Differential geometry with SageMath

Éric Gourgoulhon

Laboratoire Univers et Théories (LUTH)
CNRS / Observatoire de Paris / Université Paris Diderot
Paris Sciences et Lettres Research University
92190 Meudon, France

http://luth.obspm.fr/~luthier/gourgoulhon/

based on a collaboration with

Pablo Angulo, Michał Bejger, Marco Mancini and Travis Scrimshaw

Geometry and Computer Science
Università degli Studi "G. d'Annunzio", Pescara, Italy
8-10 February 2017

Outline

Outline

Introduction

- Computer algebra system (CAS) started to be developed in the 1960's; for instance `Macsyma` (to become `Maxima` in 1998) was initiated in 1968 at MIT

Introduction

- Computer algebra system (CAS) started to be developed in the 1960's; for instance Macsyma (to become Maxima in 1998) was initiated in 1968 at MIT
- In 1965, J.G. Fletcher developed the GEOM program, to compute the Riemann tensor of a given metric

Introduction

- Computer algebra system (CAS) started to be developed in the 1960's; for instance `Macsyma` (to become `Maxima` in 1998) was initiated in 1968 at MIT
- In 1965, J.G. Fletcher developed the `GEOM` program, to compute the Riemann tensor of a given metric
- In 1969, during his PhD under Pirani supervision, Ray d'Inverno wrote `ALAM (Atlas Lisp Algebraic Manipulator)` and used it to compute the Riemann tensor of Bondi metric. The original calculations took Bondi and his collaborators 6 months to go. The computation with ALAM took 4 minutes and yielded to the discovery of 6 errors in the original paper [J.E.F. Skea, *Applications of SHEEP* (1994)]

Introduction

- Computer algebra system (CAS) started to be developed in the 1960's; for instance Macsyma (to become Maxima in 1998) was initiated in 1968 at MIT
- In 1965, J.G. Fletcher developed the GEOM program, to compute the Riemann tensor of a given metric
- In 1969, during his PhD under Pirani supervision, Ray d'Inverno wrote ALAM (Atlas Lisp Algebraic Manipulator) and used it to compute the Riemann tensor of Bondi metric. The original calculations took Bondi and his collaborators 6 months to go. The computation with ALAM took 4 minutes and yielded to the discovery of 6 errors in the original paper [J.E.F. Skea, *Applications of SHEEP* (1994)]
- Since then, many software tools for tensor calculus have been developed... A rather exhaustive list: http://www.xact.es/links.html
  $\Longrightarrow$ cf. Maximilian Hasler's review talk on Friday.

Outline

SageMath in a few words

- SageMath (*nickname:* Sage) is a **free open-source** mathematics software system

SageMath in a few words

- SageMath (*nickname:* Sage) is a **free open-source** mathematics software system
- it is based on the Python programming language

SageMath in a few words

- SageMath (*nickname:* Sage) is a **free open-source** mathematics software system
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which

  and provides a **uniform interface** to them

SageMath in a few words

- SageMath (*nickname:* Sage) is a **free open-source** mathematics software system
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which
  - Maxima, Pynac (symbolic calculations)

and provides a **uniform interface** to them

SageMath in a few words

- SageMath (*nickname:* Sage) is a **free open-source** mathematics software system
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which
  - Maxima, Pynac (symbolic calculations)
  - GAP (group theory)

and provides a **uniform interface** to them

SageMath in a few words

- SageMath (*nickname: Sage*) is a **free open-source** mathematics software system
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which
  - Maxima, Pynac (symbolic calculations)
  - GAP (group theory)
  - PARI/GP (number theory)

and provides a **uniform interface** to them

SageMath in a few words

- SageMath (*nickname:* Sage) is a **free open-source** mathematics software system
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which
  - Maxima, Pynac (symbolic calculations)
  - GAP (group theory)
  - PARI/GP (number theory)
  - Singular (polynomial computations)

and provides a **uniform interface** to them

SageMath in a few words

- SageMath (*nickname:* Sage) is a **free open-source** mathematics software system
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which
  - Maxima, Pynac (symbolic calculations)
  - GAP (group theory)
  - PARI/GP (number theory)
  - Singular (polynomial computations)
  - matplotlib (high quality 2D figures)

  and provides a **uniform interface** to them

SageMath in a few words

- SageMath (*nickname:* Sage) is a **free open-source** mathematics software system
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which
  - Maxima, Pynac (symbolic calculations)
  - GAP (group theory)
  - PARI/GP (number theory)
  - Singular (polynomial computations)
  - matplotlib (high quality 2D figures)

  and provides a **uniform interface** to them
- William Stein (Univ. of Washington) created SageMath in 2005; since then, ∼**100 developers** (mostly mathematicians) have joined the SageMath team

SageMath in a few words

- SageMath (*nickname:* Sage) is a **free open-source** mathematics software system
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which
  - Maxima, Pynac (symbolic calculations)
  - GAP (group theory)
  - PARI/GP (number theory)
  - Singular (polynomial computations)
  - matplotlib (high quality 2D figures)

  and provides a **uniform interface** to them
- William Stein (Univ. of Washington) created SageMath in 2005; since then, ~**100 developers** (mostly mathematicians) have joined the SageMath team
- SageMath is now supported by European Union via the open-math project OpenDreamKit (2015-2019, within the *Horizon 2020* program)

SageMath in a few words

- SageMath (*nickname:* Sage) is a **free open-source** mathematics software system
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which
  - Maxima, Pynac (symbolic calculations)
  - GAP (group theory)
  - PARI/GP (number theory)
  - Singular (polynomial computations)
  - matplotlib (high quality 2D figures)

  and provides a **uniform interface** to them
- William Stein (Univ. of Washington) created SageMath in 2005; since then, $\sim$**100 developers** (mostly mathematicians) have joined the SageMath team
- SageMath is now supported by European Union via the open-math project OpenDreamKit (2015-2019, within the *Horizon 2020* program)

SageMath in a few words

- SageMath (*nickname: Sage*) is a **free open-source** mathematics software system
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which
  - Maxima, Pynac (symbolic calculations)
  - GAP (group theory)
  - PARI/GP (number theory)
  - Singular (polynomial computations)
  - matplotlib (high quality 2D figures)

  and provides a **uniform interface** to them
- William Stein (Univ. of Washington) created SageMath in 2005; since then, ∼**100 developers** (mostly mathematicians) have joined the SageMath team
- SageMath is now supported by European Union via the open-math project OpenDreamKit (2015-2019, within the *Horizon 2020* program)

The mission

Create a viable free open source alternative to Magma, Maple, Mathematica and Matlab.

Some advantages of SageMath

SageMath is free

Freedom means

1. everybody can use it, by downloading the software from `http://sagemath.org`
2. everybody can examine the source code and improve it

Some advantages of SageMath

SageMath is free

Freedom means

1. everybody can use it, by downloading the software from http://sagemath.org
2. everybody can examine the source code and improve it

SageMath is based on Python

- no need to learn any specific syntax to use it
- easy access for students
- Python is a very powerful *object oriented language*, with a neat syntax

Some advantages of SageMath

SageMath is free

Freedom means

1. everybody can use it, by downloading the software from `http://sagemath.org`
2. everybody can examine the source code and improve it

SageMath is based on Python

- no need to learn any specific syntax to use it
- easy access for students
- Python is a very powerful *object oriented language*, with a neat syntax

SageMath is developing and spreading fast

...sustained by an enthusiastic community of developers

Object-oriented notation in Python

As an object-oriented language, Python (and hence SageMath) makes use of the following **postfix notation** (same in C++, Java, etc.):

$$\texttt{result} = \texttt{object.function(arguments)}$$

In a procedural language, this would be written as

$$\texttt{result} = \texttt{function(object,arguments)}$$

Object-oriented notation in Python

As an object-oriented language, Python (and hence SageMath) makes use of the following **postfix notation** (same in C++, Java, etc.):

```
result = object.function(arguments)
```

In a procedural language, this would be written as

```
result = function(object,arguments)
```

Examples

1. `riem = g.riemann()`
2. `lie_t_v = t.lie_der(v)`

NB: no argument in example 1

SageMath approach to computer mathematics

SageMath relies on a Parent / Element scheme: each object x on which some calculus is performed has a "parent", which is another SageMath object X representing the set to which x belongs.

The calculus rules on x are determined by the *algebraic structure* of X.

Conversion rules prior to an operation, e.g. $x + y$ with x and y having different parents, are defined at the level of the parents

SageMath approach to computer mathematics

SageMath relies on a `Parent` / `Element` scheme: each object x on which some calculus is performed has a "parent", which is another SageMath object X representing the set to which x belongs.
The calculus rules on x are determined by the *algebraic structure* of X.
Conversion rules prior to an operation, e.g. $x + y$ with x and y having different parents, are defined at the level of the parents

Example

```
sage: x = 4 ; x.parent()
Integer Ring
sage: y = 4/3 ; y.parent()
Rational Field
sage: s = x + y ; s.parent()
Rational Field
sage: y.parent().has_coerce_map_from(x.parent())
True
```

SageMath approach to computer mathematics

SageMath relies on a `Parent` / `Element` scheme: each object x on which some calculus is performed has a "parent", which is another SageMath object X representing the set to which x belongs.
The calculus rules on x are determined by the *algebraic structure* of X.
Conversion rules prior to an operation, e.g. $x + y$ with x and y having different parents, are defined at the level of the parents

Example

```
sage: x = 4 ; x.parent()
Integer Ring
sage: y = 4/3 ; y.parent()
Rational Field
sage: s = x + y ; s.parent()
Rational Field
sage: y.parent().has_coerce_map_from(x.parent())
True
```

This approach is similar to that of Magma and is different from that of Mathematica, in which everything is a tree of symbols

Outline

The SageManifolds project

http://sagemanifolds.obspm.fr/

Aim

Implement smooth manifolds of arbitrary dimension in SageMath and tensor calculus on them

In particular:

- one should be able to introduce an arbitrary number of coordinate charts on a given manifold, with the relevant transition maps
- tensor fields must be manipulated as such and not through their components with respect to a specific (possibly coordinate) vector frame

The SageManifolds project

http://sagemanifolds.obspm.fr/

Aim

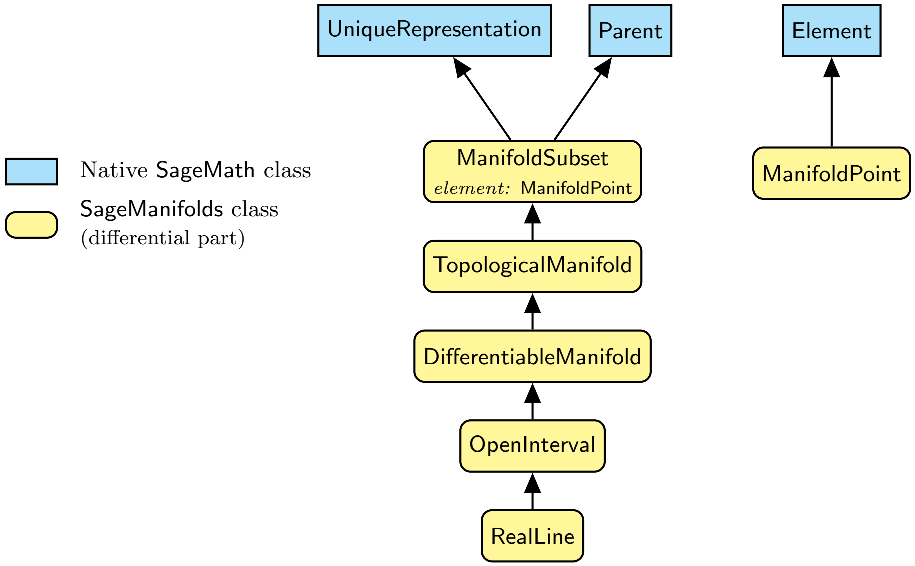
Implement smooth manifolds of arbitrary dimension in SageMath and tensor calculus on them

In particular:

- one should be able to introduce an arbitrary number of coordinate charts on a given manifold, with the relevant transition maps
- tensor fields must be manipulated as such and not through their components with respect to a specific (possibly coordinate) vector frame

Concretely, the project amounts to creating new Python classes, such as `TopologicalManifold`, `DifferentiableManifold`, `Chart`, `TensorField` or `Metric`, within SageMath's **Parent/Element framework**.

Implementing manifolds and their subsets



Implementing coordinate charts

Given a (topological) manifold M of dimension $n \geq 1$, a **coordinate chart** is a homeomorphism $\varphi : U \to V$, where U is an open subset of M and V is an open subset of $\mathbb{R}^n$.

Implementing coordinate charts

Given a (topological) manifold M of dimension $n \geq 1$, a **coordinate chart** is a homeomorphism $\varphi : U \to V$, where U is an open subset of M and V is an open subset of $\mathbb{R}^n$.

In general, more than one chart is required to cover the entire manifold:

Examples:

- at least 2 charts are necessary to cover the n-dimensional sphere $\mathbb{S}^n$ $(n \geq 1)$ and the torus $\mathbb{T}^2$
- at least 3 charts are necessary to cover the real projective plane $\mathbb{RP}^2$

Implementing coordinate charts

Given a (topological) manifold M of dimension $n \geq 1$, a **coordinate chart** is a homeomorphism $\varphi : U \to V$, where U is an open subset of M and V is an open subset of $\mathbb{R}^n$.

In general, more than one chart is required to cover the entire manifold:

Examples:

- at least 2 charts are necessary to cover the n-dimensional sphere $\mathbb{S}^n$ $(n \geq 1)$ and the torus $\mathbb{T}^2$
- at least 3 charts are necessary to cover the real projective plane $\mathbb{RP}^2$

In SageManifolds, an arbitrary number of charts can be introduced

To fully specify the manifold, one shall also provide the *transition maps* on overlapping chart domains (SageManifolds class `CoordChange`)

Implementing scalar fields

A **scalar field** on manifold M is a smooth mapping

$$\begin{array}{llcl} f: & U \subset M & \longrightarrow & \mathbb{R} \\ & p & \longmapsto & f(p) \end{array}$$

where U is an open subset of M.

Implementing scalar fields

A **scalar field** on manifold M is a smooth mapping

$$\begin{array}{llcl} f: & U \subset M & \longrightarrow & \mathbb{R} \\ & p & \longmapsto & f(p) \end{array}$$

where U is an open subset of M.

A scalar field maps *points*, not *coordinates*, to real numbers
$\Longrightarrow$ an object f in the ScalarField class has different **coordinate representations** in different charts defined on U.

Implementing scalar fields

A **scalar field** on manifold M is a smooth mapping

$$\begin{array}{cccc} f: & U \subset M & \longrightarrow & \mathbb{R} \\ & p & \longmapsto & f(p) \end{array}$$

where U is an open subset of M.

A scalar field maps *points*, not *coordinates*, to real numbers
$\Longrightarrow$ an object f in the `ScalarField` class has different **coordinate representations** in different charts defined on U.

The various coordinate representations F, $\hat{F}$, ... of f are stored as a *Python dictionary* whose keys are the charts C, $\hat{C}$, ...:

$$f.\texttt{\_express} = \left\{ C : F, \ \hat{C} : \hat{F}, \ldots \right\}$$

with $\underbrace{f(\ p\ )}_{\text{point}} = F(\underbrace{x^1, \ldots, x^n}_{\substack{\text{coord. of } p \\ \text{in chart } C}}) = \hat{F}(\underbrace{\hat{x}^1, \ldots, \hat{x}^n}_{\substack{\text{coord. of } p \\ \text{in chart } \hat{C}}}) = \ldots$

The scalar field algebra

The parent of the scalar field $f : U \to \mathbb{R}$ is the set $C^\infty(U)$ of scalar fields defined on the open subset U.

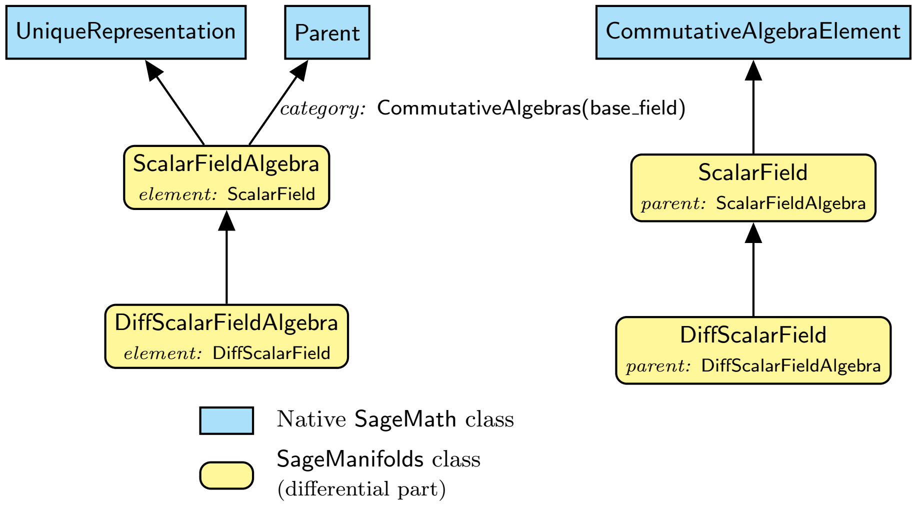
$C^\infty(U)$ has naturally the structure of a **commutative algebra over** $\mathbb{R}$:

1. it is clearly a vector space over $\mathbb{R}$
2. it is endowed with a commutative ring structure by pointwise multiplication:

$$\forall f, g \in C^\infty(U), \quad \forall p \in U, \quad (f.g)(p) := f(p)g(p)$$

The algebra $C^\infty(U)$ is implemented in SageManifolds via the class `ScalarFieldAlgebra`.

Classes for scalar fields



Vector field modules

Given an open subset $U \subset M$, the set $\mathcal{X}(U)$ of smooth vector fields defined on U has naturally the structure of a **module over the scalar field algebra** $C^\infty(U)$.

$\mathcal{X}(U)$ is a free module $\iff$ U admits a global vector frame $(\boldsymbol{e}_a)_{1\leq a\leq n}$:

$$\forall \boldsymbol{v} \in \mathcal{X}(U), \quad \boldsymbol{v} = v^a \boldsymbol{e}_a, \quad \text{with } v^a \in C^\infty(U)$$

At any point $p \in U$, the above translates into an identity in the *tangent vector space* T_pM:

$$\boldsymbol{v}(p) = v^a(p)\, \boldsymbol{e}_a(p), \quad \text{with } v^a(p) \in \mathbb{R}$$

Example:

If U is the domain of a coordinate chart $(x^a)_{1\leq a\leq n}$, $\mathcal{X}(U)$ is a free module of rank n over $C^\infty(U)$, a basis of it being the coordinate frame $(\partial/\partial x^a)_{1\leq a\leq n}$.

Parallelizable manifolds

M is a **parallelizable manifold** $\iff$ M admits a global vector frame
$\iff$ $\mathcal{X}(M)$ is a free module
$\iff$ M's tangent bundle is trivial: $TM \simeq M \times \mathbb{R}^n$

Parallelizable manifolds

M is a **parallelizable manifold** $\iff$ M admits a global vector frame
$\iff$ $\mathcal{X}(M)$ is a free module
$\iff$ M's tangent bundle is trivial: $TM \simeq M \times \mathbb{R}^n$

Examples of parallelizable manifolds

- $\mathbb{R}^n$ (global coordinate charts $\Rightarrow$ global vector frames)
- the circle $\mathbb{S}^1$ (NB: no global coordinate chart)
- the torus $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$
- the 3-sphere $\mathbb{S}^3 \simeq \mathrm{SU}(2)$, as any Lie group
- the 7-sphere $\mathbb{S}^7$
- any orientable 3-manifold (Steenrod theorem)

Parallelizable manifolds

M is a **parallelizable manifold** $\iff$ M admits a global vector frame
$\iff$ $\mathcal{X}(M)$ is a free module
$\iff$ M's tangent bundle is trivial: $TM \simeq M \times \mathbb{R}^n$

Examples of parallelizable manifolds

- $\mathbb{R}^n$ (global coordinate charts $\Rightarrow$ global vector frames)
- the circle $\mathbb{S}^1$ (NB: no global coordinate chart)
- the torus $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$
- the 3-sphere $\mathbb{S}^3 \simeq \mathrm{SU}(2)$, as any Lie group
- the 7-sphere $\mathbb{S}^7$
- any orientable 3-manifold (Steenrod theorem)

Examples of non-parallelizable manifolds

- the sphere $\mathbb{S}^2$ (hairy ball theorem!) and any n-sphere $\mathbb{S}^n$ with $n \notin \{1, 3, 7\}$
- the real projective plane $\mathbb{RP}^2$

Implementing vector fields

Ultimately, in SageManifolds, vector fields are to be described by their components w.r.t. various vector frames.

If the manifold M is not parallelizable, we assume that it can be covered by a finite number N of parallelizable open subsets U_i $(1 \leq i \leq N)$ (OK for M compact). We then consider **restrictions** of vector fields to these domains:

Implementing vector fields

Ultimately, in SageManifolds, vector fields are to be described by their components w.r.t. various vector frames.

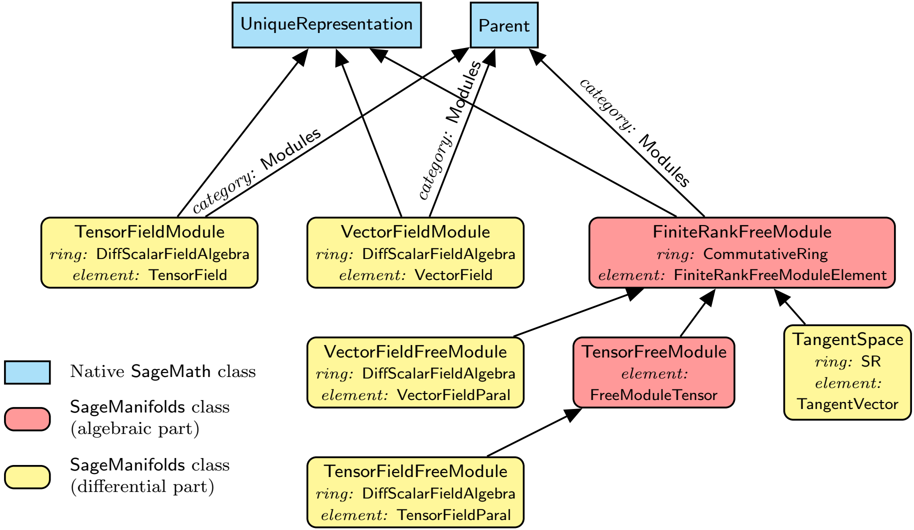
If the manifold M is not parallelizable, we assume that it can be covered by a finite number N of parallelizable open subsets U_i $(1 \leq i \leq N)$ (OK for M compact). We then consider **restrictions** of vector fields to these domains:

For each i, $\mathcal{X}(U_i)$ is a free module of rank $n = \dim M$ and is implemented in SageManifolds as an instance of `VectorFieldFreeModule`, which is a subclass of `FiniteRankFreeModule`.

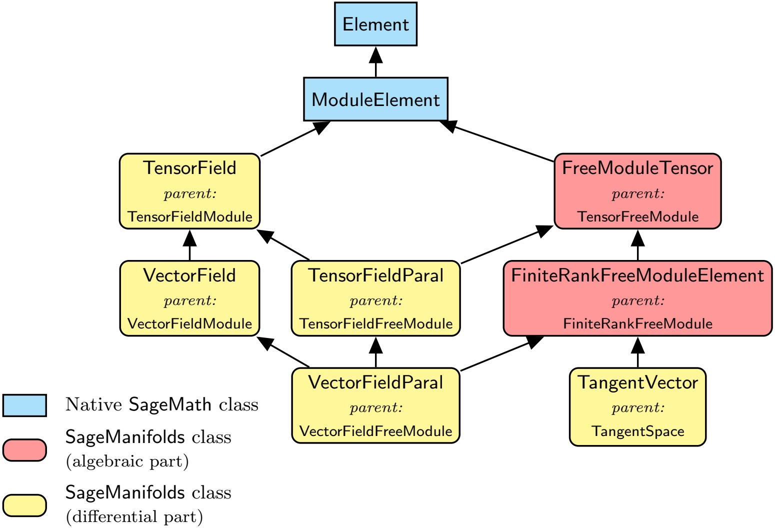
Each vector field $\boldsymbol{v} \in \mathcal{X}(U_i)$ has different set of components $(v^a)_{1 \leq a \leq n}$ in different vector frames $(\boldsymbol{e}_a)_{1 \leq a \leq n}$ introduced on U_i. They are stored as a *Python dictionary* whose keys are the vector frames:

$$\boldsymbol{v}.\texttt{\_components} = \{(\boldsymbol{e}) : (v^a),\ (\hat{\boldsymbol{e}}) : (\hat{v}^a), \ldots\}$$

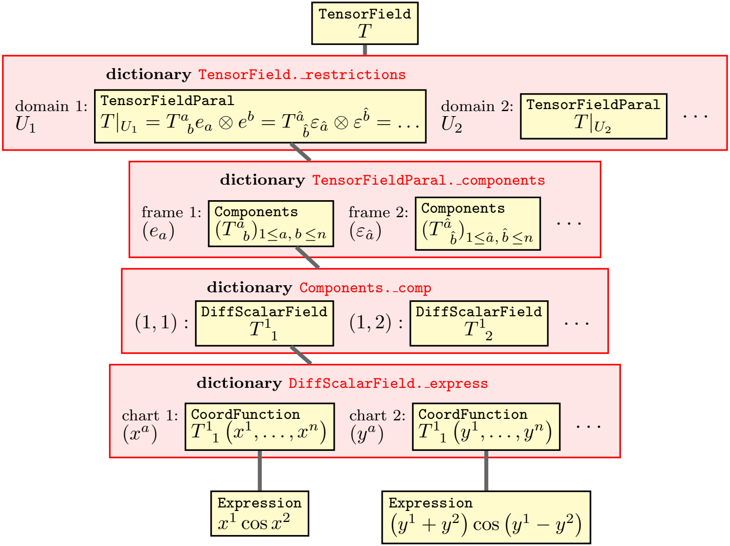
Module classes in SageManifolds



Tensor field classes

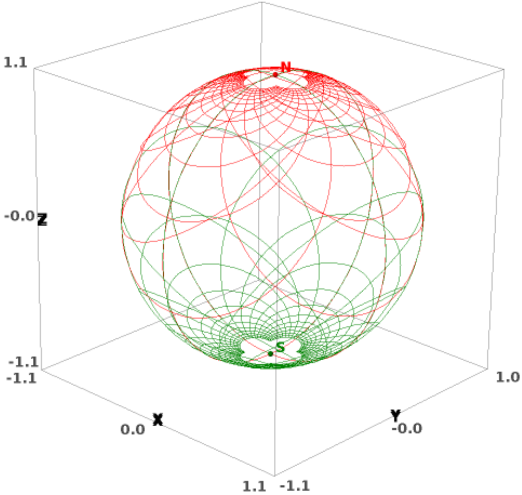


Tensor field storage



Outline

The 2-sphere example



Stereographic coordinates on the 2-sphere

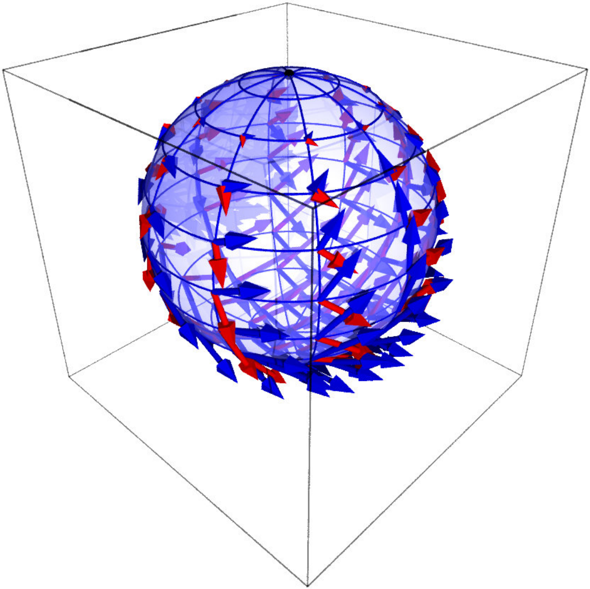
Two charts:

- X_1: $\mathbb{S}^2 \setminus \{N\} \to \mathbb{R}^2$
- X_2: $\mathbb{S}^2 \setminus \{S\} \to \mathbb{R}^2$

← picture obtained via function `RealChart.plot()`

See the worksheet at `http://sagemanifolds.obspm.fr/examples.html`

The 2-sphere example



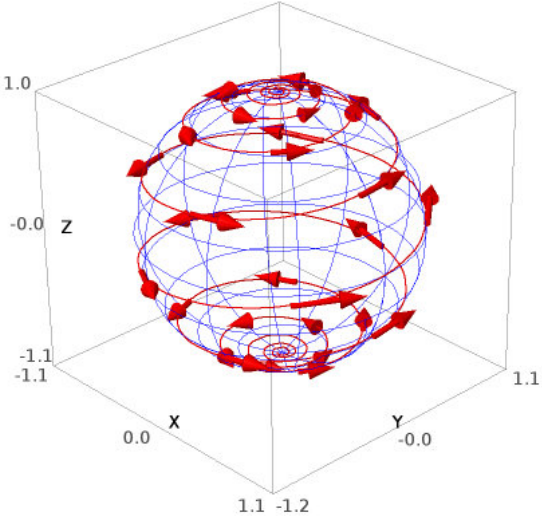
Vector frame associated with the stereographic coordinates (x, y) from the North pole

- $\frac{\partial}{\partial x}$
- $\frac{\partial}{\partial y}$

← picture obtained via the function
`VectorField.plot()`

See the worksheet at
http://sagemanifolds.obspm.fr/examples.html

The 2-sphere example

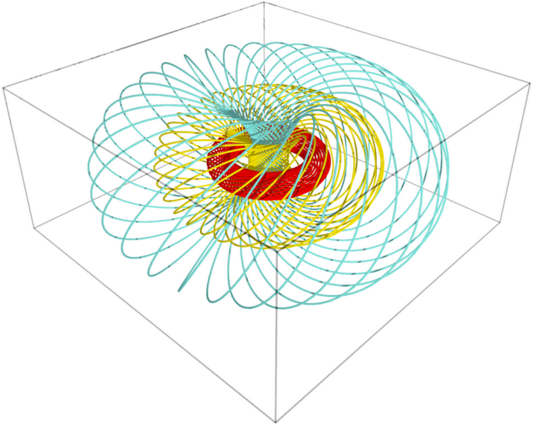


A curve in $\mathbb{S}^2$: a loxodrome and its tangent vector field

← picture obtained via the functions `DifferentiableCurve.plot()` and `VectorField.plot()`

See the worksheet at http://sagemanifolds.obspm.fr/examples.html

The 3-sphere example

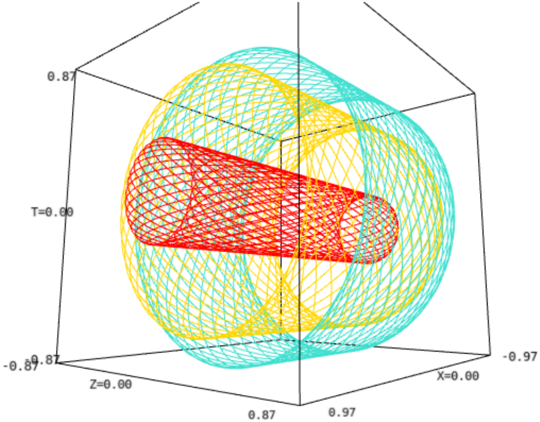


Some fibers of the **Hopf fibration** of $\mathbb{S}^3$ viewed in stereographic coordinates

← picture obtained via the function `DifferentiableCurve.plot()`

See the worksheet at http://nbviewer.jupyter.org/github/sagemanifolds/SageManifolds/blob/master/Worksheets/v1.0/SM_sphere_S3_Hopf.ipynb

The 3-sphere example

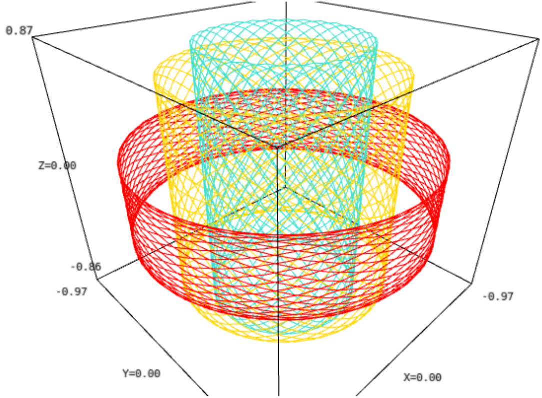


The same fibers but viewed in the Cartesian coordinates (T, X, Y) of $\mathbb{R}^4$ via the canonical embedding $\mathbb{S}^3 \to \mathbb{R}^4$

← picture obtained via the function DifferentiableCurve.plot()

See the worksheet at http://nbviewer.jupyter.org/github/sagemanifolds/SageManifolds/blob/master/Worksheets/v1.0/SM_sphere_S3_Hopf.ipynb

The 3-sphere example

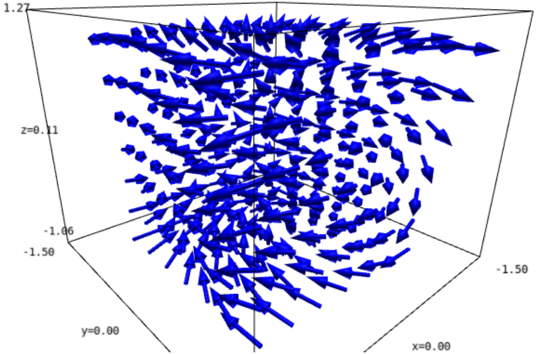


Again the same fibers but viewed in the Cartesian coordinates (X, Y, Z) of $\mathbb{R}^4$ via the canonical embedding $\mathbb{S}^3 \to \mathbb{R}^4$

$\leftarrow$ picture obtained via the function `DifferentiableCurve.plot()`

See the worksheet at http://nbviewer.jupyter.org/github/sagemanifolds/SageManifolds/blob/master/Worksheets/v1.0/SM_sphere_S3_Hopf.ipynb

The 3-sphere example

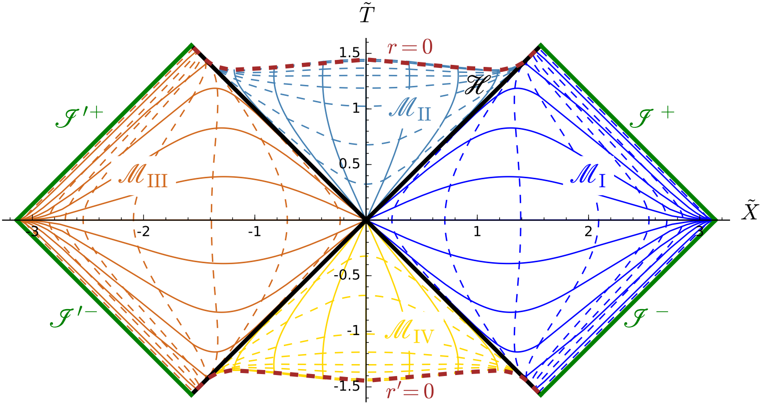


One of the vector fields of a left-invariant global vector frame of $\mathbb{S}^3$, viewed in stereographic coordinates

← picture obtained via the function `VectorField.plot()`

See the worksheet at http://nbviewer.jupyter.org/github/sagemanifolds/SageManifolds/blob/master/Worksheets/v1.0/SM_sphere_S3_vectors.ipynb

Charts on Schwarzschild spacetime
The Carter-Penrose diagram



Two charts of standard Schwarzschild-Droste coordinates (t, r, θ, φ) plotted in terms of Frolov-Novikov compactified coordinates $(\tilde{T}, \tilde{X}, \theta, \varphi)$; see the worksheet at http://luth.obspm.fr/~luthier/gourgoulhon/bh16/sage.html

Outline

Summary

SageManifolds: extends the modern computer algebra system SageMath towards differential geometry and tensor calculus

- http://sagemanifolds.obspm.fr/
- free software (GPL), as SageMath
- $\sim$ 65,000 lines of Python code (including comments and doctests)
- submitted to SageMath community as a sequence of 14 tickets
  $\rightarrow$ first ticket accepted in March 2015,
  the 14th one in Nov. 2016
- 5 developers, 3 reviewers

SageManifolds 1.0 released on 11 Jan. 2017 and fully included in SageMath 7.5

Current status

Already present (v1.0):

- topological manifolds: charts, open subsets, maps between manifolds, scalar fields
- differentiable manifolds: tangent spaces, vector frames, tensor fields, curves, pullback and pushforward operators
- standard tensor calculus (tensor product, contraction, symmetrization, etc.), even on non-parallelizable manifolds
- taking into account any monoterm tensor symmetry
- exterior calculus (wedge product, exterior derivative, Hodge duality)
- Lie derivatives of tensor fields
- affine connections (curvature, torsion)
- pseudo-Riemannian metrics
- some plotting capabilities (charts, points, curves, vector fields)
- parallelization (on tensor components) of CPU demanding computations, via the Python library `multiprocessing`

Current status

Future prospects:

- extrinsic geometry of pseudo-Riemannian submanifolds
- computation of geodesics (numerical integration via SageMath/GSL or `Gyoto`)
- integrals on submanifolds
- more graphical outputs
- more functionalities: symplectic forms, fibre bundles, spinors, variational calculus, etc.
- connection with numerical relativity: using SageMath to explore numerically-generated spacetimes

Current status

Future prospects:

- extrinsic geometry of pseudo-Riemannian submanifolds
- computation of geodesics (numerical integration via SageMath/GSL or `Gyoto`)
- integrals on submanifolds
- more graphical outputs
- more functionalities: symplectic forms, fibre bundles, spinors, variational calculus, etc.
- connection with numerical relativity: using SageMath to explore numerically-generated spacetimes

Want to join the project or simply to stay tuned?

visit http://sagemanifolds.obspm.fr/
(download, documentation, example worksheets, mailing list)