Tensor calculus with free softwares: the SageManifolds project

Éric Gourgoulhon[1], Michał Bejger[2]

[1] Laboratoire Univers et Théories (LUTH)
CNRS / Observatoire de Paris / Université Paris Diderot
92190 Meudon, France
http://luth.obspm.fr/~luthier/gourgoulhon/

[2] Centrum Astronomiczne im. M. Kopernika (CAMK)
Warsaw, Poland
http://users.camk.edu.pl/bejger/

Encuentros Relativistas Españoles 2014

Valencia

1-5 September 2014

Outline

1. Differential geometry and tensor calculus on a computer
2. Sage: a free mathematics software
3. The SageManifolds project
4. SageManifolds at work: the Mars-Simon tensor example
5. Conclusion and perspectives

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 1969, during his PhD under Pirani supervision at King's College, 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 yield 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 1969, during his PhD under Pirani supervision at King's College, 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 yield to the discovery of 6 errors in the original paper [J.E.F. Skea, *Applications of SHEEP* (1994)]
- In the early 1970's, ALAM was rewritten in the LISP programming language, thereby becoming machine independent and renamed `LAM`

Introduction

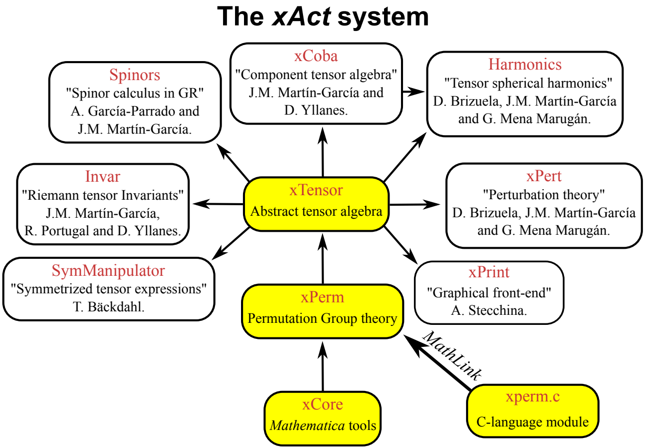
- Computer algebra system (CAS) started to be developed in the 1960's; for instance `Macsyma` (to become `Maxima` in 1998) was initiated in 1968 at MIT
- In 1969, during his PhD under Pirani supervision at King's College, Ray d'Inverno wrote `ALAM (Atlas Lisp Algebraic Manipulator)` and used it to compute 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 yield to the discovery of 6 errors in the original paper [J.E.F. Skea, *Applications of SHEEP* (1994)]
- In the early 1970's, ALAM was rewritten in the LISP programming language, thereby becoming machine independent and renamed `LAM`
- The descendant of LAM, called `SHEEP` (!), was initiated in 1977 by Inge Frick

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 1969, during his PhD under Pirani supervision at King's College, 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 yield to the discovery of 6 errors in the original paper [J.E.F. Skea, *Applications of SHEEP* (1994)]
- In the early 1970's, ALAM was rewritten in the LISP programming language, thereby becoming machine independent and renamed `LAM`
- The descendant of LAM, called `SHEEP` (!), was initiated in 1977 by Inge Frick
- Since then, many softwares for tensor calculus have been developed...

An example of modern software: The xAct suite

Free packages for tensor computer algebra in Mathematica, developed by José Martín-García et al. http://www.xact.es/



[García-Parrado Gómez-Lobo & Martín-García, Comp. Phys. Comm. **183**, 2214 (2012)]

Software for differential geometry

Packages for general purpose computer algebra systems:

- xAct free package for Mathematica [J.-M. Martin-Garcia]
- Ricci free package for Mathematica [J. L. Lee]
- MathTensor package for Mathematica [S. M. Christensen & L. Parker]
- DifferentialGeometry included in Maple [I. M. Anderson & E. S. Cheb-Terrab]
- Atlas 2 for Maple and Mathematica
- ...

Standalone applications:

- SHEEP, Classi, STensor, based on Lisp, developed in 1970's and 1980's (free) [R. d'Inverno, I. Frick, J. Åman, J. Skea, et al.]
- Cadabra field theory (free) [K. Peeters]
- SnapPy topology and geometry of 3-manifolds, based on Python (free) [M. Culler, N. M. Dunfield & J. R. Weeks]
- ...

cf. the complete list on `http://www.xact.es/links.html`

Software for differential geometry

Two types of **tensor computations**:

Abstract computations

- xAct/xTensor
- MathTensor
- Ricci
- Cadabra

Component computations

- xAct/xCoba
- Atlas 2
- DifferentialGeometry
- SageManifolds

Outline

Sage in a few words

- 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 (symbolic calculations, since 1968!)
  - 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 Sage in 2005; since then, ~**100 developers** (mostly mathematicians) have joined the Sage team

The mission

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

Some advantages of Sage

Sage 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

Sage 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

Sage is developing and spreading fast

...sustained by an important community of developers

Sage approach to computer mathematics

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

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

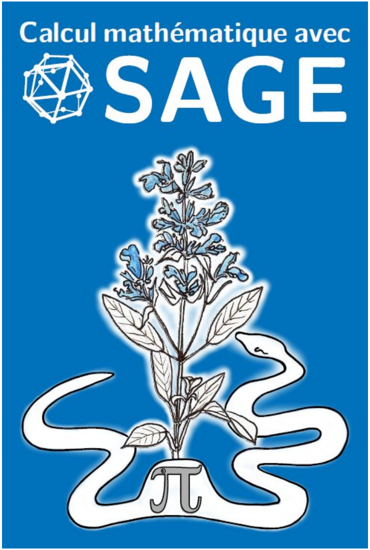
Conversion rules prior to an operation, e.g. $x + y$ with x and y having different parents, are defined at the level of the parents

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 (a CAS quite well spread in pure mathematics) and different from that of Mathematica, in which everything is a tree of symbols

The Sage book



by Paul Zimmermann et al. (2013)

Released under *Creative Commons* license:

- freely downloadable from http://sagebook.gforge.inria.fr/
- printed copies can be ordered at moderate price (10 €)

English translation in progress...

Outline

Differential geometry in Sage

Sage is well developed in many domains of mathematics:
number theory, group theory, linear algebra, combinatorics, etc.

...but not too much in the area of differential geometry:

Already in Sage

- differential forms on an open subset of Euclidean space *(with a fixed set of coordinates)* (J. Vankerschaver)
- parametrized 2-surfaces in 3-dim. Euclidean space (M. Malakhaltsev, J. Vankerschaver, V. Delecroix)

Proposed extensions (Sage Trac)

- 2-D hyperbolic geometry (V. Delecroix, M. Raum, G. Laun, trac ticket #9439)

The SageManifolds project

http://sagemanifolds.obspm.fr/

Aim

Implement the concept of real smooth manifolds of arbitrary dimension in Sage and tensor calculus on them, in a coordinate/frame-independent manner

The SageManifolds project

http://sagemanifolds.obspm.fr/

Aim

Implement the concept of real smooth manifolds of arbitrary dimension in Sage and tensor calculus on them, in a coordinate/frame-independent manner

In practice, this amounts to introducing new Python classes in Sage, basically one class per mathematical concept, for instance:

- `Manifold`: differentiable manifolds over $\mathbb{R}$, of arbitrary dimension
- `Chart`: coordinate charts
- `Point`: points on a manifold
- `DiffMapping`: differential mappings between manifolds
- `ScalarField`, `VectorField`, `TensorField`: tensor fields on a manifold
- `DiffForm`: p-forms
- `AffConnection`, `LeviCivitaConnection`: affine connections
- `Metric`: pseudo-Riemannian metrics

Implementing coordinate charts

Given a manifold $\mathcal{M}$ of dimension n, a coordinate chart on an open subset $U \subset \mathcal{M}$ is implemented in SageManifolds via the class `Chart`, whose main data is a n-uple of *Sage symbolic variables* `x`, `y`, ..., each of them representing a coordinate

Implementing coordinate charts

Given a manifold $\mathcal{M}$ of dimension n, a coordinate chart on an open subset $U \subset \mathcal{M}$ is implemented in SageManifolds via the class `Chart`, whose main data is a n-uple of *Sage symbolic variables* `x`, `y`, ..., each of them representing a coordinate

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

Examples:

- at least 2 charts are necessary to cover the circle $\mathbb{S}^1$, the sphere $\mathbb{S}^2$, and more generally the n-dimensional sphere $\mathbb{S}^n$
- at least 3 charts are necessary to cover the real projective plane $\mathbb{RP}^2$

Implementing coordinate charts

Given a manifold $\mathcal{M}$ of dimension n, a coordinate chart on an open subset $U \subset \mathcal{M}$ is implemented in SageManifolds via the class `Chart`, whose main data is a n-uple of *Sage symbolic variables* `x`, `y`, ..., each of them representing a coordinate

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

Examples:

- at least 2 charts are necessary to cover the circle $\mathbb{S}^1$, the sphere $\mathbb{S}^2$, and more generally the n-dimensional sphere $\mathbb{S}^n$
- 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 $\mathcal{M}$ is a smooth mapping

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

where U is an open subset of $\mathcal{M}$

Implementing scalar fields

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

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

where U is an open subset of $\mathcal{M}$

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 $\mathcal{M}$ is a smooth mapping

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

where U is an open subset of $\mathcal{M}$

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, $\hat{C}$, ...:

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

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

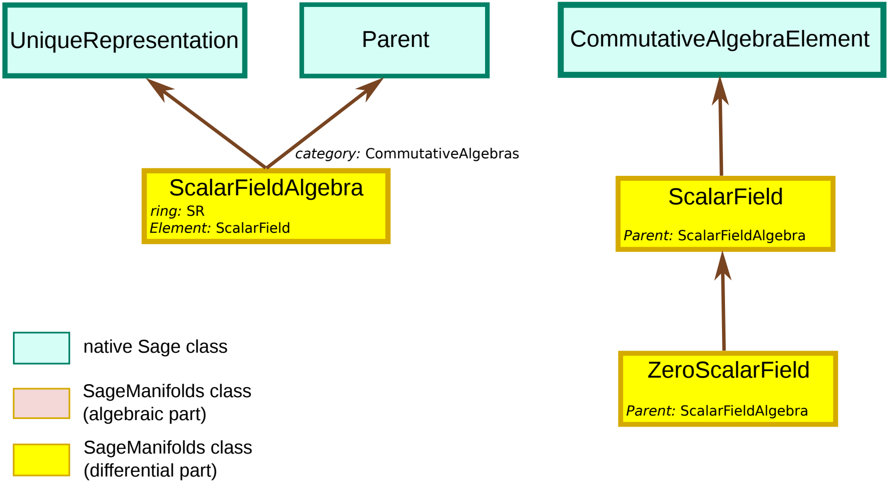
The scalar field algebra

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:

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

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

Classes for scalar fields



Vector fields

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

Vector fields

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

Reminder from linear algebra

A **module** is $\sim$ **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 physics)

An importance difference: a vector space always has a **basis**, *while a module does not necessarily have any*

$\to$ A module with a basis is called a **free module**

Vector fields

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

Reminder from linear algebra

A **module** is $\sim$ **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 physics)

An importance difference: a vector space always has a **basis**, *while a module does not necessarily have any*

$\rightarrow$ A module with a basis is called a **free module**

When $\mathcal{X}(U)$ is a free module, a basis is a **vector frame** $(\boldsymbol{e}_a)_{1\leq a\leq n}$ on U:

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

At a point $p \in U$, the above translates into an identity in the *tangent vector space* $T_p\mathcal{M}$:

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

Vector fields

A manifold $\mathcal{M}$ that admits a global vector frame (or equivalently, such that $\mathcal{X}(\mathcal{M})$ is a free module) is called a **parallelizable manifold**

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 $\mathbb{S}^7$

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 $\mathbb{RP}^2$
- most manifolds...

Implementing vector fields

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

If the manifold $\mathcal{M}$ is not parallelizable, one has to decompose it in parallelizable open subsets U_i $(1 \leq i \leq N)$ and 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.

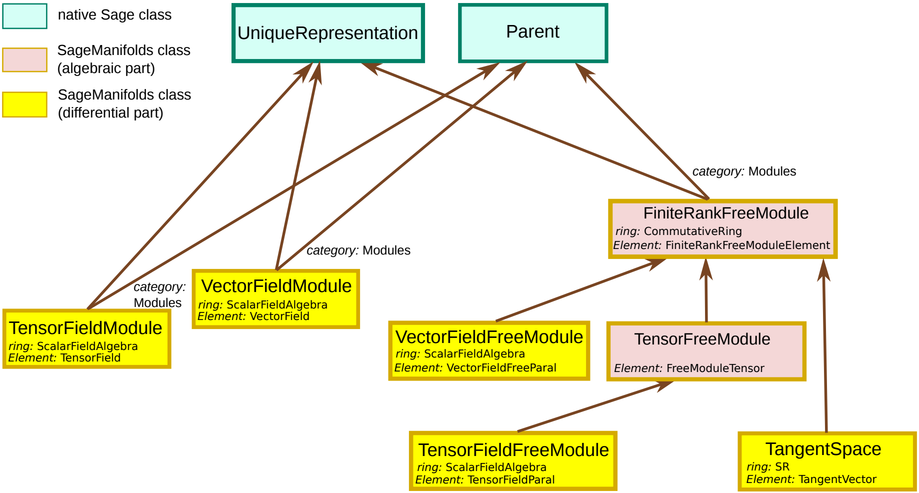
If the manifold $\mathcal{M}$ is not parallelizable, one has to decompose it in parallelizable open subsets U_i $(1 \leq i \leq N)$ and consider **restrictions** of vector fields to these domains.

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

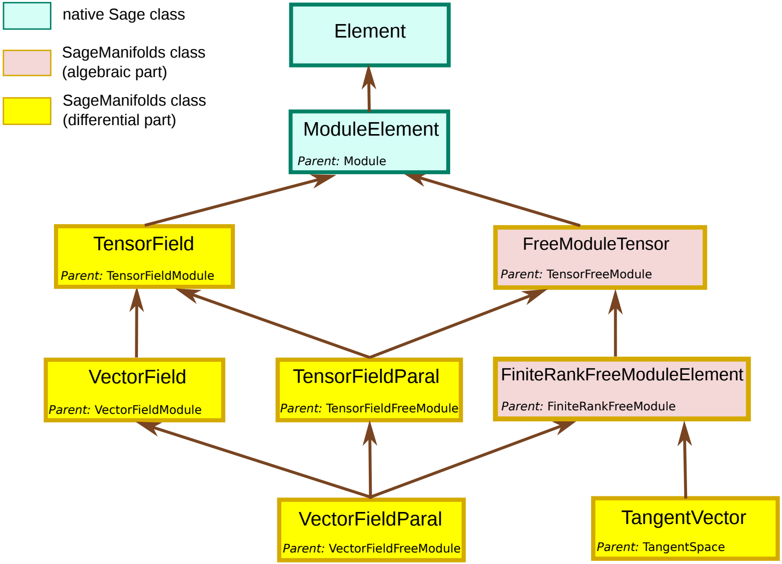
Each vector field $\boldsymbol{v} \in \mathcal{X}(U_i)$ has different set of components $(v^a)_{1 \leq a \leq n}$ in different vector frames $(\boldsymbol{e}_a)_{1 \leq a \leq n}$ introduced on U_i. They are stored as a *Python dictionary* whose keys are the vector frames:

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

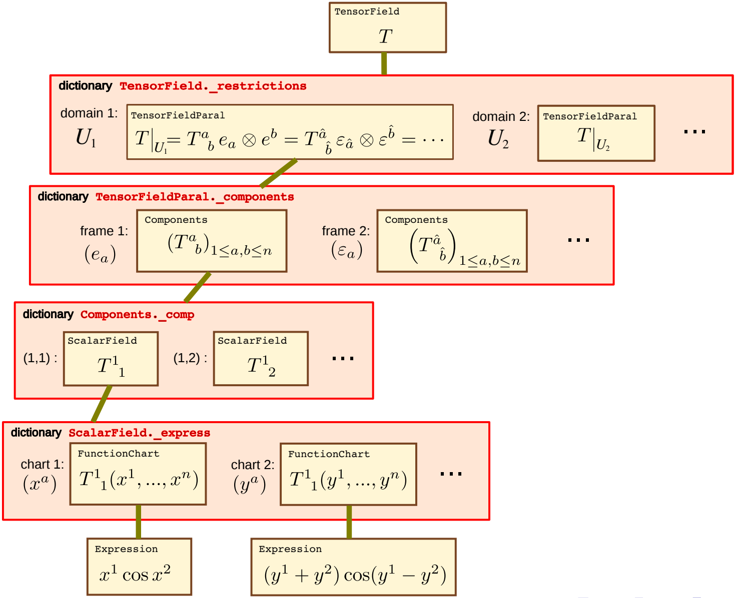
Module classes in SageManifolds



Tensor field classes in SageManifolds



Tensor field storage



Outline

Mars-Simon tensor

Definition [M. Mars, CQG **16**, 2507 (1999)]

Given a 4-dimensional spacetime $(\mathcal{M}, \boldsymbol{g})$ endowed with a Killing vector field ξ, the **Mars-Simon tensor w.r.t.** ξ is the type-(0,3) tensor $\boldsymbol{S}$ defined by

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

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 tensor and $\epsilon_{\alpha\beta\mu\nu}$ the Levi-Civita volume form
- $\mathcal{F}_{\alpha\beta} := F_{\alpha\beta} + i\, {}^*\!F_{\alpha\beta}$, with $F_{\alpha\beta} := \nabla_\alpha \xi_\beta$ (Killing 2-form) and ${}^*\!F_{\alpha\beta} := \frac{1}{2} \epsilon^{\mu\nu}{}_{\alpha\beta} F_{\mu\nu}$ (Hodge dual of $F_{\alpha\beta}$)
- $\sigma_\alpha := 2\mathcal{F}_{\mu\alpha} \xi^\mu$ (Ernst 1-form)

Mars-Simon tensor

The Mars-Simon tensor provides a nice characterization of Kerr spacetime:

Theorem (Mars, 1999)

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

$$\boldsymbol{S} = 0 \iff (\mathcal{M}, \boldsymbol{g}) \text{ is locally isometric to a Kerr spacetime}$$

Mars-Simon tensor

The Mars-Simon tensor provides a nice characterization of Kerr spacetime:

Theorem (Mars, 1999)

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

$$\boldsymbol{S} = 0 \iff (\mathcal{M}, \boldsymbol{g}) \text{ is locally isometric to a Kerr spacetime}$$

Let us use SageManifolds...

...to check the $\Leftarrow$ part of the theorem, namely that the Mars-Simon tensor is identically zero in Kerr spacetime.

NB: what follows illustrates only certain features of SageManifolds; other ones, like the multi-chart and multi-frame capabilities on non-parallelizable manifolds, are not considered in this example. $\Longrightarrow$ *More examples are provided at*
http://sagemanifolds.obspm.fr/examples.html

Object-oriented notation

To understand what follows, be aware that

as an object-oriented language, Python (and hence Sage) makes use of the following postfix notation:

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

In a functional language, this would be written as

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

Object-oriented notation

To understand what follows, be aware 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 be written as

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

Examples

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

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

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

    4-dimensional manifold 'M'

We introduce the standard **Boyer-Lindquist coordinates** as follows:

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

    chart (M, (t, r, th, ph))

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

Metric tensor

The 2 parameters m and a of the Kerr spacetime are declared as symbolic variables:

```
var('m, a')
```

(m, a)

Let us introduce the spacetime metric g and set its components in the coordinate frame associated with Boyer-Lindquist coordinates, which is the current manifold's default frame:

```
g = M.lorentz_metric('g')
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()
```

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

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

```
g[:]
```

$$\begin{pmatrix} -\frac{a^2\cos(\theta)^2 - 2mr + r^2}{a^2\cos(\theta)^2 + r^2} & 0 & 0 & -\frac{2amr\sin(\theta)^2}{a^2\cos(\theta)^2 + r^2} \\ 0 & \frac{a^2\cos(\theta)^2 + r^2}{a^2 - 2mr + r^2} & 0 & 0 \\ 0 & 0 & a^2\cos(\theta)^2 + r^2 & 0 \\ -\frac{2amr\sin(\theta)^2}{a^2\cos(\theta)^2 + r^2} & 0 & 0 & \frac{2a^2mr\sin(\theta)^4 + \left(a^2r^2 + r^4 + \left(a^4 + a^2r^2\right)\cos(\theta)^2\right)\sin(\theta)^2}{a^2\cos(\theta)^2 + r^2} \end{pmatrix}$$

The Levi-Civita connection ∇ associated with g:

```
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 g with respect to ∇ vanishes identically:

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

$$\nabla_g g = 0$$

Killing vector

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()
```

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

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

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'
```

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

Let us check that the Killing equation is satisfied:

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

$$0$$

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

Equivalently, we check that the Lie derivative of the metric along ξ vanishes:

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

0

Thank to Killing equation, $\nabla_g \xi$ is antisymmetric. We may therefore define a 2-form by $F := -\nabla_g \xi$. Here we enforce the antisymmetry by calling the function `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'
```

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

We check that

```
F == - nab_xi
```

True

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'
```

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

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

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

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

True

or, in index notation,

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

True

Curvature

The Riemann curvature tensor associated with g is:

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

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

The component $R^0{}_{123}$ is

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

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

The Ricci tensor:

```
Ric = g.ricci()
print Ric
```

```
field of symmetric bilinear forms 'Ric(g)' on the 4-dimensional manifold
'M'
```

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

Let us check that we are dealing with a solution of Einstein equation:

```
Ric.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 two of its components $C^0{}_{123}$ and $C^0{}_{101}$:

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

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

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

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

To form the Mars-Simon tensor, we need the fully covariant (type-(0,4) tensor) form 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):

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

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, as we can check:

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

True

To take this symmetry into account explicitely, we set

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

Hence we have now

```
Cd.symmetries()
```

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

Mars-Simon tensor

The Mars-Simon tensor with respect to the Killing vector ξ is a rank-3 tensor introduced by Marc Mars in 1999 (Class. Quantum Grav. **16**, 2507). It has the remarkable property to vanish identically if, and only if, the spacetime $(\mathcal{M}, g)$ is locally isometric to a Kerr spacetime.

Let us evaluate the Mars-Simon tensor by following the formulas given in Mars' article. The starting point is the self-dual complex 2-form associated with the Killing 2-form F, i.e. the object $\mathcal{F} := F + i\,{}^*F$, where ${}^*F$ is the Hodge dual of F:

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

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

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

```
FF.view()
```

$$\mathcal{F} = \left(-\frac{a^2 m \cos(\theta)^2 + 2i\, amr \cos(\theta) - mr^2}{a^4 \cos(\theta)^4 + 2\, a^2 r^2 \cos(\theta)^2 + r^4} \right) \mathrm{d}t \wedge \mathrm{d}r + \left(\frac{\left(i\, a^3 m \cos(\theta)^2 - 2\, a^2 mr \cos(\theta) - i\, amr^2 \right) \sin(\theta)}{a^4 \cos(\theta)^4 + 2\, a^2 r^2 \cos(\theta)^2 + r^4} \right) \mathrm{d}t \wedge \mathrm{d}\theta + \left(\frac{-4i\, a^4 m^2 r^2 \cos(\theta) \sin(\theta)^4 + \left(a^3 mr^4 - 2\, am^2 \cdots \right.}{a^2 r^6 - 2\, m \cdots} \right.$$

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

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

True

Let us form the right self-dual of the Weyl tensor as follows

$$\mathcal{C}_{\alpha\beta\mu\nu} = C_{\alpha\beta\mu\nu} + \frac{i}{2}\, \epsilon^{\rho\sigma}{}_{\mu\nu}\, C_{\alpha\beta\rho\sigma}$$

where $\epsilon^{\rho\sigma}{}_{\mu\nu}$ is associated to the Levi-Civita tensor $\epsilon_{\rho\sigma\mu\nu}$ and is obtained by

```
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
```

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

```
CC.symmetries()
```

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

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

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

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

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

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

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

Instead of invoking the function `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 $\gamma = \lambda\, g + \underline{\xi} \otimes \underline{\xi}$:

```
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'
```

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

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

Final computation leading to the Mars-Simon tensor:

First, we evaluate

$$S^{(1)}_{\alpha\beta\gamma} = 4\,\mathcal{C}_{\mu\alpha\nu\beta}\,\xi^\mu\xi^\nu\,\sigma_\gamma$$

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

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

Then we form the tensor

$$S^{(2)}_{\alpha\beta\gamma} = \gamma_{\alpha\beta}\,\mathcal{C}_{\gamma\rho\mu\nu}\,\xi^\rho\,\mathcal{F}^{\mu\nu}$$

by first computing $\mathcal{C}_{\gamma\rho\mu\nu}\,\xi^\rho$:

```
xiCC = CC['_.r..']*xi['^r']
print xiCC
```

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

and evaluating $\mathcal{F}^{\alpha\beta} = g^{\alpha\mu}g^{\beta\nu}\mathcal{F}_{\mu\nu}$:

```
FFuu = FF.up(g)
```

We use the index notation to perform the double contraction $\mathcal{C}_{\gamma\rho\mu\nu}\mathcal{F}^{\mu\nu}$:

```
S2 = gamma * ( xiCC['_.mn']*FFuu['^mn'] )
print S2
S2.symmetries()
```

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

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

The Mars-Simon tensor with respect to ξ is obtained by antisymmetrizing $S^{(1)}$ and $S^{(2)}$ on their last two indices and adding them:

$$S_{\alpha\beta\gamma} = S^{(1)}_{\alpha[\beta\gamma]} + S^{(2)}_{\alpha[\beta\gamma]}$$

We use the index notation for the antisymmetrization:

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

An equivalent writing would have been (the last two indices being in position 1 and 2):

```
# S1A = S1.antisymmetrize((1,2))
# S2A = S2.antisymmetrize((1,2))
```

The Mars-Simon tensor is

```
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)
```

```
S.view()
```

$$S = 0$$

We thus recover the fact that the Mars-Simon tensor vanishes identically in Kerr spacetime.

To check that the above computation was not trival, here is the component 112=$rr\theta$ for each of the two parts of the Mars-Simon tensor:

```
S1A[1,1,2]
```

SM_Mars-Simon -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Mars-Simon – Sage

The Mars-Simon tensor is

```
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)
```

```
S.view()
```

$$S = 0$$

We thus recover the fact that the Mars-Simon tensor vanishes identically in Kerr spacetime.

To check that the above computation was not trival, here is the component $112 = rr\theta$ for each of the two parts of the Mars-Simon tensor:

```
S1A[1,1,2]
```

$$-\frac{\left(4\,a^{8}m^{2}\cos(\theta)^{7}+20i\,a^{7}m^{2}r\cos(\theta)^{6}-8i\,am^{3}r^{6}+4i\,am^{2}r^{7}-4\left(2\,a^{6}m^{3}r+9\,a^{6}m^{2}r^{2}\right)\cos(\theta)^{5}+\left(-40i\,a^{5}m^{3}r^{2}-20i\,a^{5}m^{2}r^{3}\right)\cos(\theta)^{4}+20\left(4\,a^{4}m^{3}r^{3}-a^{4}m^{2}r^{4}\right)\cos(\theta)^{3}+(80i\,a^{3}m^{3}r^{4}-\cdots}{a^{2}r^{10}-2\,mr^{11}+r^{12}+\left(a^{12}-2\,a^{10}mr+a^{10}r^{2}\right)\cos(\theta)^{10}+5\left(a^{10}r^{2}-2\,a^{8}mr^{3}+a^{8}r^{4}\right)\cos(\theta)^{8}+10\left(a^{8}r^{4}-2\,a^{6}mr^{5}+a^{6}r^{6}\right)\cos(\theta)^{6}+10\left(a^{6}r^{6}-2\,a^{4}mr^{7}+a^{4}r^{8}\right)\cos(\cdots}$$

```
S2A[1,1,2]
```

$$\frac{\left(4\,a^{8}m^{2}\cos(\theta)^{7}+20i\,a^{7}m^{2}r\cos(\theta)^{6}-8i\,am^{3}r^{6}+4i\,am^{2}r^{7}-4\left(2\,a^{6}m^{3}r+9\,a^{6}m^{2}r^{2}\right)\cos(\theta)^{5}+\left(-40i\,a^{5}m^{3}r^{2}-20i\,a^{5}m^{2}r^{3}\right)\cos(\theta)^{4}+20\left(4\,a^{4}m^{3}r^{3}-a^{4}m^{2}r^{4}\right)\cos(\theta)^{3}+(80i\,a^{3}m^{3}r^{4}-36\cdots}{a^{2}r^{10}-2\,mr^{11}+r^{12}+\left(a^{12}-2\,a^{10}mr+a^{10}r^{2}\right)\cos(\theta)^{10}+5\left(a^{10}r^{2}-2\,a^{8}mr^{3}+a^{8}r^{4}\right)\cos(\theta)^{8}+10\left(a^{8}r^{4}-2\,a^{6}mr^{5}+a^{6}r^{6}\right)\cos(\theta)^{6}+10\left(a^{6}r^{6}-2\,a^{4}mr^{7}+a^{4}r^{8}\right)\cos(\theta)^{4}\cdots}$$

```
S1A[1,1,2] + S2A[1,1,2]
```

$$0$$

evaluate

Outline

Conclusion and perspectives

- **SageManifolds** is a work in progress
  $\sim$ 34,000 lines of Python code up to now (including comments and doctests)
- A preliminary version (v0.5) is freely available (GPL) at
  `http://sagemanifolds.obspm.fr/`
  and the development version (to become v0.6 soon) is available from the Git repository `https://github.com/sagemanifolds/sage`
- *Already present:*
  - maps between manifolds, pullback operator
  - submanifolds, pushforward operator
  - standard tensor calculus (tensor product, contraction, symmetrization, etc.), even on non-parallelizable manifolds
  - all monoterm tensor symmetries
  - exterior calculus, Hodge duality
  - Lie derivatives
  - affine connections, curvature, torsion
  - pseudo-Riemannian metrics, Weyl tensor

Conclusion and perspectives

- *Not implemented yet (but should be soon):*
  - extrinsic geometry of pseudo-Riemannian submanifolds
  - computation of geodesics (numerical integration via Sage/GSL or `Gyoto`)
  - integrals on submanifolds
- *To do:*
  - add more graphical outputs
  - add more functionalities: symplectic forms, fibre bundles, spinors, variational calculus, etc.
  - connection with `Lorene`, `CoCoNuT`, ...

Want to join the project or simply to stay tuned?

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