[[1]].expr(X_U)
```

Out[143… $y+1$

A vector field can be defined with components being unspecified functions of the coordinates:

In [144…
```
u = U.vector_field(function('u_x')(x,y,z),
                   function('u_y')(x,y,z),
                   function('u_z')(x,y,z), name='u')
u.display()
```

Out[144…
$$u = u_x(x,y,z)\frac{\partial}{\partial x} + u_y(x,y,z)\frac{\partial}{\partial y} + u_z(x,y,z)\frac{\partial}{\partial z}$$

In [145…
```
s = v + u
s.set_name('s')
s.display()
```

Out[145…
$$s = (y + u_x(x,y,z) + 1)\frac{\partial}{\partial x} + \left(-x + u_y(x,y,z)\right)\frac{\partial}{\partial y} + (xyz + u_z(x,y,z))\frac{\partial}{\partial z}$$

Values of vector field at a given point

The value of a vector field at some point of the manifold is obtained via the method `at()` :

In [146…
```
vp = v.at(p)
print(vp)
```

```
Tangent vector v at Point p on the 3-dimensional differentiable manifold M
```

In [147…
```
vp.display()
```

Out[147…
$$v = 3\frac{\partial}{\partial x} - \frac{\partial}{\partial y} - 2\frac{\partial}{\partial z}$$

Indeed, recall that, w.r.t. chart `X_U` =(x,y,z), the coordinates of the point p and the components of the vector field v are

In [148…
```
p.coord(X_U)
```

Out[148… $(1,2,-1)$

In [149…
```
v.display(X_U.frame(), X_U)
```

Out[149…
$$v = (y+1)\frac{\partial}{\partial x} - x\frac{\partial}{\partial y} + xyz\frac{\partial}{\partial z}$$

Note that to simplify the writing, the symbol used to denote the value of the vector field at point p is the same as that of the vector field itself (namely v); this can be changed by the method `set_name()` :

In [150…
```
vp.set_name(latex_name='v|_p')
vp.display()
```

Out[150…
$$v|_p = 3\frac{\partial}{\partial x} - \frac{\partial}{\partial y} - 2\frac{\partial}{\partial z}$$

Of course, $v|_p$ belongs to the tangent space at p:

In [151…
```
vp.parent()
```

Out[151… $T_p\,\mathcal{M}$

In [152…
```
vp in M.tangent_space(p)
```

Out[152… True

In [153…
```
up = u.at(p)
print(up)
```

```
Tangent vector u at Point p on the 3-dimensional differentiable manifold M
```

In [154…
```
up.display()
```

Out[154…
$$u = u_x\left(1, 2, -1\right)\frac{\partial}{\partial x} + u_y\left(1, 2, -1\right)\frac{\partial}{\partial y} + u_z\left(1, 2, -1\right)\frac{\partial}{\partial z}$$

1-forms

A **1-form** on $\mathcal{M}$ is a field of linear forms. For instance, it can be the *differential of a scalar field*:

In [155…
```
df = f.differential()
print(df)
```

```
1-form df on the Open subset U of the 3-dimensional differentiable manifold M
```

An equivalent writing is

In [156…
```
df = diff(f)
```

The method `display()` shows the expansion on the default coframe:

In [157…
```
df.display()
```

Out[157… $\mathrm{d}f = \mathrm{d}x + 2\,y\mathrm{d}y + 3\,z^2\mathrm{d}z$

In the above writing, the 1-form is expanded over the basis $(\mathrm{d}x, \mathrm{d}y, \mathrm{d}z)$ associated with the chart (x, y, z). This basis can be accessed via the method `coframe()` :

In [158…
```
dX = X.coframe()
dX
```

Out[158… $\left(\mathcal{M}, \left(\mathrm{d}x, \mathrm{d}y, \mathrm{d}z\right)\right)$

The list of all coframes defined on a given manifold open subset is returned by the method `coframes()` :

In [159…
```
M.coframes()
```

Out[159… $\left[\left(\mathcal{M}, (\mathrm{d}x, \mathrm{d}y, \mathrm{d}z)\right), \left(U, (\mathrm{d}x, \mathrm{d}y, \mathrm{d}z)\right), \left(U, (\mathrm{d}r, \mathrm{d}\theta, \mathrm{d}\phi)\right)\right]$

As for a vector field, the value of the differential form at some point on the manifold is obtained by the method `at()` :

In [160…
```
dfp = df.at(p)
print(dfp)
```

```
Linear form df on the Tangent space at Point p on the 3-dimensional differentiable m
anifold M
```

In [161…
```
dfp.display()
```

Out[161… $\mathrm{d}f = \mathrm{d}x + 4\mathrm{d}y + 3\mathrm{d}z$

Recall that

In [162…
```
p.coord()
```

Out[162… $(1, 2, -1)$

The linear form $\left.\mathrm{d}f\right|_p$ belongs to the dual of the tangent vector space at p:

In [163…
```
dfp.parent()
```

Out[163… $T_p\,\mathcal{M}^*$

In [164…
```
dfp.parent() is M.tangent_space(p).dual()
```

Out[164… True

As such, it is acting on vectors at p, yielding a real number:

In [165…
```
print(vp)
vp.display()
```

```
Tangent vector v at Point p on the 3-dimensional differentiable manifold M
```

Out[165… $\left.v\right|_p = 3\frac{\partial}{\partial x} - \frac{\partial}{\partial y} - 2\frac{\partial}{\partial z}$

In [166…
```
dfp(vp)
```

Out[166… -7

In [167…
```
print(up)
up.display()
```

```
Tangent vector u at Point p on the 3-dimensional differentiable manifold M
```

Out[167… $u = u_x\left(1, 2, -1\right)\frac{\partial}{\partial x} + u_y\left(1, 2, -1\right)\frac{\partial}{\partial y} + u_z\left(1, 2, -1\right)\frac{\partial}{\partial z}$

In [168…
```
dfp(up)
```

Out[168… $u_x\left(1, 2, -1\right) + 4\,u_y\left(1, 2, -1\right) + 3\,u_z\left(1, 2, -1\right)$

The differential 1-form of the unspecified scalar field h:

In [169…
```
dh = h.differential()
dh.display()
```

Out[169… $$dh = \frac{\partial H}{\partial x} \mathrm{d}x + \frac{\partial H}{\partial y} \mathrm{d}y + \frac{\partial H}{\partial z} \mathrm{d}z$$

A 1-form can also be defined from scratch:

In [170…
```
om = U.one_form(name='omega', latex_name=r'\omega')
print(om)
```

```
1-form omega on the Open subset U of the 3-dimensional differentiable manifold M
```

It can be specified by providing its components in a given coframe:

In [171…
```
om[:] = [x^2+y^2, z, x-z]    # components in the default coframe (dx,dy,dz)
om.display()
```

Out[171… $$\omega = \left(x^2 + y^2\right) \mathrm{d}x + z \mathrm{d}y + (x - z) \mathrm{d}z$$

It is also possible to initialize the components of the 1-form while declaring it, so that the above is equivalent to

In [172…
```
om = U.one_form(x^2+y^2, z, x-z, name='omega',
                latex_name=r'\omega')
om.display()
```

Out[172… $$\omega = \left(x^2 + y^2\right) \mathrm{d}x + z \mathrm{d}y + (x - z) \mathrm{d}z$$

Of course, one may set the components in a frame different from the default one:

In [173…
```
om[Y.frame(), :, Y] = [r*sin(th)*cos(ph), 0, r*sin(th)*sin(ph)]
om.display(Y.frame(), Y)
```

Out[173… $$\omega = r \cos(\phi) \sin(\theta) \mathrm{d}r + r \sin(\phi) \sin(\theta) \mathrm{d}\phi$$

The components in the coframe $(\mathrm{d}x, \mathrm{d}y, \mathrm{d}z)$ are updated automatically:

In [174…
```
om.display()
```

Out[174… $$\omega = \left(\frac{x^4 + x^2y^2 - \sqrt{x^2+y^2+z^2}y^2}{\sqrt{x^2+y^2+z^2}(x^2+y^2)}\right) \mathrm{d}x + \left(\frac{x^3y + xy^3 + \sqrt{x^2+y^2+z^2}xy}{\sqrt{x^2+y^2+z^2}(x^2+y^2)}\right) \mathrm{d}y + \left(\frac{}{\sqrt{x^2 +}}\right.$$

Let us revert to the values set previously:

In [175…
```
om[:] = [x^2+y^2, z, x-z]
om.display()
```

Out[175… $$\omega = \left(x^2 + y^2\right) \mathrm{d}x + z \mathrm{d}y + (x - z) \mathrm{d}z$$

This time, the components in the coframe $(\mathrm{d}r, \mathrm{d}\theta, \mathrm{d}\phi)$ are those that are updated:

In [176…
```
om.display(Y.frame(), Y)
```

Out[176… $$\omega = \left(r^2 \cos(\phi) \sin(\theta)^3 + r(\cos(\phi) + \sin(\phi)) \cos(\theta) \sin(\theta) - r \cos(\theta)^2\right) \mathrm{d}r$$
$$+ \left(r^2 \cos(\theta)^2 \sin(\phi) + r^2 \cos(\theta) \sin(\theta) + \left(r^3 \cos(\phi) \cos(\theta) - r^2 \cos(\phi)\right) \sin(\theta)^2\right) \mathrm{d}\theta + \left(-r^3 \sin(\right.$$

A 1-form acts on vector fields, resulting in a scalar field:

In [177…
```
print(om(v))
om(v).display()
```

```
Scalar field omega(v) on the Open subset U of the 3-dimensional differentiable manif
old M
```

Out[177…
$$\begin{array}{llcl} \omega\left(v\right): & U & \longrightarrow & \mathbb{R} \\ & \left(x, y, z\right) & \longmapsto & -xyz^{2} + x^{2}y + y^{3} + x^{2} + y^{2} + \left(x^{2}y - x\right)z \\ & \left(r, \theta, \phi\right) & \longmapsto & -r^{2}\cos(\phi)\cos(\theta)\sin(\theta) + \left(r^{4}\cos\left(\phi\right)^{2}\cos(\theta)\sin(\phi) + r^{3}\sin(\phi)\right)\sin\left(\theta\right. \end{array}$$

In [178…
```
print(df(v))
df(v).display()
```

```
Scalar field df(v) on the Open subset U of the 3-dimensional differentiable manifold
M
```

Out[178…
$$\begin{array}{llcl} \mathrm{d}f\left(v\right): & U & \longrightarrow & \mathbb{R} \\ & \left(x, y, z\right) & \longmapsto & 3\,xyz^{3} - (2\,x - 1)y + 1 \\ & \left(r, \theta, \phi\right) & \longmapsto & r\sin(\phi)\sin(\theta) + \left(3\,r^{5}\cos(\phi)\cos\left(\theta\right)^{3}\sin(\phi) - 2\,r^{2}\cos(\phi)\sin(\phi)\right)\sin\left(\right. \end{array}$$

In [179…
```
om(u).display()
```

Out[179…
$$\begin{array}{llcl} \omega\left(u\right): & U & \longrightarrow & \mathbb{R} \\ & \left(x, y, z\right) & \longmapsto & x^{2}u_{x}\left(x, y, z\right) + y^{2}u_{x}\left(x, y, z\right) + z\left(u_{y}\left(x, y, z\right) - u_{z}\left(x, y, z\right)\right) + xu_{z}\left(x, y\right. \\ & \left(r, \theta, \phi\right) & \longmapsto & r^{2}\sin\left(\theta\right)^{2}u_{x}\left(r\cos(\phi)\sin(\theta), r\sin(\phi)\sin(\theta), r\cos(\theta)\right) + r\cos(\theta)u_{y}\left(r\cos\right. \\ & & & \quad + \left(r\cos(\phi)\sin(\theta) - r\cos(\theta)\right)u_{z}\left(r\cos(\phi)\sin(\theta), r\sin(\right. \end{array}$$

In the case of a differential 1-form, the following identity holds:

In [180…
```
df(v) == v(f)
```

Out[180… True

1-forms are Sage *element* objects, whose *parent* is the $C^{\infty}(U)$-module $\Omega^{1}(U)$ of all 1-forms defined on U:

In [181…
```
df.parent()
```

Out[181… $\Omega^{1}\left(U\right)$

In [182…
```
print(df.parent())
```

```
Free module Omega^1(U) of 1-forms on the Open subset U of the 3-dimensional differen
tiable manifold M
```

In [183…
```
print(om.parent())
```

```
Free module Omega^1(U) of 1-forms on the Open subset U of the 3-dimensional differen
tiable manifold M
```

$\Omega^{1}(U)$ is actually the dual of the free module $\mathfrak{X}(U)$:

In [184…
```
df.parent() is v.parent().dual()
```

Out[184… True

Differential forms and exterior calculus

The **exterior product** of two 1-forms is taken via the method `wedge()` and results in a 2-form:

In [185…
```
a = om.wedge(df)
print(a)
a.display()
```

```
2-form omega∧df on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[185… $\omega\wedge\mathrm{d}f = \left(2\,x^{2}y + 2\,y^{3} - z\right)\mathrm{d}x\wedge\mathrm{d}y + \left(3\,\left(x^{2} + y^{2}\right)z^{2} - x + z\right)\mathrm{d}x\wedge\mathrm{d}z + \left(3\,z^{3} - 2\,xy + 2\,yz\right)\mathrm{c}$

A matrix view of the components:

In [186…
```
a[:]
```

Out[186…

$$\begin{pmatrix} 0 & 2\,x^{2}y + 2\,y^{3} - z & 3\,\left(x^{2} + y^{2}\right)z^{2} - x + z \\ -2\,x^{2}y - 2\,y^{3} + z & 0 & 3\,z^{3} - 2\,xy + 2\,yz \\ -3\,\left(x^{2} + y^{2}\right)z^{2} + x - z & -3\,z^{3} + 2\,xy - 2\,yz & 0 \end{pmatrix}$$

Displaying only the non-vanishing components, skipping the redundant ones (i.e. those that can be deduced by antisymmetry):

In [187…
```
a.display_comp(only_nonredundant=True)
```

Out[187…

$$\begin{array}{lcl} \omega\wedge \mathrm{d}f_{\,x\,y} & = & 2\,x^{2}y + 2\,y^{3} - z \\ \omega\wedge \mathrm{d}f_{\,x\,z} & = & 3\,\left(x^{2} + y^{2}\right)z^{2} - x + z \\ \omega\wedge \mathrm{d}f_{\,y\,z} & = & 3\,z^{3} - 2\,xy + 2\,yz \end{array}$$

The 2-form $\omega\wedge \mathrm{d}f$ can be expanded on the $(\mathrm{d}r, \mathrm{d}\theta, \mathrm{d}\phi)$ coframe:

In [188…
```
a.display(Y.frame(), Y)
```

Out[188…

$$\omega\wedge \mathrm{d}f = \Big(3\,r^{5}\cos(\phi)\sin(\theta)^{4} - \Big(3\,r^{5}\cos(\phi) - 3\,r^{4}\cos(\theta)\sin(\phi) - 2\,r^{3}\cos(\phi)\sin(\phi)^{2}\Big)\sin(\theta)^{2}$$

$$- \Big(2\,r^{3}\cos(\theta)\sin(\phi)^{2} - r^{2}\cos(\phi)^{2}\Big)\sin(\theta)\Big)\,\mathrm{d}r\wedge \mathrm{d}\theta$$

$$+ \Big(2\,r^{4}\sin(\phi)\sin(\theta)^{5} + \Big(3\,r^{5}\cos(\theta)^{3}\sin(\phi) + 2\,r^{3}\cos(\phi)^{2}\cos(\theta)\sin(\phi)\Big)\sin(\theta)^{3} - \Big(2\,r^{3}\cos(\phi$$

$$(\theta)^{2} - \Big(3\,r^{4}\cos(\phi)\cos(\theta)^{4} - r^{2}\cos(\theta)^{2}\sin(\phi)\Big)\sin(\theta)\Big)\,\mathrm{d}r\wedge \mathrm{d}\phi$$

$$+ \Big(-r^{3}\cos(\theta)^{2}\sin(\theta) - \Big(3\,r^{6}\cos(\theta)^{2}\sin(\phi) + 2\,r^{4}\cos(\phi)^{2}\sin(\phi) - 2\,r^{5}\cos(\theta)\sin(\phi)\Big)\sin(\theta)^{4}$$

$$+ \Big(3\,r^{5}\cos(\phi)\cos(\theta)^{3} - r^{3}\cos(\theta)\sin(\phi)\Big)\sin(\theta)^{2}\Big)\,\mathrm{d}\theta\wedge \mathrm{d}\phi$$

As a 2-form, $A := \omega\wedge \mathrm{d}f$ can be applied to a pair of vectors and is antisymmetric:

In [189…
```
a.set_name('A')
print(a(u,v))
a(u,v).display()
```

```
Scalar field A(u,v) on the Open subset U of the 3-dimensional differentiable manifol
d M
```

Out[189…

$$
\begin{array}{llll}
A\left(u,v\right): & U & \longrightarrow & \mathbb{R} \\
& (x,y,z) & \longmapsto & 3\,xyz^4 u_y\left(x,y,z\right) - 2\,x^2y^2 u_y\left(x,y,z\right) - \\
& & & - \left(3\,y^3 u_z\left(x,y,z\right) - (2 \\
& & & - \left(2\,x^2y^2 u_y\left(x,y,z\right) + \left(x^2 u_x\left(x,y,\right.\right.\right. \\
& (r,\theta,\phi) & \longmapsto & \left(r^4\cos(\phi)\cos\left(\theta\right)^2\sin(\phi)\sin\left(\theta\right)^2 + \left(\sin\left(\phi\right)^3 - \sin(\phi)\right)r^4\cos(\theta)\sin\right. \\
& & & + \left(3\,r^6\cos(\phi)\cos\left(\theta\right)^4\sin(\phi)\sin\left(\theta\right)^2 + r^2\cos(\theta)\sin(\phi)\sin(\theta) + 2\left(\right.\right. \\
& & & \left(r^5\cos(\phi)\cos\left(\theta\right)^2\sin\left(\phi\right)^2 - r^3\sin(\phi)\right)\sin\left(\theta\right)^3 + r\cos(\theta)\Big) \\
& & & - \left(\left(3\,r^5\cos\left(\theta\right)^2\sin(\phi) - 2\left(\sin\left(\phi\right)^3 - \sin(\phi)\right)r^3\right)\sin\left(\theta\right)^3 + \left(3\,r^4\right.\right. \\
& & & - \left(3\,r^4\cos(\phi)\cos\left(\theta\right)^3 - r^2\cos(\theta)\sin(\phi) + r\cos(\phi)\right)\sin(\theta)\Big)
\end{array}
$$

In [190…

```
a(u,v) == - a(v,u)
```

Out[190… True

In [191…

```
a.symmetries()
```

```
no symmetry; antisymmetry: (0, 1)
```

The **exterior derivative** of a differential form:

In [192…

```
dom = om.exterior_derivative()
print(dom)
dom.display()
```

```
2-form domega on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[192… $\mathrm{d}\omega = -2\,y\mathrm{d}x\wedge\mathrm{d}y + \mathrm{d}x\wedge\mathrm{d}z - \mathrm{d}y\wedge\mathrm{d}z$

Instead of invoking the method `exterior_derivative()`, one can use the function `diff()` (available in SageMath 9.2 or higher):

In [193…

```
dom = diff(om)
```

In [194…

```
da = diff(a)
print(da)
da.display()
```

```
3-form dA on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[194… $\mathrm{d}A = \left(-6\,yz^2 - 2\,y - 1\right)\mathrm{d}x\wedge\mathrm{d}y\wedge\mathrm{d}z$

The exterior derivative is nilpotent:

In [195…

```
ddf = diff(df)
ddf.display()
```

Out[195… $\mathrm{d}\mathrm{d}f = 0$

In [196…

```
ddom = diff(dom)
ddom.display()
```

Out[196… $\mathrm{d}\mathrm{d}\omega = 0$

Lie derivative

The Lie derivative of any tensor field with respect to a vector field is computed by the method `lie_derivative()` , with the vector field as the argument:

In [197…
```
lv_om = om.lie_derivative(v)
print(lv_om)
lv_om.display()
```

```
1-form on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[197…
$$\left(-yz^2 + (xy-1)z + 2\,x\right) \mathrm{d}x + \left(-xz^2 + x^2 + y^2 + (x^2+xy)z\right) \mathrm{d}y + \left(-2\,xyz + (x^2+1)y + 1\right)$$

In [198…
```
lu_dh = dh.lie_derivative(u)
print(lu_dh)
lu_dh.display()
```

```
1-form on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[198…
$$\left(u_x(x,y,z)\frac{\partial^2 H}{\partial x^2} + u_y(x,y,z)\frac{\partial^2 H}{\partial x\partial y} + u_z(x,y,z)\frac{\partial^2 H}{\partial x\partial z} + \frac{\partial H}{\partial x}\frac{\partial u_x}{\partial x} + \frac{\partial H}{\partial y}\frac{\partial u_y}{\partial x} + \frac{\partial H}{\partial z}\frac{\partial u}{\partial x}\right.$$
$$+ \left(u_x(x,y,z)\frac{\partial^2 H}{\partial x\partial y} + u_y(x,y,z)\frac{\partial^2 H}{\partial y^2} + u_z(x,y,z)\frac{\partial^2 H}{\partial y\partial z} + \frac{\partial H}{\partial x}\frac{\partial u_x}{\partial y} + \frac{\partial H}{\partial y}\frac{\partial u_y}{\partial y} + \frac{\partial H}{\partial z}\right.$$
$$+ \left(u_x(x,y,z)\frac{\partial^2 H}{\partial x\partial z} + u_y(x,y,z)\frac{\partial^2 H}{\partial y\partial z} + u_z(x,y,z)\frac{\partial^2 H}{\partial z^2} + \frac{\partial H}{\partial x}\frac{\partial u_x}{\partial z} + \frac{\partial H}{\partial y}\frac{\partial u_y}{\partial z} + \frac{\partial H}{\partial z}\right.$$

Let us check **Cartan identity** on the 1-form ω:

$$\mathcal{L}_v\omega = v\cdot \mathrm{d}\omega + \mathrm{d}\langle\omega, v\rangle$$

and on the 2-form A:

$$\mathcal{L}_v A = v\cdot \mathrm{d}A + \mathrm{d}(v\cdot A)$$

In [199…
```
om.lie_derivative(v) == v.contract(diff(om)) + diff(om(v))
```

Out[199… True

In [200…
```
a.lie_derivative(v) == v.contract(diff(a)) + diff(v.contract(a))
```

Out[200… True

The Lie derivative of a vector field along another one is the **commutator** of the two vectors fields:

In [201…
```
v.lie_derivative(u)(f) == u(v(f)) - v(u(f))
```

Out[201… True

Tensor fields of arbitrary rank

Up to now, we have encountered tensor fields

- of type (0,0) (i.e. scalar fields),
- of type (1,0) (i.e. vector fields),
- of type (0,1) (i.e. 1-forms),
- of type (0,2) and antisymmetric (i.e. 2-forms).

More generally, tensor fields of any type (p, q) can be introduced in SageMath. For instance a tensor field of type (1,2) on the open subset U is declared as follows:

In [202…
```
t = U.tensor_field(1, 2, name='T')
print(t)
```

```
Tensor field T of type (1,2) on the Open subset U of the 3-dimensional differentiabl
e manifold M
```

As for vectors or 1-forms, the tensor's components with respect to the domain's default frame are set by means of square brackets:

In [203…
```
t[1,2,1] = 1 + x^2
t[3,2,1] = x*y*z
```

Unset components are zero:

In [204…
```
t.display()
```

Out[204…
$$T = \left(x^2 + 1\right) \frac{\partial}{\partial x} \otimes \mathrm{d}y \otimes \mathrm{d}x + xyz \frac{\partial}{\partial z} \otimes \mathrm{d}y \otimes \mathrm{d}x$$

In [205…
```
t[:]
```

Out[205…
$$\left[\left[[0,0,0], \left[x^2+1,0,0\right], [0,0,0]\right], [[0,0,0], [0,0,0], [0,0,0]], [[0,0,0], [xyz,0,0], [0,0,0]]\right]$$

Display of the nonzero components:

In [206…
```
t.display_comp()
```

Out[206…
$$\begin{array}{lcl} T^{\,x}_{\,y\,x} & = & x^2 + 1 \\ T^{\,z}_{\,y\,x} & = & xyz \end{array}$$

Double square brackets return the component (still w.r.t. the default frame) as a scalar field, while single square brackets return the expression of this scalar field in terms of the domain's default coordinates:

In [207…
```
print(t[[1,2,1]])
t[[1,2,1]].display()
```

```
Scalar field on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[207…
$$\begin{array}{llcl} & U & \longrightarrow & \mathbb{R} \\ & (x,y,z) & \longmapsto & x^2 + 1 \\ & (r,\theta,\phi) & \longmapsto & r^2 \cos\left(\phi\right)^2 \sin\left(\theta\right)^2 + 1 \end{array}$$

In [208…
```
print(t[1,2,1])
t[1,2,1]
```

```
x^2 + 1
```

Out[208…
$$x^2 + 1$$

A tensor field of type (1,2) maps a 3-tuple (1-form, vector field, vector field) to a scalar field:

In [209…
```
print(t(om, u, v))
t(om, u, v).display()
```

```
Scalar field T(omega,u,v) on the Open subset U of the 3-dimensional differentiable m
anifold M
```

Out[209… $T\left(\omega ,u,v\right): \quad U \longrightarrow \mathbb{R}$

$$(x,y,z) \longmapsto \left(x^2+1\right)y^3u_y\left(x,y,z\right) + \left(x^2+1\right)y^2u_y\left(x,y,z\right) - \left(xy^2u_y\left(x,y,z\right)\right.$$
$$+$$
$$(r,\theta,\phi) \longmapsto \left(r^5\cos\left(\phi\right)^2\sin(\phi)\sin\left(\theta\right)^5 - \left(\left(\cos\left(\phi\right)^4 - \cos\left(\phi\right)^2\right)r^5\cos(\theta) - r^4\right.\right.$$
$$+\left(\left(\cos\left(\phi\right)^3 - \cos(\phi)\right)r^5\cos\left(\theta\right)^2 + r^4\cos\left(\phi\right)^2\cos(\theta)\sin(\phi) + r^3\right.$$
$$($$

As for vectors and differential forms, the tensor components can be taken in any frame defined on the manifold:

In [210…
```
t[Y.frame(), 1,1,1, Y]
```

Out[210… $r^2\cos\left(\phi\right)^4\sin(\phi)\sin\left(\theta\right)^5 + \left(\cos\left(\phi\right)^4 - \cos\left(\phi\right)^2\right)r^3\sin\left(\theta\right)^6 - \left(\cos\left(\phi\right)^4 - \cos\left(\phi\right)^2\right)r^3\sin\left(\theta\right)^4 +$

Tensor calculus

The **tensor product** $\otimes$ is denoted by `*` :

In [211…
```
print(v.tensor_type())
print(a.tensor_type())
```

```
(1, 0)
(0, 2)
```

In [212…
```
b = v*a
print(b)
b
```

```
Tensor field v⊗A of type (1,2) on the Open subset U of the 3-dimensional differentia
ble manifold M
```

Out[212… $v\otimes A$

The tensor product preserves the (anti)symmetries: since A is a 2-form, it is antisymmetric with respect to its two arguments (positions 0 and 1); as a result, b is antisymmetric with respect to its last two arguments (positions 1 and 2):

In [213…
```
a.symmetries()
```

```
no symmetry; antisymmetry: (0, 1)
```

In [214…
```
b.symmetries()
```

```
no symmetry; antisymmetry: (1, 2)
```

Standard **tensor arithmetics** is implemented:

In [215…
```
s = - t + 2*f* b
print(s)
```

```
Tensor field of type (1,2) on the Open subset U of the 3-dimensional differentiable
manifold M
```

Tensor contractions are dealt with by the methods `trace()` and `contract()` : for instance, let us contract the tensor T w.r.t. its first two arguments (positions 0 and 1), i.e. let us form the tensor c of components $c_i = T^k_{\ \ ki}$:

In [216…
```
c = t.trace(0,1)
print(c)
```

```
1-form on the Open subset U of the 3-dimensional differentiable manifold M
```

An alternative to the writing `trace(0,1)` is to use the **index notation** to denote the contraction: the indices are given in a string inside the `[]` operator, with `'^'` in front of the contravariant indices and `'_'` in front of the covariant ones:

In [217…
```
c1 = t['^k_ki']
print(c1)
c1 == c
```

```
1-form on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[217… True

The contraction is performed on the repeated index (here `k`); the letter denoting the remaining index (here `i`) is arbitrary:

In [218…
```
t['^k_kj'] == c
```

Out[218… True

In [219…
```
t['^b_ba'] == c
```

Out[219… True

It can even be replaced by a dot:

In [220…
```
t['^k_k.'] == c
```

Out[220… True

LaTeX notations are allowed:

In [221…
```
t['^{k}_{ki}'] == c
```

Out[221… True

as well as Greek letters (only for SageMath 9.2 or higher):

In [222…
```
t['^μ_μα'] == c
```

Out[222… True

The contraction $T^i_{\ jk} v^k$ of the tensor fields T and v is taken as follows (2 refers to the last index position of T and 0 to the only index position of v):

In [223…
```
tv = t.contract(2, v, 0)
print(tv)
```

```
Tensor field of type (1,1) on the Open subset U of the 3-dimensional differentiable
manifold M
```

Since 2 corresponds to the last index position of T and 0 to the first index position of v, a shortcut for the above is

In [224…
```
tv1 = t.contract(v)
print(tv1)
```

```
Tensor field of type (1,1) on the Open subset U of the 3-dimensional differentiable
manifold M
```

In [225…
```
tv1 == tv
```

Out[225… True

Instead of `contract()`, the **index notation**, combined with the * operator, can be used to denote the contraction:

In [226…
```
t['^i_jk']*v['^k'] == tv
```

Out[226… True

The non-repeated indices can be replaced by dots:

In [227…
```
t['^._.k']*v['^k'] == tv
```

Out[227… True

Metric structures

A **Riemannian metric** on the manifold $\mathcal{M}$ is declared as follows:

In [228…
```
g = M.riemannian_metric('g')
print(g)
```

```
Riemannian metric g on the 3-dimensional differentiable manifold M
```

It is a symmetric tensor field of type (0,2):

In [229…
```
g.parent()
```

Out[229… $\mathcal{T}^{(0,2)}\left(\mathcal{M}\right)$

In [230…
```
print(g.parent())
```

```
Free module T^(0,2)(M) of type-(0,2) tensors fields on the 3-dimensional differentia
ble manifold M
```

In [231…
```
g.symmetries()
```

```
symmetry: (0, 1); no antisymmetry
```

The metric is initialized by its components with respect to some vector frame. For instance, using the default frame of $\mathcal{M}$:

In [232…
```
g[1,1], g[2,2], g[3,3] = 1, 1, 1
g.display()
```

Out[232… $g = \mathrm{d} x\otimes \mathrm{d} x+\mathrm{d} y\otimes \mathrm{d} y+\mathrm{d} z\otimes \mathrm{d} z$

The components w.r.t. another vector frame are obtained as for any tensor field:

In [233…
```
g.display(Y.frame(), Y)
```

Out[233… $g = \mathrm{d} r\otimes \mathrm{d} r + r^{2} \mathrm{d} \theta\otimes \mathrm{d} \theta + r^{2} \sin\left(\theta\right)^{2} \mathrm{d} \phi\otimes \mathrm{d} \phi$

For a coordinate frame, as a shortcut, one may provide only the name of the chart:

In [234…
```
g.display(Y)
```

Out[234… $g = \mathrm{d} r\otimes \mathrm{d} r + r^{2} \mathrm{d} \theta\otimes \mathrm{d} \theta + r^{2} \sin\left(\theta\right)^{2} \mathrm{d} \phi\otimes \mathrm{d} \phi$

Of course, the metric acts on vector pairs:

In [235…
```
print(g(u,v))
g(u,v).display()
```

```
Scalar field g(u,v) on the Open subset U of the 3-dimensional differentiable manifol
d M
```

Out[235…
$$\begin{array}{llcl} g(u,v): & U & \longrightarrow & \mathbb{R} \\ & (x,y,z) & \longmapsto & xyzu_z(x,y,z) + yu_x(x,y,z) - xu_y(x,y,z) + u_x(x,y,z) \\ & (r,\theta,\phi) & \longmapsto & r^3\cos(\phi)\cos(\theta)\sin(\phi)\sin(\theta)^2 u_z(r\cos(\phi)\sin(\theta), r\sin(\phi)\sin(\theta), r\cos \\ & & & + (r\sin(\phi)\sin(\theta)+1)u_x(r\cos(\phi) \end{array}$$

The **Levi-Civita connection** associated to the metric g:

In [236…
```
nabla = g.connection()
print(nabla)
nabla
```

```
Levi-Civita connection nabla_g associated with the Riemannian metric g on the 3-dime
nsional differentiable manifold M
```

Out[236… ∇_g

The Christoffel symbols with respect to the manifold's default coordinates:

In [237…
```
nabla.coef()[:]
```

Out[237… $[[[0,0,0],[0,0,0],[0,0,0]],[[0,0,0],[0,0,0],[0,0,0]],[[0,0,0],[0,0,0],[0,0,0]]]$

The Christoffel symbols with respect to the coordinates (r,θ,ϕ):

In [238…
```
nabla.coef(Y.frame())[:, Y]
```

Out[238…
$$\left[\left[[0,0,0],[0,-r,0],\left[0,0,-r\sin(\theta)^2\right]\right],\left[\left[0,\frac{1}{r},0\right],\left[\frac{1}{r},0,0\right],[0,0,-\cos(\theta)\sin(\theta)]\right],\left[\left[0,0,\frac{1}{r}\right.\right.$$

A nice view is obtained via the method `display()` (by default, only the nonzero connection coefficients are shown):

In [239…
```
nabla.display(frame=Y.frame(), chart=Y)
```

Out[239…
$$\begin{array}{lcl} \Gamma^{r}{}_{\theta\theta} & = & -r \\ \Gamma^{r}{}_{\phi\phi} & = & -r\sin(\theta)^2 \\ \Gamma^{\theta}{}_{r\theta} & = & \frac{1}{r} \\ \Gamma^{\theta}{}_{\theta r} & = & \frac{1}{r} \\ \Gamma^{\theta}{}_{\phi\phi} & = & -\cos(\theta)\sin(\theta) \\ \Gamma^{\phi}{}_{r\phi} & = & \frac{1}{r} \\ \Gamma^{\phi}{}_{\theta\phi} & = & \frac{\cos(\theta)}{\sin(\theta)} \\ \Gamma^{\phi}{}_{\phi r} & = & \frac{1}{r} \\ \Gamma^{\phi}{}_{\phi\theta} & = & \frac{\cos(\theta)}{\sin(\theta)} \end{array}$$

One may also use the method `christoffel_symbols_display()` of the metric, which (by default) displays only the non-redundant Christoffel symbols:

In [240…
```
g.christoffel_symbols_display(Y)
```

Out[240…
$$\Gamma^{r}{}_{\theta\,\theta} = -r$$
$$\Gamma^{r}{}_{\phi\,\phi} = -r\sin(\theta)^2$$
$$\Gamma^{\theta}{}_{r\,\theta} = \frac{1}{r}$$
$$\Gamma^{\theta}{}_{\phi\,\phi} = -\cos(\theta)\sin(\theta)$$
$$\Gamma^{\phi}{}_{r\,\phi} = \frac{1}{r}$$
$$\Gamma^{\phi}{}_{\theta\,\phi} = \frac{\cos(\theta)}{\sin(\theta)}$$

The connection acting as a covariant derivative:

In [241…
```
nab_v = nabla(v)
print(nab_v)
nab_v.display()
```

```
Tensor field nabla_g(v) of type (1,1) on the Open subset U of the 3-dimensional diff
erentiable manifold M
```

Out[241…
$$\nabla_g v = \frac{\partial}{\partial x}\otimes \mathrm{d}y - \frac{\partial}{\partial y}\otimes \mathrm{d}x + yz\frac{\partial}{\partial z}\otimes \mathrm{d}x + xz\frac{\partial}{\partial z}\otimes \mathrm{d}y + xy\frac{\partial}{\partial z}\otimes \mathrm{d}z$$

Being a Levi-Civita connection, ∇_g is torsion.free:

In [242…
```
print(nabla.torsion())
nabla.torsion().display()
```

```
Tensor field of type (1,2) on the 3-dimensional differentiable manifold M
```

Out[242… 0

In the present case, it is also flat:

In [243…
```
print(nabla.riemann())
nabla.riemann().display()
```

```
Tensor field Riem(g) of type (1,3) on the 3-dimensional differentiable manifold M
```

Out[243… $\mathrm{Riem}\left(g\right) = 0$

Let us consider a non-flat metric, by changing g_{rr} to $1/(1+r^2)$:

In [244…
```
g[Y.frame(), 1,1, Y] = 1/(1+r^2)
g.display(Y.frame(), Y)
```

Out[244…
$$g = \left(\frac{1}{r^2+1}\right)\mathrm{d}r\otimes \mathrm{d}r + r^2\mathrm{d}\theta\otimes \mathrm{d}\theta + r^2\sin(\theta)^2\mathrm{d}\phi\otimes \mathrm{d}\phi$$

For convenience, we change the default chart on the domain U to Y=$(U,(r,\theta,\phi))$:

In [245…
```
U.set_default_chart(Y)
```

In this way, we do not have to specify Y when asking for coordinate expressions in terms of (r,θ,ϕ):

In [246…
```
g.display(Y.frame())
```

Out[246…
$$g = \left(\frac{1}{r^2+1}\right)\mathrm{d}r\otimes \mathrm{d}r + r^2\mathrm{d}\theta\otimes \mathrm{d}\theta + r^2\sin(\theta)^2\mathrm{d}\phi\otimes \mathrm{d}\phi$$

We recognize the metric of the hyperbolic space $\mathbb{H}^3$. Its expression in terms of the chart $(U,(x,y,z))$ is

In [247…
```
g.display(X_U.frame(), X_U)
```

Out[247…
$$g = \left(\frac{y^2 + z^2 + 1}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}x \otimes \mathrm{d}x + \left(-\frac{xy}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}x \otimes \mathrm{d}y + \left(-\frac{xz}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}$$
$$+ \left(\frac{x^2 + z^2 + 1}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}y \otimes \mathrm{d}y + \left(-\frac{yz}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}y \otimes \mathrm{d}z + \left(-\frac{xz}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}z$$
$$+ \left(\frac{x^2 + y^2 + 1}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}z \otimes \mathrm{d}z$$

A matrix view of the components may be more appropriate:

In [248…
```
g[X_U.frame(), :, X_U]
```

Out[248…
$$\begin{pmatrix} \frac{y^2+z^2+1}{x^2+y^2+z^2+1} & -\frac{xy}{x^2+y^2+z^2+1} & -\frac{xz}{x^2+y^2+z^2+1} \\ -\frac{xy}{x^2+y^2+z^2+1} & \frac{x^2+z^2+1}{x^2+y^2+z^2+1} & -\frac{yz}{x^2+y^2+z^2+1} \\ -\frac{xz}{x^2+y^2+z^2+1} & -\frac{yz}{x^2+y^2+z^2+1} & \frac{x^2+y^2+1}{x^2+y^2+z^2+1} \end{pmatrix}$$

We extend these components, a priori defined only on U, to the whole manifold $\mathcal{M}$, by demanding the same coordinate expressions in the frame associated to the chart X=$(\mathcal{M}, (x, y, z))$:

In [249…
```
g.add_comp_by_continuation(X.frame(), U, X)
g.display()
```

Out[249…
$$g = \left(\frac{y^2 + z^2 + 1}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}x \otimes \mathrm{d}x + \left(-\frac{xy}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}x \otimes \mathrm{d}y + \left(-\frac{xz}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}$$
$$+ \left(\frac{x^2 + z^2 + 1}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}y \otimes \mathrm{d}y + \left(-\frac{yz}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}y \otimes \mathrm{d}z + \left(-\frac{xz}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}z$$
$$+ \left(\frac{x^2 + y^2 + 1}{x^2 + y^2 + z^2 + 1}\right) \mathrm{d}z \otimes \mathrm{d}z$$

The Levi-Civita connection is automatically recomputed, after the change in g:

In [250…
```
nabla = g.connection()
```

In particular, the Christoffel symbols are different:

In [251…
```
nabla.display(only_nonredundant=True)
```

Out[251…
$$\Gamma^{x}{}_{x\,x} = -\frac{xy^2+xz^2+x}{x^2+y^2+z^2+1}$$
$$\Gamma^{x}{}_{x\,y} = \frac{x^2y}{x^2+y^2+z^2+1}$$
$$\Gamma^{x}{}_{x\,z} = \frac{x^2z}{x^2+y^2+z^2+1}$$
$$\Gamma^{x}{}_{y\,y} = -\frac{x^3+xz^2+x}{x^2+y^2+z^2+1}$$
$$\Gamma^{x}{}_{y\,z} = \frac{xyz}{x^2+y^2+z^2+1}$$
$$\Gamma^{x}{}_{z\,z} = -\frac{x^3+xy^2+x}{x^2+y^2+z^2+1}$$
$$\Gamma^{y}{}_{x\,x} = -\frac{y^3+yz^2+y}{x^2+y^2+z^2+1}$$
$$\Gamma^{y}{}_{x\,y} = \frac{xy^2}{x^2+y^2+z^2+1}$$
$$\Gamma^{y}{}_{x\,z} = \frac{xyz}{x^2+y^2+z^2+1}$$
$$\Gamma^{y}{}_{y\,y} = -\frac{yz^2+(x^2+1)y}{x^2+y^2+z^2+1}$$
$$\Gamma^{y}{}_{y\,z} = \frac{y^2z}{x^2+y^2+z^2+1}$$
$$\Gamma^{y}{}_{z\,z} = -\frac{y^3+(x^2+1)y}{x^2+y^2+z^2+1}$$
$$\Gamma^{z}{}_{x\,x} = -\frac{z^3+(y^2+1)z}{x^2+y^2+z^2+1}$$
$$\Gamma^{z}{}_{x\,y} = \frac{xyz}{x^2+y^2+z^2+1}$$
$$\Gamma^{z}{}_{x\,z} = \frac{xz^2}{x^2+y^2+z^2+1}$$
$$\Gamma^{z}{}_{y\,y} = -\frac{z^3+(x^2+1)z}{x^2+y^2+z^2+1}$$
$$\Gamma^{z}{}_{y\,z} = \frac{yz^2}{x^2+y^2+z^2+1}$$
$$\Gamma^{z}{}_{z\,z} = -\frac{(x^2+y^2+1)z}{x^2+y^2+z^2+1}$$

In [252…
```
nabla.display(frame=Y.frame(), chart=Y, only_nonredundant=True)
```

Out[252…
$$\Gamma^{r}{}_{r\,r} = -\frac{r}{r^2+1}$$
$$\Gamma^{r}{}_{\theta\,\theta} = -r^3 - r$$
$$\Gamma^{r}{}_{\phi\,\phi} = -\left(r^3 + r\right)\sin\left(\theta\right)^2$$
$$\Gamma^{\theta}{}_{r\,\theta} = \frac{1}{r}$$
$$\Gamma^{\theta}{}_{\phi\,\phi} = -\cos(\theta)\sin(\theta)$$
$$\Gamma^{\phi}{}_{r\,\phi} = \frac{1}{r}$$
$$\Gamma^{\phi}{}_{\theta\,\phi} = \frac{\cos(\theta)}{\sin(\theta)}$$

The **Riemann tensor** is now

In [253…
```
Riem = nabla.riemann()
print(Riem)
Riem.display(Y.frame())
```

```
Tensor field Riem(g) of type (1,3) on the 3-dimensional differentiable manifold M
```

Out[253…

$$\mathrm{Riem}\left(g\right) = -r^2 \frac{\partial}{\partial r} \otimes \mathrm{d}\theta \otimes \mathrm{d}r \otimes \mathrm{d}\theta + r^2 \frac{\partial}{\partial r} \otimes \mathrm{d}\theta \otimes \mathrm{d}\theta \otimes \mathrm{d}r - r^2 \sin\left(\theta\right)^2 \frac{\partial}{\partial r} \otimes \mathrm{d}\phi \otimes \mathrm{d}r \otimes \mathrm{d}\phi + \otimes \mathrm{d}r \otimes \mathrm{d}\theta + \left(-\frac{1}{r^2+1}\right) \frac{\partial}{\partial \theta} \otimes \mathrm{d}r \otimes \mathrm{d}\theta \otimes \mathrm{d}r - r^2 \sin\left(\theta\right)^2 \frac{\partial}{\partial \theta} \otimes \mathrm{d}\phi \otimes \mathrm{d}\theta \otimes \mathrm{d}\phi + r^2 \sin\left(\theta\right)^2 \frac{\partial}{\partial \theta} + \left(-\frac{1}{r^2+1}\right) \frac{\partial}{\partial \phi} \otimes \mathrm{d}r \otimes \mathrm{d}\phi \otimes \mathrm{d}r + r^2 \frac{\partial}{\partial \phi} \otimes \mathrm{d}\theta \otimes \mathrm{d}\theta \otimes \mathrm{d}\phi - r^2 \frac{\partial}{\partial \phi} \otimes \mathrm{d}\theta \otimes \mathrm{d}\phi \otimes \mathrm{d}\theta$$

Note that it can be accessed directely via the metric, without any explicit mention of the connection:

In [254…

```
g.riemann() is nabla.riemann()
```

Out[254… True

The **Ricci tensor** is

In [255…

```
Ric = g.ricci()
print(Ric)
Ric.display(Y.frame())
```

```
Field of symmetric bilinear forms Ric(g) on the 3-dimensional differentiable manifol
d M
```

Out[255…

$$\mathrm{Ric}\left(g\right) = \left(-\frac{2}{r^2+1}\right) \mathrm{d}r \otimes \mathrm{d}r - 2\,r^2 \mathrm{d}\theta \otimes \mathrm{d}\theta - 2\,r^2 \sin\left(\theta\right)^2 \mathrm{d}\phi \otimes \mathrm{d}\phi$$

The **Weyl tensor** is:

In [256…

```
C = g.weyl()
print(C)
C.display()
```

```
Tensor field C(g) of type (1,3) on the 3-dimensional differentiable manifold M
```

Out[256… $\mathrm{C}\left(g\right) = 0$

The Weyl tensor vanishes identically because the dimension of $\mathcal{M}$ is 3.

Finally, the **Ricci scalar** is

In [257…

```
R = g.ricci_scalar()
print(R)
R.display()
```

```
Scalar field r(g) on the 3-dimensional differentiable manifold M
```

Out[257…

$$\begin{array}{llcl} \mathrm{r}\left(g\right): & \mathcal{M} & \longrightarrow & \mathbb{R} \\ & (x, y, z) & \longmapsto & -6 \\ \text{on } U: & (r, \theta, \phi) & \longmapsto & -6 \end{array}$$

We recover that $\mathbb{H}^3$ is a Riemannian manifold of constant negative curvature.

Tensor transformations induced by a metric

The most important tensor transformation induced by the metric g is the so-called **musical isomorphism**, or **index raising** and **index lowering**:

In [258…

```
print(t)
```

```
Tensor field T of type (1,2) on the Open subset U of the 3-dimensional differentiabl
e manifold M
```

In [259…

```
t.display()
```

Out[259…] $T = \left(r^2 \cos(\phi)^2 \sin(\theta)^2 + 1\right) \frac{\partial}{\partial x} \otimes \mathrm{d}y \otimes \mathrm{d}x + r^3 \cos(\phi) \cos(\theta) \sin(\phi) \sin(\theta)^2 \frac{\partial}{\partial z} \otimes \mathrm{d}y \otimes \mathrm{d}x$

In [260…]
```
t.display(X_U.frame(), X_U)
```

Out[260…] $T = \left(x^2 + 1\right) \frac{\partial}{\partial x} \otimes \mathrm{d}y \otimes \mathrm{d}x + xyz \frac{\partial}{\partial z} \otimes \mathrm{d}y \otimes \mathrm{d}x$

Raising the last index (position 2) of T with g:

In [261…]
```
s = t.up(g, 2)
print(s)
```

```
Tensor field of type (2,1) on the Open subset U of the 3-dimensional differentiable
manifold M
```

Note that the raised index becomes the *last* one among the contravariant indices, i.e. the tensor s returned by the method `up` is

$$s^{ab}{}_c = g^{bi} T^a{}_{ic}$$

See the up() documentation for more details.

Raising all the covariant indices of T (i.e. those at the positions 1 and 2):

In [262…]
```
s = t.up(g)
print(s)
```

```
Tensor field of type (3,0) on the Open subset U of the 3-dimensional differentiable
manifold M
```

Lowering all contravariant indices of T (i.e. the index at position 0):

In [263…]
```
s = t.down(g)
print(s)
```

```
Tensor field of type (0,3) on the Open subset U of the 3-dimensional differentiable
manifold M
```

Note that the lowered index becomes the *first* one among the covariant indices, i.e. the tensor s returned by the method `down` is

$$s_{abc} = g_{ai} T^i{}_{bc}$$

See the down() documentation for more details.

Hodge duality

The volume 3-form (Levi-Civita tensor) associated with the metric g is

In [264…]
```
epsilon = g.volume_form()
print(epsilon)
epsilon.display()
```

```
3-form eps_g on the 3-dimensional differentiable manifold M
```

Out[264…] $\epsilon_g = \left(\frac{1}{\sqrt{x^2 + y^2 + z^2 + 1}}\right) \mathrm{d}x \wedge \mathrm{d}y \wedge \mathrm{d}z$

In [265…]
```
epsilon.display(Y.frame())
```

Out[265...

$$\epsilon_g = \left(\frac{r^2 \sin(\theta)}{\sqrt{r^2+1}} \right) \mathrm{d}r \wedge \mathrm{d}\theta \wedge \mathrm{d}\phi$$

In [266...

```
print(f)
f.display()
```

```
Scalar field f on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[266...

$$\begin{array}{llcl} f: & U & \longrightarrow & \mathbb{R} \\ & (x, y, z) & \longmapsto & z^3 + y^2 + x \\ & (r, \theta, \phi) & \longmapsto & r^3 \cos(\theta)^3 + r^2 \sin(\phi)^2 \sin(\theta)^2 + r\cos(\phi)\sin(\theta) \end{array}$$

In [267...

```
sf = f.hodge_dual(g.restrict(U))
print(sf)
sf.display()
```

```
3-form *f on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[267...

$$\star f = \left(\frac{r^3 \cos(\theta)^3 + r^2 \sin(\phi)^2 \sin(\theta)^2 + r\cos(\phi)\sin(\theta)}{\sqrt{r^2+1}} \right) \mathrm{d}x \wedge \mathrm{d}y \wedge \mathrm{d}z$$

We check the classical formula $\star f = f\,\epsilon_g$, or, more precisely, $\star f = f\,\epsilon_g|_U$ (for f is defined on U only):

In [268...

```
sf == f * epsilon.restrict(U)
```

Out[268...

True

The Hodge dual of a 1-form is a 2-form:

In [269...

```
print(om)
om.display()
```

```
1-form omega on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[269...

$$\omega = r^2 \sin(\theta)^2 \mathrm{d}x + r\cos(\theta)\mathrm{d}y + \left(r\cos(\phi)\sin(\theta) - r\cos(\theta)\right)\mathrm{d}z$$

In [270...

```
som = om.hodge_dual(g)
print(som)
som.display()
```

```
2-form *omega on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[270...

$$\star\omega = \left(\frac{r^4 \cos(\phi)\cos(\theta)\sin(\theta)^3 - r^3\cos(\theta)^3 - r\cos(\theta) + \left(r^3(\cos(\phi)+\sin(\phi))\cos(\theta)^2 + r\cos(\cdots}{\sqrt{r^2+1}} \right.$$

$$+ \left(-\frac{r^4\cos(\phi)\sin(\phi)\sin(\theta)^4 - r^3\cos(\theta)^2\sin(\phi)\sin(\theta) + \left(\cos(\phi)\sin(\phi) + \sin(\phi)^2\right) r^3 \cos(\theta)\,\mathrm{si} \cdots}{\sqrt{r^2+1}} \right.$$

$$+ \left(\frac{r^4\cos(\phi)^2\sin(\theta)^4 - r^3\cos(\phi)\cos(\theta)^2\sin(\theta) + \left(\left(\cos(\phi)^2 + \cos(\phi)\sin(\phi)\right) r^3\cos(\theta) + r^2\right) \cdots}{\sqrt{r^2+1}} \right.$$

The Hodge dual of a 2-form is a 1-form:

In [271...

```
print(a)
```

```
2-form A on the Open subset U of the 3-dimensional differentiable manifold M
```

In [272...

```
sa = a.hodge_dual(g)
print(sa)
sa.display()
```

```
1-form *A on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[272…

$$\star A = \left(\frac{3\,r^5 \cos\left(\theta\right)^5 + 3\,r^3 \cos\left(\theta\right)^3 + \left(3\,r^6 \cos(\phi) \cos\left(\theta\right)^2 \sin(\phi) - 2\,r^5 \cos(\phi) \cos(\theta) \sin(\phi) - 2 \ldots + \left(2\,r^4 \cos(\theta) \sin\left(\phi\right)^3 + \left(\sin\left(\phi\right)^3 - \sin(\phi)\right) r^3\right) \sin\left(\theta\right)^3 + \left(3\,r^5 \cos\left(\theta\right)^3 \sin\left(\phi\right)^2 - 2\,r^4 \cos(\phi) \cos\left(\theta\right)^2 \sin(\phi) + r^3 \cos(\phi) \cos(\theta) \sin(\phi) - 2\,r^2 \cos \ldots + \left(2\,r^4 \cos\left(\theta\right)^3 \sin(\phi) + r^3 \cos(\phi) \cos\left(\theta\right)^2 + 2\,r^2 \cos(\theta) \sin(\phi)\right) \sin(\theta)}{\sqrt{r^2+1}}\right.$$

$$+ \left(-\frac{r^3 \cos\left(\theta\right)^3 + \left(3\,r^6 \cos\left(\phi\right)^2 \cos\left(\theta\right)^2 - 2\left(\cos\left(\phi\right)^2 - 1\right) r^5 \cos(\theta) + 2\left(\cos\left(\phi\right)^4 - \cos\left(\phi\right)^2\right) r \ldots - \left(r^3 \cos\left(\phi\right)^3 + 2\left(\cos\left(\phi\right)^3 - \cos(\phi)\right) r^4 \cos(\theta)\right) \sin\left(\theta\right)^3 + \left(3\,r^6 \cos\left(\theta\right)^4 + 3\,r^5 \cos(\phi) \cos \ldots + r\cos(\theta) - \left(r^3(\cos(\phi) + \sin(\phi)) \cos\left(\theta\right)^2 + r\cos(\phi)\right) \sin(\theta)}{\sqrt{r^2+1}}\right.$$

$$+ \left(\frac{2\,r^5 \sin(\phi) \sin\left(\theta\right)^5 + \left(3\,r^6 \cos\left(\theta\right)^3 \sin(\phi) + 2\,r^4 \cos\left(\phi\right)^2 \cos(\theta) \sin(\phi) + 2\,r^3 \sin(\phi)\right) \sin\left(\theta\right)^3 - \left(2\,r^4 \cos(\phi) \cos\left(\theta\right)^2 \sin(\phi) + (\cos(\phi)\sin(\phi) + 1) r^3 \cos(\theta)\right) \sin\left(\theta\right)^2 - r\cos(\theta) - \left(3\,r^5 \cos \ldots\right.}{\sqrt{r^2+1}}\right.$$

Finally, the Hodge dual of a 3-form is a 0-form, i.e. a scalar field:

In [273…

```
print(da)
da.display()
```

```
3-form dA on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[273…

$$dA = \left(-2\left(3\,r^3 \cos\left(\theta\right)^2 \sin(\phi) + r\sin(\phi)\right) \sin(\theta) - 1\right) \mathrm{d}x \wedge \mathrm{d}y \wedge \mathrm{d}z$$

In [274…

```
sda = da.hodge_dual(g)
print(sda)
sda.display()
```

```
Scalar field *dA on the Open subset U of the 3-dimensional differentiable manifold M
```

Out[274…

$$\begin{array}{llcl} \star\mathrm{d}A: & U & \longrightarrow & \mathbb{R} \\ & (x,y,z) & \longmapsto & -\left(6\,yz^2 + 2\,y + 1\right)\sqrt{x^2+y^2+z^2+1} \\ & (r,\theta,\phi) & \longmapsto & -\sqrt{r^2+1}\left(2\left(3\,r^3 \cos\left(\theta\right)^2 \sin(\phi) + r\sin(\phi)\right) \sin(\theta) + 1\right) \end{array}$$

In dimension 3 and for a Riemannian metric, the Hodge star is idempotent:

In [275…

```
sf.hodge_dual(g) == f
```

Out[275… True

In [276…
```
som.hodge_dual(g) == om
```
Out[276… True

In [277…
```
sa.hodge_dual(g) == a
```
Out[277… True

In [278…
```
sda.hodge_dual(g.restrict(U)) == da
```
Out[278… True

Getting help

To get the list of functions (methods) that can be called on a object, type the name of the object, followed by a dot and press the TAB key, e.g. `sa.<TAB>` .

To get information on an object or a method, use the question mark:

In [279…
```
nabla?
```

```
Signature:      nabla(tensor)
Type:           LeviCivitaConnection
String form:    Levi-Civita connection nabla_g associated with the Riemannian metric
g on the 3-dimensional differentiable manifold M
File:           ~/sage/10.5/src/sage/manifolds/differentiable/levi_civita_connectio
n.py
Docstring:
   Levi-Civita connection on a pseudo-Riemannian manifold.

   Let M be a differentiable manifold of class C^\infty (smooth
   manifold) over \RR endowed with a pseudo-Riemannian metric g. Let
   C^\infty(M) be the algebra of smooth functions M--> \RR
   (cf. "DiffScalarFieldAlgebra") and let \mathfrak{X}(M) be the
   C^\infty(M)-module of vector fields on M (cf. "VectorFieldModule").
   The *Levi-Civita connection associated with* g is the unique
   operator

      \begin{array}{cccc} \nabla: & \mathfrak{X}(M) x
      \mathfrak{X}(M) & ---> &          \mathfrak{X}(M) \\
      & (u,v) &  |--->  & \nabla_u v \end{array}

   that

   * is \RR-bilinear, i.e. is bilinear when considering
     \mathfrak{X}(M) as a vector space over \RR

   * is C^\infty(M)-linear w.r.t. the first argument: \forall f\in
     C^\infty(M),\ \nabla_{fu} v = f\nabla_u v

   * obeys Leibniz rule w.r.t. the second argument: \forall f\in
     C^\infty(M),\ \nabla_u (f v) = df(u)\, v + f  \nabla_u v

   * is torsion-free

   * is compatible with g: \forall (u,v,w)\in \mathfrak{X}(M)^3,\
     u(g(v,w)) = g(\nabla_u v, w) + g(v, \nabla_u w)

   The Levi-Civita connection \nabla gives birth to the *covariant
   derivative operator* acting on tensor fields, denoted by the same
   symbol:

      \begin{array}{cccc} \nabla: &  T^{(k,l)}(M) & ---> &
      T^{(k,l+1)}(M)\\         & t &  |--->  & \nabla t
      \end{array}

   where T^{(k,l)}(M) stands for the C^\infty(M)-module of tensor
   fields of type (k,l) on M (cf. "TensorFieldModule"), with the
   convention T^{(0,0)}(M):=C^\infty(M). For a vector field v,  the
   covariant derivative \nabla v is a type-(1,1) tensor field such
   that

      \forall u \in\mathfrak{X}(M), \   \nabla_u v = \nabla v(., u)

   More generally for any tensor field t\in T^{(k,l)}(M), we have

      \forall u \in\mathfrak{X}(M), \   \nabla_u t = \nabla t(...,
      u)

   Note:

     The above convention means that, in terms of index notation, the
     "derivation index" in \nabla t is the *last* one:

        \nabla_c t^{a_1... a_k}_{\quad\quad b_1... b_l} =
        (\nabla t)^{a_1... a_k}_{\quad\quad b_1... b_l c}

   INPUT:
```

```
    * "metric" -- the metric g defining the Levi-Civita connection, as
      an instance of class "PseudoRiemannianMetric"

    * "name" -- name given to the connection

    * "latex_name" -- (default: "None") LaTeX symbol to denote the
      connection

    * "init_coef" -- boolean (default: "True"); determines whether the
      Christoffel symbols are initialized (in the top charts on the
      domain, i.e. disregarding the subcharts)

    EXAMPLES:

    Levi-Civita connection associated with the Euclidean metric on
    \RR^3 expressed in spherical coordinates:

       sage: forget() # for doctests only
       sage: M = Manifold(3, 'R^3', start_index=1)
       sage: c_spher.<r,th,ph> = M.chart(r'r:(0,+oo) th:(0,pi):\theta ph:(0,2*pi):\ph
i')
       sage: g = M.metric('g')
       sage: g[1,1], g[2,2], g[3,3] = 1, r^2 , (r*sin(th))^2
       sage: g.display()
       g = dr⊗dr + r^2 dth⊗dth + r^2*sin(th)^2 dph⊗dph
       sage: nab = g.connection(name='nabla', latex_name=r'\nabla') ; nab
       Levi-Civita connection nabla associated with the Riemannian metric g on
        the 3-dimensional differentiable manifold R^3

    Let us check that the connection is compatible with the metric:

       sage: Dg = nab(g) ; Dg
       Tensor field nabla(g) of type (0,3) on the 3-dimensional
        differentiable manifold R^3
       sage: Dg == 0
       True

    and that it is torsionless:

       sage: nab.torsion() == 0
       True

    As a check, let us enforce the computation of the torsion:

       sage: sage.manifolds.differentiable.affine_connection.AffineConnection.torsion
(nab) == 0
       True

    The connection coefficients in the manifold's default frame are
    Christoffel symbols, since the default frame is a coordinate frame:

       sage: M.default_frame()
       Coordinate frame (R^3, (∂/∂r,∂/∂th,∂/∂ph))
       sage: nab.coef()
       3-indices components w.r.t. Coordinate frame (R^3, (∂/∂r,∂/∂th,∂/∂ph)),
        with symmetry on the index positions (1, 2)

    We note that the Christoffel symbols are symmetric with respect to
    their last two indices (positions (1,2)); their expression is:

       sage: nab.coef()[:]  # display as a array
       [[[0, 0, 0], [0, -r, 0], [0, 0, -r*sin(th)^2]],
        [[0, 1/r, 0], [1/r, 0, 0], [0, 0, -cos(th)*sin(th)]],
        [[0, 0, 1/r], [0, 0, cos(th)/sin(th)], [1/r, cos(th)/sin(th), 0]]]
       sage: nab.display()  # display only the non-vanishing symbols
       Gam^r_th,th = -r
```

```
   Gam^r_ph,ph = -r*sin(th)^2
   Gam^th_r,th = 1/r
   Gam^th_th,r = 1/r
   Gam^th_ph,ph = -cos(th)*sin(th)
   Gam^ph_r,ph = 1/r
   Gam^ph_th,ph = cos(th)/sin(th)
   Gam^ph_ph,r = 1/r
   Gam^ph_ph,th = cos(th)/sin(th)
   sage: nab.display(only_nonredundant=True)  # skip redundancy due to symmetry
   Gam^r_th,th = -r
   Gam^r_ph,ph = -r*sin(th)^2
   Gam^th_r,th = 1/r
   Gam^th_ph,ph = -cos(th)*sin(th)
   Gam^ph_r,ph = 1/r
   Gam^ph_th,ph = cos(th)/sin(th)

The same display can be obtained via the function
"christoffel_symbols_display()" acting on the metric:

   sage: g.christoffel_symbols_display(chart=c_spher)
   Gam^r_th,th = -r
   Gam^r_ph,ph = -r*sin(th)^2
   Gam^th_r,th = 1/r
   Gam^th_ph,ph = -cos(th)*sin(th)
   Gam^ph_r,ph = 1/r
   Gam^ph_th,ph = cos(th)/sin(th)
```
Init docstring: Construct a Levi-Civita connection.

Call docstring:

```
Action of the connection on a tensor field.

INPUT:

* "tensor" -- a tensor field T, of type (k,\ell)

OUTPUT: tensor field \nabla T
```

In [280…

```
g.ricci_scalar?
```

```
Signature:      g.ricci_scalar(name=None, latex_name=None)
Docstring:
   Return the metric's Ricci scalar.

   The Ricci scalar is the scalar field r defined from the Ricci
   tensor Ric and the metric tensor g by

      r = g^{ij} Ric_{ij}

   INPUT:

   * "name" -- (default: "None") name given to the Ricci scalar; if
     none, it is set to "r(g)", where "g" is the metric's name

   * "latex_name" -- (default: "None") LaTeX symbol to denote the
     Ricci scalar; if none, it is set to "r(g)", where "g" is the
     metric's name

   OUTPUT:

   * the Ricci scalar r, as an instance of "DiffScalarField"

   EXAMPLES:

   Ricci scalar of the standard metric on the 2-sphere:

      sage: M = Manifold(2, 'S^2', start_index=1)
      sage: U = M.open_subset('U') # the complement of a meridian (domain of spheric
al coordinates)
      sage: c_spher.<th,ph> = U.chart(r'th:(0,pi):\theta ph:(0,2*pi):\phi')
      sage: a = var('a') # the sphere radius
      sage: g = U.metric('g')
      sage: g[1,1], g[2,2] = a^2, a^2*sin(th)^2
      sage: g.display() # standard metric on the 2-sphere of radius a:
      g = a^2 dth⊗dth + a^2*sin(th)^2 dph⊗dph
      sage: g.ricci_scalar()
      Scalar field r(g) on the Open subset U of the 2-dimensional
       differentiable manifold S^2
      sage: g.ricci_scalar().display() # The Ricci scalar is constant:
      r(g): U → ℝ
         (th, ph) ↦ 2/a^2
Init docstring: Initialize self.  See help(type(self)) for accurate signature.
File:           ~/sage/10.5/src/sage/manifolds/differentiable/metric.py
Type:           method
```

Using a double question mark leads directly to the **Python source code** (SageMath is **open source**, isn't it?):

In [281…
```
g.ricci_scalar??
```

```
Signature: g.ricci_scalar(name=None, latex_name=None)
Docstring:
   Return the metric's Ricci scalar.

   The Ricci scalar is the scalar field r defined from the Ricci
   tensor Ric and the metric tensor g by

      r = g^{ij} Ric_{ij}

   INPUT:

   * "name" -- (default: "None") name given to the Ricci scalar; if
     none, it is set to "r(g)", where "g" is the metric's name

   * "latex_name" -- (default: "None") LaTeX symbol to denote the
     Ricci scalar; if none, it is set to "r(g)", where "g" is the
     metric's name

   OUTPUT:

   * the Ricci scalar r, as an instance of "DiffScalarField"

   EXAMPLES:

   Ricci scalar of the standard metric on the 2-sphere:

      sage: M = Manifold(2, 'S^2', start_index=1)
      sage: U = M.open_subset('U') # the complement of a meridian (domain of spheric
al coordinates)
      sage: c_spher.<th,ph> = U.chart(r'th:(0,pi):\theta ph:(0,2*pi):\phi')
      sage: a = var('a') # the sphere radius
      sage: g = U.metric('g')
      sage: g[1,1], g[2,2] = a^2, a^2*sin(th)^2
      sage: g.display() # standard metric on the 2-sphere of radius a:
      g = a^2 dth⊗dth + a^2*sin(th)^2 dph⊗dph
      sage: g.ricci_scalar()
      Scalar field r(g) on the Open subset U of the 2-dimensional
       differentiable manifold S^2
      sage: g.ricci_scalar().display() # The Ricci scalar is constant:
      r(g): U → ℝ
         (th, ph) ↦ 2/a^2
Source:
    def ricci_scalar(self, name=None, latex_name=None):
        r"""
        Return the metric's Ricci scalar.

        The Ricci scalar is the scalar field `r` defined from the Ricci tensor
        `Ric` and the metric tensor `g` by

        .. MATH::

            r = g^{ij} Ric_{ij}

        INPUT:

        - ``name`` -- (default: ``None``) name given to the Ricci scalar;
          if none, it is set to "r(g)", where "g" is the metric's name
        - ``latex_name`` -- (default: ``None``) LaTeX symbol to denote the
          Ricci scalar; if none, it is set to "\\mathrm{r}(g)", where "g"
          is the metric's name

        OUTPUT:

        - the Ricci scalar `r`, as an instance of
          :class:`~sage.manifolds.differentiable.scalarfield.DiffScalarField`

        EXAMPLES:
```

```
        Ricci scalar of the standard metric on the 2-sphere::

            sage: M = Manifold(2, 'S^2', start_index=1)
            sage: U = M.open_subset('U') # the complement of a meridian (domain of s
pherical coordinates)
            sage: c_spher.<th,ph> = U.chart(r'th:(0,pi):\theta ph:(0,2*pi):\phi')
            sage: a = var('a') # the sphere radius
            sage: g = U.metric('g')
            sage: g[1,1], g[2,2] = a^2, a^2*sin(th)^2
            sage: g.display() # standard metric on the 2-sphere of radius a:
            g = a^2 dth⊗dth + a^2*sin(th)^2 dph⊗dph
            sage: g.ricci_scalar()
            Scalar field r(g) on the Open subset U of the 2-dimensional
             differentiable manifold S^2
            sage: g.ricci_scalar().display() # The Ricci scalar is constant:
            r(g): U → ℝ
               (th, ph) ↦ 2/a^2

        """
        if self._ricci_scalar is None:
            manif = self._ambient_domain
            ric = self.ricci()
            ig = self.inverse()
            frame = ig.common_basis(ric)
            cric = ric._components[frame]
            cig = ig._components[frame]
            rsum1 = 0
            for i in manif.irange():
                rsum1 += cig[[i,i]] * cric[[i,i]]
            rsum2 = 0
            for i in manif.irange():
                for j in manif.irange(start=i+1):
                    rsum2 += cig[[i,j]] * cric[[i,j]]
            self._ricci_scalar = rsum1 + 2*rsum2
            if name is None:
                self._ricci_scalar._name = "r(" + self._name + ")"
            else:
                self._ricci_scalar._name = name
            if latex_name is None:
                self._ricci_scalar._latex_name = r"\mathrm{r}\left(" + \
                                                 self._latex_name + r"\right)"
            else:
                self._ricci_scalar._latex_name = latex_name
        return self._ricci_scalar
File:      ~/sage/10.5/src/sage/manifolds/differentiable/metric.py
Type:      method
```

Going further

Have a look at the examples on SageManifolds page, especially the 2-dimensional sphere for usage on a non-parallelizable manifold (each scalar field has to be defined in at least two coordinate charts, the $C^\infty(\mathcal{M})$-module $\mathfrak{X}(\mathcal{M})$ is no longer free and each tensor field has to be defined in at least two vector frames).

It is also a good idea to take a look at the tutorial videos by Christian Bär.