Sphere $\mathbb{S}^2$

This notebook demonstrates some differential geometry capabilities of SageMath on the example of the 2-dimensional sphere. The corresponding tools have been developed within the SageManifolds project.

NB: a version of SageMath at least equal to 9.3 is required to run this notebook:

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

Out[1]:
```
'SageMath version 9.7, Release Date: 2022-09-19'
```

First we set up the notebook to display math formulas using LaTeX formatting:

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

$\mathbb{S}^2$ from the manifold catalog

The 2-sphere, with predefined charts and embedding in the Euclidean 3-space, can be obtained directly from SageMath's manifold catalog:

In [3]:
```
S2 = manifolds.Sphere(2)
S2
```

Out[3]: $\mathbb{S}^2$

In [4]:
```
print(S2)
```

```
2-sphere S^2 of radius 1 smoothly embedded in the Euclidean space E^3
```

In [5]:
```
S2.spherical_coordinates()
```

Out[5]: $(A, (\theta, \phi))$

In [6]:
```
S2.metric().display()
```

Out[6]: $g = \mathrm{d}\theta \otimes \mathrm{d}\theta + \sin\left(\theta\right)^2 \mathrm{d}\phi \otimes \mathrm{d}\phi$

$\mathbb{S}^2$ defined from scratch as a 2-dimensional smooth manifold

For the purpose of introducing generic smooth manifolds in SageMath, we shall not use the above predefined object. Instead we shall construct $\mathbb{S}^2$ from scratch, by invoking the generic function `Manifold`:

In [7]:
```
S2 = Manifold(2, 'S^2', latex_name=r'\mathbb{S}^2', start_index=1)
```

The first argument, `2`, is the dimension of the manifold, while the second argument is the symbol used to label the manifold.

The argument `start_index` sets the index range to be used on the manifold for labelling components w.r.t. a basis or a frame: `start_index=1` corresponds to $\{1, 2\}$; the default value is `start_index=0`

and yields $\{0, 1\}$.

The function `Manifold` has actually many options, which are displayed via the command `Manifold?`:

In [8]:
```
# Manifold?
```

By default `Manifold` constructs a smooth manifold over $\mathbb{R}$:

In [9]:
```
print(S2)
```

```
2-dimensional differentiable manifold S^2
```

In [10]:
```
S2
```

Out[10]: $\mathbb{S}^2$

$\mathbb{S}^2$ is in the category of smooth manifolds over $\mathbb{R}$:

In [11]:
```
S2.category()
```

Out[11]: $\mathrm{Smooth}_{\mathrm{R}}$

In [12]:
```
print(S2.category())
```

```
Category of smooth manifolds over Real Field with 53 bits of precision
```

At the moment, the real field $\mathbb{R}$ is modeled by 53-bit floating-point approximations, but this plays no role in the manifold implementation:

In [13]:
```
print(S2.base_field())
```

```
Real Field with 53 bits of precision
```

In [14]:
```
S2.base_field() is RR
```

Out[14]: True

Coordinate charts on $\mathbb{S}^2$

The function `Manifold` generates a manifold with no-predefined coordinate chart, so that the manifold (user) **atlas** is empty:

In [15]:
```
S2.atlas()
```

Out[15]: $[]$

Let us introduce some charts. At least two charts are necessary to cover the sphere. Let us choose the charts associated with the **stereographic projections** to the equatorial plane from the North pole and the South pole respectively. We first introduce the open subsets covered by these two charts:

$$U := \mathbb{S}^2 \setminus \{N\},$$

$$V := \mathbb{S}^2 \setminus \{S\},$$

where N is a point of $\mathbb{S}^2$, which we shall call the **North pole**, and S is the point of U of stereographic coordinates $(0, 0)$, which we call the **South pole**:

To find the method to create an open subset, we type `U = S2.<TAB>` to get the list of possible methods by autocompletion:

In [16]:
```
#U = S2.
```

In [17]:
```
U = S2.open_subset('U')
print(U)
```

```
Open subset U of the 2-dimensional differentiable manifold S^2
```

In [18]:
```
V = S2.open_subset('V')
print(V)
```

```
Open subset V of the 2-dimensional differentiable manifold S^2
```

As an open subset of a smooth manifold, U is itself a smooth manifold:

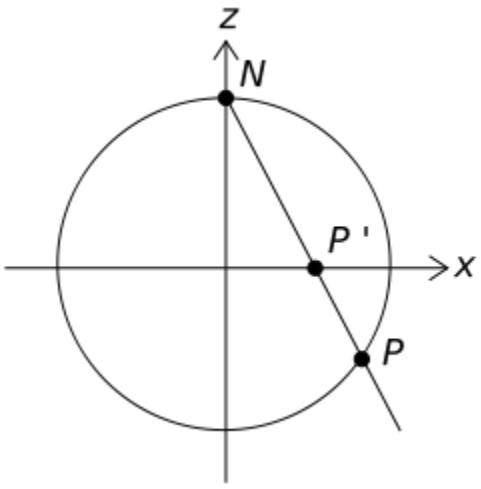
In [19]:
```
print(U.category())
```

```
Join of Category of subobjects of sets and Category of smooth manifolds over Real Field
with 53 bits of precision
```

We declare that $\mathbb{S}^2 = U \cup V$:

In [20]:
```
S2.declare_union(U, V)
```

The **stereographic chart** on U is constructed from the stereographic projection from the North pole onto the equatorial plane: in the Wikipedia figure below, the stereographic coordinates (x, y) of the point $P \in U$ are the Cartesian coordinates of the point P' in the equatorial plane.



We call this chart `stereoN` and construct it via the method `chart`:

In [21]:
```
stereoN.<x,y> = U.chart()
```

The syntax `.<x,y>` in the left-hand side implies that the Python names `x` and `y` are added to the global namespace, to access to the two coordinates of the chart as symbolic variables. This allows one to refer subsequently to the coordinates by these names. Besides, in the present case, the function `chart()` has no argument, which implies that the coordinate symbols will be `x` and `y` (i.e. exactly the characters appearing in the `<...>` operator) and that each coordinate range is $(-\infty, +\infty)$. As we will

see below, for other cases, an argument must be passed to `chart()` to specify each coordinate symbol and range, as well as some specific LaTeX symbol.

Note: the notation `.<x,y>` is not standard Python syntax, but a "SageMath enhanced" syntax. Actually the SageMath kernel preparses the cell entries before sending them to the Python interpreter. The outcome of the preparser is shown by the function `preparse`. In the present case:

In [22]:
```
print(preparse("stereoN.<x,y> = U.chart()"))
```

```
stereoN = U.chart(names=('x', 'y',)); (x, y,) = stereoN._first_ngens(2)
```

Another example of preparsing:

In [23]:
```
preparse("2^3")
```

Out[23]:
```
Integer(2)**Integer(3)
```

The chart created by the above command:

In [24]:
```
stereoN
```

Out[24]: $(U, (x, y))$

In [25]:
```
print(stereoN)
```

```
Chart (U, (x, y))
```

In [26]:
```
stereoN.coord_range()
```

Out[26]: $x: \ (-\infty, +\infty); \quad y: \ (-\infty, +\infty)$

The coordinates can be accessed individually, either by means of their indices in the chart (following the convention `start_index=1` set in the manifold's definition) or by their names as Python variables:

In [27]:
```
stereoN[1]
```

Out[27]: x

In [28]:
```
y is stereoN[2]
```

Out[28]: True

The coordinates are SageMath symbolic expressions:

In [29]:
```
type(y)
```

Out[29]:
```
<class 'sage.symbolic.expression.Expression'>
```

In [30]:
```
y.parent()
```

Out[30]: SR

Stereographic coordinates from the South Pole

We introduce on V the coordinates (x', y') corresponding to the stereographic projection from the South pole:

In [31]:
```
stereoS.<xp,yp> = V.chart("xp:x' yp:y'")
```

In this case, the string argument passed to `chart` stipulates that the text-only names of the coordinates are xp and yp (same as the Python variables names defined within the `<...>` operator in the left-hand side), while their LaTeX names are x' and y'.

In [32]:
```
stereoS
```

Out[32]: $(V, (x', y'))$

At this stage, the user's atlas on the manifold is made of two charts:

In [33]:
```
S2.atlas()
```

Out[33]: $\left[(U, (x, y)), (V, (x', y'))\right]$

To complete the construction of the manifold structure, we have to specify the **transition map** between the charts `stereoN` = $(U, (x, y))$ and `stereoS` = $(V, (x', y'))$; it is given by standard inversion formulas:

In [34]:
```
stereoN_to_S = stereoN.transition_map(stereoS,
                                      (x/(x^2+y^2), y/(x^2+y^2)),
                                      intersection_name='W',
                                      restrictions1= x^2+y^2!=0,
                                      restrictions2= xp^2+yp^2!=0)
stereoN_to_S.display()
```

Out[34]:
$$\begin{cases} x' &= \frac{x}{x^2+y^2} \\ y' &= \frac{y}{x^2+y^2} \end{cases}$$

In the above declaration, 'W' is the name given to the chart-overlap subset: $W := U \cap V$, the condition $x^2 + y^2 \neq 0$ defines W as a subset of U, and the condition $x'^2 + y'^2 \neq 0$ defines W as a subset of V.

The inverse coordinate transformation is computed by means of the method `inverse()`:

In [35]:
```
stereoS_to_N = stereoN_to_S.inverse()
stereoS_to_N.display()
```

Out[35]:
$$\begin{cases} x &= \frac{x'}{x'^2+y'^2} \\ y &= \frac{y'}{x'^2+y'^2} \end{cases}$$

In the present case, the situation is of course perfectly symmetric regarding the coordinates (x, y) and (x', y').

At this stage, the user's atlas has four charts:

In [36]:
```
S2.atlas()
```

Out[36]: $\left[(U, (x, y)), (V, (x', y')), (W, (x, y)), (W, (x', y'))\right]$

Let us store $W = U \cap V$ into a Python variable for future use:

In [37]:
```
W = U.intersection(V)
```

Similarly we store the charts $(W, (x, y))$ (the restriction of $(U, (x, y))$ to W) and $(W, (x', y'))$ (the restriction of $(V, (x', y'))$ to W) into Python variables:

In [38]:
```
stereoN_W = stereoN.restrict(W)
stereoN_W
```

Out[38]: $(W, (x, y))$

In [39]:
```
stereoN_W is S2.atlas()[2]
```

Out[39]: True

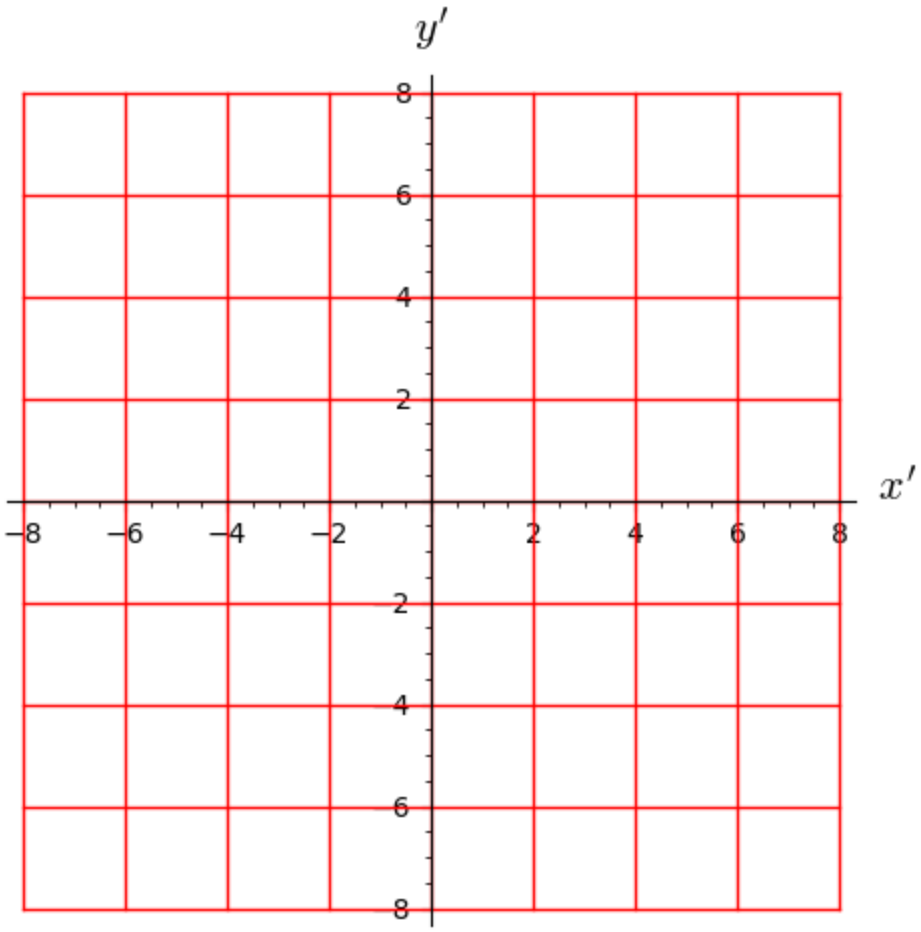
In [40]:
```
stereoS_W = stereoS.restrict(W)
stereoS_W
```

Out[40]: $(W, (x', y'))$

Coordinate charts are endoved with a method `plot`. For instance, we may plot the chart $(W, (x', y'))$ in terms of itself, as a grid:

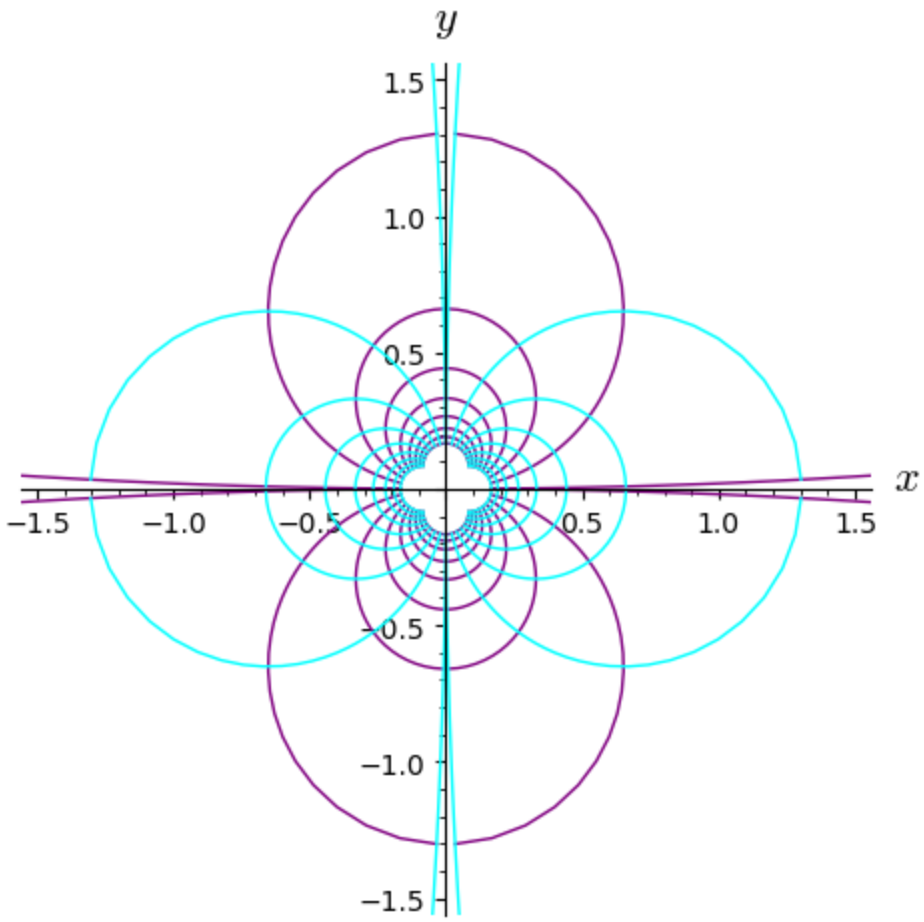
In [41]:
```
stereoS_W.plot()
```

Out[41]:



More interestingly, let us plot the stereographic chart (x', y') in terms of the stereographic chart (x, y) on the domain W where both systems overlap. We split the plot in four parts to avoid the singularity at $(x', y') = (0, 0)$ and ask for the coordinate lines along which x' (resp. y') varies to be colored in purple (resp. cyan):

In [42]:
```
graph = (stereoS_W.plot(stereoN, ranges={xp:[-6,-0.02], yp:[-6,-0.02]},
                        color={xp: 'purple', yp: 'cyan'})
         + stereoS_W.plot(stereoN, ranges={xp:[-6,-0.02], yp:[0.02,6]},
                        color={xp: 'purple', yp: 'cyan'})
         + stereoS_W.plot(stereoN, ranges={xp:[0.02,6], yp:[-6,-0.02]},
                        color={xp: 'purple', yp: 'cyan'})
         + stereoS_W.plot(stereoN, ranges={xp:[0.02,6], yp:[0.02,6]},
                        color={xp: 'purple', yp: 'cyan'}))
graph.show(xmin=-1.5, xmax=1.5, ymin=-1.5, ymax=1.5)
```



Spherical coordinates

The standard **spherical coordinates** (θ, ϕ) are defined on the open domain $A \subset W \subset \mathbb{S}^2$ that is the complement of the "origin meridian"; since the latter is the half-circle defined by $y = 0$ and $x \geq 0$, we declare:

In [43]:
```
A = W.open_subset('A', coord_def={stereoN_W: (y!=0, x<0),
                                  stereoS_W: (yp!=0, xp<0)})
print(A)
```

```
Open subset A of the 2-dimensional differentiable manifold S^2
```

The restriction of the stereographic chart from the North pole to A is

In [44]:
```
stereoN_A = stereoN_W.restrict(A)
stereoN_A
```

Out[44]: $(A, (x, y))$

We then declare the chart $(A, (\theta, \phi))$ by specifying the intervals $(0, \pi)$ and $(0, 2\pi)$ spanned by respectively θ and ϕ:

In [45]:
```
spher.<th,ph> = A.chart(r'th:(0,pi):\theta ph:(0,2*pi):\phi')
spher
```

Out[45]: $(A, (\theta, \phi))$

In [46]:
```
spher.coord_range()
```

Out[46]: $\theta: \ (0, \pi); \quad \phi: \ (0, 2\,\pi)$

The specification of the spherical coordinates is completed by providing the transition map with the stereographic chart $(A, (x, y))$:

In [47]:
```
spher_to_stereoN = spher.transition_map(stereoN_A,
                                        (sin(th)*cos(ph)/(1-cos(th)),
                                         sin(th)*sin(ph)/(1-cos(th))))
spher_to_stereoN.display()
```

Out[47]:
$$\begin{cases} x &= -\frac{\cos(\phi)\sin(\theta)}{\cos(\theta)-1} \\ y &= -\frac{\sin(\phi)\sin(\theta)}{\cos(\theta)-1} \end{cases}$$

We also provide the inverse transition map:

In [48]:
```
spher_to_stereoN.set_inverse(2*atan(1/sqrt(x^2+y^2)), atan2(-y,-x)+pi)
```

```
Check of the inverse coordinate transformation:
  th == 2*arctan(sqrt(-cos(th) + 1)/sqrt(cos(th) + 1))  **failed**
  ph == pi + arctan2(sin(ph)*sin(th)/(cos(th) - 1), cos(ph)*sin(th)/(cos(th) - 1))  **fa
iled**
  x == x  *passed*
  y == y  *passed*
NB: a failed report can reflect a mere lack of simplification.
```

The check is passed, modulo some lack of trigonometric simplifications in the first two lines.

In [49]:
```
spher_to_stereoN.inverse().display()
```

Out[49]:
$$\begin{cases} \theta &= 2\,\arctan\left(\frac{1}{\sqrt{x^2+y^2}}\right) \\ \phi &= \pi + \arctan(-y, -x) \end{cases}$$

The transition map $(A, (\theta, \phi)) \to (A, (x', y'))$ is obtained by combining the transition maps $(A, (\theta, \phi)) \to (A, (x, y))$ and $(A, (x, y)) \to (A, (x', y'))$ via the operator `*`:

In [50]:
```
stereoN_to_S_A = stereoN_to_S.restrict(A)
spher_to_stereoS = stereoN_to_S_A * spher_to_stereoN
spher_to_stereoS.display()
```

Out[50]:
$$\begin{cases} x' &= -\frac{\cos(\phi)\cos(\theta)-\cos(\phi)}{\sin(\theta)} \\ y' &= -\frac{\cos(\theta)\sin(\phi)-\sin(\phi)}{\sin(\theta)} \end{cases}$$

Similarly, the transition map $(A, (x', y')) \to (A, (\theta, \phi))$ is obtained by combining the transition maps $(A, (x', y')) \to (A, (x, y))$ and $(A, (x, y)) \to (A, (\theta, \phi))$:

In [51]:
```
stereoS_to_N_A = stereoN_to_S.inverse().restrict(A)
```

```
stereoS_to_spher = spher_to_stereoN.inverse() * stereoS_to_N_A
stereoS_to_spher.display()
```

Out[51]:

$$\begin{cases} \theta &= 2\arctan\left(\sqrt{x'^2+y'^2}\right) \\ \phi &= \pi - \arctan\left(\frac{y'}{x'^2+y'^2}, -\frac{x'}{x'^2+y'^2}\right) \end{cases}$$

The user atlas of $\mathbb{S}^2$ is now

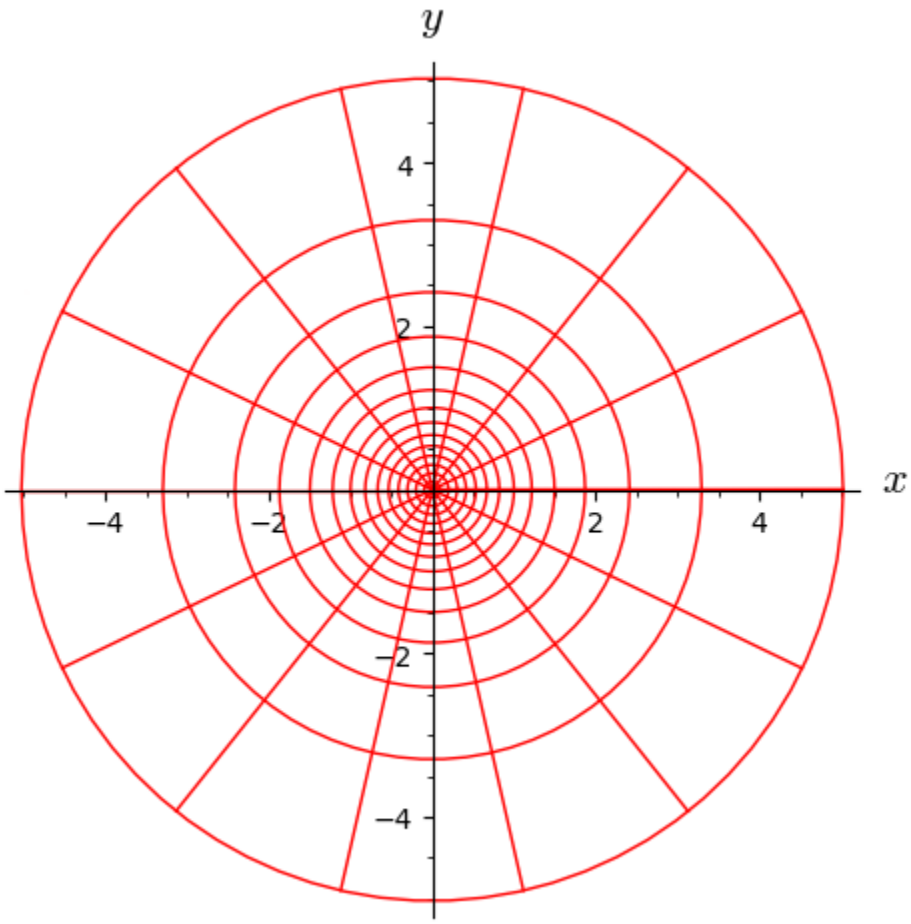
In [52]:
```
S2.atlas()
```

Out[52]: $\left[(U,(x,y)), (V,(x',y')), (W,(x,y)), (W,(x',y')), (A,(x,y)), (A,(x',y')), (A,(\theta,\phi))\right]$

Let us draw the grid of spherical coordinates (θ, ϕ) in terms of stereographic coordinates from the North pole (x, y):

In [53]:
```
spher.plot(stereoN, number_values=15, ranges={th: (pi/8,pi)})
```

Out[53]:



Points on $\mathbb{S}^2$

To create a point on $\mathbb{S}^2$, we use SageMath's ***parent / element*** syntax, i.e. the call operator `S2(...)` acting on the parent `S2`, with the point's coordinates in some chart as argument.

For instance, we declare the **North pole** (resp. the **South pole**) as the point of coordinates $(0, 0)$ in the chart $(V, (x', y'))$ (resp. in the chart $(U, (x, y))$):

In [54]:
```
N = S2((0,0), chart=stereoS, name='N')
print(N)
```

```
Point N on the 2-dimensional differentiable manifold S^2
```

In [55]:
```
S = S2((0,0), chart=stereoN, name='S')
print(S)
```

```
Point S on the 2-dimensional differentiable manifold S^2
```

In [56]:
```
N.parent()
```

Out[56]: $\mathbb{S}^2$

In [57]:
```
S.parent()
```

Out[57]: $\mathbb{S}^2$

We have of course

In [58]:
```
N in S2
```

Out[58]: True

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

Out[59]: False

In [60]:
```
N in V
```

Out[60]: True

In [61]:
```
N in A
```

Out[61]: False

Let us introduce some point p of stereographic coordinates $(x, y) = (1, 2)$:

In [62]:
```
p = S2((1,2), chart=stereoN, name='p')
```

p lies in the open subset A:

In [63]:
```
p in A
```

Out[63]: True

Charts acting on points

By definition, a chart maps points to pairs of real numbers (the point's coordinates):

In [64]:
```
stereoN(p)  # by definition of p
```

Out[64]: $(1, 2)$

In [65]:
```
stereoS(p)
```

Out[65]: $\left(\frac{1}{5}, \frac{2}{5}\right)$

In [66]:
```
spher(p)
```

Out[66]: $\left(2\,\arctan\left(\frac{1}{5}\,\sqrt{5}\right), \arctan(2)\right)$

In [67]:
```
stereoS(N)
```

Out[67]: $(0, 0)$

In [68]:
```
#stereoN(N)    ## returns an error
```

Maps between manifolds: the embedding of $\mathbb{S}^2$ into $\mathbb{R}^3$

Let us first declare $\mathbb{R}^3$ as the 3-dimensional Euclidean space, denoting the Cartesian coordinates by (X, Y, Z):

In [69]:
```
R3.<X,Y,Z> = EuclideanSpace(name='R^3', latex_name=r'\mathbb{R}^3', metric_name='h')
cartesian = R3.cartesian_coordinates()
cartesian
```

Out[69]: $\left(\mathbb{R}^3, (X, Y, Z)\right)$

As an Euclidean space, `R3` is considered by Sage as a smooth manifold:

In [70]:
```
print(R3.category())
```

```
Join of Category of smooth manifolds over Real Field with 53 bits of precision and Categ
ory of connected manifolds over Real Field with 53 bits of precision and Category of com
plete metric spaces
```

The embedding $\Phi : \mathbb{S}^2 \longrightarrow \mathbb{R}^3$ is then defined via the method `diff_map` by providing the standard formulas relating the stereographic coordinates to the ambient Cartesian ones when considering the **stereographic projection** from the point $(0, 0, 1)$ (North pole) or $(0, 0, -1)$ (South pole) to the equatorial plane $Z = 0$:

In [71]:
```
Phi = S2.diff_map(R3, {(stereoN, cartesian):
                       [2*x/(1+x^2+y^2), 2*y/(1+x^2+y^2),
                        (x^2+y^2-1)/(1+x^2+y^2)],
                       (stereoS, cartesian):
                       [2*xp/(1+xp^2+yp^2), 2*yp/(1+xp^2+yp^2),
                        (1-xp^2-yp^2)/(1+xp^2+yp^2)]},
                  name='Phi', latex_name=r'\Phi')
```

In [72]:
```
Phi.display()
```

Out[72]:
$$\begin{aligned}
\Phi : \quad \mathbb{S}^2 &\longrightarrow \mathbb{R}^3 \\
\text{on } U : \quad (x, y) &\longmapsto (X, Y, Z) = \left(\frac{2\,x}{x^2+y^2+1}, \frac{2\,y}{x^2+y^2+1}, \frac{x^2+y^2-1}{x^2+y^2+1}\right) \\
\text{on } V : \quad (x', y') &\longmapsto (X, Y, Z) = \left(\frac{2\,x'}{x'^2+y'^2+1}, \frac{2\,y'}{x'^2+y'^2+1}, -\frac{x'^2+y'^2-1}{x'^2+y'^2+1}\right)
\end{aligned}$$

In [73]:
```
Phi.parent()
```

Out[73]: $\mathrm{Hom}\left(\mathbb{S}^2, \mathbb{R}^3\right)$

In [74]:
```
print(Phi.parent())
```

```
Set of Morphisms from 2-dimensional differentiable manifold S^2 to Euclidean space R^3 i
n Category of smooth manifolds over Real Field with 53 bits of precision
```

In [75]:
```
Phi.parent() is Hom(S2, R3)
```

Out[75]: True

Φ maps points of $\mathbb{S}^2$ to points of $\mathbb{R}^3$:

In [76]:
```
N1 = Phi(N)
print(N1)
N1
```

```
Point Phi(N) on the Euclidean space R^3
```

Out[76]: $\Phi\left(N\right)$

In [77]:
```
cartesian(N1)
```

Out[77]: $(0, 0, 1)$

In [78]:
```
S1 = Phi(S)
print(S1)
S1
```

```
Point Phi(S) on the Euclidean space R^3
```

Out[78]: $\Phi\left(S\right)$

In [79]:
```
cartesian(S1)
```

Out[79]: $(0, 0, -1)$

In [80]:
```
p1 = Phi(p)
print(p1)
p1
```

```
Point Phi(p) on the Euclidean space R^3
```

Out[80]: $\Phi\left(p\right)$

In [81]:
```
cartesian(p1)
```

Out[81]: $\left(\frac{1}{3}, \frac{2}{3}, \frac{2}{3}\right)$

Φ has been defined in terms of the stereographic charts $(U, (x, y))$ and $(V, (x', y'))$, but we may ask its expression in terms of spherical coordinates. This triggers a computation involving the transition map $(A, (x, y)) \to (A, (\theta, \phi))$:

In [82]:
```
Phi.display(stereoN_A, cartesian)
```

Out[82]:
$$\begin{aligned}\Phi : \quad & \mathbb{S}^2 & \longrightarrow \quad & \mathbb{R}^3 \\ \text{on } A : \quad & (x, y) & \longmapsto \quad & (X, Y, Z) = \left(\frac{2\,x}{x^2+y^2+1}, \frac{2\,y}{x^2+y^2+1}, \frac{x^2+y^2-1}{x^2+y^2+1}\right)\end{aligned}$$

In [83]:
```
Phi.display(spher, cartesian)
```

Out[83]:

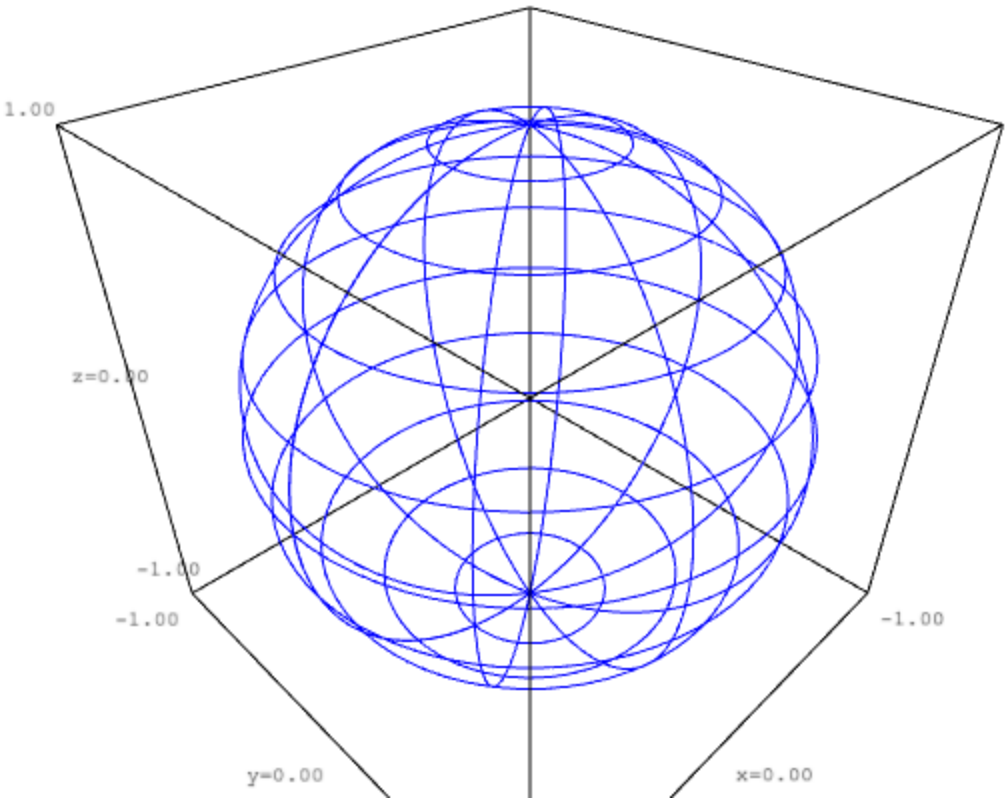
$$\begin{array}{lccc} \Phi: & \mathbb{S}^2 & \longrightarrow & \mathbb{R}^3 \\ \text{on } A: & (\theta, \phi) & \longmapsto & (X, Y, Z) = (\cos(\phi)\sin(\theta), \sin(\phi)\sin(\theta), \cos(\theta)) \end{array}$$

Φ (θ, ϕ)

(X, Y, Z) $\mathbb{R}^3$

In [84]:
```
graph_spher = spher.plot(chart=cartesian, mapping=Phi, number_values=11,
                         color='blue', label_axes=False)
graph_spher
```

Out[84]:



Φ

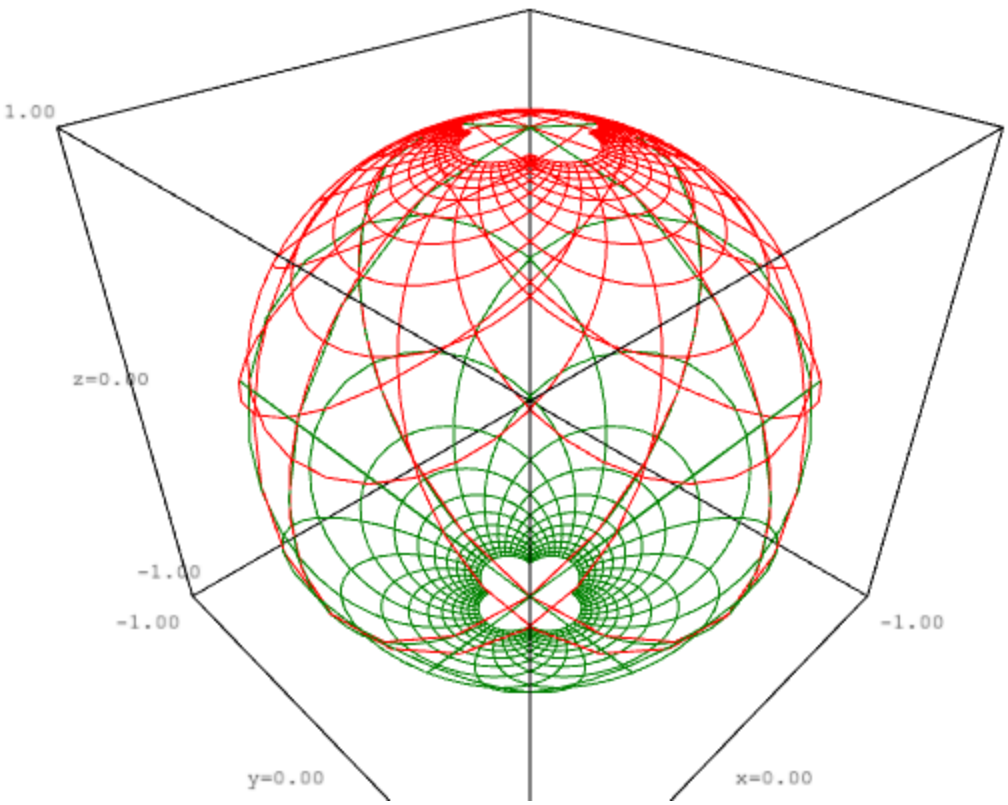
$\mathbb{R}^3$

In [85]:
```
graph = stereoN.plot(chart=cartesian, mapping=Phi, number_values=25,
                     label_axes=False)
graph
```

Out[85]:

In [86]:
```
graph += stereoS.plot(chart=cartesian, mapping=Phi, number_values=25,
                      color='green', label_axes=False)
graph
```

Out[86]:

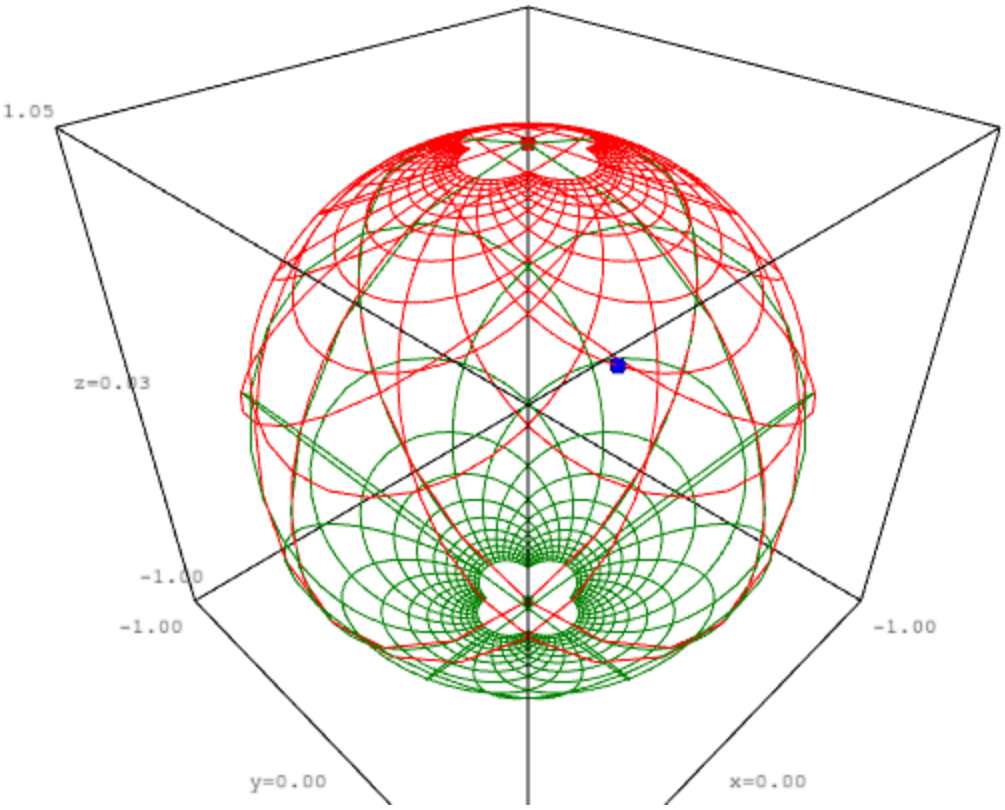


$N \quad S \quad p$

In [87]:
```
graph += N.plot(chart=cartesian, mapping=Phi, color='red',
                label_offset=0.05)
graph += S.plot(chart=cartesian, mapping=Phi, color='green',
```

```
                label_offset=0.05)
graph += p.plot(chart=cartesian, mapping=Phi, color='blue',
                label_offset=0.05)
graph
```

Out[87]:



$\mathbb{S}^2$ p

In [88]:
```
Tp = S2.tangent_space(p)
print(Tp)
Tp
```

Out[88]: $T_p\,\mathbb{S}^2$

$T_p\mathbb{S}^2$ $\mathbb{R}$

In [89]:
```
print(Tp.category())
```

In [90]:
```
dim(Tp)
```

Out[90]: 2

In [91]:
```
dim(Tp) == dim(S2)
```

Out[91]: True

p p $T\mathbb{S}^2$

In [92]:
```
Tp is S2.tangent_bundle().fiber(p)
```

Out[92]: True

The vector space $T_p\mathbb{S}^2$ is endowed with bases inherited from the coordinate frames defined around p:

In [93]:
```
Tp.bases()
```

Out[93]: $\left[\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right), \left(\frac{\partial}{\partial x'}, \frac{\partial}{\partial y'}\right), \left(\frac{\partial}{\partial \theta}, \frac{\partial}{\partial \phi}\right)\right]$

On the contrary, since $(V, (x', y'))$ is the only chart defined so far around the point N, we have a unique predefined basis in $T_N\mathbb{S}^2$:

In [94]:
```
T_N = S2.tangent_space(N)
T_N.bases()
```

Out[94]: $\left[\left(\frac{\partial}{\partial x'}, \frac{\partial}{\partial y'}\right)\right]$

To shorten some writings, there is the concept of default basis:

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

Out[95]: $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$

An element of $T_p\mathbb{S}^2$ is constructed via SageMath's *parent/element* syntax, i.e. via the call method of the parent:

In [96]:
```
v = Tp((-2, 3), name='v')
print(v)
```

```
Tangent vector v at Point p on the 2-dimensional differentiable manifold S^2
```

Equivalently, one can use the method `tangent_vector` of manifolds:

In [97]:
```
v == S2.tangent_vector(p, -2, 3, name='v')
```

Out[97]: True

One has of course:

In [98]:
```
v in Tp
```

Out[98]: True

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

Out[99]: $T_p\,\mathbb{S}^2$

The vector v expanded in the default basis of $T_p\mathbb{S}^2$:

In [100…
```
v.display()
```

Out[100]: $v = -2\frac{\partial}{\partial x} + 3\frac{\partial}{\partial y}$

In [101…
```
v.display(Tp.bases()[1])
```

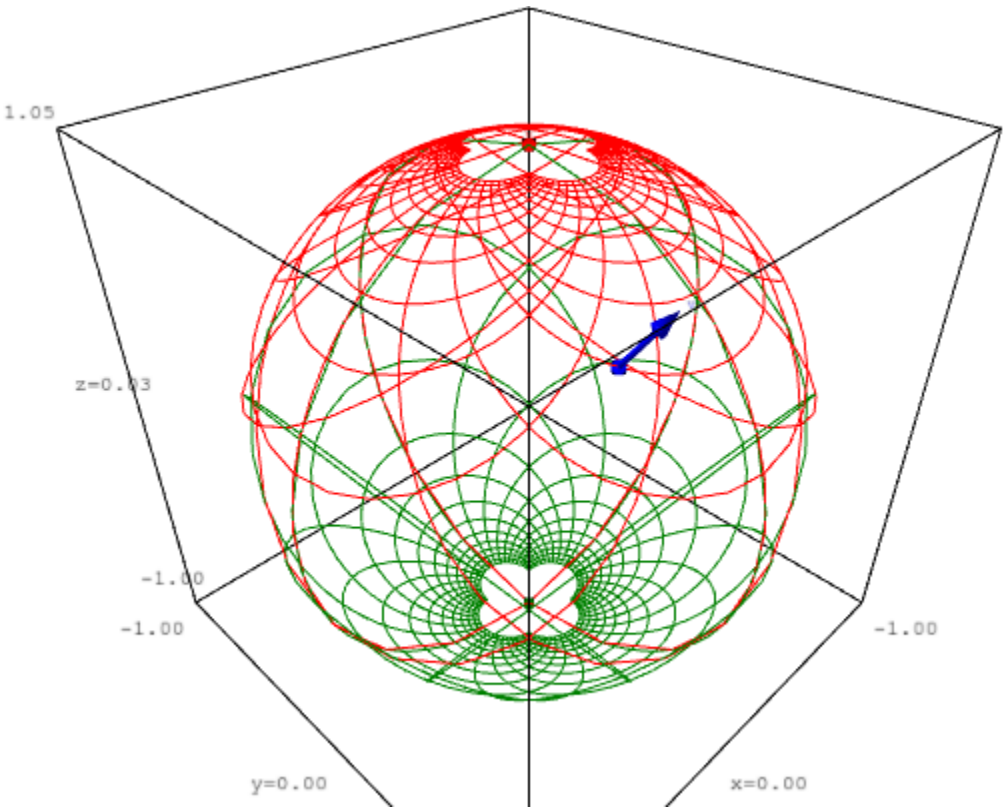
Out[101]:
$$v = -\frac{18}{25}\frac{\partial}{\partial x'} - \frac{1}{25}\frac{\partial}{\partial y'}$$

In [102…
```
v.display(Tp.bases()[2])
```

Out[102]:
$$v = -\frac{4}{15}\sqrt{5}\frac{\partial}{\partial \theta} + \frac{7}{5}\frac{\partial}{\partial \phi}$$

In [103…
```
graph += v.plot(chart=cartesian, mapping=Phi, scale=0.2, width=0.5)
graph
```

Out[103]:



Φ p

In [104…
```
dPhi_p = Phi.differential(p)
print(dPhi_p)
dPhi_p
```

Out[104]:
$\mathrm{d}\Phi_p$

In [105…
```
dPhi_p.domain()
```

Out[105]:
$T_p\,\mathbb{S}^2$

In [106…
```
dPhi_p.codomain()
```

Out[106]: $T_{\Phi(p)}\,\mathbb{R}^3$

In [107…
```
dPhi_p.parent()
```

Out[107]: $\mathrm{Hom}\left(T_p\,\mathbb{S}^2, T_{\Phi(p)}\,\mathbb{R}^3\right)$

The image by $\mathrm{d}\Phi_p$ of the vector $v \in T_p\mathbb{S}^2$ introduced above is

In [108…
```
dPhi_p(v)
```

Out[108]: $\mathrm{d}\Phi_p\,(v)$

In [109…
```
print(dPhi_p(v))
```

```
Vector dPhi_p(v) at Point Phi(p) on the Euclidean space R^3
```

In [110…
```
dPhi_p(v) in R3.tangent_space(Phi(p))
```

Out[110]: True

In [111…
```
dPhi_p(v).display()
```

Out[111]: $\mathrm{d}\Phi_p\,(v) = -\frac{10}{9}\,e_X + \frac{1}{9}\,e_Y + \frac{4}{9}\,e_Z$

Algebra of scalar fields

The set $C^\infty(\mathbb{S}^2)$ of all smooth functions $\mathbb{S}^2 \to \mathbb{R}$ has naturally the structure of a commutative algebra over $\mathbb{R}$. $C^\infty(\mathbb{S}^2)$ is therefore returned by the method `scalar_field_algebra()` of manifolds:

In [112…
```
CS = S2.scalar_field_algebra()
CS
```

Out[112]: $C^\infty\left(\mathbb{S}^2\right)$

Since the algebra internal product is the pointwise multiplication, it is clearly commutative, so that $C^\infty(\mathbb{S}^2)$ belongs to Sage's category of commutative algebras:

In [113…
```
print(CS.category())
```

```
Join of Category of commutative algebras over Symbolic Ring and Category of homsets of t
opological spaces
```

The base ring of the algebra $C^\infty(\mathbb{S}^2)$ is the field $\mathbb{R}$, which is represented here by Sage's Symbolic Ring (SR):

In [114…
```
CS.base_ring()
```

Out[114]: SR

Elements of $C^\infty(\mathbb{S}^2)$ are of course (smooth) scalar fields:

In [115…
```
print(CS.an_element())
```

```
Scalar field on the 2-dimensional differentiable manifold S^2
```

This example element is the constant scalar field that takes the value 2:

In [116…
```
CS.an_element().display()
```

Out[116]:

$$\begin{array}{llcl} & \mathbb{S}^2 & \longrightarrow & \mathbb{R} \\ \text{on } U: & (x, y) & \longmapsto & 2 \\ \text{on } V: & (x', y') & \longmapsto & 2 \\ \text{on } A: & (\theta, \phi) & \longmapsto & 2 \end{array}$$

A specific element is the zero one:

In [117…
```
f = CS.zero()
print(f)
```

```
Scalar field zero on the 2-dimensional differentiable manifold S^2
```

Scalar fields map points of $\mathbb{S}^2$ to real numbers:

In [118…
```
f(N), f(S), f(p)
```

Out[118]: $(0, 0, 0)$

In [119…
```
f.display()
```

Out[119]:

$$\begin{array}{llcl} 0: & \mathbb{S}^2 & \longrightarrow & \mathbb{R} \\ \text{on } U: & (x, y) & \longmapsto & 0 \\ \text{on } V: & (x', y') & \longmapsto & 0 \\ \text{on } A: & (\theta, \phi) & \longmapsto & 0 \end{array}$$

Another specific element is the algebra unit element, i.e. the constant scalar field 1:

In [120…
```
f = CS.one()
print(f)
```

```
Scalar field 1 on the 2-dimensional differentiable manifold S^2
```

In [121…
```
f(N), f(S), f(p)
```

Out[121]: $(1, 1, 1)$

A generic scalar field is defined by its coordinate expression in some chart(s); for instance:

In [122…
```
f = S2.scalar_field({stereoN: 1/(1+x^2+y^2)}, name='f')
f.display()
```

Out[122]:

$$\begin{array}{llcl} f: & \mathbb{S}^2 & \longrightarrow & \mathbb{R} \\ \text{on } U: & (x, y) & \longmapsto & \frac{1}{x^2+y^2+1} \\ \text{on } W: & (x', y') & \longmapsto & \frac{x'^2+y'^2}{x'^2+y'^2+1} \\ \text{on } A: & (\theta, \phi) & \longmapsto & -\frac{1}{2}\cos(\theta) + \frac{1}{2} \end{array}$$

We see that Sage has used the transition map between the two stereographic charts on W to express f in terms of the coordinates (x', y') on W. Let us this expression to extend f to the whole of V:

In [123…
```
f.add_expr_by_continuation(stereoS, W)
```

Then f is well defined in all $\mathbb{S}^2 = U \cup V$:

In [124…
```
f.display()
```

Out[124]:
$$\begin{array}{lccc} f: & \mathbb{S}^2 & \longrightarrow & \mathbb{R} \\ \text{on } U: & (x, y) & \longmapsto & \frac{1}{x^2+y^2+1} \\ \text{on } V: & (x', y') & \longmapsto & \frac{x'^2+y'^2}{x'^2+y'^2+1} \\ \text{on } A: & (\theta, \phi) & \longmapsto & -\frac{1}{2}\cos(\theta) + \frac{1}{2} \end{array}$$

In [125…
```
f(N)
```

Out[125]:
$$0$$

In [126…
```
f.parent()
```

Out[126]:
$$C^\infty\left(\mathbb{S}^2\right)$$

Scalar fields map the manifold's points to real numbers:

In [127…
```
f(N)
```

Out[127]:
$$0$$

In [128…
```
f(S)
```

Out[128]:
$$1$$

In [129…
```
f(p)
```

Out[129]:
$$\frac{1}{6}$$

We may define the restrictions of f to the open subsets U and V:

In [130…
```
fU = f.restrict(U)
fU.display()
```

Out[130]:
$$\begin{array}{lccc} f: & U & \longrightarrow & \mathbb{R} \\ & (x, y) & \longmapsto & \frac{1}{x^2+y^2+1} \\ \text{on } W: & (x', y') & \longmapsto & \frac{x'^2+y'^2}{x'^2+y'^2+1} \\ \text{on } A: & (\theta, \phi) & \longmapsto & -\frac{1}{2}\cos(\theta) + \frac{1}{2} \end{array}$$

In [131…
```
fV = f.restrict(V)
fV.display()
```

Out[131]:

$$\begin{array}{llcl} f: & V & \longrightarrow & \mathbb{R} \\ & (x', y') & \longmapsto & \frac{x'^2+y'^2}{x'^2+y'^2+1} \\ \text{on } W: & (x, y) & \longmapsto & \frac{1}{x^2+y^2+1} \\ \text{on } A: & (\theta, \phi) & \longmapsto & -\frac{1}{2}\cos(\theta) + \frac{1}{2} \end{array}$$

In [132…
```
fU(p), fU(S)
```

Out[132]:
$$\left(\frac{1}{6}, 1\right)$$

In [133…
```
fU.parent()
```

Out[133]:
$$C^\infty(U)$$

In [134…
```
fV.parent()
```

Out[134]:
$$C^\infty(V)$$

In [135…
```
CU = U.scalar_field_algebra()
fU.parent() is CU
```

Out[135]:
$$\text{True}$$

A scalar field on $\mathbb{S}^2$ can be coerced to a scalar field on U, the coercion being simply the restriction:

In [136…
```
CU.has_coerce_map_from(CS)
```

Out[136]:
$$\text{True}$$

In [137…
```
fU == CU(f)
```

Out[137]:
$$\text{True}$$

The arithmetic of scalar fields (operations in the algebra $C^\infty(\mathbb{S}^2)$):

In [138…
```
g = f*f - 2*f
g.set_name('g')
g.display()
```

Out[138]:
$$\begin{array}{llcl} g: & \mathbb{S}^2 & \longrightarrow & \mathbb{R} \\ \text{on } U: & (x, y) & \longmapsto & -\frac{2\,x^2+2\,y^2+1}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1} \\ \text{on } V: & (x', y') & \longmapsto & -\frac{x'^4+y'^4+2\left(x'^2+1\right)y'^2+2\,x'^2}{x'^4+y'^4+2\,(x'^2+1)y'^2+2\,x'^2+1} \\ \text{on } A: & (\theta, \phi) & \longmapsto & \frac{1}{4}\cos(\theta)^2 + \frac{1}{2}\cos(\theta) - \frac{3}{4} \end{array}$$

Vector fields

The set $\mathfrak{X}(\mathbb{S}^2)$ of all smooth vector fields on $\mathbb{S}^2$ is a module over the algebra $C^\infty(\mathbb{S}^2)$. It is obtained by the method `vector_field_module()`:

In [139…
```
XS = S2.vector_field_module()
XS
```

Out[139]: $\mathfrak{X}\left(\mathbb{S}^2\right)$

In [140…
```
print(XS)
```

```
Module X(S^2) of vector fields on the 2-dimensional differentiable manifold S^2
```

In [141…
```
XS.base_ring()
```

Out[141]: $C^\infty\left(\mathbb{S}^2\right)$

In [142…
```
XS.category()
```

Out[142]: $\mathrm{Modules}_{C^\infty\left(\mathbb{S}^2\right)}$

$\mathfrak{X}(\mathbb{S}^2)$ is not a free module:

In [143…
```
isinstance(XS, FiniteRankFreeModule)
```

Out[143]: False

because $\mathbb{S}^2$ is not a parallelizable manifold:

In [144…
```
S2.is_manifestly_parallelizable()
```

Out[144]: False

On the contrary, the set $\mathfrak{X}(U)$ of smooth vector fields on U is a free module:

In [145…
```
XU = U.vector_field_module()
isinstance(XU, FiniteRankFreeModule)
```

Out[145]: True

because U is parallelizable:

In [146…
```
U.is_manifestly_parallelizable()
```

Out[146]: True

Due to the introduction of the stereographic coordinates (x, y) on U, a basis has already been defined on the free module $\mathfrak{X}(U)$, namely the coordinate basis $(\partial/\partial x, \partial/\partial y)$:

In [147…
```
XU.print_bases()
```

```
Bases defined on the Free module X(U) of vector fields on the Open subset U of the 2-dim
ensional differentiable manifold S^2:
 - (U, (∂/∂x,∂/∂y)) (default basis)
```

In [148…
```
eU = XU.default_basis()
eU
```

Out[148]: $\left(U, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)\right)$

Similarly

In [149…
```
XV = V.vector_field_module()
eV = XV.default_basis()
eV
```

Out[149]: $\left(V, \left(\frac{\partial}{\partial x'}, \frac{\partial}{\partial y'}\right)\right)$

From the point of view of the open set U, `eU` is also the default vector frame:

In [150…
```
eU is U.default_frame()
```

Out[150]: True

Similarly:

In [151…
```
eV is V.default_frame()
```

Out[151]: True

`eU` is also the default vector frame on $\mathbb{S}^2$ (although not defined on the whole $\mathbb{S}^2$), for it is the first vector frame defined on an open subset of $\mathbb{S}^2$:

In [152…
```
eU is S2.default_frame()
```

Out[152]: True

In [153…
```
S2.frames()
```

Out[153]: $\left[\left(U, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)\right), \left(V, \left(\frac{\partial}{\partial x'}, \frac{\partial}{\partial y'}\right)\right), \left(W, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)\right), \left(W, \left(\frac{\partial}{\partial x'}, \frac{\partial}{\partial y'}\right)\right), \left(A, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)\right), \left(A, \left(\frac{\partial}{\partial x'}, \frac{\partial}{\partial y'}\right)\right), \left(A, \left(\frac{\partial}{\partial \theta}, \frac{\partial}{\partial \phi}\right)\right)\right]$

Let us introduce a vector field on $\mathbb{S}^2$ by providing its components in the frame `eU`:

In [154…
```
v = S2.vector_field(1, -2, frame=eU, name='v')
v.display(eU)
```

Out[154]: $v = \frac{\partial}{\partial x} - 2\frac{\partial}{\partial y}$

In [155…
```
v.parent()
```

Out[155]: $\mathfrak{X}\left(\mathbb{S}^2\right)$

On W, we can express v in terms of the (x', y') coordinates:

In [156…
```
v.restrict(W).display(stereoS.restrict(W).frame(), stereoS.restrict(W))
```

Out[156]: $v = \left(-x'^2 + 4\,x'y' + y'^2\right)\frac{\partial}{\partial x'} + \left(-2\,x'^2 - 2\,x'y' + 2\,y'^2\right)\frac{\partial}{\partial y'}$

We extend the definition of v to V thanks to the above expression:

In [157…]
```
v.add_comp_by_continuation(eV, W, chart=stereoS)
v.display(eV)
```

Out[157]:
$$v = \left(-x'^2 + 4\,x'y' + y'^2\right) \frac{\partial}{\partial x'} + \left(-2\,x'^2 - 2\,x'y' + 2\,y'^2\right) \frac{\partial}{\partial y'}$$

At this stage, the vector field v is defined on the whole manifold $\mathbb{S}^2$: it has expressions in each of the two frames `eU` and `eV`, which cover $\mathbb{S}^2$.

According to the hairy ball theorem, v has to vanish somewhere. This occurs at the North pole:

In [158…]
```
vN = v.at(N)
print(vN)
```

```
Tangent vector v at Point N on the 2-dimensional differentiable manifold S^2
```

In [159…]
```
vN.display()
```

Out[159]:
$$v = 0$$

$v|_N$ is the zero vector of the tangent vector space $T_N\mathbb{S}^2$:

In [160…]
```
vN.parent()
```

Out[160]:
$$T_N\,\mathbb{S}^2$$

In [161…]
```
vN.parent() is S2.tangent_space(N)
```

Out[161]:
$$\text{True}$$

In [162…]
```
vN == S2.tangent_space(N).zero()
```

Out[162]:
$$\text{True}$$

On the contrary, v is non-zero at the South pole:

In [163…]
```
vS = v.at(S)
print(v)
```

```
Vector field v on the 2-dimensional differentiable manifold S^2
```

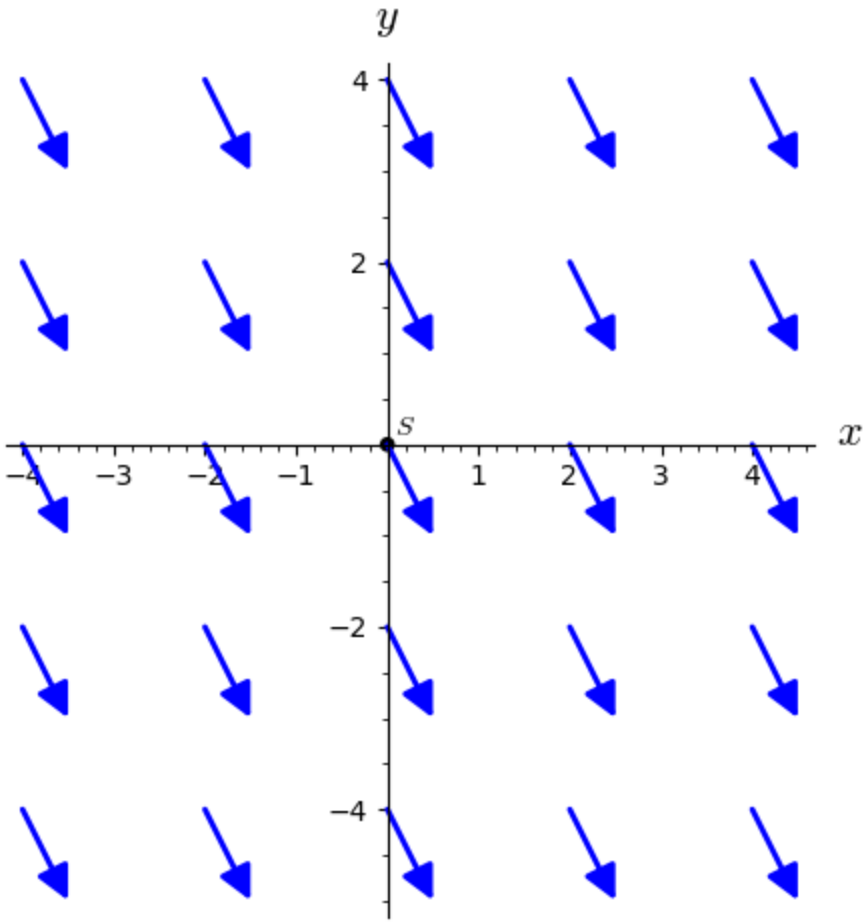
In [164…]
```
vS.display()
```

Out[164]:
$$v = \frac{\partial}{\partial x} - 2\frac{\partial}{\partial y}$$

Let us plot the vector field v is terms of the stereographic chart $(U, (x, y))$, with the South pole S superposed:

In [165…]
```
v.plot(chart=stereoN, chart_domain=stereoN, max_range=4,
       number_values=5, scale=0.5, aspect_ratio=1) \
+ S.plot(stereoN, size=30, label_offset=0.2)
```

Out[165]:



The vector field appears homogeneous because its components w.r.t. the frame $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$ are constant:

In [166…]
```
v.display(stereoN.frame())
```

Out[166]:
$$v = \frac{\partial}{\partial x} - 2\frac{\partial}{\partial y}$$

On the contrary, once drawn in terms of the stereographic chart $(V, (x', y'))$, v does no longer appears homogeneous:

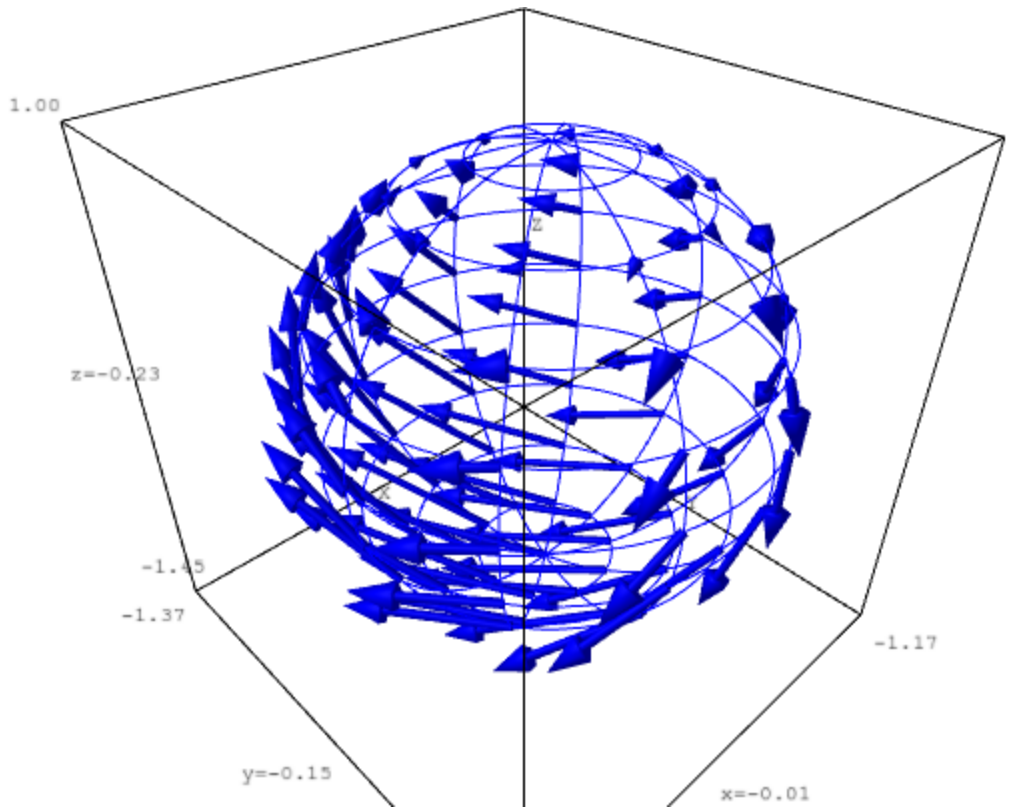
In [167…]
```
v.plot(chart=stereoS, chart_domain=stereoS, max_range=4, scale=0.02,
       aspect_ratio=1) \
+ N.plot(chart=stereoS, size=30, label_offset=0.2)
```

Out[167]:

v Φ

In [168…
```
graph_v = v.plot(chart=cartesian, mapping=Phi, chart_domain=spher,
                 number_values=11, scale=0.2)
graph_spher + graph_v
```

Out[168]:



$\frac{\partial}{\partial x}$

In [169…
```
stereoN.frame()
```

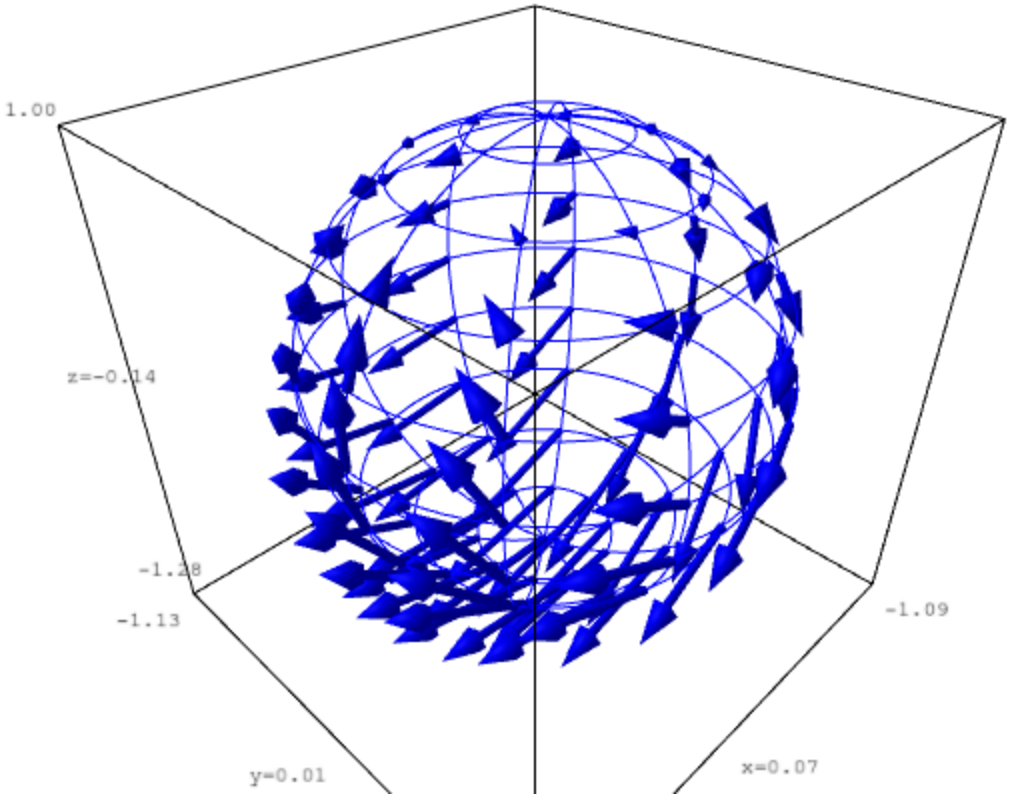
Out[169]:
$$\left(U, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)\right)$$

In [170…
```
ex = stereoN.frame()[1]
ex
```

Out[170]:
$$\frac{\partial}{\partial x}$$

In [171…
```
graph_ex = ex.plot(chart=cartesian, mapping=Phi, chart_domain=spher,
                   number_values=11, scale=0.4, width=1,
                   label_axes=False)
graph_spher + graph_ex
```

Out[171]:



$$\frac{\partial}{\partial y}$$

In [172…
```
ey = stereoN.frame()[2]
ey
```

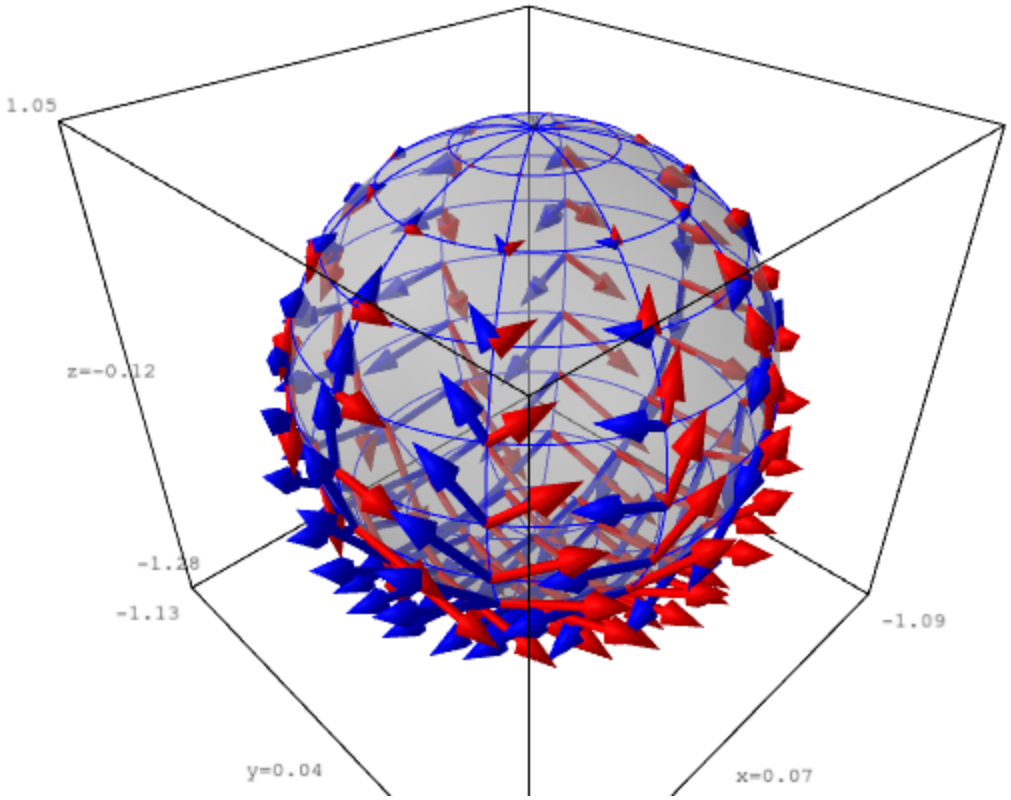
Out[172]:
$$\frac{\partial}{\partial y}$$

In [173…
```
graph_ey = ey.plot(chart=cartesian, mapping=Phi, chart_domain=spher,
                   number_values=11, scale=0.4, width=1, color='red',
                   label_axes=False)
graph_spher + graph_ey
```

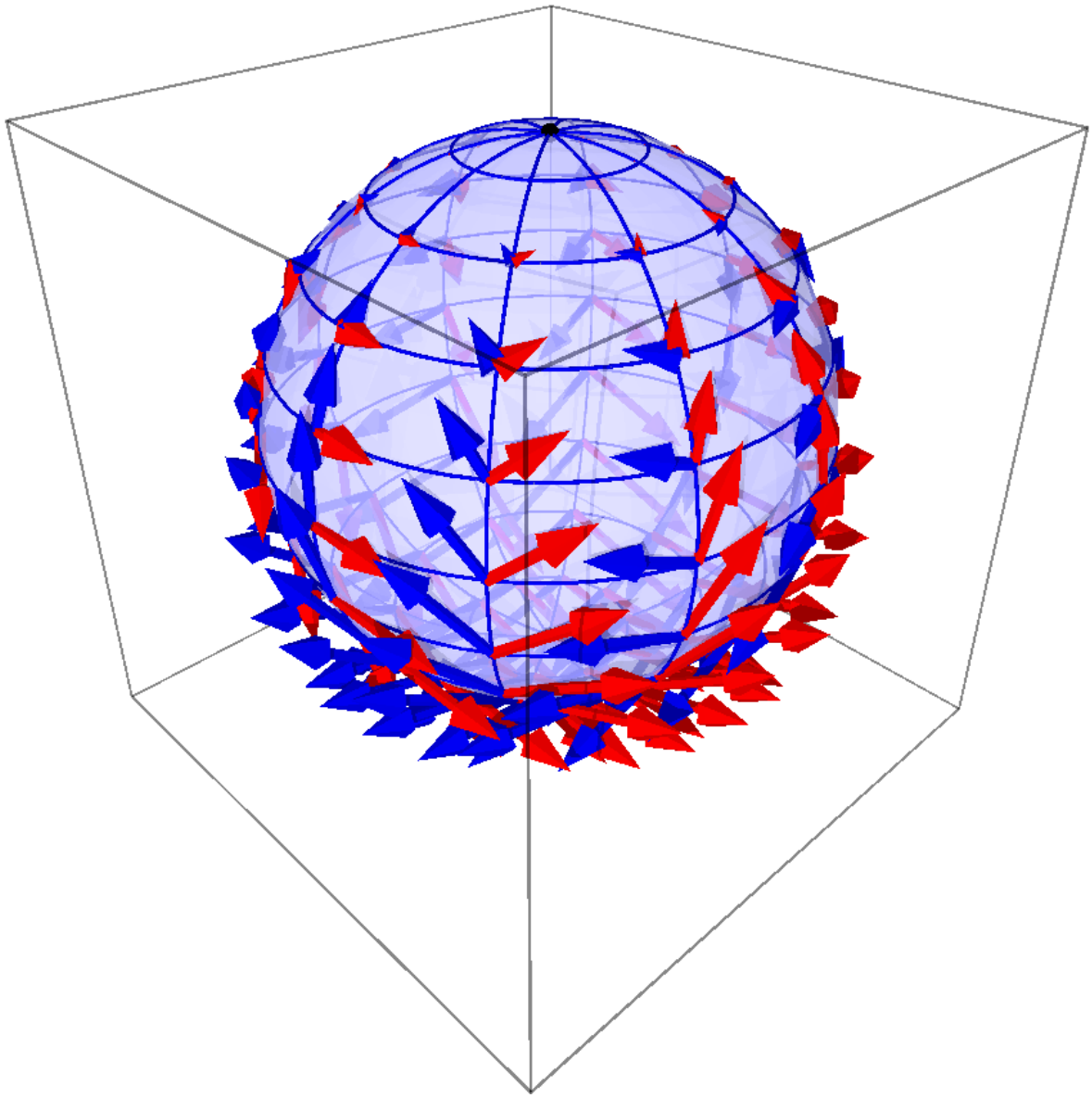
Out[173]:

In [174…
```
graph_frame = graph_spher + graph_ex + graph_ey \
              + N.plot(cartesian, mapping=Phi, label_offset=0.05, size=5) \
              + S.plot(cartesian, mapping=Phi, label_offset=0.05, size=5)
graph_frame + sphere(color='lightgrey', opacity=0.4)
```

Out[174]:



In [175…
```
show(graph_frame + sphere(opacity=0.5), viewer='tachyon', figsize=10)
```



Vector fields acting on scalar fields

v and f are both fields defined on the whole sphere (respectively a vector field and a scalar field). By the very definition of a vector field, v acts on f:

In [176…
```
vf = v(f)
print(vf)
vf.display()
```

```
Scalar field v(f) on the 2-dimensional differentiable manifold S^2
```

Out[176]:

$$
\begin{aligned}
v(f):\quad & \mathbb{S}^2 && \longrightarrow && \mathbb{R} \\
\text{on } U:\quad & (x, y) && \longmapsto && -\frac{2\,(x-2\,y)}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1} \\
\text{on } V:\quad & (x', y') && \longmapsto && -\frac{2\left(x'^3-2\,x'^2y'+x'y'^2-2\,y'^3\right)}{x'^4+y'^4+2\,(x'^2+1)y'^2+2\,x'^2+1} \\
\text{on } A:\quad & (\theta, \phi) && \longmapsto && \frac{1}{2}\left((\cos(\phi)-2\,\sin(\phi))\cos(\theta)-\cos(\phi)+2\,\sin(\phi)\right)\sin(\theta)
\end{aligned}
$$

Values of $v(f)$ at the North pole, at the South pole and at point p:

In [177…
```
vf(N)
```

Out[177]: 0

In [178…
```
vf(S)
```

Out[178]: 0

In [179…
```
vf(p)
```

Out[179]: $\frac{1}{6}$

1-forms

A 1-form on $\mathbb{S}^2$ is a field of linear forms on the tangent spaces. For instance it can be the differential of a scalar field:

In [180…
```
f.display()
```

Out[180]:
$$\begin{array}{lccc} f: & \mathbb{S}^2 & \longrightarrow & \mathbb{R} \\ \text{on } U: & (x, y) & \longmapsto & \frac{1}{x^2+y^2+1} \\ \text{on } V: & (x', y') & \longmapsto & \frac{x'^2+y'^2}{x'^2+y'^2+1} \\ \text{on } A: & (\theta, \phi) & \longmapsto & -\frac{1}{2}\cos(\theta) + \frac{1}{2} \end{array}$$

In [181…
```
df = diff(f)
print(df)
```

```
1-form df on the 2-dimensional differentiable manifold S^2
```

In [182…
```
df.display()  # display w.r.t. the default frame
```

Out[182]:
$$\mathrm{d}f = \left(-\frac{2\,x}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1}\right)\mathrm{d}x + \left(-\frac{2\,y}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1}\right)$$

In [183…
```
df.display(eV)
```

Out[183]:
$$\mathrm{d}f = \left(\frac{2\,x'}{x'^4+y'^4+2\,(x'^2+1)y'^2+2\,x'^2+1}\right)\mathrm{d}x' + \left(\frac{2\,y'}{x'^4+y'^4+2\,(x'^2+1)y'^2+2\,x'^2+1}\right)\mathrm{d}y'$$

In [184…
```
df.display(spher.frame())
```

Out[184]:
$$\mathrm{d}f = \left(\frac{\sqrt{x^2+y^2}}{x^2+y^2+1}\right)\mathrm{d}\theta$$

In [185…
```
df.display(spher.frame(), spher)  # asking for the components to be shown in the spheric
```

Out[185]: $\mathrm{d}f = \frac{1}{2}\sin(\theta)\mathrm{d}\theta$

In [186…
```
print(df.parent())
```

```
Module Omega^1(S^2) of 1-forms on the 2-dimensional differentiable manifold S^2
```

In [187…
```
df.parent()
```

Out[187]: $\Omega^1\left(\mathbb{S}^2\right)$

The 1-form acting on a vector field:

In [188…
```
print(df(v))
df(v).display()
```

```
Scalar field df(v) on the 2-dimensional differentiable manifold S^2
```

Out[188]:
$$\begin{aligned}
\mathrm{d}f(v):\quad & \mathbb{S}^2 && \longrightarrow && \mathbb{R} \\
\text{on } U:\quad & (x, y) && \longmapsto && -\frac{2\,(x-2\,y)}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1} \\
\text{on } V:\quad & (x', y') && \longmapsto && -\frac{2\left(x'^3-2\,x'^2y'+x'y'^2-2\,y'^3\right)}{x'^4+y'^4+2\,(x'^2+1)y'^2+2\,x'^2+1} \\
\text{on } A:\quad & (\theta, \phi) && \longmapsto && \frac{1}{2}\left(\left(\cos(\phi)-2\,\sin(\phi)\right)\cos(\theta)-\cos(\phi)+2\,\sin(\phi)\right)\sin(\theta)
\end{aligned}$$

Let us check the identity $\mathrm{d}f(v) = v(f)$:

In [189…
```
df(v) == v(f)
```

Out[189]: True

Similarly, we have $\mathcal{L}_v f = v(f)$:

In [190…
```
f.lie_derivative(v) == v(f)
```

Out[190]: True

Curves in $\mathbb{S}^2$

In order to define curves in $\mathbb{S}^2$, we first introduce the field of real numbers $\mathbb{R}$ as a 1-dimensional smooth manifold with a canonical coordinate chart:

In [191…
```
R.<t> = manifolds.RealLine()
print(R)
```

```
Real number line ℝ
```

In [192…
```
print(R.category())
```

```
Category of smooth connected manifolds over Real Field with 53 bits of precision
```

In [193…
```
dim(R)
```

Out[193]: 1

In [194…
```
R.atlas()
```

Out[194]: $[(\mathrm{R}, (t))]$

Let us define a **loxodrome of the sphere** in terms of its parametric equation with respect to the chart `spher` = $(A, (\theta, \phi))$

In [195…
```
c = S2.curve({spher: [2*atan(exp(-t/10)), t]}, (t, -oo, +oo), name='c')
```

Curves in $\mathbb{S}^2$ are considered as morphisms from the manifold $\mathbb{R}$ to the manifold $\mathbb{S}^2$:

In [196…
```
c.parent()
```

Out[196]: $\mathrm{Hom}\left(\mathrm{R}, \mathbb{S}^2\right)$

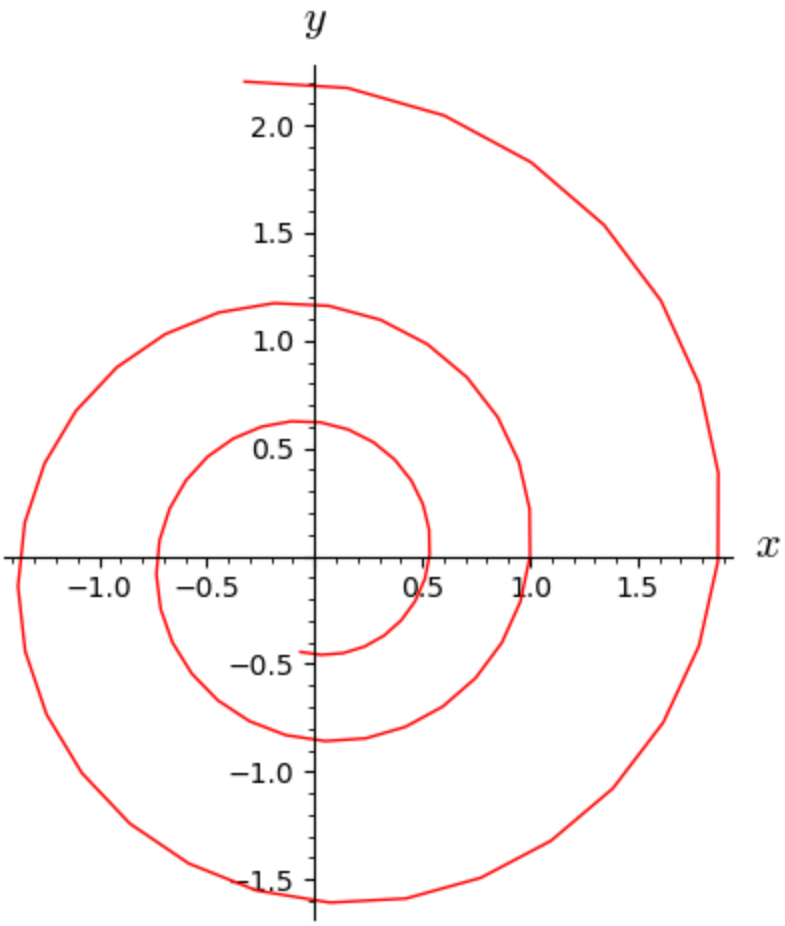
In [197…
```
c.display()
```

Out[197]:
$$\begin{aligned} c: \ \mathrm{R} &\longrightarrow \mathbb{S}^2 \\ t &\longmapsto (x, y) = \left(\cos(t) e^{\left(\frac{1}{10} t\right)}, e^{\left(\frac{1}{10} t\right)} \sin(t)\right) \\ t &\longmapsto (x', y') = \left(\cos(t) e^{\left(-\frac{1}{10} t\right)}, e^{\left(-\frac{1}{10} t\right)} \sin(t)\right) \\ t &\longmapsto (\theta, \phi) = \left(2 \arctan\left(e^{\left(-\frac{1}{10} t\right)}\right), t\right) \end{aligned}$$

The curve c can be plotted in terms of stereographic coordinates (x, y):

In [198…
```
c.plot(chart=stereoN, aspect_ratio=1)
```

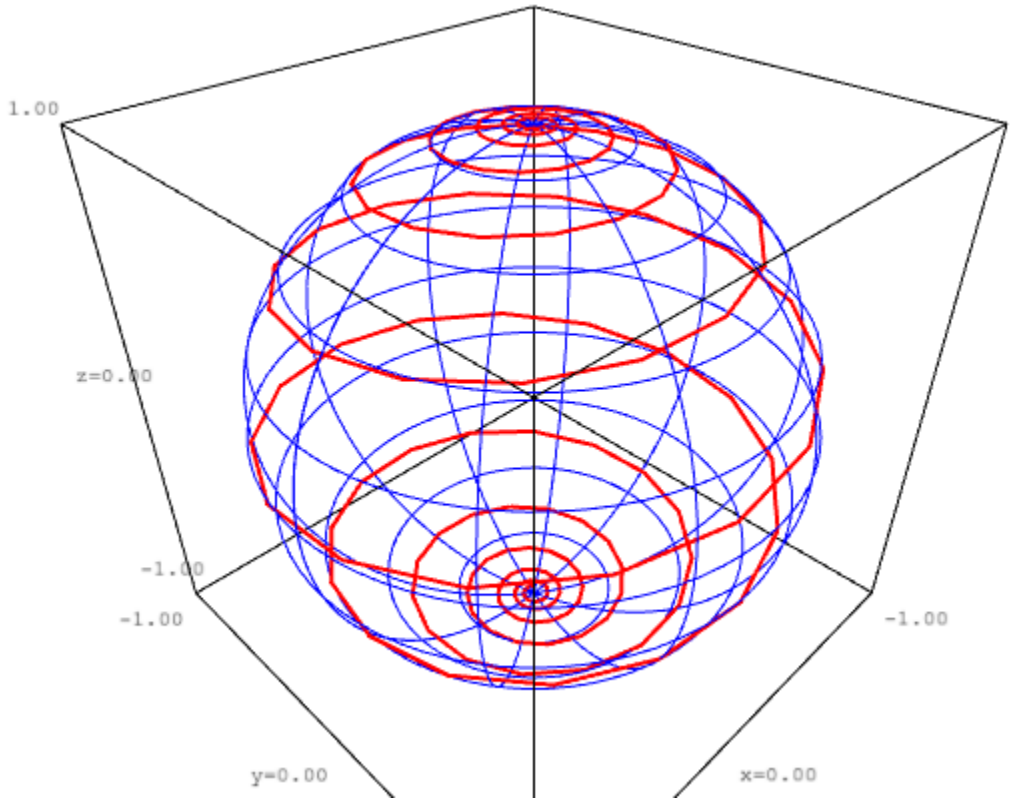
Out[198]:



Φ c $\mathbb{R}^3$

In [199…
```
graph_c = c.plot(mapping=Phi, max_range=40, plot_points=200,
                 thickness=2, label_axes=False)
graph_spher + graph_c
```

Out[199]:



c

In [200…
```
vc = c.tangent_vector_field()
vc
```

Out[200]: c'

c' $\mathbb{R}$ $\mathbb{S}^2$

In [201…
```
print(vc)
```

$\mathbb{R}$ $\mathbb{S}^2$ $c : \mathbb{R} \to \mathbb{S}^2$
$\mathfrak{X}(\mathbb{R}, c)$ $C^\infty(\mathbb{R})$

In [202…
```
vc.parent()
```

Out[202]: $\mathfrak{X}\left(\mathrm{R}, c\right)$

In [203…
```
vc.parent().category()
```

Out[203]: $\mathrm{Modules}_{C^\infty(\mathrm{R})}$

In [204…
```
vc.parent().base_ring()
```

Out[204]: $C^\infty(\mathrm{R})$

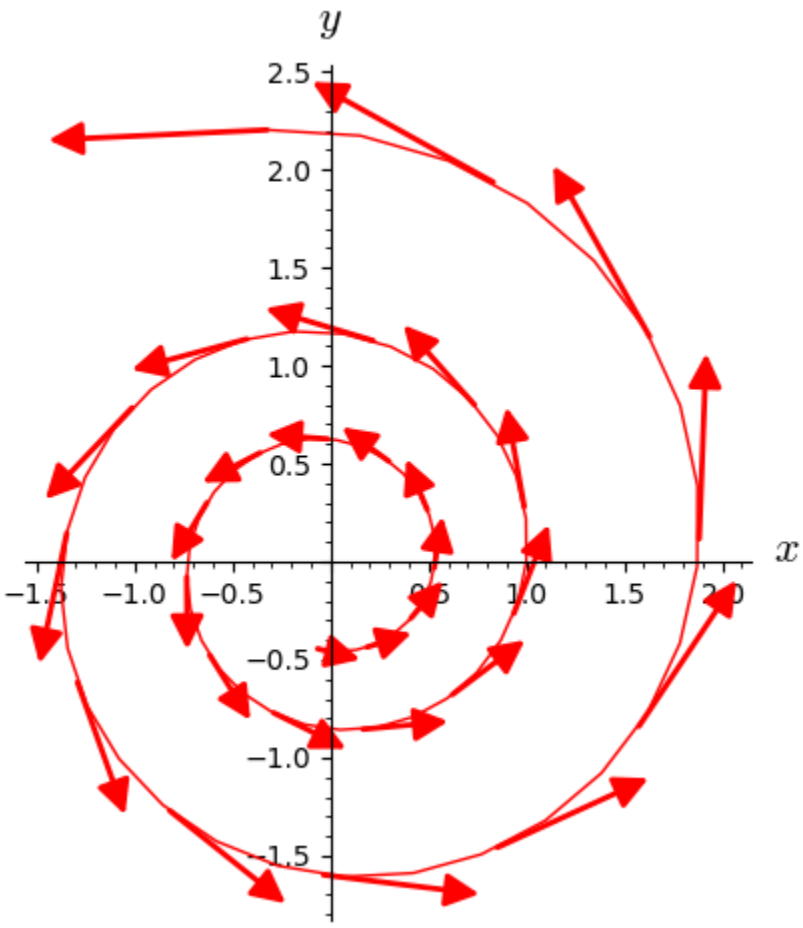
A coordinate view of c':

In [205…
```
vc.display()
```

Out[205]: $$c' = \left(\frac{1}{10}\cos(t)e^{\left(\frac{1}{10}t\right)} - e^{\left(\frac{1}{10}t\right)}\sin(t)\right)\frac{\partial}{\partial x} + \left(\cos(t)e^{\left(\frac{1}{10}t\right)} + \frac{1}{10}e^{\left(\frac{1}{10}t\right)}\sin(t)\right)\frac{\partial}{\partial y}$$

Let us plot the vector field c' in terms of the stereographic chart $(U, (x, y))$:

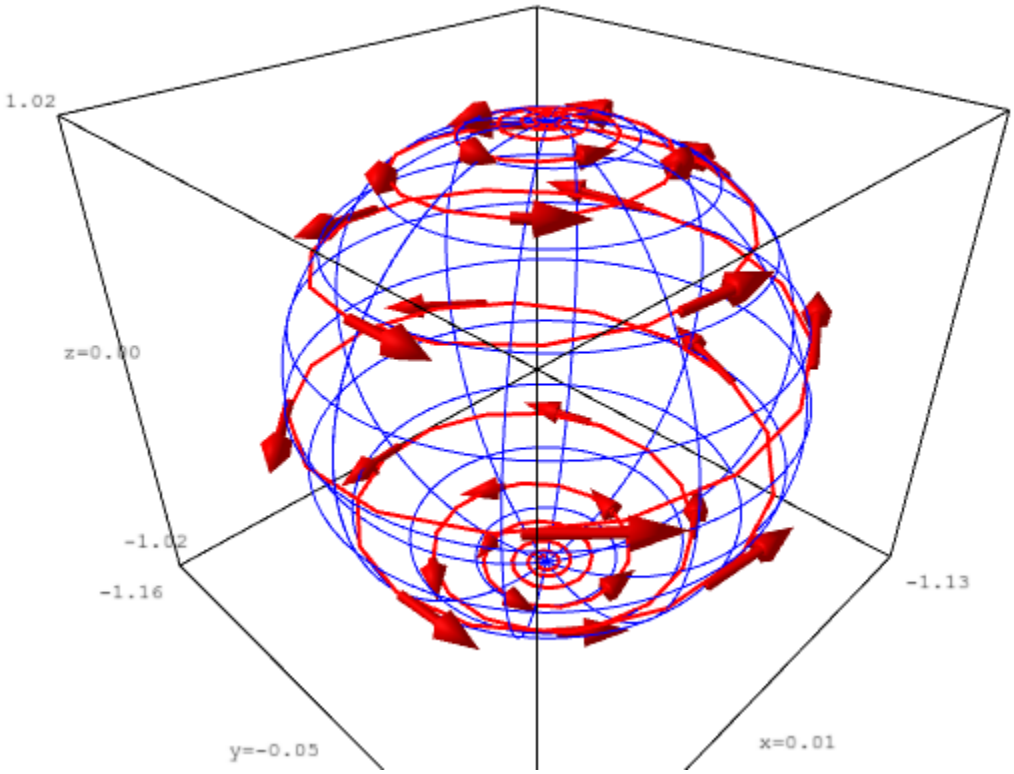
In [206…
```
show(vc.plot(chart=stereoN, number_values=30, scale=0.5, color='red') +
     c.plot(chart=stereoN), aspect_ratio=1)
```



A 3D view of c' is obtained via the embedding Φ:

In [207…
```
graph_vc = vc.plot(chart=cartesian, mapping=Phi, ranges={t: (-20, 20)},
                   number_values=30, scale=0.5, color='red',
                   label_axes=False)
graph_spher + graph_c + graph_vc
```

Out[207]:



$\mathbb{S}^2$

$\mathbb{S}^2$ $\mathbb{R}^3$

In [208…
```
h = R3.metric()
h.display()
```

Out[208]: $h = \mathrm{d}X \otimes \mathrm{d}X + \mathrm{d}Y \otimes \mathrm{d}Y + \mathrm{d}Z \otimes \mathrm{d}Z$

g $\mathbb{S}^2$ h Φ

In [209…
```
g = S2.metric('g')
g.set( Phi.pullback(h) )
print(g)
```

g

h

In [210…
```
print(g.parent())
```

In [211…
```
g.tensor_type()
```

Out[211]: $(0, 2)$

In [212…
```
g.symmetries()
```

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

The expression of the metric in terms of the default frame on $\mathbb{S}^2$ (stereoN):

In [213…]
```
g.display()
```

Out[213]:
$$g = \left(\frac{4}{x^4 + y^4 + 2\,(x^2+1)y^2 + 2\,x^2 + 1}\right) \mathrm{d}x \otimes \mathrm{d}x + \left(\frac{4}{x^4 + y^4 + 2\,(x^2+1)y^2 + 2\,x^2 + 1}\right) \mathrm{d}y \otimes \mathrm{d}y$$

We may factorize the metric components:

In [214…]
```
g.apply_map(factor, frame=eU, keep_other_components=True)
g.display()
```

Out[214]:
$$g = \frac{4}{\left(x^2 + y^2 + 1\right)^2} \mathrm{d}x \otimes \mathrm{d}x + \frac{4}{\left(x^2 + y^2 + 1\right)^2} \mathrm{d}y \otimes \mathrm{d}y$$

A matrix view of the components of g in the manifold's default frame:

In [215…]
```
g[:]
```

Out[215]:
$$\begin{pmatrix} \frac{4}{\left(x^2+y^2+1\right)^2} & 0 \\ 0 & \frac{4}{\left(x^2+y^2+1\right)^2} \end{pmatrix}$$

In [216…]
```
g[1,1]
```

Out[216]:
$$\frac{4}{\left(x^2 + y^2 + 1\right)^2}$$

Display in terms of the vector frame `eV` = $(V, (\partial_{x'}, \partial_{y'}))$:

In [217…]
```
g.apply_map(factor, frame=eV, keep_other_components=True)
g.display(eV)
```

Out[217]:
$$g = \frac{4}{\left(x'^2 + y'^2 + 1\right)^2} \mathrm{d}x' \otimes \mathrm{d}x' + \frac{4}{\left(x'^2 + y'^2 + 1\right)^2} \mathrm{d}y' \otimes \mathrm{d}y'$$

Expression of the metric tensor in terms of spherical coordinates:

In [218…]
```
g.display(spher.frame(), chart=spher)
```

Out[218]:
$$g = \mathrm{d}\theta \otimes \mathrm{d}\theta + \sin\left(\theta\right)^2 \mathrm{d}\phi \otimes \mathrm{d}\phi$$

The metric acts on vector field pairs, resulting in a scalar field:

In [219…]
```
print(g(v,v))
```

```
Scalar field g(v,v) on the 2-dimensional differentiable manifold S^2
```

In [220…]
```
g(v,v).parent()
```

Out[220]:
$$C^\infty\left(\mathbb{S}^2\right)$$

In [221…
```
g(v,v).display()
```

Out[221]:

$$\begin{aligned} g(v,v): \quad \mathbb{S}^2 &\longrightarrow \mathbb{R} \\ \text{on } U: \quad (x, y) &\longmapsto \frac{20}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1} \\ \text{on } V: \quad (x', y') &\longmapsto \frac{20\left(x'^4+2\,x'^2y'^2+y'^4\right)}{x'^4+y'^4+2\,\left(x'^2+1\right)y'^2+2\,x'^2+1} \\ \text{on } A: \quad (\theta, \phi) &\longmapsto 5\,\cos(\theta)^2 - 10\,\cos(\theta) + 5 \end{aligned}$$

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

In [222…
```
nabla = g.connection()
print(nabla)
nabla
```

```
Levi-Civita connection nabla_g associated with the Riemannian metric g on the 2-dimensio
nal differentiable manifold S^2
```

Out[222]: ∇_g

As a test, we verify that ∇_g acting on g results in zero:

In [223…
```
nabla(g).display()
```

Out[223]: $\nabla_g g = 0$

The nonzero Christoffel symbols of g (skipping those that can be deduced by symmetry on the last two indices) w.r.t. two charts:

In [224…
```
g.christoffel_symbols_display(chart=stereoN)
```

Out[224]:

$$\begin{aligned} \Gamma^{x}{}_{x\,x} &= -\frac{2\,x}{x^2+y^2+1} \\ \Gamma^{x}{}_{x\,y} &= -\frac{2\,y}{x^2+y^2+1} \\ \Gamma^{x}{}_{y\,y} &= \frac{2\,x}{x^2+y^2+1} \\ \Gamma^{y}{}_{x\,x} &= \frac{2\,y}{x^2+y^2+1} \\ \Gamma^{y}{}_{x\,y} &= -\frac{2\,x}{x^2+y^2+1} \\ \Gamma^{y}{}_{y\,y} &= -\frac{2\,y}{x^2+y^2+1} \end{aligned}$$

In [225…
```
g.christoffel_symbols_display(chart=spher)
```

Out[225]:

$$\begin{aligned} \Gamma^{\theta}{}_{\phi\,\phi} &= -\cos(\theta)\sin(\theta) \\ \Gamma^{\phi}{}_{\theta\,\phi} &= \frac{\cos(\theta)}{\sin(\theta)} \end{aligned}$$

∇_g acting on the vector field v:

In [226…
```
print(nabla(v))
```

```
Tensor field nabla_g(v) of type (1,1) on the 2-dimensional differentiable manifold S^2
```

In [227…
```
nabla(v).display(stereoN.frame())
```

Out[227]:
$$\nabla_g v = \left(-\frac{2\,(x-2\,y)}{x^2+y^2+1}\right)\frac{\partial}{\partial x}\otimes \mathrm{d}x + \left(-\frac{2\,(2\,x+y)}{x^2+y^2+1}\right)\frac{\partial}{\partial x}\otimes \mathrm{d}y + \left(\frac{2\,(2\,x+y)}{x^2+y^2+1}\right)\frac{\partial}{\partial y}\otimes \mathrm{d}x + \left(-\frac{2\,(x-2\,y)}{x^2+y^2+1}\right)\frac{\partial}{\partial y}\otimes \mathrm{d}y$$

In [228…]
```
nabla(v)[:]
```

Out[228]:
$$\begin{pmatrix} -\frac{2\,(x-2\,y)}{x^2+y^2+1} & -\frac{2\,(2\,x+y)}{x^2+y^2+1} \\ \frac{2\,(2\,x+y)}{x^2+y^2+1} & -\frac{2\,(x-2\,y)}{x^2+y^2+1} \end{pmatrix}$$

In [229…]
```
nabla(v)[1,2]
```

Out[229]:
$$-\frac{2\,(2\,x+y)}{x^2+y^2+1}$$

Curvature

The Riemann tensor associated with the metric g:

In [230…]
```
Riem = g.riemann()
print(Riem)
Riem.display()
```

```
Tensor field Riem(g) of type (1,3) on the 2-dimensional differentiable manifold S^2
```

Out[230]:
$$\mathrm{Riem}\,(g) = \left(\frac{4}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1}\right)\frac{\partial}{\partial x}\otimes \mathrm{d}y\otimes \mathrm{d}x\otimes \mathrm{d}y + \left(-\frac{4}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1}\right)\frac{\partial}{\partial x}\otimes \mathrm{d}y\otimes \mathrm{d}y\otimes \mathrm{d}x + \left(-\frac{4}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1}\right)\frac{\partial}{\partial y}\otimes \mathrm{d}x\otimes \mathrm{d}x\otimes \mathrm{d}y + \left(\frac{4}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1}\right)\frac{\partial}{\partial y}\otimes \mathrm{d}x\otimes \mathrm{d}y\otimes \mathrm{d}x$$

The components of the Riemann tensor in the default frame on $\mathbb{S}^2$:

In [231…]
```
Riem.display_comp()
```

Out[231]:
$$\begin{aligned} \mathrm{Riem}(g)^{x}{}_{y\,x\,y} &= \frac{4}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1} \\ \mathrm{Riem}(g)^{x}{}_{y\,y\,x} &= -\frac{4}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1} \\ \mathrm{Riem}(g)^{y}{}_{x\,x\,y} &= -\frac{4}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1} \\ \mathrm{Riem}(g)^{y}{}_{x\,y\,x} &= \frac{4}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1} \end{aligned}$$

The components in the frame associated with spherical coordinates:

In [232…]
```
Riem.display_comp(spher.frame(), chart=spher)
```

Out[232]:

$$\begin{aligned}
\mathrm{Riem}(g)^{\theta}{}_{\phi\theta\phi} &= \sin(\theta)^2 \\
\mathrm{Riem}(g)^{\theta}{}_{\phi\phi\theta} &= -\sin(\theta)^2 \\
\mathrm{Riem}(g)^{\phi}{}_{\theta\theta\phi} &= -1 \\
\mathrm{Riem}(g)^{\phi}{}_{\theta\phi\theta} &= 1
\end{aligned}$$

In [233…]
```
print(Riem.parent())
```

```
Module T^(1,3)(S^2) of type-(1,3) tensors fields on the 2-dimensional differentiable man
ifold S^2
```

In [234…]
```
Riem.symmetries()
```

```
no symmetry; antisymmetry: (2, 3)
```

The Riemann tensor associated with the Euclidean metric h on $\mathbb{R}^3$ is identically zero:

In [235…]
```
h.riemann().display()
```

Out[235]:
$$\mathrm{Riem}\,(h) = 0$$

The Ricci tensor and the Ricci scalar:

In [236…]
```
Ric = g.ricci()
Ric.display()
```

Out[236]:
$$\mathrm{Ric}\,(g) = \left(\frac{4}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1}\right) \mathrm{d}x\otimes\mathrm{d}x + \left(\frac{4}{x^4+y^4+2\,(x^2+1)y^2+2\,x^2+1}\right) \mathrm{d}y\otimes\mathrm{d}y$$

In [237…]
```
print(g.inverse())
```

```
Tensor field inv_g of type (2,0) on the 2-dimensional differentiable manifold S^2
```

In [238…]
```
g.inverse().display()
```

Out[238]:
$$g^{-1} = \left(\frac{1}{4}\,x^4+\frac{1}{4}\,y^4+\frac{1}{2}\,(x^2+1)y^2+\frac{1}{2}\,x^2+\frac{1}{4}\right)\frac{\partial}{\partial x}\otimes\frac{\partial}{\partial x} + \left(\frac{1}{4}\,x^4+\frac{1}{4}\,y^4+\frac{1}{2}\,(x^2+1)y^2+\frac{1}{2}\,x^2+\frac{1}{4}\right)\frac{\partial}{\partial y}\otimes\frac{\partial}{\partial y}$$

In [239…]
```
R = g.ricci_scalar()
R.display()
```

Out[239]:
$$\begin{aligned}
\mathrm{r}\,(g):\quad & \mathbb{S}^2 && \longrightarrow && \mathbb{R} \\
\text{on } U:\quad & (x,y) && \longmapsto && 2 \\
\text{on } V:\quad & (x',y') && \longmapsto && 2 \\
\text{on } A:\quad & (\theta,\phi) && \longmapsto && 2
\end{aligned}$$

Hence we recover the fact that $(\mathbb{S}^2, g)$ is a Riemannian manifold of constant positive curvature.

In dimension 2, the Riemann curvature tensor is entirely determined by the Ricci scalar R according to

$$R^i{}_{jlk} = \frac{R}{2}\left(\delta^i{}_k g_{jl} - \delta^i{}_l g_{jk}\right)$$

Let us check this formula here, under the form $R^i{}_{jlk} = -R g_{j[k}\delta^i{}_{l]}$:

In [240…]
```
delta = S2.tangent_identity_field()
Riem == - R*(g*delta).antisymmetrize(2,3)
```

Out[240]: True

Similarly the relation $\mathrm{Ric} = (R/2)\, g$ must hold:

In [241…]
```
Ric == (R/2)*g
```

Out[241]: True

Manifold orientation and volume 2-form

In order to introduce the volume 2-form associated with the metric g, we need to define an orientation on $\mathbb{S}^2$ first. We choose the orientation so that the vector frame $(\partial/\partial x', \partial/\partial y')$ of the stereographic coordinates from the South pole is right-handed. This is somewhat natural, because the triplet $(\partial/\partial x', \partial/\partial y', n)$, where n is the unit outward normal to $\mathbb{S}^2$, is right-handed with respect to the standard orientation of $\mathbb{R}^3$. On the contrary the triplet $(\partial/\partial x, \partial/\partial y, n)$ formed from stereographic coordinates from the North pole is left-handed (see the above plot). Actually, we can check that $(\partial/\partial x, \partial/\partial y)$ and $(\partial/\partial x', \partial/\partial y')$ lead to two opposite orientations, because the transition map $(x, y) \mapsto (x', y')$ has a negative Jacobian determinant:

In [242…]
```
stereoN_to_S.jacobian_det()
```

Out[242]: $$-\frac{1}{x^4 + 2\,x^2y^2 + y^4}$$

We define the orientation via the method `set_orientation()` with a list of right-handed vector frames, whose domains form an open cover of $\mathbb{S}^2$. We therefore provide `eV` = $(\partial/\partial x', \partial/\partial y')$ (domain: V) and the "reversed" frame $(\partial/\partial y, \partial/\partial x)$ on U:

In [243…]
```
reU = U.vector_frame('f', (eU[2], eU[1]))
reU[1].display(eU), reU[2].display(eU)
```

Out[243]: $$\left(f_1 = \frac{\partial}{\partial y}, f_2 = \frac{\partial}{\partial x}\right)$$

In [244…]
```
S2.set_orientation([eV, reU])
```

The **volume 2-form** or **Levi-Civita tensor** associated with g is then

In [245…]
```
eps = g.volume_form()
print(eps)
eps.display()
```

```
2-form eps_g on the 2-dimensional differentiable manifold S^2
```

Out[245]: $$\epsilon_g = \left(-\frac{4}{x^4 + y^4 + 2\,(x^2+1)y^2 + 2\,x^2 + 1}\right) \mathrm{d}x \wedge \mathrm{d}y$$

Notice the minus sign in the the above expression, which reflects the fact that the default frame

$(\partial/\partial x, \partial/\partial y)$ is left-handed. On the contrary, we have

In [246…]:
```
eps.display(eV)
```

Out[246]:
$$\epsilon_g = \left(\frac{4}{x'^4 + y'^4 + 2\left(x'^2+1\right)y'^2 + 2\,x'^2 + 1} \right) \mathrm{d}x' \wedge \mathrm{d}y'$$

A nicer display is obtained by factorizing the components:

In [247…]:
```
eps.apply_map(factor, frame=eV, keep_other_components=True)
eps.display(stereoS.frame())
```

Out[247]:
$$\epsilon_g = \frac{4}{\left(x'^2 + y'^2 + 1\right)^2} \mathrm{d}x' \wedge \mathrm{d}y'$$

The frame associated with spherical coordinates is right-handed and we recover the standard expression of the volume 2-form:

In [248…]:
```
eps.display(spher.frame(), chart=spher)
```

Out[248]:
$$\epsilon_g = \sin(\theta) \mathrm{d}\theta \wedge \mathrm{d}\phi$$

The exterior derivative of the 2-form ϵ_g:

In [249…]:
```
print(diff(eps))
```

```
3-form deps_g on the 2-dimensional differentiable manifold S^2
```

Of course, since $\mathbb{S}^2$ has dimension 2, all 3-forms vanish identically:

In [250…]:
```
diff(eps).display()
```

Out[250]:
$$\mathrm{d}\epsilon_g = 0$$

Non-holonomic frames

Up to know, all the vector frames introduced on $\mathbb{S}^2$ have been coordinate frames. Let us introduce a non-coordinate frame on the open subset A. To ease the manipulations, we change first the default chart and default frame on A to the spherical coordinate ones:

In [251…]:
```
A.default_chart()
```

Out[251]:
$$(A, (x, y))$$

In [252…]:
```
A.default_frame()
```

Out[252]:
$$\left(A, \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)\right)$$

In [253…]:
```
A.set_default_chart(spher)
A.set_default_frame(spher.frame())
A.default_chart()
```

Out[253]:
$$(A, (\theta, \phi))$$

In [254…
```
A.default_frame()
```

Out[254]: $\left(A, \left(\frac{\partial}{\partial\theta}, \frac{\partial}{\partial\phi}\right)\right)$

We introduce the new vector frame $e = \left(\frac{\partial}{\partial\theta}, \frac{1}{\sin\theta}\frac{\partial}{\partial\phi}\right)$:

In [255…
```
spher.frame()[:]
```

Out[255]: $\left(\frac{\partial}{\partial\theta}, \frac{\partial}{\partial\phi}\right)$

In [256…
```
d_dth, d_dph = spher.frame()[:]
e = A.vector_frame('e', (d_dth, 1/sin(th)*d_dph))
print(e)
e
```

```
Vector frame (A, (e_1,e_2))
```

Out[256]: $(A, (e_1, e_2))$

In [257…
```
(e[1].display(), e[2].display())
```

Out[257]: $\left(e_1 = \frac{\partial}{\partial\theta}, e_2 = \frac{1}{\sin(\theta)}\frac{\partial}{\partial\phi}\right)$

The new frame is an orthonormal frame for the metric g:

In [258…
```
g(e[1],e[1]).expr()
```

Out[258]: 1

In [259…
```
g(e[1],e[2]).expr()
```

Out[259]: 0

In [260…
```
g(e[2],e[2]).expr()
```

Out[260]: 1

In [261…
```
g[e,:]
```

Out[261]: $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

In [262…
```
g.display(e)
```

Out[262]: $g = e^1 \otimes e^1 + e^2 \otimes e^2$

In [263…
```
eps.display(e)
```

Out[263]: $\epsilon_g = e^1 \wedge e^2$

It is non-holonomic, since its structure coefficients are not identically zero:

In [264…
```
e.structure_coeff()[:]
```

Out[264]:

$$\left[[[0,0],[0,0]], \left[\left[0, -\frac{\cos(\theta)}{\sin(\theta)}\right], \left[\frac{\cos(\theta)}{\sin(\theta)}, 0\right]\right]\right]$$

In [265…
```
e[2].lie_derivative(e[1]).display(e)
```

Out[265]:
$$-\frac{\cos(\theta)}{\sin(\theta)} e_2$$

while we have of course

In [266…
```
spher.frame().structure_coeff()[:]
```

Out[266]: $[[[0,0],[0,0]],[[0,0],[0,0]]]$

Using SymPy as the symbolic backend

By default, the symbolic backend used in calculus on manifolds is SageMath's one (Pynac + Maxima), implemented via the symbolic ring `SR`. We can choose to use SymPy instead:

In [267…
```
S2.set_calculus_method('sympy')
```

In [268…
```
F = 2*f
F.display()
```

Out[268]:
$$\begin{array}{lccc} & \mathbb{S}^2 & \longrightarrow & \mathbb{R} \\ \text{on } U: & (x, y) & \longmapsto & \frac{2}{x^2+y^2+1} \\ \text{on } V: & (x', y') & \longmapsto & \frac{2\left(xp^2+yp^2\right)}{xp^2+yp^2+1} \\ \text{on } A: & (\theta, \phi) & \longmapsto & 1 - \cos\left(th\right) \end{array}$$

In [269…
```
F.expr()
```

Out[269]:
```
2/(x**2 + y**2 + 1)
```

In [270…
```
type(F.expr())
```

Out[270]:
```
<class 'sympy.core.mul.Mul'>
```

Back to Sage's default:

In [271…
```
S2.set_calculus_method('SR')
```

In [272…
```
F.expr()
```

Out[272]:
$$\frac{2}{x^2 + y^2 + 1}$$

In [273…
```
type(F.expr())
```

Out[273]:
```
<class 'sage.symbolic.expression.Expression'>
```

Going further

See the notebooks Smooth manifolds, charts and scalar fields and Smooth manifolds, vector fields and tensor fields from the lectures Symbolic tensor calculus on manifolds. Many example notebooks are provided at the SageManifolds page.

See also the series of notebooks by Andrzej Chrzeszczyk: Introduction to manifolds in SageMath, as well as the tutorial videos by Christian Bär: Manifolds in SageMath.