SageManifolds

A free package for differential geometry and tensor calculus

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/

20th International Conference on General Relativity and Gravitation
Warsaw, Poland
7-13 July 2013

Outline

Outline

Sage in a few words

- Sage is a **free open-source** mathematics software
- it is based on the Python programming language
- it makes use of **many pre-existing open-sources packages**, among which
  - Maxima (symbolic calculations, since 1967 !)
  - GAP (group theory)
  - PARI/GP (number theory)
  - Singular (polynomial computations)
  - matplotlib (high quality figures)

  and provides a uniform interface to them
- William Stein (Univ. of Washington) created Sage in 2005; since then, ∼**150 developers** have joined the Sage team

The mission

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

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 a 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

Outline

Existing softwares for differential geometry

Packages for proprietary softwares:

- xAct free package for Mathematica
- DifferentialGeometry included in Maple
- Atlas 2 for Maple
- ...

Standalone softwares:

- Cadabra field theory (free)
- SnapPy topology and geometry of 3-manifolds (Python) (free)
- ...

The situation in Sage

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

but nothing is implemented for differential geometry, except for differential forms on an open subset of Euclidean space *(with a fixed set of coordinates)*.

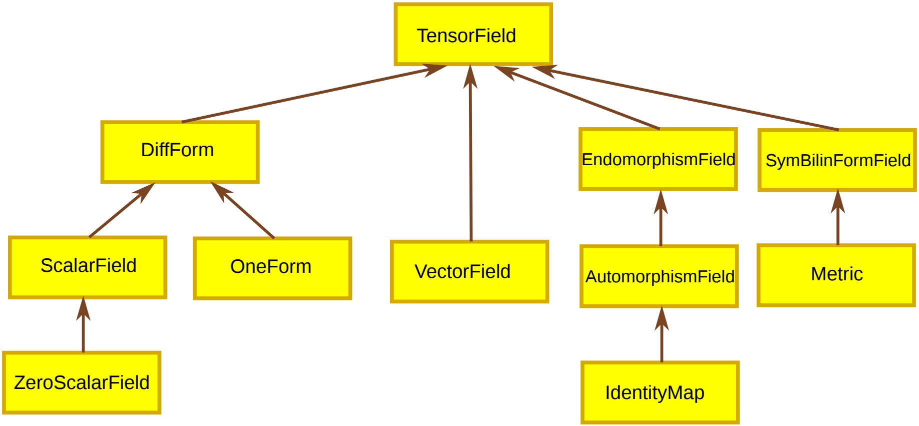
Hence the SageManifolds project

SageManifolds

A new set of *Python classes* implementing differential geometry in Sage:

- `Manifold`: differentiable manifolds over $\mathbb{R}$, of arbitrary dimension
- `SubManifold`, `Curves`: submanifolds
- `Point`: points on a manifold
- `Chart`: charts
- `DiffMapping`, `Diffeomorphism`: differential mappings between manifolds
- `ScalarField`: differential mappings to $\mathbb{R}$
- `TensorField`, `VectorField`, `SymBilinFormField`, etc.: tensor fields on a manifold
- `DiffForm`, `OneForm`: p-forms
- `VectorFrame`, `CoordBasis`: vector frames on a manifold, including tetrads and coordinate bases
- `Components`, `CompWithSym`, etc.: components of a tensor field in a given vector frame
- `AffConnection`, `LeviCivitaConnection`: affine connections
- `Metric`, `RiemannMetric`, `LorentzMetric`: pseudo-Riemannian metrics

Inheritance diagram of the tensor field classes



Basic SageManifolds objects are coordinate-free

1. The scalar field case

Scalar field on the manifold $\mathcal{M}$: (differentiable) mapping $f : \mathcal{M} \to \mathbb{R}$

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 $\mathcal{M}$.

The various coordinate representations are stored as a *Python dictionary* whose keys are the names of the various charts:

$$f.\mathrm{express} = \left\{ C : F, \ \hat{C} : \hat{F}, \ldots \right\}$$

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

Basic SageManifolds objects are coordinate-free

2. The tensor field case

Given a vector frame $(\boldsymbol{e}_i)$ with dual coframe $(\boldsymbol{e}^i)$, the components of a tensor field $\boldsymbol{T}$ in this frame are *scalar fields*, since

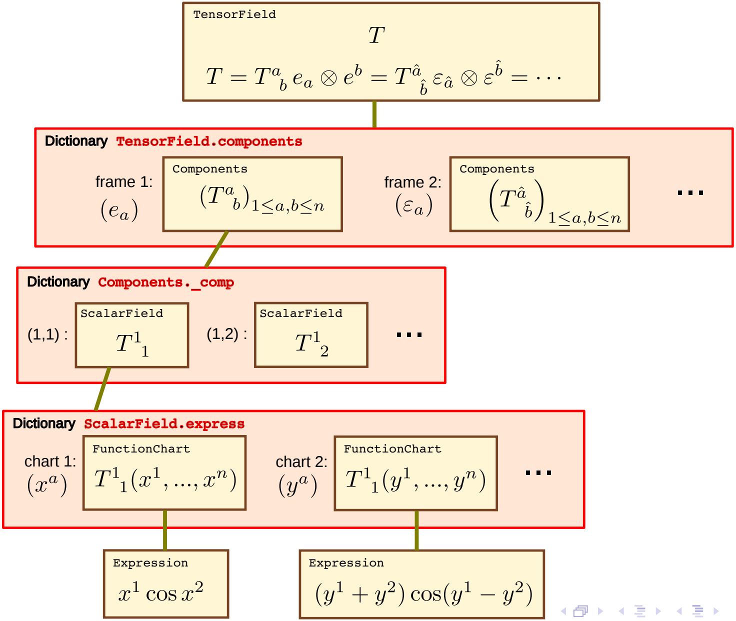
$$T^{i\ldots}_{\quad j\ldots} = \boldsymbol{T}(\boldsymbol{e}^i, \ldots, \boldsymbol{e}_j, \ldots)$$

$\Longrightarrow$ an object $\boldsymbol{T}$ in the `TensorField` class has different sets of components $T^{i\ldots}_{\quad j\ldots}$ in different vector frames, each component in a given set being an object of the `ScalarField` class

The various sets of components are stored as a *Python dictionary* whose keys are the names of the various vector frames:

$$\boldsymbol{T}.\mathrm{components} = \left\{ (\boldsymbol{e}) : (T^{i\ldots}_{\quad j\ldots}),\ (\hat{\boldsymbol{e}}) : (\hat{T}^{i\ldots}_{\quad j\ldots}), \ldots \right\}$$

Tensor field implementation



SageManifolds at work: the Kerr-Newman example

1. Checking Maxwell equations

SM_Kerr_Newman -- Sage - Mozilla Firefox

SageManifolds: examples | SM_Kerr_Newman – Sage

localhost:8080/home/admin/80/

```
dX = M.coframe('BL_b') ; dX
```

$$(dt, dr, d\theta, d\phi)$$

The electromagnetic field tensor F is formed as [cf. e.g. Eq. (33.5) of Misner, Thorne & Wheeler (1973)]

```
F = q/rho2^2 * (r^2-a^2*cos(th)^2)* dX(1).wedge( dX(0) - a*sin(th)^2* dX(3) ) + \
    2*q/rho2^2 * a*r*cos(th)*sin(th)* dX(2).wedge( (r^2+a^2)* dX(3) - a* dX(0) )
```

```
F.set_name('F') ; F.show()
```

$$F = \left(\frac{a^2 q \cos(\theta)^2 - q r^2}{a^4 \cos(\theta)^4 + 2a^2r^2\cos(\theta)^2 + r^4}\right) dt \wedge dr + \left(\frac{2a^2 q r \sin(\theta)\cos(\theta)}{a^4 \cos(\theta)^4 + 2a^2r^2\cos(\theta)^2 + r^4}\right) dt \wedge d\theta + \left(\frac{\left(a^3 q \cos(\theta)^2 - a q r^2\right)\sin(\theta)^2}{a^4 \cos(\theta)^4 + 2a^2r^2\cos(\theta)^2 + r^4}\right) dr \wedge d\phi + \left(\frac{2\left(a^3 q r + a q r\right.}{a^4 \cos(\theta)^4 + 2\ldots}\right.$$

The Hodge dual of F:

```
star_F = F.hodge_star(g) ; star_F.show()
```

$$\star F = \left(\frac{2aqr\cos(\theta)}{a^4 \cos(\theta)^4 + 2a^2r^2\cos(\theta)^2 + r^4}\right) dt \wedge dr + \left(-\frac{\left(a^3 q \cos(\theta)^2 - a q r^2\right)\sin(\theta)}{a^4 \cos(\theta)^4 + 2a^2r^2\cos(\theta)^2 + r^4}\right) dt \wedge d\theta + \left(-\frac{2\left(a^4 q r \sin(\theta)^4 \cos(\theta) - \left(a^4 q r + a^2 q r^3\right)\sin(\theta)^2\cos(\theta)\right)}{a^6\cos(\theta)^6 + 3a^4r^2\cos(\theta)^4 + 3a^2r^4\cos(\theta)^2 + r^6}\right)$$

Maxwell equations

Let us check that F obeys the two (source-free) Maxwell equations:

```
xder(F).show()
```

$$dF = 0$$

```
xder(star_F).show()
```

$$d \star F = 0$$

SageManifolds at work: the Kerr-Newman example
2. Checking Einstein equations

Copy of SM_Kerr_Newman -- Sage - Mozilla Firefox

SageManifolds: examples | Copy of SM_Kerr_Newman -...

localhost:8080/home/admin/86/

The Einstein tensor is

```
G = g.ricci() - 1/2*g.ricci_scalar()*g ; print G
    field of symmetric bilinear forms '+Ric(g)' on the 4-dimensional
    manifold 'M'
```

Since the Ricci scalar is zero, the Einstein tensor reduces to the Ricci tensor:

```
G == g.ricci()
    True
```

The invariant $F_{\mu\nu}F^{\mu\nu}$ of the electromagnetic field:

```
F2 = F.contract(0, F.up(g), 0).self_contract(0, 1) ; print F2
    scalar field on the 4-dimensional manifold 'M'
```

```
F2.show()
```

$$(t, r, \theta, \phi) \mapsto -\frac{2\left(a^4 q^2 \cos(\theta)^4 - 6\,a^2 q^2 r^2 \cos(\theta)^2 + q^2 r^4\right)}{a^8 \cos(\theta)^8 + 4\,a^6 r^2 \cos(\theta)^6 + 6\,a^4 r^4 \cos(\theta)^4 + 4\,a^2 r^6 \cos(\theta)^2 + r^8}$$

The energy-momentum tensor of the electromagnetic field:

```
T = 1/(4*pi)*( F.contract(0, F.up(g, 0), 0) - 1/4*F2 * g ); print T
    tensor field of type (0,2) on the 4-dimensional manifold 'M'
```

Check of the Einstein equation:

```
G == 8*pi*T
    True
```

Outline

Perspectives

- **SageManifolds** is a work in progress
  ($\sim$ 16,000 lines of Python code up to now)
- A preliminary version is available at `http://sagemanifolds.obspm.fr/`
- *Already present:* standard tensor calculus (tensor product, contraction, symmetrization, etc.), exterior calculus, Lie derivatives, affine connections, curvature, torsion, pseudo-Riemannian metrics, Weyl tensor, Hodge duality
- *Not implemented yet (but should be soon):* pullback and pushforward operators, extrinsic geometry of submanifolds
- *To do:* convert some parts to *Cython* in order to compile them (*C* code) and increase the computational speed
- *For future releases:* symplectic forms, fibre bundles, spinors, variational calculus