Tensor calculus with open-source software: the SageManifolds project

Tensor calculus with open-source software: the SageManifolds project

Eric Gourgoulhon[1], **Michał Bejger**[2], **Marco Mancini**[1]

[1] Laboratoire Univers et Théories, UMR 8102 du CNRS, Observatoire de Paris, Université Paris Diderot, 92190 Meudon, France

[2] Centrum Astronomiczne im. M. Kopernika, ul. Bartycka 18, 00-716 Warsaw, Poland

E-mail: eric.gourgoulhon@obspm.fr, bejger@camk.edu.pl, marco.mancini@obspm.fr

Abstract. The SageManifolds project aims at extending the mathematics software system Sage towards differential geometry and tensor calculus. Like Sage, SageManifolds is free, open-source and is based on the Python programming language. We discuss here some details of the implementation, which relies on Sage's parent/element framework, and present a concrete example of use.

1. Introduction

Computer algebra for general relativity (GR) has a long history, which started almost as soon as computer algebra itself in the 1960s. The first GR program was GEOM, written by J.G. Fletcher in 1965 [1]. Its main capability was to compute the Riemann tensor of a given metric. In 1969, R.A. d'Inverno developed ALAM (for *Atlas Lisp Algebraic Manipulator*) and used it to compute the Riemann and Ricci tensors of the Bondi metric. According to [2], the original calculations took Bondi and collaborators 6 months to finish, while the computation with ALAM took 4 minutes and yielded the discovery of 6 errors in the original paper. Since then, numerous packages have been developed: the reader is referred to [3] for a review of computer algebra systems for GR prior to 2002, and to [4] for a more recent review focused on tensor calculus. It is also worth to point out the extensive list of tensor calculus packages maintained by J. M. Martin-Garcia at [5].

2. Software for differential geometry

Software packages for differential geometry and tensor calculus can be classified in two categories:

(i) Applications atop some general purpose computer algebra system. Notable examples are the xAct suite [6] and Ricci [7], both running atop Mathematica, DifferentialGeometry [8] integrated into Maple, GRTensorII [9] atop Maple and Atlas 2 [10] for Mathematica and Maple.

(ii) Standalone applications. Recent examples are Cadabra (field theory) [11], SnapPy (topology and geometry of 3-manifolds) [12] and Redberry (tensors) [13].

All applications listed in the second category are free software. In the first category, xAct and Ricci are also free software, but they require a proprietary product, the source code of which is closed (Mathematica).

As far as tensor calculus is concerned, the above packages can be distinguished by the type of computation that they perform: abstract calculus (xAct/xTensor, Ricci, Cadabra, Redberry), or component calculus (xAct/xCoba, DifferentialGeometry, GRTensorII, Atlas 2). In the first category, tensor operations such as contraction or covariant differentiation are performed by manipulating the indices themselves rather than the components to which they correspond. In the second category, vector frames are explicitly introduced on the manifold and tensor operations are carried out on the components in a given frame.

3. An overview of Sage

Sage [14] is a free, open-source mathematics software system, which is based on the Python programming language. It makes use of over 90 open-source packages, among which are Maxima and Pynac (symbolic calculations), GAP (group theory), PARI/GP (number theory), Singular (polynomial computations), and matplotlib (high quality 2D figures). Sage provides a uniform Python interface to all these packages; however, Sage is much more than a mere interface: it contains a large and increasing part of original code (more than 750,000 lines of Python and Cython, involving 5344 classes). Sage was created in 2005 by W. Stein [15] and since then its development has been sustained by more than a hundred researchers (mostly mathematicians). Very good introductory textbooks about Sage are [16, 17, 18].

Apart from the syntax, which is based on a popular programming language and not a custom script language, a difference between Sage and, e.g., Maple or Mathematica is the usage of the *parent/element pattern*. This framework more closely reflects actual mathematics. For instance, in Mathematica, all objects are trees of symbols and the program is essentially a set of sophisticated rules to manipulate symbols. On the contrary, in Sage each object has a given type (i.e. is an instance of a given Python class[1]), and one distinguishes *parent* types, which model mathematical sets with some structure (e.g. algebraic structure), from *element* types, which model set elements. Moreover, each parent belongs to some dynamically generated class that encodes informations about its *category*, in the mathematical sense of the word (see [19] for a discussion of Sage's category framework). Automatic conversion rules, called *coercions*, prior to a binary operation, e.g. $x + y$ with x and y having different parents, are implemented.

4. The SageManifolds project

4.1. Aim and scope

Sage is well developed in many areas of mathematics but very little exists for differential geometry and tensor calculus. One may mention differential forms defined on a fixed coordinate patch, implemented by J. Vankerschaver [20], and the 2-dimensional parametrized surfaces of the 3-dimensional Euclidean space recently added by M. Malakhaltsev and J. Vankerschaver [21].

The aim of SageManifolds [22] is to introduce smooth manifolds and tensor fields in Sage, with the following requirements: (i) one should be able to introduce various coordinate charts on a manifold, with the relevant transition maps; (ii) 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 `Manifold`, `Chart`, `TensorField` or `Metric`, to implement them within the parent/element pattern and to code mathematical operations as class methods. For instance the class `Manifold`, devoted to real smooth manifolds, is a parent class, i.e. it inherits from Sage's class `Parent`. On the other hand, the class devoted to manifold points, `ManifoldPoint`, is an element class and therefore inherits from Sage's class `Element`. This is illustrated by the inheritance diagram of Fig. 1. In this

[1] Let us recall that within an object-oriented programming language (as Python), a *class* is a structure to declare and store the properties common to a set of objects. These properties are data (called *attributes* or *state variables*) and functions acting on the data (called *methods*). A specific realization of an object within a given class is called an *instance* of that class.

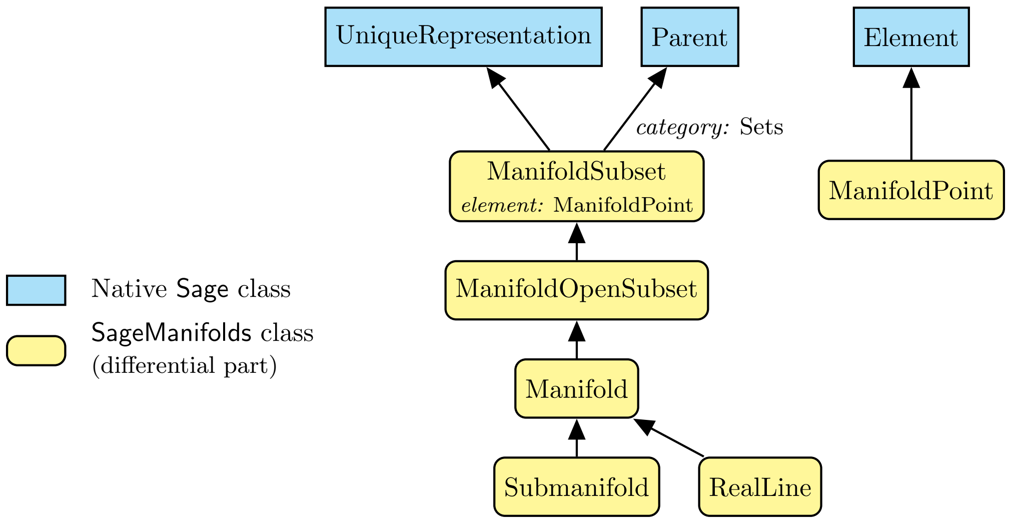

Figure 1. Python classes for smooth manifolds (`Manifold`), generic subsets of them (`ManifoldSubset`), open subsets of them (`ManifoldOpenSubset`) and points on them (`ManifoldPoint`).

diagram, each class at the base of some arrow is a subclass (also called *derived class*) of the class at the arrowhead. Note however that the actual type of a parent is a dynamically generated class taking into account the mathematical category to which it belongs. For instance, the actual type of a smooth manifold is not `Manifold`, but a subclass of it named `Manifold_with_category`, reflecting the fact that `Manifold` is declared in the category of `Sets`[2]. Note also that the class `Manifold` inherits from Sage's class `UniqueRepresentation`, which ensures that there is a unique manifold instance for a given dimension and given name.

4.2. Implementation of charts

Given a smooth manifold $\mathcal{M}$ of dimension n, a coordinate chart on some open subset $U \subset \mathcal{M}$ is implemented in SageManifolds via the class `Chart`, whose main data is a n-tuple of Sage symbolic variables $(x_1, \ldots, x_n)$, each of them representing a coordinate. In general, more than one (regular) chart is required to cover the entire manifold. For instance, at least 2 charts are necessary for the n-dimensional sphere $\mathbb{S}^n$ $(n \geq 1)$ and the torus $\mathbb{T}^2$ and 3 charts for the real projective plane $\mathbb{RP}^2$ (see Fig. 6 below). Accordingly, SageManifolds allows for an arbitrary number of charts. To fully specify the manifold, one shall also provide the transition maps (changes of coordinates) on overlapping chart domains (SageManifolds class `CoordChange`).

4.3. Implementation of scalar fields

A *scalar field* on manifold $\mathcal{M}$ is a smooth mapping

$$\begin{array}{rccl} f: & U \subset \mathcal{M} & \longrightarrow & \mathbb{R} \\ & p & \longmapsto & f(p), \end{array} \tag{1}$$

[2] A tighter category would be topological spaces, but such a category has been not implemented in Sage yet.

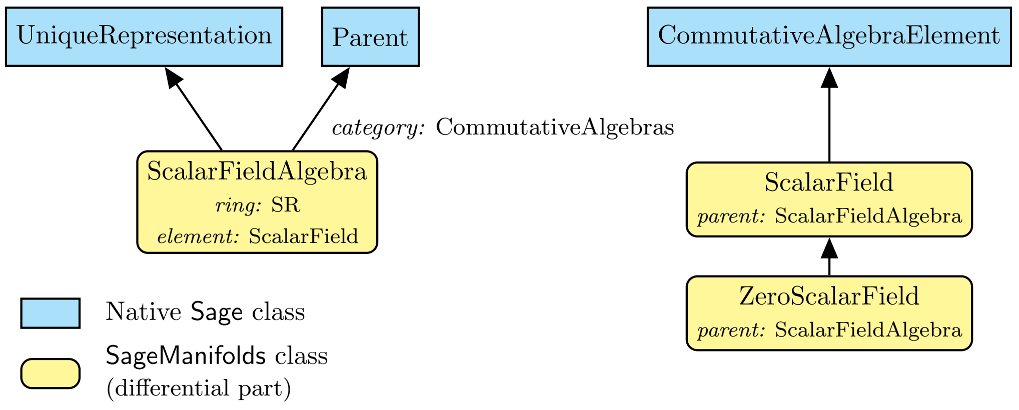
UniqueRepresentation
Parent
CommutativeAlgebraElement
category: CommutativeAlgebras
ScalarFieldAlgebra
ring: SR
element: ScalarField
ScalarField
parent: ScalarFieldAlgebra
ZeroScalarField
parent: ScalarFieldAlgebra
Native Sage class
SageManifolds class (differential part)

Figure 2. Python classes for scalar fields on a manifold.

where U is some open subset of $\mathcal{M}$. A scalar field has different coordinate representations F, $\hat{F}$, etc. in different charts X, $\hat{X}$, etc. defined on U:

$$f(p) = F(\underbrace{x^1, \ldots, x^n}_{\substack{\text{coord. of } p\\ \text{in chart } X}}) = \hat{F}(\underbrace{\hat{x}^1, \ldots, \hat{x}^n}_{\substack{\text{coord. of } p\\ \text{in chart } \hat{X}}}) = \ldots \tag{2}$$

These representations are stored in some attribute of the class `ScalarField`, namely a Python dictionary[3] whose keys are the various charts defined on U:

$$f.\_\mathtt{express} = \left\{X : F,\ \hat{X} : \hat{F}, \ldots\right\}. \tag{3}$$

Each representation F is an instance of the class `FunctionChart`, which resembles Sage native symbolic functions, but involves automatic simplifications in all arithmetic operations.

Given an open subset $U \subset \mathcal{M}$, the set $C^\infty(U)$ of scalar fields defined on U has naturally the structure of a commutative algebra over $\mathbb{R}$: it is clearly a vector space over $\mathbb{R}$ and it is endowed with a commutative ring structure by pointwise multiplication:

$$\forall f, g \in C^\infty(U), \quad \forall p \in U, \quad (f.g)(p) := f(p)g(p). \tag{4}$$

The algebra $C^\infty(U)$ is implemented in SageManifolds via the parent class `ScalarFieldAlgebra`, in the category `CommutativeAlgebras`. The corresponding element class is of course `ScalarField` (cf. Fig. 2).

4.4. Modules and free modules

Given an open subset $U \subset \mathcal{M}$, the set $\mathscr{X}(U)$ of all smooth vector fields defined on U has naturally the structure of a *module over the algebra* $C^\infty(U)$. Let us recall that a *module* is similar to a *vector space*, except that it is based on a *ring* (here $C^\infty(U)$) instead of a *field* (usually $\mathbb{R}$ or $\mathbb{C}$ in physical applications). Of course, every vector space is a module, since every

[3] A *dictionary*, also known as *associative array*, is a data structure that generalizes the concept of array in the sense that the key to access to some element is not restricted to an integer or a tuple of integers.

field is a ring. There is an important difference though: every vector space has a basis (as a consequence of the axiom of choice), while a module does not necessarily have any. When it possesses one, it is called a *free module*. Moreover, if the module's base ring is commutative, it can be shown that all bases have the same cardinality, which is called the *rank* of the module (for vector spaces, which are free modules, the word *dimension* is used instead of *rank*).

If $\mathscr{X}(U)$ is a free module (examples are provided in Sec. 4.5 below), a basis of it is nothing but a *vector frame* $(\boldsymbol{e}_a)_{1\leq a\leq n}$ on U (often called a *tetrad* in the context of 4-dimensional GR):

$$\forall \boldsymbol{v} \in \mathscr{X}(U), \quad \boldsymbol{v} = v^a \boldsymbol{e}_a, \quad \text{with } v^a \in C^\infty(U). \tag{5}$$

The rank of $\mathscr{X}(U)$ is thus n, i.e. the manifold's dimension[4]. At any point $p \in U$, Eq. (5) gives birth to an identity in the tangent vector space $T_p\mathcal{M}$:

$$\boldsymbol{v}(p) = v^a(p)\ \boldsymbol{e}_a(p), \quad \text{with } v^a(p) \in \mathbb{R}, \tag{6}$$

which means that the set $(\boldsymbol{e}_a(p))_{1\leq a\leq n}$ is a basis of $T_p\mathcal{M}$. Note that if U is covered by a chart $(x^a)_{1\leq a\leq n}$, then $(\partial/\partial x^a)_{1\leq a\leq n}$ is a vector frame on U, usually called *coordinate frame* or *natural basis*. Note also that, being a vector space over $\mathbb{R}$, the tangent space $T_p\mathcal{M}$ represents another kind of free module which occurs naturally in the current context.

It turns out that so far only free modules *with a distinguished basis* were implemented in Sage. This means that, given a free module M of rank n, all calculations refer to a single basis of M. This amounts to identifying M with R^n, where R is the ring over which M is based. This is unfortunately not sufficient for dealing with smooth manifolds in a coordinate-independent way. For instance, there is no canonical isomorphism between $T_p\mathcal{M}$ and $\mathbb{R}^n$ when no coordinate system is privileged in the neighborhood of p. Therefore we have started a pure algebraic part of SageManifolds to implement generic free modules, with an arbitrary number of bases, none of them being distinguished. This resulted in (i) the parent class `FiniteRankFreeModule`, within Sage's category `Modules`, and (ii) the element class `FiniteRankFreeModuleElement`. Then both classes `VectorFieldFreeModule` (for $\mathscr{X}(U)$, when it is a free module) and `TangentSpace` (for $T_p\mathcal{M}$) inherit from `FiniteRankFreeModule` (see Fig. 3).

4.5. Implementation of vector fields

Ultimately, in SageManifolds, vector fields are described by their components with respect to various vector frames, according to Eq. (5), but without any vector frame being privileged, leaving the freedom to select one to the user, as well as to change coordinates. A key point is that not every manifold admits a global vector frame. A manifold $\mathcal{M}$, or more generally an open subset $U \subset \mathcal{M}$, that admits a global vector frame is called *parallelizable*. Equivalently, $\mathcal{M}$ is parallelizable if, and only if, $\mathscr{X}(\mathcal{M})$ is a free module. In terms of tangent bundles, parallelizable manifolds are those for which the tangent bundle is trivial: $T\mathcal{M} \simeq \mathcal{M} \times \mathbb{R}^n$. Examples of parallelizable manifolds are [23]

- the Cartesian space $\mathbb{R}^n$ for $n = 1, 2, \ldots$,
- the circle $\mathbb{S}^1$,
- the torus $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$,
- the sphere $\mathbb{S}^3 \simeq \mathrm{SU}(2)$, as any Lie group,
- the sphere $\mathbb{S}^7$,
- any orientable 3-manifold (Steenrod theorem [24]).

[4] Note that the dimensionality of $\mathscr{X}(U)$ depends of the adopted structure: as a vector space over $\mathbb{R}$, the dimension of $\mathscr{X}(U)$ is infinite, while as a free module over $C^\infty(U)$, $\mathscr{X}(U)$ has a finite rank. Note also that if $\mathscr{X}(U)$ is not free (i.e. no global vector frame exists on U), the notion of rank is meaningless.

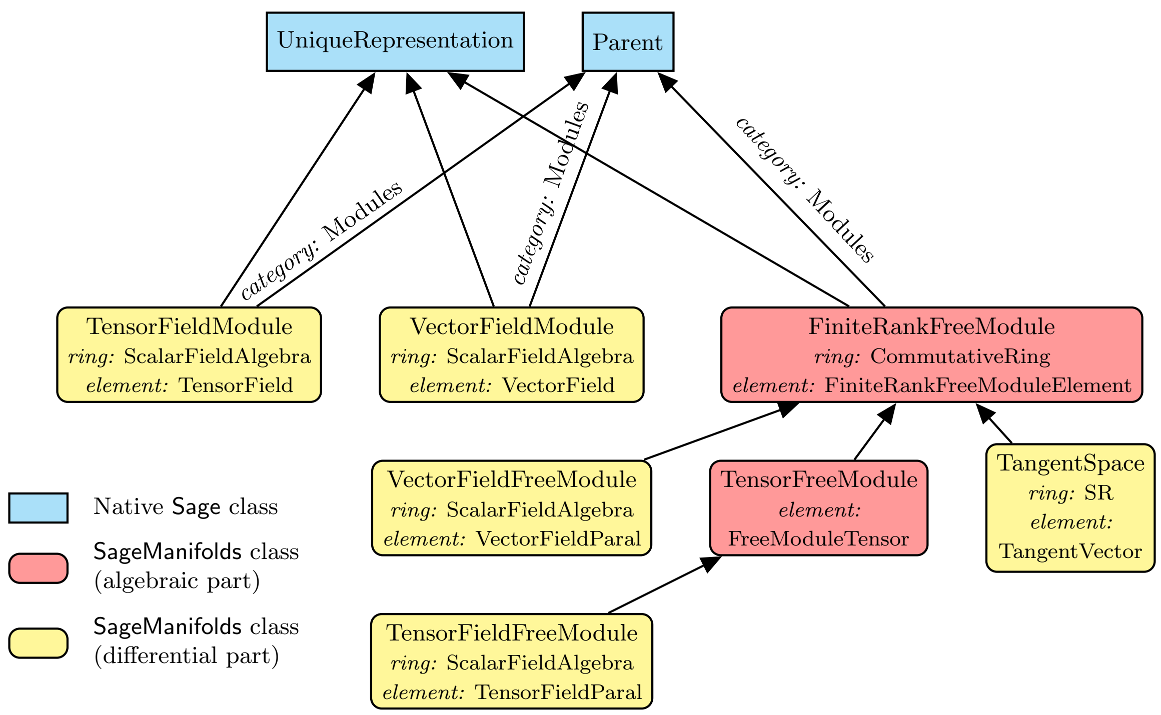

Figure 3. Python classes for modules. For each of them, the class of the base ring is indicated, as well as the class for the elements.

On the other hand, examples of non-parallelizable manifolds are

- the sphere $\mathbb{S}^2$ (as a consequence of the hairy ball theorem), as well as any sphere $\mathbb{S}^n$ with $n \notin \{1, 3, 7\}$,
- the real projective plane $\mathbb{RP}^2$.

Actually, "most" manifolds are non-parallelizable. As noticed above, if a manifold is covered by a single chart, it is parallelizable (the prototype being $\mathbb{R}^n$). But the reverse is not true: $\mathbb{S}^1$ and $\mathbb{T}^2$ are parallelizable and require at least two charts to cover them.

If the manifold $\mathcal{M}$ is not parallelizable, we assume that it can be covered by a finite number N of parallelizable open subsets U_i $(1 \leq i \leq N)$. In particular, this holds if $\mathcal{M}$ is compact, for any compact manifold admits a finite atlas. We then consider the restrictions of vector fields to the U_i's. For each i, $\mathscr{X}(U_i)$ is a free module of rank $n = \dim \mathcal{M}$ and is implemented in SageManifolds as an instance of `VectorFieldFreeModule` (cf. Sec. 4.4 and Figs. 3 and 4). Each vector field $\boldsymbol{v} \in \mathscr{X}(U_i)$ has different sets 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 [cf. Eq. (5)]. They are stored as a Python dictionary whose keys are the vector frames:

$$\boldsymbol{v}.\texttt{\_components} = \{(\boldsymbol{e}) : (v^a),\ (\hat{\boldsymbol{e}}) : (\hat{v}^a), \ldots\}\,. \tag{7}$$

4.6. Implementation of tensor fields

The implementation of tensor fields in SageManifolds follows the strategy adopted for vector fields. Consider for instance a tensor field $\boldsymbol{T}$ of type (1,1) on the manifold $\mathcal{M}$. It can

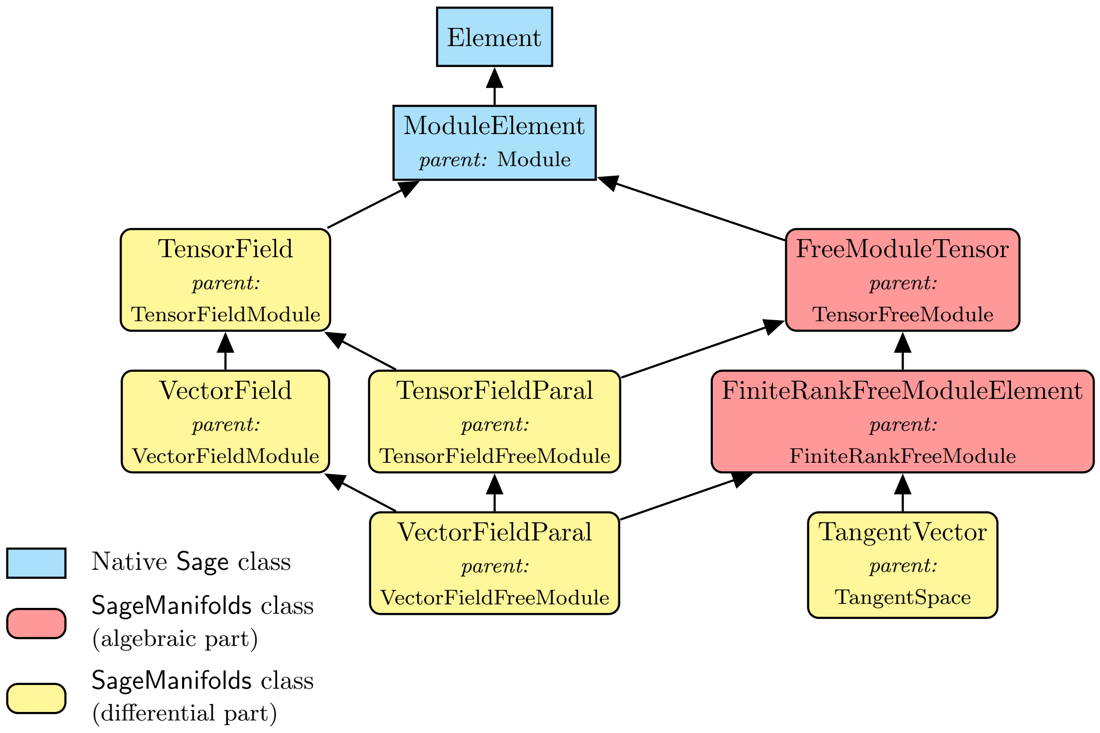

Figure 4. Python classes implementing tensors and tensor fields.

be represented by components $T^a{}_b$ only on a parallelizable open subset $U \subset \mathcal{M}$, since the decomposition

$$\boldsymbol{T}|_U = T^a{}_b \, \boldsymbol{e}_a \otimes \boldsymbol{e}^b, \tag{8}$$

which defines $T^a{}_b$, is meaningful only when a vector frame $(\boldsymbol{e}_a)$ exists[5]. Therefore, one first decomposes the tensor field $\boldsymbol{T}$ into its restrictions $\boldsymbol{T}|_{U_i}$ on parallelizable open subsets of $\mathcal{M}$, U_i $(1 \leq i \leq N)$ and then considers the components on various vector frames on each subset U_i. For each vector frame $(\boldsymbol{e}_a)$, the set of components $(T^a{}_b)$ is stored in a devoted class (named `Components`), which takes into account all the tensor monoterm symmetries: only non-redundant components are stored, the other ones being deduced by (anti)symmetry. This is illustrated in Fig. 5, which depicts the internal storage of tensor fields in SageManifolds. Note that each component $T^a{}_b$ is a scalar field on U_i, according to the formula

$$T^a{}_b = \boldsymbol{T}(\boldsymbol{e}^a, \boldsymbol{e}_b). \tag{9}$$

Accordingly, the penultimate level of Fig. 5 corresponds to the scalar field storage, as described by (3). The last level is constituted by Sage's symbolic expressions (class `Expression`).

5. Current status of SageManifolds

5.1. Functionalities

At present (version 0.6), the functionalities included in SageManifolds are as follows:

[5] Using standard notation, in Eq. (8), $(\boldsymbol{e}^b)$ stands for the coframe dual to $(\boldsymbol{e}_a)$

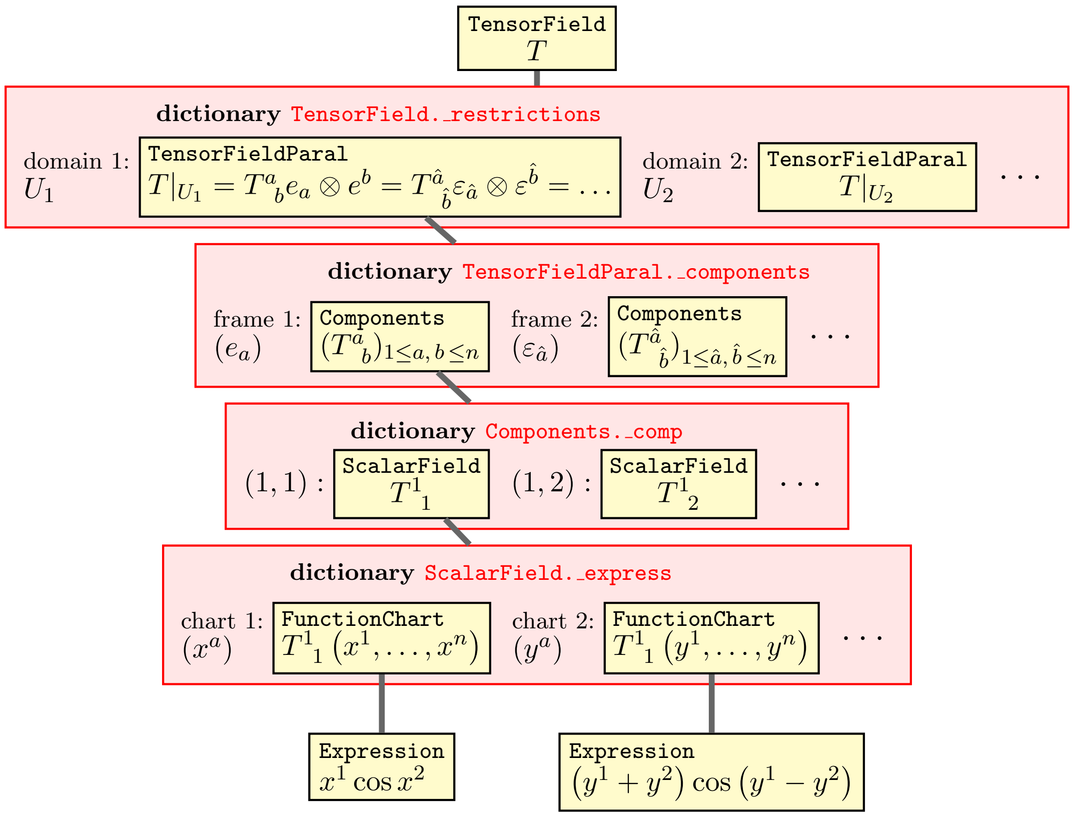
TensorField
T
dictionary TensorField._restrictions
domain 1: U_1
TensorFieldParal
$T|_{U_1} = T^a{}_b e_a \otimes e^b = T^{\hat{a}}{}_{\hat{b}} \varepsilon_{\hat{a}} \otimes \varepsilon^{\hat{b}} = \ldots$
domain 2: U_2
TensorFieldParal
$T|_{U_2}$
$\ldots$
dictionary TensorFieldParal._components
frame 1: (e_a)
Components
$(T^a{}_b)_{1\leq a,b\leq n}$
frame 2: $(\varepsilon_{\hat{a}})$
Components
$(T^{\hat{a}}{}_{\hat{b}})_{1\leq \hat{a},\hat{b}\leq n}$
$\ldots$
dictionary Components._comp
$(1,1)$:
ScalarField
$T^1{}_1$
$(1,2)$:
ScalarField
$T^1{}_2$
$\ldots$
dictionary ScalarField._express
chart 1: (x^a)
FunctionChart
$T^1{}_1\left(x^1,\ldots,x^n\right)$
chart 2: (y^a)
FunctionChart
$T^1{}_1\left(y^1,\ldots,y^n\right)$
$\ldots$
Expression
$x^1 \cos x^2$
Expression
$\left(y^1+y^2\right)\cos\left(y^1-y^2\right)$

Figure 5. Storage of tensor fields in SageManifolds. Each red box represents a Python dictionary; the dictionary values are depicted by yellow boxes, the keys being indicated at the left of each box.

- maps between manifolds and pullback operator,
- submanifolds and pushforward operator,
- standard tensor calculus (tensor product, contraction, symmetrization, etc.), even on non-parallelizable manifolds,
- arbitrary monoterm tensor symmetries,
- exterior calculus (wedge product and exterior derivative, Hodge duality),
- Lie derivatives along a vector field,
- affine connections (curvature, torsion),
- pseudo-Riemannian metrics (Levi-Civita connection, Weyl tensor),
- graphical display of charts.

5.2. Parallelization
To improve the reactivity of SageManifolds and take advantage of multicore processors, some tensorial operations are performed by parallel processes. The parallelization is implemented by means of the Python library `multiprocessing`, via the built-in Sage decorator `@parallel`. Using it permits to define a function that is run on different sub-processes. If n processes are

used, given a function and a list of arguments for it, any process will call the function with an element of the list, one at time, spanning all the list.

Currently[6], the parallelized operations are tensor algebra, tensor contractions, computation of the connection coefficient and computation of Riemann tensor.

The parallelization of an operation is achieved by first creating a function which computes the required operation on a subset of the components of a tensor; second, by creating a list of $2n$ (twice the number of used processes) arguments for this function. Then applying this function to the input list, the calculation is performed in parallel. At the end of the computation a fourth phase is needed to retrieve the results. The choice to divide the work in $2n$ is a compromise between the load balancing and the cost of creating multiple processes. The number of processors to be used in the parallelization can be controlled by the user.

6. SageManifolds at work: Kerr spacetime and Simon-Mars tensor

We give hereafter a short illustration of SageManifolds focused on tensor calculus in 4-dimensional GR. Another example, to be found at [25], is based on the manifold $\mathbb{S}^2$ and focuses more on the use of multiple charts and on the treatment of non-parallelizable manifolds. Yet another example illustrates some graphical capabilities of SageManifolds: Figure 6 shows the famous immersion of the real projective plane $\mathbb{RP}^2$ into the Euclidean space $\mathbb{R}^3$ known as the *Boy surface*. This figure has been obtained by means of the method `plot()` applied to three coordinate charts covering $\mathbb{RP}^2$, the definition of which is related to the interpretation of $\mathbb{RP}^2$ as the set of straight lines Δ through the origin of $\mathbb{R}^3$: (i) in red, the chart X_1 covering the open subset of $\mathbb{RP}^2$ defined by all lines Δ that are not parallel to the plane $z = 0$, the coordinates of X_1 being the coordinates (x, y) of the intersection of the considered line Δ with the plane $z = 1$; (ii) in green, the chart X_2 covering the open subset defined by all lines Δ that are not parallel to the plane $x = 0$, the coordinates of X_2 being the coordinates (y, z) of intersection with the plane $x = 1$; (iii) in blue, the chart X_3 covering the open subset defined by all lines Δ that are not parallel to the plane $y = 0$, the coordinates of X_2 being the coordinates (z, x) of intersection with the plane $y = 1$. Figure 6 actually shows the coordinate grids of these three charts through the Apéry map [26], which realizes an immersion of $\mathbb{RP}^2$ into $\mathbb{R}^3$. This example, as many others, can be found at [25].

Let us consider a 4-dimensional spacetime, i.e. a smooth 4-manifold $\mathcal{M}$ endowed with a Lorentzian metric $\boldsymbol{g}$. We assume that $(\mathcal{M}, \boldsymbol{g})$ is stationary and denote by $\boldsymbol{\xi}$ the corresponding Killing vector field. The *Simon-Mars tensor w.r.t.* $\boldsymbol{\xi}$ is then the type-(0,3) tensor field $\boldsymbol{S}$ defined by [27]

$$S_{\alpha\beta\gamma} := 4\mathcal{C}_{\mu\alpha\nu[\beta}\, \xi^\mu \xi^\nu\, \sigma_{\gamma]} + \gamma_{\alpha[\beta}\, \mathcal{C}_{\gamma]\rho\mu\nu}\, \xi^\rho\, \mathcal{F}^{\mu\nu}, \tag{10}$$

where

- $\gamma_{\alpha\beta} := \lambda\, g_{\alpha\beta} + \xi_\alpha \xi_\beta$, with $\lambda := -\xi_\mu \xi^\mu$;
- $\mathcal{C}_{\alpha\beta\mu\nu} := C_{\alpha\beta\mu\nu} + \frac{i}{2}\epsilon^{\rho\sigma}{}_{\mu\nu}\, C_{\alpha\beta\rho\sigma}$, with $C^\alpha{}_{\beta\mu\nu}$ being the Weyl curvature tensor and $\epsilon_{\alpha\beta\mu\nu}$ the Levi-Civita volume 4-form;
- $\mathcal{F}_{\alpha\beta} := F_{\alpha\beta} + i\, {}^*F_{\alpha\beta}$, with $F_{\alpha\beta} := \nabla_\alpha \xi_\beta$ (Killing 2-form) and ${}^*F_{\alpha\beta} := \frac{1}{2}\epsilon^{\mu\nu}{}_{\alpha\beta} F_{\mu\nu}$;
- $\sigma_\alpha := 2\mathcal{F}_{\mu\alpha}\xi^\mu$ (Ernst 1-form).

The Simon-Mars tensor provides a nice characterization of Kerr spacetime, according the following theorem proved by Mars [27]: if $\boldsymbol{g}$ satisfies the vacuum Einstein equation and $(\mathcal{M}, \boldsymbol{g})$ contains a stationary asymptotically flat end $\mathcal{M}^\infty$ such that $\boldsymbol{\xi}$ tends to a time translation at infinity in $\mathcal{M}^\infty$ and the Komar mass of $\boldsymbol{\xi}$ in $\mathcal{M}^\infty$ is non-zero, then $\boldsymbol{S} = 0$ if, and only if, $(\mathcal{M}, \boldsymbol{g})$ is locally isometric to a Kerr spacetime.

[6] in the development version of SageManifolds; this will become available in version 0.7 of the stable release.

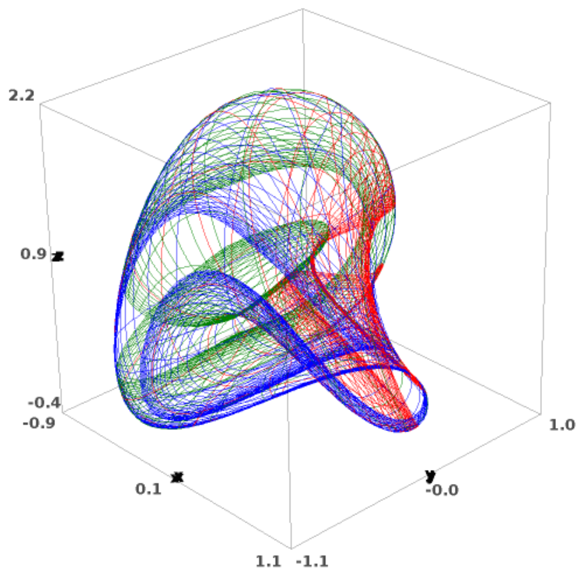

Figure 6. Boy surface depicted via the grids of 3 coordinate charts covering $\mathbb{RP}^2$ (see the text for the color code).

In what follows, we use SageManifolds to compute the Simon-Mars tensor according to formula (10) for the Kerr metric and check that we get zero (the "if" part of the above theorem). The corresponding worksheet can be downloaded from
http://sagemanifolds.obspm.fr/examples/html/SM_Simon-Mars_Kerr.html.
For the sake of clarity, let us recall that, as an object-oriented language, Python (and hence Sage) makes use of the following postfix notation:

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

In a functional language, this would correspond to `result = function(object,arguments)`. For instance, the Riemann tensor of a metric $\boldsymbol{g}$ is obtained as `riem = g.riemann()` (in this case, there is no extra argument, hence the empty parentheses). With this in mind, let us proceed with the computation by means of SageManifolds. In the text below, the blue color denotes the outputs as they appear in the Sage notebook (note that all outputs are automatically LaTeX-formatted by Sage).

The first step is to declare the Kerr spacetime (or more precisely the part of the Kerr spacetime covered by Boyer-Lindquist coordinates) as a 4-dimensional manifold:

```
M = Manifold(4, 'M', latex_name=r'\mathcal{M}')
print M
```

```
4-dimensional manifold 'M'
```

The standard Boyer-Lindquist coordinates (t, r, θ, ϕ) are introduced by declaring a chart X on $\mathcal{M}$, via the method `chart()`, the argument of which is a string expressing the coordinates names, their ranges (the default is $(-\infty, +\infty)$) and their LaTeX symbols:

```
X.<t,r,th,ph> = M.chart('t r:(0,+oo) th:(0,pi):\\theta ph:(0,2*pi):\\phi')
print X ; X
```

chart (M, (t, r, th, ph))
$(\mathcal{M}, (t, r, \theta, \phi))$

We define next the Kerr metric $\boldsymbol{g}$ by setting its components in the coordinate frame associated with Boyer-Lindquist coordinates. Since the latter is the current manifold's default frame (being the only one defined at this stage), we do not need to specify it when referring to the components by their indices:

```
g = M.lorentz_metric('g')
m = var('m') ; a = var('a')
rho2 = r^2 + (a*cos(th))^2
Delta = r^2 -2*m*r + a^2
g[0,0] = -(1-2*m*r/rho2)
g[0,3] = -2*a*m*r*sin(th)^2/rho2
g[1,1], g[2,2] = rho2/Delta, rho2
g[3,3] = (r^2+a^2+2*m*r*(a*sin(th))^2/rho2)*sin(th)^2
g.view()
```

$$g = \left(-\frac{a^2\cos(\theta)^2 - 2\,mr + r^2}{a^2\cos(\theta)^2 + r^2}\right)\mathrm{d}t\otimes\mathrm{d}t + \left(-\frac{2\,amr\sin(\theta)^2}{a^2\cos(\theta)^2 + r^2}\right)\mathrm{d}t\otimes\mathrm{d}\phi + \left(\frac{a^2\cos(\theta)^2 + r^2}{a^2 - 2\,mr + r^2}\right)\mathrm{d}r\otimes\mathrm{d}r + \left(a^2\cos(\theta)^2 + r^2\right)\mathrm{d}\theta\otimes\mathrm{d}\theta + \left(-\frac{2\,amr\sin(\theta)^2}{a^2\cos(\theta)^2 + r^2}\right)\mathrm{d}\phi\otimes\mathrm{d}t + \left(\frac{2\,a^2mr\sin(\theta)^4 + \left(a^2r^2 + r^4 + \left(a^4 + a^2r^2\right)\cos(\theta)^2\right)\sin(\theta)^2}{a^2\cos(\theta)^2 + r^2}\right)\mathrm{d}\phi\otimes\mathrm{d}\phi$$

The Levi-Civita connection $\boldsymbol{\nabla}$ associated with $\boldsymbol{g}$ is obtained by the method `connection()`:

```
nab = g.connection() ; print nab
```

Levi-Civita connection 'nabla_g' associated with the Lorentzian metric 'g' on the 4-dimensional manifold 'M'

As a check, we verify that the covariant derivative of $\boldsymbol{g}$ with respect to $\boldsymbol{\nabla}$ vanishes identically:

```
nab(g).view()
```

$\nabla_g g = 0$

As mentionned above, the default vector frame on the spacetime manifold is the coordinate basis associated with Boyer-Lindquist coordinates:

```
M.default_frame() is X.frame()
```

True

```
X.frame()
```

$\left(\mathcal{M}, \left(\frac{\partial}{\partial t}, \frac{\partial}{\partial r}, \frac{\partial}{\partial \theta}, \frac{\partial}{\partial \phi}\right)\right)$

Let us consider the first vector field of this frame:

```
xi = X.frame()[0] ; xi
```

$\frac{\partial}{\partial t}$

```
print xi
```

vector field 'd/dt' on the 4-dimensional manifold 'M'

The 1-form associated to it by metric duality is

```
xi_form = xi.down(g)
xi_form.set_name('xi_form', r'\underline{\xi}')
print xi_form ; xi_form.view()
```

1-form 'xi_form' on the 4-dimensional manifold 'M'

$$\underline{\xi} = \left(-\frac{a^2\cos(\theta)^2 - 2\,mr + r^2}{a^2\cos(\theta)^2 + r^2}\right)\mathrm{d}t + \left(-\frac{2\,amr\sin(\theta)^2}{a^2\cos(\theta)^2 + r^2}\right)\mathrm{d}\phi$$

Its covariant derivative is

```
nab_xi = nab(xi_form)
print nab_xi ; nab_xi.view()
```

tensor field 'nabla_g xi_form' of type (0,2) on the 4-dimensional manifold 'M'

$$\nabla_g\underline{\xi} = \left(\frac{a^2m\cos(\theta)^2 - mr^2}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}t\otimes\mathrm{d}r + \left(\frac{2\,a^2mr\cos(\theta)\sin(\theta)}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}t\otimes\mathrm{d}\theta +$$
$$\left(-\frac{a^2m\cos(\theta)^2 - mr^2}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}r\otimes\mathrm{d}t + \left(\frac{\left(a^3m\cos(\theta)^2 - amr^2\right)\sin(\theta)^2}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}r\otimes\mathrm{d}\phi +$$
$$\left(-\frac{2\,a^2mr\cos(\theta)\sin(\theta)}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}\theta\otimes\mathrm{d}t + \left(\frac{2\left(a^3mr + amr^3\right)\cos(\theta)\sin(\theta)}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}\theta\otimes\mathrm{d}\phi +$$
$$\left(-\frac{\left(a^3m\cos(\theta)^2 - amr^2\right)\sin(\theta)^2}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}\phi\otimes\mathrm{d}r + \left(-\frac{2\left(a^3mr + amr^3\right)\cos(\theta)\sin(\theta)}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}\phi\otimes\mathrm{d}\theta$$

Let us check that the Killing equation is satisfied:

```
nab_xi.symmetrize().view()
```

0

Equivalently, we check that the Lie derivative of the metric along $\boldsymbol{\xi}$ vanishes:

```
g.lie_der(xi).view()
```

0

Thanks to Killing equation, $\boldsymbol{\nabla}\underline{\boldsymbol{\xi}}$ is antisymmetric. We may therefore define a 2-form by $\boldsymbol{F} := -\boldsymbol{\nabla}\underline{\boldsymbol{\xi}}$. Here we enforce the antisymmetry by calling the method `antisymmetrize()` on nab_xi:

```
F = - nab_xi.antisymmetrize()
F.set_name('F')
print F ; F.view()
```

2-form 'F' on the 4-dimensional manifold 'M'

$$F = \left(-\frac{a^2m\cos(\theta)^2 - mr^2}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}t\wedge\mathrm{d}r + \left(-\frac{2\,a^2mr\cos(\theta)\sin(\theta)}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}t\wedge\mathrm{d}\theta +$$
$$\left(-\frac{\left(a^3m\cos(\theta)^2 - amr^2\right)\sin(\theta)^2}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}r\wedge\mathrm{d}\phi + \left(-\frac{2\left(a^3mr + amr^3\right)\cos(\theta)\sin(\theta)}{a^4\cos(\theta)^4 + 2\,a^2r^2\cos(\theta)^2 + r^4}\right)\mathrm{d}\theta\wedge\mathrm{d}\phi$$

The squared norm of the Killing vector is

```
lamb = - g(xi,xi)
lamb.set_name('lambda', r'\lambda')
print lamb ; lamb.view()
```

scalar field 'lambda' on the 4-dimensional manifold 'M'

$$\begin{array}{llcl}\lambda: & \mathcal{M} & \longrightarrow & \mathbb{R} \\ & (t,r,\theta,\phi) & \longmapsto & \frac{a^2\cos(\theta)^2-2\,mr+r^2}{a^2\cos(\theta)^2+r^2}\end{array}$$

Instead of invoking $\boldsymbol{g}(\boldsymbol{\xi},\boldsymbol{\xi})$, we could have evaluated λ by means of the 1-form $\underline{\boldsymbol{\xi}}$ acting on the vector field $\boldsymbol{\xi}$:

```
lamb == - xi_form(xi)
```

True

---

or, using index notation as $\lambda = -\xi_a \xi^a$:

```
lamb == - ( xi_form['_a']*xi['^a'] )
```

True

The Riemann curvature tensor associated with $\boldsymbol{g}$ is

```
Riem = g.riemann()
print Riem
```

tensor field 'Riem(g)' of type (1,3) on the 4-dimensional manifold 'M'

The component $R^0{}_{123} = R^t{}_{r\theta\phi}$ is

```
Riem[0,1,2,3]
```

$$-\frac{\left(a^7m-2\,a^5m^2r+a^5mr^2\right)\cos(\theta)\sin(\theta)^5+\left(a^7m+2\,a^5m^2r+6\,a^5mr^2-6\,a^3m^2r^3+5\,a^3mr^4\right)\cos(\theta)\sin(\theta)^3}{a^2r^6-2\,mr^7+r^8+(a^8-2\,a^6mr+a^6r^2)\cos(\theta)^6+3\,(a^6r^2-2\,a^4mr^3+a^4r^4)\cos(\theta)^4+3\,(a^4r^4-2\,a^2mr^5+a^2r^6)\cos(\theta)^2}$$
$$+\frac{2\left(a^7m-a^5mr^2-5\,a^3mr^4-3\,amr^6\right)\cos(\theta)\sin(\theta)}{a^2r^6-2\,mr^7+r^8+(a^8-2\,a^6mr+a^6r^2)\cos(\theta)^6+3\,(a^6r^2-2\,a^4mr^3+a^4r^4)\cos(\theta)^4+3\,(a^4r^4-2\,a^2mr^5+a^2r^6)\cos(\theta)^2}$$

Let us check that the Kerr metric is a vacuum solution of Einstein equation, i.e. that the Ricci tensor vanishes identically:

```
g.ricci().view()
```

$\mathrm{Ric}(g) = 0$

The Weyl conformal curvature tensor is

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

tensor field 'C(g)' of type (1,3) on the 4-dimensional manifold 'M'

Let us exhibit the component $C^0{}_{101} = C^t{}_{rtr}$:

```
C[0,1,0,1]
```

$$\frac{3\,a^4mr\cos(\theta)^4+3\,a^2mr^3+2\,mr^5-\left(9\,a^4mr+7\,a^2mr^3\right)\cos(\theta)^2}{a^2r^6-2\,mr^7+r^8+(a^8-2\,a^6mr+a^6r^2)\cos(\theta)^6+3\,(a^6r^2-2\,a^4mr^3+a^4r^4)\cos(\theta)^4+3\,(a^4r^4-2\,a^2mr^5+a^2r^6)\cos(\theta)^2}$$

To form the Simon-Mars tensor, we need the fully covariant form (type-(0,4) tensor) of the Weyl tensor (i.e. $C_{\alpha\beta\mu\nu} = g_{\alpha\sigma}C^{\sigma}{}_{\beta\mu\nu}$); we get it by lowering the first index with the metric:

```
Cd = C.down(g) ; print Cd
```

tensor field of type (0,4) on the 4-dimensional manifold 'M'

The (monoterm) symmetries of this tensor are those inherited from the Weyl tensor, i.e. the antisymmetry on the last two indices (position 2 and 3, the first index being at position 0):

```
Cd.symmetries()
```

no symmetry; antisymmetry: (2, 3)

Actually, `Cd` is also antisymmetric with respect to the first two indices (positions 0 and 1), as we can check:

```
Cd == Cd.antisymmetrize(0,1)
```

True

To take this symmetry into account explicitly, we set

```
Cd = Cd.antisymmetrize(0,1)
Cd.symmetries()
```

no symmetry; antisymmetries: [(0, 1), (2, 3)]

The starting point in the evaluation of Simon-Mars tensor is the self-dual complex 2-form associated with the Killing 2-form $\boldsymbol{F}$, i.e. the object $\boldsymbol{\mathcal{F}} := \boldsymbol{F} + i\,{}^*\boldsymbol{F}$, where ${}^*\boldsymbol{F}$ is the Hodge dual of $\boldsymbol{F}$:

```
FF = F + I * F.hodge_star(g)
FF.set_name('FF', r'\mathcal{F}')
print FF ; FF.view()
```

2-form 'FF' on the 4-dimensional manifold 'M'

$$\begin{aligned}\mathcal{F} = {} & \left(-\frac{a^{2} m \cos\left(\theta\right)^{2} + 2i \, a m r \cos\left(\theta\right) - m r^{2}}{a^{4} \cos\left(\theta\right)^{4} + 2 \, a^{2} r^{2} \cos\left(\theta\right)^{2} + r^{4}}\right) \mathrm{d} t\wedge \mathrm{d} r + \left(\frac{\left(i \, a^{3} m \cos\left(\theta\right)^{2} - 2 \, a^{2} m r \cos\left(\theta\right) - i \, a m r^{2}\right) \sin\left(\theta\right)}{a^{4} \cos\left(\theta\right)^{4} + 2 \, a^{2} r^{2} \cos\left(\theta\right)^{2} + r^{4}}\right) \mathrm{d} t\wedge \mathrm{d} \theta \\ & + \left(\frac{-4i \, a^{4} m^{2} r^{2} \cos\left(\theta\right) \sin\left(\theta\right)^{4} + \left(a^{3} m r^{4} - 2 \, a m^{2} r^{5} + a m r^{6} - \left(a^{7} m - 2 \, a^{5} m^{2} r + a^{5} m r^{2}\right) \cos\left(\theta\right)^{4}\right) \sin\left(\theta\right)^{2}}{a^{2} r^{6} - 2 \, m r^{7} + r^{8} + (a^{8} - 2 \, a^{6} m r + a^{6} r^{2}) \cos\left(\theta\right)^{6} + 3 \, (a^{6} r^{2} - 2 \, a^{4} m r^{3} + a^{4} r^{4}) \cos\left(\theta\right)^{4} + 3 \, (a^{4} r^{4} - 2 \, a^{2} m r^{5} + a^{2} r^{6}) \cos\left(\theta\right)^{2}}\right. \\ & \left. - \frac{\left(\left(2i \, a^{6} m r + 2i \, a^{4} m r^{3}\right) \cos\left(\theta\right)^{3} + \left(-4i \, a^{4} m^{2} r^{2} + 2i \, a^{4} m r^{3} - 4i \, a^{2} m^{2} r^{4} + 2i \, a^{2} m r^{5}\right) \cos\left(\theta\right)\right) \sin\left(\theta\right)^{2}}{a^{2} r^{6} - 2 \, m r^{7} + r^{8} + (a^{8} - 2 \, a^{6} m r + a^{6} r^{2}) \cos\left(\theta\right)^{6} + 3 \, (a^{6} r^{2} - 2 \, a^{4} m r^{3} + a^{4} r^{4}) \cos\left(\theta\right)^{4} + 3 \, (a^{4} r^{4} - 2 \, a^{2} m r^{5} + a^{2} r^{6}) \cos\left(\theta\right)^{2}}\right) \mathrm{d} r\wedge \mathrm{d} \phi \\ & + \left(-\frac{\left(i \, a^{4} m + i \, a^{2} m r^{2}\right) \sin\left(\theta\right)^{3} + \left(-i \, a^{4} m + i \, m r^{4} + 2 \, \left(a^{3} m r + a m r^{3}\right) \cos\left(\theta\right)\right) \sin\left(\theta\right)}{a^{4} \cos\left(\theta\right)^{4} + 2 \, a^{2} r^{2} \cos\left(\theta\right)^{2} + r^{4}}\right) \mathrm{d} \theta\wedge \mathrm{d} \phi\end{aligned}$$

Let us check that $\boldsymbol{\mathcal{F}}$ is self-dual, i.e. that it obeys ${}^*\boldsymbol{\mathcal{F}} = -i\boldsymbol{\mathcal{F}}$:

```
FF.hodge_star(g) == - I * FF
```

True

To evaluate the right self-dual of the Weyl tensor, we need the tensor $\epsilon^{\alpha\beta}{}_{\gamma\delta}$:

```
eps = g.volume_form(2)  # 2 = the first 2 indices are contravariant
print eps ; eps.symmetries()
```

tensor field of type (2,2) on the 4-dimensional manifold 'M'
no symmetry; antisymmetries: [(0, 1), (2, 3)]

The right self-dual Weyl tensor is then:

```
CC = Cd + I/2*( eps['^rs_..']*Cd['_..rs'] )
CC.set_name('CC', r'\mathcal{C}') ;
print CC ; CC.symmetries()
```

tensor field 'CC' of type (0,4) on the 4-dimensional manifold 'M'
no symmetry; antisymmetries: [(0, 1), (2, 3)]

```
CC[0,1,2,3]
```

$$\frac{\left(a^{5} m \cos(\theta)^{5} + 3i\, a^{4} m r \cos(\theta)^{4} + 3i\, a^{2} m r^{3} + 2i\, m r^{5} - \left(3\, a^{5} m + 5\, a^{3} m r^{2}\right) \cos(\theta)^{3}\right) \sin(\theta)}{a^{6} \cos(\theta)^{6} + 3\, a^{4} r^{2} \cos(\theta)^{4} + 3\, a^{2} r^{4} \cos(\theta)^{2} + r^{6}}$$
$$+ \frac{\left(\left(-9i\, a^{4} m r - 7i\, a^{2} m r^{3}\right) \cos(\theta)^{2} + 3\left(3\, a^{3} m r^{2} + 2\, a m r^{4}\right) \cos(\theta)\right) \sin(\theta)}{a^{6} \cos(\theta)^{6} + 3\, a^{4} r^{2} \cos(\theta)^{4} + 3\, a^{2} r^{4} \cos(\theta)^{2} + r^{6}}$$

The Ernst 1-form $\sigma_\alpha = 2\mathcal{F}_{\mu\alpha}\, \xi^\mu$ ($0 =$ contraction on the first index of $\boldsymbol{\mathcal{F}}$):

```
sigma = 2*FF.contract(0, xi)
```

Instead of invoking the method `contract()`, we could have used the index notation to denote the contraction:

```
sigma == 2*( FF['_ma']*xi['^m'] )
```

True

```
sigma.set_name('sigma', r'\sigma')
print sigma ; sigma.view()
```

1-form 'sigma' on the 4-dimensional manifold 'M'

$$\sigma = \left(-\frac{2\, a^{2} m \cos(\theta)^{2} + 4i\, a m r \cos(\theta) - 2\, m r^{2}}{a^{4} \cos(\theta)^{4} + 2\, a^{2} r^{2} \cos(\theta)^{2} + r^{4}}\right) \mathrm{d}r + \left(\frac{\left(2i\, a^{3} m \cos(\theta)^{2} - 4\, a^{2} m r \cos(\theta) - 2i\, a m r^{2}\right) \sin(\theta)}{a^{4} \cos(\theta)^{4} + 2\, a^{2} r^{2} \cos(\theta)^{2} + r^{4}}\right) \mathrm{d}\theta$$

The symmetric bilinear form $\boldsymbol{\gamma} = \lambda\, \boldsymbol{g} + \underline{\boldsymbol{\xi}} \otimes \underline{\boldsymbol{\xi}}$:

```
gamma = lamb*g + xi_form * xi_form
gamma.set_name('gamma', r'\gamma')
print gamma ; gamma.view()
```

field of symmetric bilinear forms 'gamma' on the 4-dimensional manifold 'M'

$$\gamma = \left(\frac{a^{2} \cos(\theta)^{2} - 2\, m r + r^{2}}{a^{2} - 2\, m r + r^{2}}\right) \mathrm{d}r \otimes \mathrm{d}r + \left(a^{2} \cos\left(\theta\right)^{2} - 2\, m r + r^{2}\right) \mathrm{d}\theta \otimes \mathrm{d}\theta +$$
$$\left(\frac{2\, a^{2} m r \sin(\theta)^{4} - \left(2\, a^{2} m r - a^{2} r^{2} + 2\, m r^{3} - r^{4} - \left(a^{4} + a^{2} r^{2}\right) \cos(\theta)^{2}\right) \sin(\theta)^{2}}{a^{2} \cos(\theta)^{2} + r^{2}}\right) \mathrm{d}\phi \otimes \mathrm{d}\phi$$

The first part of the Simon-Mars tensor is $S^{(1)}_{\alpha\beta\gamma} = 4\mathcal{C}_{\mu\alpha\nu\beta}\, \xi^\mu\, \xi^\nu\, \sigma_\gamma$:

```
S1 = 4*( CC.contract(0,xi).contract(1,xi) ) * sigma
print S1
```

tensor field of type (0,3) on the 4-dimensional manifold 'M'

The second part is $S^{(2)}_{\alpha\beta\gamma} = \gamma_{\alpha\beta}\, \mathcal{C}_{\rho\gamma\mu\nu}\, \xi^\rho\, \mathcal{F}^{\mu\nu}$, which we compute using the index notation to perform the contractions:

```
FFuu = FF.up(g)
xiCC = CC['_.r..']*xi['^r']
S2 = gamma * ( xiCC['_.mn']*FFuu['^mn'] )
print S2
```

tensor field of type (0,3) on the 4-dimensional manifold 'M'

To get the Simon-Mars tensor, we need to antisymmetrize $\boldsymbol{S}^{(1)}$ and $\boldsymbol{S}^{(2)}$ on their last two indices; we choose to use the standard index notation to perform this operation (an alternative would have been to call directly the method `antisymmetrize()`):

```
S1A = S1['_a[bc]']
S2A = S2['_a[bc]']
```

The Simon-Mars tensor is then

```
S = = S1A + S2A
S.set_name('S')
print S ; S.symmetries()
```

```
tensor field 'S' of type (0,3) on the 4-dimensional manifold 'M'
no symmetry; antisymmetry: (1, 2)
```

We check that it vanishes identically, as it should for Kerr spacetime:

```
S.view()
```

$S = 0$

---

7. Conclusion and future prospects

SageManifolds is a work in progress. It encompasses currently $\sim 35,000$ lines of Python code (including comments and doctests). The last stable version (0.6), the functionalities of which are listed in Sec. 5.1, is freely downloadable from the project page [22]. The development version (to become version 0.7 soon) is also available from that page. Among future developments are

- the extrinsic geometry of pseudo-Riemannian submanifolds,
- the computation of geodesics (numerical integration via Sage/GSL or Gyoto [28]),
- evaluating integrals on submanifolds,
- adding more graphical outputs,
- adding more functionalities: symplectic forms, fibre bundles, spinors, variational calculus, etc.
- the connection with numerical relativity.

The last point means using SageManifolds for interactive exploration of numerically generated spacetimes. Looking at the diagram in Fig. 5, one realizes that it suffices to replace only the lowest level, currently relying on Sage's symbolic expressions, by computations on numerical data.

Let us conclude by stating that, in the very spirit of free software, anybody interested in contributing to the project is very welcome!

Acknowledgments

This work has benefited from enlightening discussions with Volker Braun, Vincent Delecroix, Simon King, Sébastien Labbé, José M. Martín-García, Marc Mezzarobba, Thierry Monteil, Travis Scrimshaw, Nicolas M. Thiéry and Anne Vaugon. We also thank Stéphane Méné for his technical help and Tolga Birkandan for fruitful comments about the manuscript. EG acknowledges the warm hospitality of the Organizers of the *Encuentros Relativistas Españoles 2014*, where this work was presented.

References

[1] Fletcher J G, Clemens R, Matzner R, Thorne K S and Zimmerman B A 1967 *Astrophys. J.* **148**, L91
[2] Skea J E F 1994 Applications of SHEEP *Lecture notes available at* http://www.computeralgebra.nl/systemsoverview/special/tensoranalysis/sheep/
[3] MacCallum M A H 2002 *Int. J. Mod. Phys. A* **17**, 2707
[4] Korol'kova A V, Kulyabov D S and Sevast'yanov L A 2013 *Prog. Comput. Soft.* **39**, 135
[5] http://www.xact.es/links.html

[6] Martin-Garcia J M 2008 *Comput. Phys. Commun.* **179**, 597; http://www.xact.es
[7] http://www.math.washington.edu/~lee/Ricci/
[8] Anderson I M and Torre C G 2012 *J. Math. Phys.* **53**, 013511; http://digitalcommons.usu.edu/dg/
[9] http://grtensor.phy.queensu.ca/
[10] http://digi-area.com/Maple/atlas/
[11] Peeters K 2007 *Comput. Phys. Commun.* **15**, 550; http://cadabra.phi-sci.com/
[12] Culler M, Dunfield N M and Weeks J R, SnapPy, a computer program for studying the geometry and topology of 3-manifolds, http://snappy.computop.org
[13] Bolotin D A and Poslavsky S V 2013 Introduction to Redberry: the computer algebra system designed for tensor manipulation *Preprint* arXiv:1302.1219; http://redberry.cc/
[14] http://sagemath.org/
[15] Stein W and Joyner D 2005 *Commun. Comput. Algebra* **39**, 61
[16] Joyner D and Stein W 2014 *Sage Tutorial* (CreateSpace)
[17] Zimmermann P et al. 2013 *Calcul mathématique avec Sage* (CreateSpace); freely downloadable from http://sagebook.gforge.inria.fr/
[18] Bard G V 2015 *Sage for Undergraduates* (Americ. Math. Soc.) in press; preprint freely downloadable from http://www.gregorybard.com/
[19] http://www.sagemath.org/doc/reference/categories/sage/categories/primer.html
[20] http://sagemath.org/doc/reference/tensor/
[21] http://sagemath.org/doc/reference/riemannian_geometry/
[22] http://sagemanifolds.obspm.fr/
[23] Lee J M 2013 *Introduction to Smooth Manifolds* 2nd edition (New York: Springer)
[24] Steenrod N 1951 *The Topology of Fibre Bundles* (Princeton: Princeton Univ. Press)
[25] http://sagemanifolds.obspm.fr/examples.html
[26] Apéry F 1986 *Adv. Math.* **61**, 185
[27] Mars M 1999 *Class. Quantum Grav.* **16**, 2507
[28] http://gyoto.obspm.fr/