Exploring black hole spacetimes with SageManifolds

Éric Gourgoulhon

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

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

based on a collaboration with
Michał Bejger, Marco Mancini, Travis Scrimshaw

One Hundred Years of Strong Gravity
Instituto Superior Técnico, Lisbon
10-12 June 2015

Outline

Outline

Introduction

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

Introduction

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

Introduction

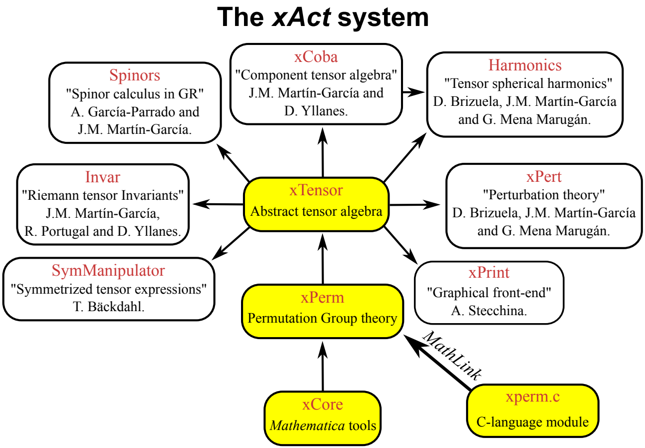
- Computer algebra system (CAS) started to be developed in the 1960's; for instance `Macsyma` (to become `Maxima` in 1998) was initiated in 1968 at MIT
- In 1965, J.G. Fletcher developed the `GEOM` program, to compute the Riemann tensor of a given metric
- In 1969, during his PhD under Pirani supervision,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

Introduction

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

Sage in a few words

- Sage (*full name:* SageMath) 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
- Sage is now supported by European Union via the Horizon 2020 project OpenDreamKit (2015-2019)

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 enthusiast community of developers

Outline

The SageManifolds project

http://sagemanifolds.obspm.fr/

Aim

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

In particular:

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

The SageManifolds project

http://sagemanifolds.obspm.fr/

Aim

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

In particular:

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

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

Implementing coordinate charts

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

Implementing coordinate charts

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

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

Examples:

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

Implementing coordinate charts

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

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

Examples:

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

In SageManifolds, an arbitrary number of charts can be introduced

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

Implementing scalar fields

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

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

where U is an open subset of M.

Implementing scalar fields

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

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

where U is an open subset of M.

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

Implementing scalar fields

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

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

where U is an open subset of M.

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

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

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

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

The scalar field algebra

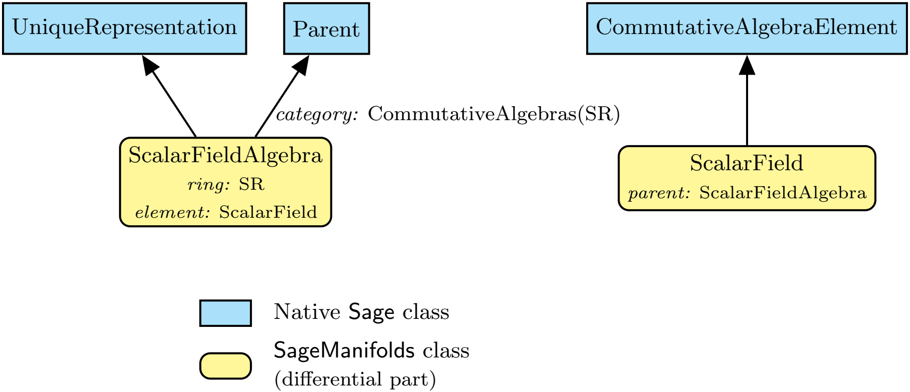
Given an open subset $U \subset M$, the set $C^\infty(U)$ of scalar fields defined on U has naturally the structure of a **commutative algebra over** $\mathbb{R}$:

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

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

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

Classes for scalar fields



Vector field modules

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

Vector field modules

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

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**

Vector field modules

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

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

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

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

Example:

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

Parallelizable manifolds

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

Parallelizable manifolds

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

Examples of parallelizable manifolds

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

Parallelizable manifolds

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

Examples of parallelizable manifolds

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

Examples of non-parallelizable manifolds

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

Implementing vector fields

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

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

Implementing vector fields

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

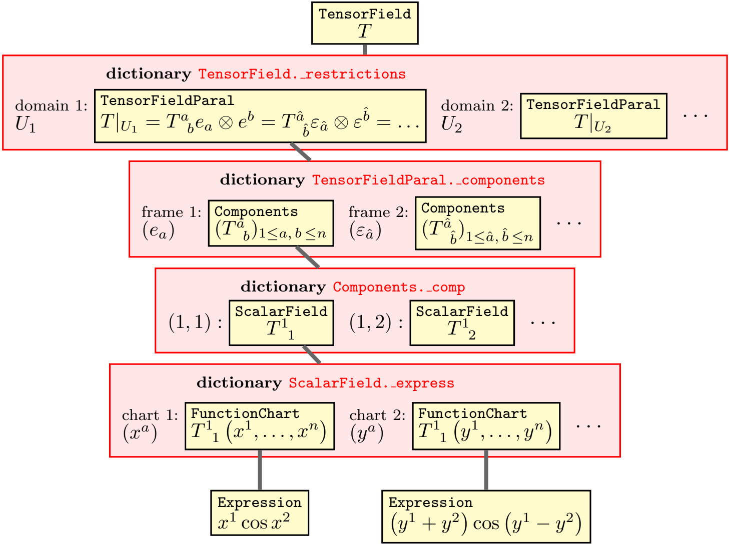
If the manifold M is not parallelizable, we assume that it can be covered by a finite number N of parallelizable open subsets U_i $(1 \leq i \leq N)$ (OK for M compact). We then consider **restrictions** of vector fields to these domains:

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

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

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

Tensor field storage



Outline

Object-oriented notation in Python

(in order to understand what follows...)

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

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

In a procedural language, this would be written as

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

Object-oriented notation in Python

(in order to understand what follows...)

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

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

In a procedural language, this would be written as

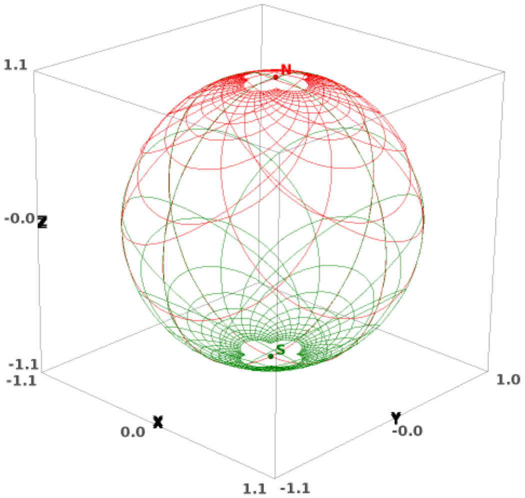
```
result = function(object,arguments)
```

Examples

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

NB: no argument in example 1

The 2-sphere example



Stereographic coordinates on the 2-sphere

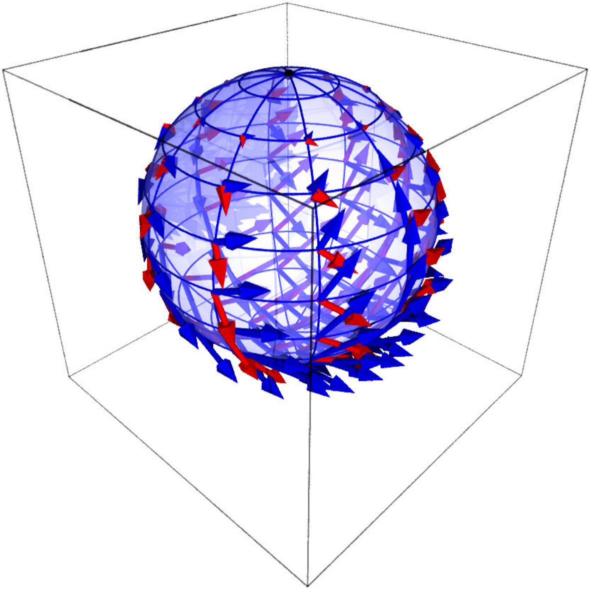
Two charts:

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

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

See the worksheet at
http://sagemanifolds.obspm.fr/examples/html/SM_sphere_S2.html

The 2-sphere example



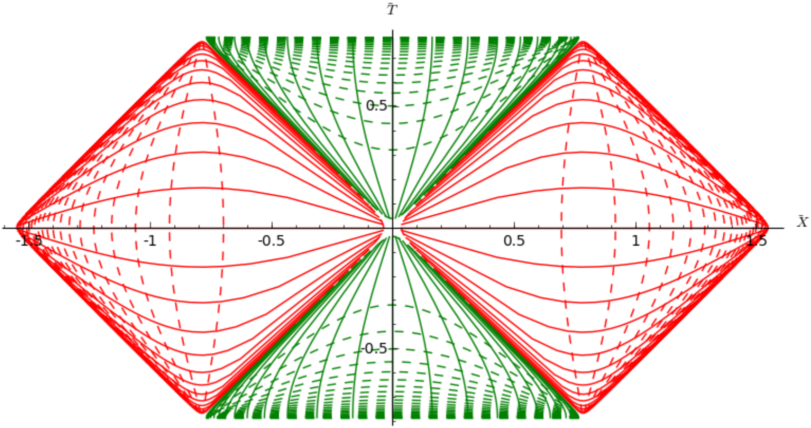
Vector frame associated with the stereographic coordinates (x, y) from the North pole

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

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

Charts on Schwarzschild spacetime

The Carter-Penrose diagram

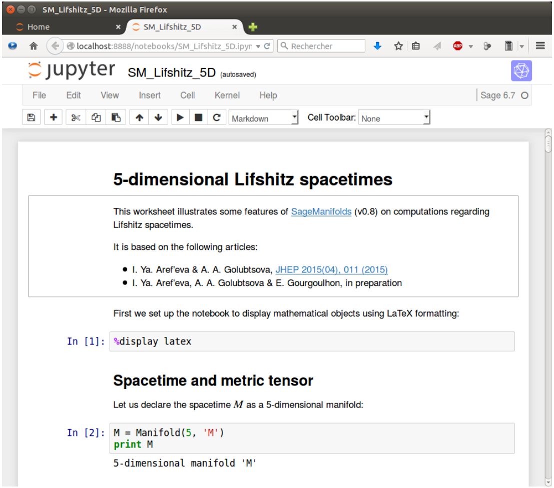


Two charts of standard Schwarzschild-Droste coordinates (t, r, θ, φ) plotted in terms of compactified coordinates $(\tilde{T}, \tilde{X}, \theta, \varphi)$

See the worksheet at

http://sagemanifolds.obspm.fr/examples/html/SM_Carter-Penrose_diag.html

5D Lifshitz spacetime example



A full example with Sage + SageManifolds running in a Jupyter notebook

Get/read the worksheet at
http://nbviewer.ipython.org/github/sagemanifolds/SageManifolds/blob/master/Worksheets/v0.8/SM_Lifshitz_5D.ipynb

5-dimensional Lifshitz spacetimes

This worksheet illustrates some features of SageManifolds (v0.8) on computations regarding Lifshitz spacetimes.

It is based on the following articles:

- I. Ya. Aref'eva & A. A. Golubtsova, JHEP 2015(04), 011 (2015)
- I. Ya. Aref'eva, A. A. Golubtsova & E. Gourgoulhon, in preparation

First we set up the notebook to display mathematical objects using LaTeX formatting:

In [1]:
```
%display latex
```

Spacetime and metric tensor

Let us declare the spacetime M as a 5-dimensional manifold:

In [2]:
```
M = Manifold(5, 'M')
print M
```

```
5-dimensional manifold 'M'
```

We introduce a first coordinate system on M:

```
In [3]: X0.<t,x,y1,y2,R> = M.chart('t x y1:y_1 y2:y_2 R:(0,+oo)')
        X0
```

Out[3]: $(M, (t, x, y_1, y_2, R))$

Let us consider the following Lifshitz-symmetric metric, parametrized by some real number ν:

```
In [4]: g = M.lorentz_metric('g')
        var('nu', latex_name=r'\nu', domain='real')
        g[0,0] = -R^(2*nu)
        g[1,1] = R^(2*nu)
        g[2,2] = R^2
        g[3,3] = R^2
        g[4,4] = 1/R^2
        g.display()
```

Out[4]:
$$g = -R^{2\nu} \mathrm{d}t \otimes \mathrm{d}t + R^{2\nu} \mathrm{d}x \otimes \mathrm{d}x + R^2 \mathrm{d}y_1 \otimes \mathrm{d}y_1 + R^2 \mathrm{d}y_2 \otimes \mathrm{d}y_2 + \frac{1}{R^2} \mathrm{d}R \otimes \mathrm{d}R$$

A matrix view of the metric components:

In [5]:
```
g[:]
```

Out[5]:
$$\begin{pmatrix} -R^{2\nu} & 0 & 0 & 0 & 0 \\ 0 & R^{2\nu} & 0 & 0 & 0 \\ 0 & 0 & R^2 & 0 & 0 \\ 0 & 0 & 0 & R^2 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{R^2} \end{pmatrix}$$

This metric is invariant under the *Lifshitz scaling*

$$(t, x, y_1, y_2, R) \longmapsto \left(\lambda^\nu t, \lambda^\nu x, \lambda y_1, \lambda y_2, \frac{R}{\lambda}\right)$$

- If $\nu = 1$ the scaling is isotropic and we recognize the metric of AdS_5 in Poincaré coordinates
- If $\nu \neq 1$, the scaling is anisotropic

Let us introduce a second coordinate system on M:

In [6]:
```
X.<t,x,y1,y2,r> = M.chart('t x y1:y_1 y2:y_2 r:(0,+oo)')
X
```

Out[6]: $(M, (t, x, y_1, y_2, r))$

and relate it to the previous one by the transformation $r = \ln R$:

In [7]:
```
X0_to_X = X0.transition_map(X, [t, x, y1, y2, ln(R)])
X0_to_X.display()
```

Out[7]:
$$\begin{cases} t & = & t \\ x & = & x \\ y_1 & = & y_1 \\ y_2 & = & y_2 \\ r & = & \log(R) \end{cases}$$

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

In [8]:
```
X_to_X0 = X0_to_X.inverse()
X_to_X0.display()
```

Out[8]:
$$\begin{cases} t & = & t \\ x & = & x \\ y_1 & = & y_1 \\ y_2 & = & y_2 \\ R & = & e^r \end{cases}$$

At this stage, the manifold's atlas defined by the user is

In [9]:
```
M.atlas()
```

Out[9]: $[(M,(t,x,y_1,y_2,R)),(M,(t,x,y_1,y_2,r))]$

and the list of defined vector frames defined is

In [10]:
```
M.frames()
```

Out[10]: $$\left[\left(M,\left(\frac{\partial}{\partial t},\frac{\partial}{\partial x},\frac{\partial}{\partial y_1},\frac{\partial}{\partial y_2},\frac{\partial}{\partial R}\right)\right),\left(M,\left(\frac{\partial}{\partial t},\frac{\partial}{\partial x},\frac{\partial}{\partial y_1},\frac{\partial}{\partial y_2},\frac{\partial}{\partial r}\right)\right)\right]$$

The expression of the metric in terms of the new coordinates is

In [11]:
```
g.display(X.frame(), X)
```

Out[11]: $$g=-e^{(2\,\nu r)}\mathrm{d}t\otimes\mathrm{d}t+e^{(2\,\nu r)}\mathrm{d}x\otimes\mathrm{d}x+e^{(2\,r)}\mathrm{d}y_1\otimes\mathrm{d}y_1+e^{(2\,r)}\mathrm{d}y_2\otimes\mathrm{d}y_2+\mathrm{d}r\otimes\mathrm{d}r$$

or, in matrix view:

In [12]:
```
g[X.frame(),:,X]
```

Out[12]:

$$\begin{pmatrix} -e^{(2\,\nu r)} & 0 & 0 & 0 & 0 \\ 0 & e^{(2\,\nu r)} & 0 & 0 & 0 \\ 0 & 0 & e^{(2\,r)} & 0 & 0 \\ 0 & 0 & 0 & e^{(2\,r)} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

To access to a particular component, we have to specify (i) the frame w.r.t. which it is defined and (ii) the coordinates in which the component is expressed:

In [13]:
```
g[X.frame(),0,0,X]
```

Out[13]: $-e^{(2\,\nu r)}$

In [14]:
```
g[X.frame(),0,0]  # the default chart is used
```

Out[14]: $-R^{2\,\nu}$

From now on, let us consider the coordinates $X = (t, x, y_1, y_2, r)$ as the default ones on the manifold M:

In [15]:
```
M.set_default_chart(X)
M.set_default_frame(X.frame())
```

Then

In [16]:
```
g.display()
```

Out[16]:

$$g = -e^{(2\,\nu r)} \mathrm{d}t \otimes \mathrm{d}t + e^{(2\,\nu r)} \mathrm{d}x \otimes \mathrm{d}x + e^{(2\,r)} \mathrm{d}y_1 \otimes \mathrm{d}y_1 + e^{(2\,r)} \mathrm{d}y_2 \otimes \mathrm{d}y_2 + \mathrm{d}r \otimes \mathrm{d}r$$

In [17]:
```
g[:]
```

Out[17]:

$$\begin{pmatrix} -e^{(2\,\nu r)} & 0 & 0 & 0 & 0 \\ 0 & e^{(2\,\nu r)} & 0 & 0 & 0 \\ 0 & 0 & e^{(2\,r)} & 0 & 0 \\ 0 & 0 & 0 & e^{(2\,r)} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

In [18]:
```
g[0,0]
```

Out[18]:

$$-e^{(2\,\nu r)}$$

In [19]:
```
g.display_comp()
```

Out[19]:

$$
\begin{aligned}
g_{t\,t} &= -e^{(2\,\nu r)} \\
g_{x\,x} &= e^{(2\,\nu r)} \\
g_{y_1\,y_1} &= e^{(2\,r)} \\
g_{y_2\,y_2} &= e^{(2\,r)} \\
g_{r\,r} &= 1
\end{aligned}
$$

Curvature

The Riemann tensor is

In [20]:
```
Riem = g.riemann()
print Riem
```

```
tensor field 'Riem(g)' of type (1,3) on the 5-dimensional manifold 'M'
```

In [21]:
```
Riem.display_comp(only_nonredundant=True)
```

Out[21]:

$$\mathrm{Riem}(g)^{t}{}_{x\,t\,x} = -\nu^2 e^{(2\,\nu r)}$$
$$\mathrm{Riem}(g)^{t}{}_{y_1\,t\,y_1} = -\nu e^{(2\,r)}$$
$$\mathrm{Riem}(g)^{t}{}_{y_2\,t\,y_2} = -\nu e^{(2\,r)}$$
$$\mathrm{Riem}(g)^{t}{}_{r\,t\,r} = -\nu^2$$
$$\mathrm{Riem}(g)^{x}{}_{t\,t\,x} = -\nu^2 e^{(2\,\nu r)}$$
$$\mathrm{Riem}(g)^{x}{}_{y_1\,x\,y_1} = -\nu e^{(2\,r)}$$
$$\mathrm{Riem}(g)^{x}{}_{y_2\,x\,y_2} = -\nu e^{(2\,r)}$$
$$\mathrm{Riem}(g)^{x}{}_{r\,x\,r} = -\nu^2$$
$$\mathrm{Riem}(g)^{y_1}{}_{t\,t\,y_1} = -\nu e^{(2\,\nu r)}$$
$$\mathrm{Riem}(g)^{y_1}{}_{x\,x\,y_1} = \nu e^{(2\,\nu r)}$$
$$\mathrm{Riem}(g)^{y_1}{}_{y_2\,y_1\,y_2} = -e^{(2\,r)}$$
$$\mathrm{Riem}(g)^{y_1}{}_{r\,y_1\,r} = -1$$
$$\mathrm{Riem}(g)^{y_2}{}_{t\,t\,y_2} = -\nu e^{(2\,\nu r)}$$
$$\mathrm{Riem}(g)^{y_2}{}_{x\,x\,y_2} = \nu e^{(2\,\nu r)}$$
$$\mathrm{Riem}(g)^{y_2}{}_{y_1\,y_1\,y_2} = e^{(2\,r)}$$
$$\mathrm{Riem}(g)^{y_2}{}_{r\,y_2\,r} = -1$$

$$\mathrm{Riem}(g)^{r}{}_{t\,t\,r} = -\nu^2 e^{(2\,\nu r)}$$
$$\mathrm{Riem}(g)^{r}{}_{x\,x\,r} = \nu^2 e^{(2\,\nu r)}$$
$$\mathrm{Riem}(g)^{r}{}_{y_1\,y_1\,r} = e^{(2\,r)}$$
$$\mathrm{Riem}(g)^{r}{}_{y_2\,y_2\,r} = e^{(2\,r)}$$

The Ricci tensor:

In [22]:
```
Ric = g.ricci()
print Ric
```

```
field of symmetric bilinear forms 'Ric(g)' on the 5-dimensional manifol
d 'M'
```

In [23]:
```
Ric.display()
```

Out[23]:

$$\mathrm{Ric}\,(g) = 2\left(\nu^2+\nu\right)e^{(2\,\nu r)}\mathrm{d}t\otimes\mathrm{d}t - 2\left(\nu^2+\nu\right)e^{(2\,\nu r)}\mathrm{d}x\otimes\mathrm{d}x - 2\,(\nu+1)e^{(2\,r)}\mathrm{d}y_1\otimes\mathrm{d}y_1 - 2\,(\nu+1)e^{(2\,r)}\mathrm{d}y_2\otimes\mathrm{d}y_2 + \left(-2\,\nu^2-2\right)\mathrm{d}r\otimes\mathrm{d}r$$

In [24]:
```
Ric.display_comp()
```

Out[24]:

$$\mathrm{Ric}(g)_{\,t\,t} = 2\left(\nu^2 + \nu\right)e^{(2\,\nu r)}$$
$$\mathrm{Ric}(g)_{\,x\,x} = -2\left(\nu^2 + \nu\right)e^{(2\,\nu r)}$$
$$\mathrm{Ric}(g)_{\,y_1\,y_1} = -2\,(\nu + 1)e^{(2\,r)}$$
$$\mathrm{Ric}(g)_{\,y_2\,y_2} = -2\,(\nu + 1)e^{(2\,r)}$$
$$\mathrm{Ric}(g)_{\,r\,r} = -2\,\nu^2 - 2$$

The Ricci scalar:

In [25]:
```
Rscal = g.ricci_scalar()
print Rscal
```

```
scalar field 'r(g)' on the 5-dimensional manifold 'M'
```

In [26]:
```
Rscal.display()
```

Out[26]:

$$\begin{array}{llcl} \mathrm{r}(g): & M & \longrightarrow & \mathbb{R} \\ & (t,x,y_1,y_2,R) & \longmapsto & -6\,\nu^2 - 8\,\nu - 6 \\ & (t,x,y_1,y_2,r) & \longmapsto & -6\,\nu^2 - 8\,\nu - 6 \end{array}$$

We note that the Ricci scalar is constant.

Source model

Let us consider a model based on the following action, involving a dilaton scalar field ϕ and a Maxwell 2-form F:

$$S = \int \left(R(g) + \Lambda - \frac{1}{2} \nabla_m \phi \nabla^m \phi - \frac{1}{4} e^{\lambda\phi} F_{mn} F^{mn} \right) \sqrt{-g}\, \mathrm{d}^5 x \qquad (1)$$

The dilaton scalar field

We consider the following ansatz for the dilaton scalar field ϕ:

$$\phi = \frac{1}{\lambda}(4r + \ln \mu)\,,$$

where λ and μ are two constants.

In [27]:
```
var('mu', latex_name=r'\mu')
var('lamb', latex_name=r'\lambda')
phi = M.scalar_field({X: (4*r + ln(mu))/lamb},
                     name='phi', latex_name=r'\phi')
phi.display()
```

Out[27]:

$$\begin{array}{llcl} \phi: & M & \longrightarrow & \mathbb{R} \\ & (t, x, y_1, y_2, R) & \longmapsto & \frac{4\ \log(R)+\log(\mu)}{\lambda} \\ & (t, x, y_1, y_2, r) & \longmapsto & \frac{4\, r+\log(\mu)}{\lambda} \end{array}$$

The 1-form $\mathrm{d}\phi$ is

In [28]:
```
dphi = phi.differential()
print dphi
```

```
1-form 'dphi' on the 5-dimensional manifold 'M'
```

In [29]:
```
dphi.display()
```

Out[29]:

$$\mathrm{d}\phi = \frac{4}{\lambda}\mathrm{d}r$$

In [30]:
```
dphi[:]  # all the components in the default frame
```

Out[30]:

$$\left[0, 0, 0, 0, \frac{4}{\lambda}\right]$$

The 2-form field

We consider the following ansatz for F:

$$F = \frac{1}{2} q \, \mathrm{d}y_1 \wedge \mathrm{d}y_2,$$

where q is a constant.

Let us first get the 1-forms $\mathrm{d}y_1$ and $\mathrm{d}y_2$:

In [31]:
```
X.coframe()
```

Out[31]: $(M, (\mathrm{d}t, \mathrm{d}x, \mathrm{d}y_1, \mathrm{d}y_2, \mathrm{d}r))$

In [32]:
```
dy1 = X.coframe()[2]
dy2 = X.coframe()[3]
print dy1
print dy2
dy1, dy2
```

```
1-form 'dy1' on the 5-dimensional manifold 'M'
1-form 'dy2' on the 5-dimensional manifold 'M'
```

Out[32]: $(\mathrm{d}y_1, \mathrm{d}y_2)$

We can then form F according to the above ansatz:

In [33]:
```
var('q')
F = q/2 * dy1.wedge(dy2)
F.set_name('F')
print F
F.display()
```

```
2-form 'F' on the 5-dimensional manifold 'M'
```

Out[33]:
$$F = \frac{1}{2} q \mathrm{d}y_1 \wedge \mathrm{d}y_2$$

By construction, the 2-form F is closed (since q is constant):

In [34]:
```
print xder(F)
```

```
3-form 'dF' on the 5-dimensional manifold 'M'
```

In [35]:
```
xder(F).display()
```

Out[35]: $\mathrm{d}F = 0$

Let us evaluate the square $F_{mn}F^{mn}$ of F:

In [36]:
```
Fu = F.up(g)
print Fu
Fu.display()
```

```
tensor field of type (2,0) on the 5-dimensional manifold 'M'
```

Out[36]:
$$\frac{1}{2}\, q e^{(-4\, r)} \frac{\partial}{\partial y_1} \otimes \frac{\partial}{\partial y_2} - \frac{1}{2}\, q e^{(-4\, r)} \frac{\partial}{\partial y_2} \otimes \frac{\partial}{\partial y_1}$$

In [37]:
```
F2 = F['_{mn}']*Fu['^{mn}']  # using LaTeX notations to denote contracti
print F2
F2.display()
```

```
scalar field on the 5-dimensional manifold 'M'
```

Out[37]:

$$\begin{array}{llcl} & M & \longrightarrow & \mathbb{R} \\ & (t,x,y_1,y_2,R) & \longmapsto & \frac{q^2}{2\,R^4} \\ & (t,x,y_1,y_2,r) & \longmapsto & \frac{1}{2}\,q^2e^{(-4\,r)} \end{array}$$

We shall also need the tensor $\mathcal{F}_{mn} = F_{mp}F_n{}^p$:

In [38]:
```
FF = F['_mp'] * F.up(g,1)['^p_n']
print FF
FF.display()
```

```
tensor field of type (0,2) on the 5-dimensional manifold 'M'
```

Out[38]: $\frac{1}{4}\,q^2e^{(-2\,r)}\mathrm{d}y_1 \otimes \mathrm{d}y_1 + \frac{1}{4}\,q^2e^{(-2\,r)}\mathrm{d}y_2 \otimes \mathrm{d}y_2$

The tensor field $\mathcal{F}$ is symmetric:

In [39]:
```
FF == FF.symmetrize()
```
Out[39]: True

Therefore, from now on, we set

In [40]:
```
FF = FF.symmetrize()
```

Field equations

Einstein equation

Let us first introduce the cosmological constant:

In [41]:
```
var('Lamb', latex_name=r'\Lambda')
```
Out[41]: Λ

From the action (1), the field equation for the metric g is

$$R_{mn} + \frac{\Lambda}{3} g - \frac{1}{2}\partial_m\phi\partial_n\phi - \frac{1}{2} e^{\lambda\phi} F_{mp} F_n{}^p + \frac{1}{12} e^{\lambda\phi} F_{rs} F^{rs} g_{mn} = 0$$

We write it as

```
EE == 0
```

with EE defined by

```
In [42]: EE = Ric + Lamb/3*g - 1/2* (dphi*dphi) -  1/2*exp(lamb*phi)*FF \
              + 1/12*exp(lamb*phi)*F2*g
         EE.set_name('E')
         print EE
```

field of symmetric bilinear forms 'E' on the 5-dimensional manifold 'M'

```
In [43]: EE.display_comp(only_nonredundant=True)
```

Out[43]:

$$E_{tt} = -\frac{1}{24}\left(\mu q^2 - 48\,\nu^2 + 8\,\Lambda - 48\,\nu\right)e^{(2\,\nu r)}$$

$$E_{xx} = \frac{1}{24}\left(\mu q^2 - 48\,\nu^2 + 8\,\Lambda - 48\,\nu\right)e^{(2\,\nu r)}$$

$$E_{y_1 y_1} = -\frac{1}{12}\left(\mu q^2 - 4\,\Lambda + 24\,\nu + 24\right)e^{(2\,r)}$$

$$E_{y_2 y_2} = -\frac{1}{12}\left(\mu q^2 - 4\,\Lambda + 24\,\nu + 24\right)e^{(2\,r)}$$

$$E_{rr} = \frac{\lambda^2\mu q^2 - 48\,\lambda^2\nu^2 + 8\,(\Lambda-6)\lambda^2 - 192}{24\,\lambda^2}$$

We note that EE==0 leads to only 3 independent equations:

In [44]:
```
eq1 = (EE[0,0]/exp(2*nu*r)).expr()
eq1
```

Out[44]:
$$-\frac{1}{24}\,\mu q^2 + 2\,\nu^2 - \frac{1}{3}\,\Lambda + 2\,\nu$$

In [45]:
```
eq2 = (EE[2,2]/exp(2*r)).expr()
eq2
```

Out[45]:
$$-\frac{1}{12}\,\mu q^2 + \frac{1}{3}\,\Lambda - 2\,\nu - 2$$

In [46]:
```
eq3 = EE[4,4].expr().expand()
eq3
```

Out[46]:
$$\frac{1}{24}\,\mu q^2 - 2\,\nu^2 + \frac{1}{3}\,\Lambda - \frac{8}{\lambda^2} - 2$$

Dilaton field equation

First we evaluate $\nabla_m \nabla^m \phi$:

In [47]:
```
nab = g.connection()
print nab
nab
```

```
Levi-Civita connection 'nabla_g' associated with the Lorentzian metric
'g' on the 5-dimensional manifold 'M'
```

Out[47]: ∇_g

In [48]:
```
box_phi = nab(nab(phi).up(g)).trace()
print box_phi
box_phi.display()
```

```
scalar field on the 5-dimensional manifold 'M'
```

Out[48]:

$$\begin{array}{llcl} & M & \longrightarrow & \mathbb{R} \\ & (t,x,y_1,y_2,R) & \longmapsto & \frac{8\,(\nu+1)}{\lambda} \\ & (t,x,y_1,y_2,r) & \longmapsto & \frac{8\,(\nu+1)}{\lambda} \end{array}$$

From the action (1), the field equation for ϕ is

$$\nabla_m \nabla^m \phi = \frac{\lambda}{4} e^{\lambda\phi} F_{mn} F^{mn}$$

We write it as

```
DE == 0
```

with DE defined by

In [49]:
```
DE = box_phi - lamb/4*exp(lamb*phi) * F2
print DE
```

```
scalar field on the 5-dimensional manifold 'M'
```

In [50]:
```
DE.display()
```

Out[50]:

$$\begin{array}{llcl} M & \longrightarrow & \mathbb{R} \\ (t, x, y_1, y_2, R) & \longmapsto & -\frac{\lambda^2 \mu q^2 - 64\,\nu - 64}{8\,\lambda} \\ (t, x, y_1, y_2, r) & \longmapsto & -\frac{\lambda^2 \mu q^2 - 64\,\nu - 64}{8\,\lambda} \end{array}$$

Hence the dilaton field equation provides a fourth equation:

In [51]:
```
eq4 = DE.expr().expand()
eq4
```

Out[51]: $$-\frac{1}{8}\lambda\mu q^2 + \frac{8\nu}{\lambda} + \frac{8}{\lambda}$$

Maxwell equation

From the action (1), the field equation for F is

$$\nabla_m\left(e^{\lambda\phi}F^{mn}\right) = 0$$

We write it as

```
ME == 0
```

with ME defined by

In [52]:
```
ME = nab(exp(lamb*phi)*Fu).trace(0,2)
print ME
ME.display()
```

```
vector field on the 5-dimensional manifold 'M'
```

Out[52]: 0

We get identically zero; indeed the tensor $\nabla_p(e^{\lambda\phi}F^{mn})$ has a vanishing trace, as we can check:

In [53]:
```
nab(exp(lamb*phi)*Fu).display()
```

Out[53]:

$$\mu q \frac{\partial}{\partial y_1} \otimes \frac{\partial}{\partial y_2} \otimes \mathrm{d}r - \frac{1}{2}\mu q e^{(2r)} \frac{\partial}{\partial y_1} \otimes \frac{\partial}{\partial r} \otimes \mathrm{d}y_2 - \mu q \frac{\partial}{\partial y_2} \otimes \frac{\partial}{\partial y_1} \otimes \mathrm{d}r + \frac{1}{2}\mu q e^{(2r)} \frac{\partial}{\partial y_2} \otimes \frac{\partial}{\partial r} \otimes \mathrm{d}y_1 + \frac{1}{2}\mu q e^{(2r)} \frac{\partial}{\partial r} \otimes \frac{\partial}{\partial y_1} \otimes \mathrm{d}y_2 - \frac{1}{2}\mu q e^{(2r)} \frac{\partial}{\partial r} \otimes \frac{\partial}{\partial y_2} \otimes \mathrm{d}y_1$$

Summary

We have 4 equations involving the constants λ, μ, ν, q and Λ:

```
In [54]: eq1 == 0
```

Out[54]:

$$-\frac{1}{24}\,\mu q^2 + 2\,\nu^2 - \frac{1}{3}\,\Lambda + 2\,\nu = 0$$

```
In [55]: eq2 == 0
```

Out[55]:

$$-\frac{1}{12}\,\mu q^2 + \frac{1}{3}\,\Lambda - 2\,\nu - 2 = 0$$

```
In [56]: eq3 == 0
```

Out[56]:

$$\frac{1}{24}\,\mu q^2 - 2\,\nu^2 + \frac{1}{3}\,\Lambda - \frac{8}{\lambda^2} - 2 = 0$$

```
In [57]: eq4 == 0
```

Out[57]:

$$-\frac{1}{8}\,\lambda\mu q^2 + \frac{8\,\nu}{\lambda} + \frac{8}{\lambda} = 0$$

Solution for $\nu = 1$ (AdS_5)

In [58]:
```
eqs = [eq1, eq2, eq3, eq4]
neqs = [eq.subs(nu=1) for eq in eqs]
```

In [59]:
```
[eq == 0 for eq in neqs]
```

Out[59]:

$$\left[-\frac{1}{24}\,\mu q^2 - \frac{1}{3}\,\Lambda + 4 = 0, -\frac{1}{12}\,\mu q^2 + \frac{1}{3}\,\Lambda - 4 = 0, \frac{1}{24}\,\mu q^2 + \frac{1}{3}\,\Lambda - \frac{8}{\lambda^2} - 4 = 0, -\frac{1}{8}\,\lambda\mu q^2 + \frac{16}{\lambda} = 0\right]$$

In [60]:
```
solve([eq == 0 for eq in neqs], lamb, mu, Lamb, q)
```

Out[60]: []

Hence there is no solution for AdS_5 with the above ansatz.

Remark

This is related to the **positive energy theorem of AdS** mentioned by Akihiro Ishibashi in his talk: $(M, g) = \mathrm{AdS} \iff E = 0$.

Solution for $\nu = 2$

In [61]:
```
neqs = [eq.subs(nu=2) for eq in eqs]
[eq == 0 for eq in neqs]
```

Out[61]:

$$\left[-\frac{1}{24}\,\mu q^2 - \frac{1}{3}\,\Lambda + 12 = 0, -\frac{1}{12}\,\mu q^2 + \frac{1}{3}\,\Lambda - 6 = 0, \frac{1}{24}\,\mu q^2 + \frac{1}{3}\,\Lambda - \frac{8}{\lambda^2} - 10 = 0, -\frac{1}{8}\,\lambda\mu q^2 + \frac{24}{\lambda} = 0\right]$$

In [62]:
```
solve([eq == 0 for eq in neqs], lamb, mu, Lamb, q)
```

Out[62]:

$$\left[\left[\lambda = 2, \mu = \frac{48}{r_1^2}, \Lambda = 30, q = r_1\right], \left[\lambda = (-2), \mu = \frac{48}{r_2^2}, \Lambda = 30, q = r_2\right]\right]$$

Hence there are two families of solutions, each famility being parametrized by e.g. q.

Solution for $\nu = 4$

```
In [63]: neqs = [eq.subs(nu=4) for eq in eqs]
         [eq == 0 for eq in neqs]
```

Out[63]:

$$\left[-\frac{1}{24}\,\mu q^2 - \frac{1}{3}\,\Lambda + 40 = 0, -\frac{1}{12}\,\mu q^2 + \frac{1}{3}\,\Lambda - 10 = 0, \frac{1}{24}\,\mu q^2 + \frac{1}{3}\,\Lambda - \frac{8}{\lambda^2} - 34 = 0, -\frac{1}{8}\,\lambda\mu q^2 + \frac{40}{\lambda} = 0\right]$$

```
In [64]: solve([eq == 0 for eq in neqs], lamb, mu, Lamb, q)
```

Out[64]:

$$\left[\left[\lambda = \frac{2}{3}\,\sqrt{3}, \mu = \frac{240}{r_3^2}, \Lambda = 90, q = r_3\right], \left[\lambda = -\frac{2}{3}\,\sqrt{3}, \mu = \frac{240}{r_4^2}, \Lambda = 90, q = r_4\right]\right]$$

Hence there are two families of solutions, each family being parametrized by e.g. q.

Outline

Conclusion and perspectives

- **SageManifolds** is a work in progress
  $\sim$ 51,000 lines of Python code up to now (including comments and doctests)
- A preliminary version (v0.8) is freely available (GPL) at
  `http://sagemanifolds.obspm.fr/`
  and the development version is available from the Git repository
  `https://github.com/sagemanifolds/sage`

Current status

Already present (v0.8):

- maps between manifolds, pullback operator
- submanifolds, pushforward operator
- curves in manifolds
- standard tensor calculus (tensor product, contraction, symmetrization, etc.), even on non-parallelizable manifolds
- all monoterm tensor symmetries
- exterior calculus (wedge product, exterior derivative, Hodge duality)
- Lie derivatives of tensor fields
- affine connections, curvature, torsion
- pseudo-Riemannian metrics, Weyl tensor
- some plotting capabilities (charts, points, curves, vector fields)
- parallelization (on tensor components) of CPU demanding computations, via the Python library `multiprocessing`

Current status

- *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

Current status

- *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
- *Future prospects:*
  - add more graphical outputs
  - add more functionalities: symplectic forms, fibre bundles, spinors, variational calculus, etc.
  - connection with numerical relativity: using Sage to explore numerically-generated spacetimes

Integration into Sage

SageManifolds is aimed to be fully integrated into Sage

- The algebraic part (tensors on free modules of finite rank) has been submitted to Sage Trac as ticket #15916 and got a positive review $\Longrightarrow$ integrated in Sage 6.6
- The differential part is being split in various tickets for submission to Sage Trac (cf. the metaticket #18528); meanwhile, one has to download it from http://sagemanifolds.obspm.fr/
- SageManifolds v0.8 is installed in the SageMathCloud $\Longrightarrow$ open a free account and use it online: https://cloud.sagemath.com/

Acknowledgements: the SageManifolds project has benefited from many discussions with Sage developers around the world, and especially in Paris area

Want to join the project or simply to stay tuned?

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