Cadabra and `python` algorithms in general relativity and cosmology I: Generalities

O. Castillo-Felisola

*Departamento de Física and Centro Científico y Tecnológico de Valparaíso,
Universidad Técnica Federico Santa María, Casilla 110-V, Valparaíso, Chile.*

D. T. Price

*Department of Mathematical Sciences, Durham University,
South Road, Durham DH1 3LE, United Kingdom.*

M. Scomparin

Via del Grano 33, Mogliano Veneto, Italy.

The aim of this work is to present a series of concrete examples which illustrate how the computer algebra system `Cadabra` can be used to manipulate expressions appearing in General Relativity and other gravitational theories. We highlight the way in which `Cadabra`'s philosophy differs from other systems with related functionality. The use of various new built-in packages is discussed, and we show how such packages can also be created by end-users directly using the notebook interface.

The current paper focuses on fairly generic applications in gravitational theories, including the use of differential forms, the derivation of field equations and the construction of their solutions.

1. Introduction

Algebraic manipulation of mathematical expressions is a common but tedious part of most research in physics. Symbolic computer algebra software has been used from the early days to help with this, and many special-purpose systems have been built to deal with expression manipulations specific to particular areas in physics. This is in particular true for research in gravity; for a recent review of the many uses of symbolic computer algebra in this field, see Ref. [1].

`Cadabra` is a relatively new, free and open-source standalone computer algebra system[i], which was designed from the ground up to manipulate mathematical expressions which occur in classical and quantum field theory [2-5]. In contrast to many special-purpose systems written for sometimes very specific tasks, it aims to provide a wide variety of basic field-theory building blocks, not only to tackle gravity computations but also to provide support for things like fermions and anti-commuting variables, algebra-valued objects, component and abstract computations, tensor symmetries and various others. Its main philosophy is to provide a simple-to-use 'scratchpad' for computations in field theory in its widest sense, to help with computations which are too tedious to do by hand, while keeping them close in form to what those computations would look like on paper. It is programmable in `Python`, yet also accepts mathematical expressions in standard LaTeX notation. It has been used in a wide variety of computations in high-energy physics and gravity, but also in different fields such as nuclear physics [6].

Because `Cadabra` tries to encourage and support a work-flow which is close to how computations are done with pencil and paper, it sometimes differs quite strongly from other computer algebra systems with a wide scope. In the present paper, and its followup companion [7], our goal is to show how gravity computations (which will at least in spirit be familiar to many readers) can be done with `Cadabra`. For a deeper look into using the system for advanced gravity computations, see *e.g.* [9].

The approach of `Cadabra` is that (geometrical) objects are first declared by assigning `properties` to objects, after which they can then be manipulated with `algorithms`, which act according to the previously assigned properties. A brief example is in order[ii],

```
1    {a,b}::NonCommuting.
2    {a,b}::Distributable.
3    {b}::SelfAntiCommuting.
4    expr := b ( a + b + a b );
5    distribute(_);
```

$$b\,(a + b + ab), \qquad ba + bab, \tag{1}$$

In the above example, a and b are the *objects*, and we assign the properties `NonCommuting` and `Distributable` to them, and additionally assign the property `SelfAntiCommuting` to the object b. The first assignment forbids the rearrangement of bab as ab^2, while the

last assignment ensures that $b^2 = 0$. Notice that output is shown only when a command ends with a semicolon (;).

This paper is organised as follows. Section 2 briefly introduces the basic concepts of General Relativity (see Sec. 2.1) and their implementation in Cadabra (see Sec. 2.2). The code in Sec. 2.2 is intended to serve as a *header* file, which can be called from other Cadabra notebooks in order to avoid the declaration of "standard" properties. In Sec. 3 we exemplify the manipulation of tensor expressions by writing down explicit expressions for the Lanczos–Lovelock Lagrangians in Sec. 3.1, and their field equations in Sec. 3.2. In Sec. 4 we explore the capabilities of Cadabra to manipulate differential forms. Specifically, we obtain the Bianchi identities from the structural equations[iii]. Next, in Sec. 5 the variational principle is exemplified by extremising the Einstein–Hilbert action. For reasons of space, the variation of the Lanczos–Lovelock action is not addressed in this paper. We then deal with the resolution of Einstein field equations in Sec. 6, solving in particular the Schwarzschild spacetime. Some conclusions are drawn in the Sec. 7. In Appendix A we introduce a tool that could help to improve the performance of calculations or long routines.

2. Formalism

2.1. Introduction to the formalism of General Relativity

Let us start by giving a brief reminder of the ingredients of General Relativity, both to set our conventions and to prepare for the discussion of its properties formulated in Cadabra's language. General Relativity is currently the best model of gravitational interactions, and was proposed in 1915 by A. Einstein [9,10], as an attempt to conciliate the concepts introduced by the special theory of relativity with those of gravitation. In his model, Einstein proposed that the gravitational interaction is an effect of the curvature of the spacetime. Meanwhile, the matter distribution determines how the spacetime curves. This is sometimes called the geometrisation of gravity.

Since the theory has to be invariant under general coordinate transformations, its building blocks are *tensors* (or more generally *tensor densities*). In General Relativity the spacetime is assumed to be a pseudo-Riemannian manifold, whose geometry is completely characterised by the metric tensor, $g_{\mu\nu}$. In order for the derivative of a tensor to be a tensor, the concept of connection ($\Gamma^\lambda{}_{\mu\nu}$) has to be introduced, allowing to define a *covariant derivative* ($\nabla_\mu = \partial_\mu + \Gamma^\bullet{}_{\mu\bullet}$). The condition of metricity, *i.e.* $\nabla g = 0$, relates the connection (the Levi-Civita connection) to the metric and partial derivatives of it,

$$\Gamma^\mu{}_{\nu\tau} = \frac{1}{2}g^{\mu\sigma}(\partial_\tau g_{\nu\sigma} + \partial_\nu g_{\tau\sigma} - \partial_\sigma g_{\nu\tau}). \tag{2}$$

The Levi-Civita connection is symmetric in its lower indices, $\Gamma^\mu{}_{\nu\tau} = \Gamma^\mu{}_{\tau\nu}$, this property is referred as *torsion-free condition*.

The action of the commutator of covariant derivatives on a vector yields an algebraic operator, dubbed the *curvature tensor*,

$$[\nabla_\mu, \nabla_\nu]V^\tau = R^\tau{}_{\sigma\mu\nu}V^\sigma,$$

where

$$R^\tau{}_{\sigma\mu\nu} = \partial_\mu\Gamma^\tau{}_{\nu\sigma} - \partial_\nu\Gamma^\tau{}_{\mu\sigma} + \Gamma^\tau{}_{\mu\lambda}\Gamma^\lambda{}_{\nu\sigma} - \Gamma^\tau{}_{\nu\lambda}\Gamma^\lambda{}_{\mu\sigma}. \tag{3}$$

The curvature tensor, also known as the Riemann tensor, is skew-symmetric in the last two indices, and additionally satisfies the algebraic and differential Bianchi identities,

$$R^\tau{}_{\sigma\mu\nu} + R^\tau{}_{\mu\nu\sigma} + R^\tau{}_{\nu\sigma\mu} = 0,$$

$$\nabla_\lambda R^\tau{}_{\sigma\mu\nu} + \nabla_\mu R^\tau{}_{\sigma\nu\lambda} + \nabla_\nu R^\tau{}_{\sigma\lambda\mu} = 0.$$

The contraction of the Riemann tensor are interesting geometrical quantities,

$$R_{\sigma\nu} = R^\tau{}_{\sigma\tau\nu} \quad \text{and} \quad R = g^{\mu\nu}R_{\mu\nu}, \tag{4}$$

called the Ricci tensor and Ricci scalar (curvature) respectively.

From a physics perspective, the relevant geometrical object is the *Einstein tensor*,

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R. \tag{5}$$

The field equations of General Relativity are obtained by extremising the Einstein–Hilbert action,

$$S = \frac{1}{2\kappa}\int \mathrm{d}^4x\,\sqrt{-g}\,(R - 2\Lambda), \tag{6}$$

where κ is the coupling constant of gravity (inversely proportional to the gravitational Newton constant G_N), Λ is the cosmological constant, and the symbol g stands for the determinant of the metric tensor[iv].

The interaction between matter and gravity is (formally) achieved through the minimal coupling mechanism, *i.e.* starting from the action on a flat spacetime and replacing the partial derivative by covariant derivative, the Minkowski metric by the curved metric, and the flat volume measure d^4x by the invariant volume measure $\mathrm{d}^4x\sqrt{-g}$. Hence, to the action in Eq. (6) one adds the matter action for a generic field ψ,

$$S_{\mathrm{mat}} = \int \mathrm{d}^4x\sqrt{-g}\mathcal{L}_{\mathrm{mat}}(\psi, g, \nabla\psi). \tag{7}$$

This then leads to Einstein's equations, which set the Einstein tensor $G_{\mu\nu}$ proportional to the energy-momentum tensor, $T_{\mu\nu}$, which encodes the properties of the matter distribution.
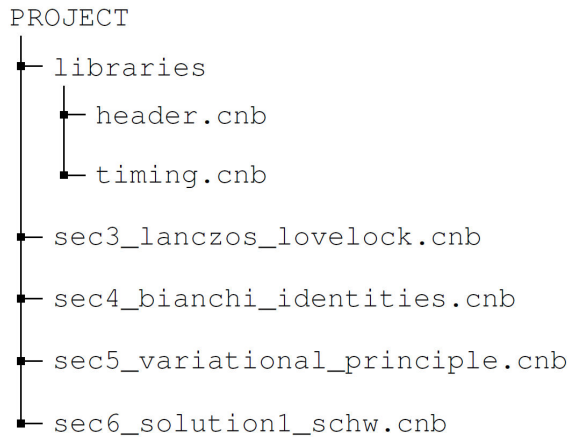
```
PROJECT
├── libraries
│   ├── header.cnb
│   └── timing.cnb
├── sec3_lanczos_lovelock.cnb
├── sec4_bianchi_identities.cnb
├── sec5_variational_principle.cnb
└── sec6_solution1_schw.cnb
```

FIGURE 1. Organisation of notebooks discussed in the present paper.

2.2. Introduction to the formalism of Cadabra: The header.cnb library

The code presented in this article is organised as a project containing a notebook for each chapter, as well as a separate library with two Cadabra packages which can be re-used in other computations[v]. The structure of the project is depicted in Fig. 1.

The *header.cnb* notebook is included at the start of some of the notebooks and defines a set of objects and related properties which will be used throughout the discussion of tensor perturbations in General Relativity. Providing this structure also allows us to define a useful workflow for the follow-up to this paper [7]. The purpose of this is to avoid repetitive declarations and ensure consistency between the notebooks. The file begins by importing the global dependencies:

```
1  import sympy
2  import cdb.core.manip as manip
3  import cdb.core.component as comp
4  import cdb.sympy.solvers as solv
```

The last three imports are from the Cadabra standard library [11], and provide common operations. The cdb.core.comp library is useful for component calculations and cdb.sympy.solvers is a simple Cadabra wrapper for the equation solvers provided in the SymPy library [12].

Although these imports appear to be regular Python packages, they are in fact Cadabra notebooks and can be found at <cadabra-root-dir>/lib/python3.x/site-packages/cdb. When Cadabra finds an import statement, it will not only do a standard search in sys.path for Python packages, but it will also search for notebook files, which are automatically converted into Python scripts and imported using the native functionality. Not only does this make writing packages for Cadabra very natural, but it also makes these packages very easy to read, as the documentation is written next to the code in LaTeX cells using the \algorithm command, and test code can be written

under the exported functions in ghost cells which are ignored when imported, similarly to how if __ name__ == " __main __ " statements are used in Python.

After this, the *header.cnb* file defines one main function init_properties, which accepts a coordinates parameter containing the range of coordinates used through the notebooks and a metrics parameter with the names of the metric tensors required, and uses this to inject appropriate property declarations into the Cadabra kernel. It begins by declaring the coordinates and indices used in the notebooks:

```
5   def init_properties(*, coordinates, metrics=[$g_{\mu
        \nu}$], signature=-1):
6       """
7       Initialise common property declarations which are
            shared between the
8       two papers
9       """
10      @(coordinates)::Coordinate.
11      index_list := {\mu,\nu,\rho,\sigma,\alpha,\beta,\
            gamma,\tau,\chi,\psi,\lambda,\lambda#}.
12      @(index_list)::Indices(position=independent,
            values=@(coordinates)).
13      Integer(index_list, Ex(rf"1..{len(coordinates)}"))
```

The use of the *pull-in* syntax @(...) allows us to use Cadabra expressions inside other expressions, similarly to how curly brackets are used in Python strings to include other objects (*e.g.* 'This is some {}'.format('text')). As the ::Property syntax expects a Cadabra expression on the left-hand side, not a variable name, we use it here to declare properties on these expressions which are not hard-coded. As well as assigning the Indices property to our index list, we also assign the Integer property which makes the number of coordinates countable, allowing functions like eliminate_kronecker which makes the substitution $\delta_\mu^\mu \to D$ to work. One final thing to note is the #, which declares an infinite set of labelled indices (*i.e.* \lambda1, \lambda2) which is useful to ensure that spare indices are always available (useful when running code in loops and when doing higher perturbative orders of a computation, for which it is not always easy to estimate how many dummy indices will be required).

The function then associates the relevant properties to the metrics defined in the metrics parameter:

```
15      sig = Ex(signature)
16      for metric in metrics:
17          # Both lower indices: Metric
18          for index in metric.top().indices():
19              index.parent_rel = parent_rel_t.sub
20          @(metric)::Metric(signature=@(sig)).
21
22          # Both upper indices: InverseMetric
23          for index in metric.top().indices():
24              index.parent_rel = parent_rel_t.super
25          @(metric)::InverseMetric(signature=@(sig)).
```

```
26
27       # Mixed indices: KroneckerDelta
28       for index in metric.top().indices():
29           index.parent_rel = parent_rel_t.sub
30           @(metric)::KroneckerDelta.
31           index.parent_rel = parent_rel_t.super
```

Here the `for` loops act over the two indices of the metric, lowering them for the `Metric` declaration and raising them for the `InverseMetric` declaration. Note that the $(1,1)$ forms of the metric tensor (*e.g.* $g^\mu{}_\nu$) must be declared as `KroneckerDelta`. After this, some standard symbols are defined

```
33       # Symbols
34       i::ImaginaryI.
35       \delta{#}::KroneckerDelta.
36       \partial{#}::PartialDerivative.
37       \nabla{#}::Derivative.
38       d{#}::Derivative.
```

Finally, we introduce some of the standard General Relativity objects. For each one, multiple properties must be defined—we begin by assigning a `Cadabra` identifier to each one and assigning a `LaTeXForm` to it, which controls how it will be rendered in LaTeX. The symmetries and dependencies of each are then defined; as `Cadabra` makes very few assumptions about the objects one uses any dependence on derivatives must be explicitly stated.

```
40       # LaTeX Typography
41       ch{#}::LaTeXForm("\Gamma").
42       {rm{#},rc{#},sc}::LaTeXForm("R").
43       ei{#}::LaTeXForm("G").
44       Lmb{#}::LaTeXForm("\Lambda").
45
46       # Symmetries
47       ch^{\rho}_{\mu\nu}::TableauSymmetry(shape={2},
             indices={1,2}).
48       rm^{\rho}_{\sigma\mu\nu}::TableauSymmetry(shape={1
             ,1}, indices={2,3}).
49       rc_{\mu\nu}::Symmetric.
50       ei_{\mu\nu}::Symmetric.
```

This completes the `init_properties` function, and the remainder of the `header.cnb` file are some algebraic definitions of the General Relativity objects that are used throughout the notebooks, allowing them to be substituted for each other. One can check the sanity of the definition with a few testing lines,

```
init_properties(coordinates=$t,x,y,z$, metrics=[$g_
    {\mu\nu}$, $\eta_{\mu\nu}$])
# Test that properties are declared correctly
assert eliminate_metric($g_{\mu\rho}x^{\rho\nu}$) ==
    $x_{\mu}^{\nu}$
assert eliminate_kronecker($g_{\mu}^{\rho}x_{\rho\nu
    }$) == $x_{\mu\nu}$
assert eliminate_kronecker($\eta^{\mu}_{\mu}$) ==
    $4$
assert eliminate_kronecker($g_{\mu}^{\mu}$) == $4$
```

As some of them depend on definitions of symbols as derivatives in order for index consistency to be maintained, they are defined as functions returning expression objects so that they are only parsed when called (after `init_properties` has been invoked) and not during import where they would raise a parsing error

```
58   def ch():
59       return $ch^{\mu}_{\nu\tau} = \frac{1}{2} g^{\mu\
             sigma} (\partial_{\tau}{g_{\nu\sigma}} + \
             partial_{\nu}{g_{\tau\sigma}} - \partial_{\
             sigma}{g_{\nu\tau}})$
60   def rm():
61       return $rm^{\tau}_{\sigma\mu\nu} = \partial_{\mu}{
             ch^{\tau}_{\nu\sigma}} - \partial_{\nu}{ch^{\
             tau}_{\mu\sigma}} + \Gamma^{\tau}_{\mu\lambda} ch
             ^{\lambda}_{\nu\sigma} - ch^{\tau}_{\nu\
             lambda} ch^{\lambda}_{\mu\sigma}$
62   def rc():
63       return $rc_{\sigma\nu} = rm^{\tau}_{\sigma\tau\nu}
             $
64   def rs():
65       return $sc = g^{\mu\nu} rc_{\mu\nu}$
66   def ei():
67       return $ei_{\mu\nu} = rc_{\mu\nu} - \frac{1}{2} g_
             {\mu\nu} sc$
```

These correspond to the relations/definitions

$$\Gamma^\mu{}_{\nu\rho} = \frac{1}{2} g^{\mu\sigma}(\partial_\rho g_{\nu\sigma} + \partial_\nu g_{\rho\sigma} - \partial_\sigma g_{\nu\rho}), \qquad (8a)$$

$$R^\tau{}_{\sigma\mu\nu} = \partial_\mu \Gamma^\tau{}_{\nu\sigma} - \partial_\nu \Gamma^\tau{}_{\mu\sigma}$$
$$+ \Gamma^\tau{}_{\mu\lambda}\Gamma^\lambda{}_{\nu\sigma} - \Gamma^\tau{}_{\nu\lambda}\Gamma^\lambda{}_{\mu\sigma}, \qquad (8b)$$

$$R_{\sigma\nu} = R^\tau{}_{\sigma\tau\nu}, \qquad (8c)$$

$$R = g^{\mu\nu} R_{\mu\nu}, \qquad (8d)$$

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2} g_{\mu\nu} R. \qquad (8e)$$

3. Manipulation of tensorial expressions

3.1. Explicit form of the Lanczos–Lovelock Lagrangian

The Lanczos–Lovelock Lagrangians [13,14] are build in arbitrary dimensions to satisfy the same requirements as the Einstein–Hilbert Lagrangian in General Relativity: (i) Invariance under general coordinate transformations; (ii) local Lorentz symmetry, and; (iii) Provide field equations that are second order partial differential equations.

In four dimensions, the Lanczos–Lovelock Lagrangian is that of Einstein–Hilbert (with cosmological constant) with the addition of topological terms, the Gauß–Bonnet Lagrangian [14]. The generic Lanczos–Lovelock Lagrangian density in D dimensions is given by a sum [15,16]

$$\mathcal{L} = \sum_{p=0}^{n} a_p \mathcal{L}^{(D,p)}, \qquad (9)$$

with $n = [D/2]$, a_p denoted the coupling constant of the p-th term, and

$$\mathcal{L}^{(D,p)} = \sqrt{g} \, \frac{1}{2^p} \, \delta^{b_1 \cdots b_{2p}}_{a_1 \cdots a_{2p}}$$
$$\times R^{a_1 a_2}{}_{b_1 b_2} \cdots R^{a_{2p-1} a_{2p}}{}_{b_{2p-1} b_{2p}}. \qquad (10)$$

The purpose of this example is to find the form of the elements of the series, *i.e.* Eq. (10) after eliminating the generalised delta.

Notation and definitions

In this example, the position of the indices (unlike in General Relativity) carry no meaning, and we shall write all of them as upper indices. Hence, for this example, the default behaviour of indices will be enough. In this example, we do not use the `header.cnb` file from Sec. 2.2, consequently, we shall explain some of the declarations as they appear.

We start by declaring the indices, Kronecker delta, Levi-Civita epsilon, and the partial derivative.

```
1   {a#,b#,s#,t#,m,n}::Indices.
2   {a#,b#,s#,t#,m,n}::Integer(1..D).
3   \delta{#}::KroneckerDelta.
4   \epsilon{#}::EpsilonTensor(delta=-\delta).
```

The hash symbol after the `a`, `b`, `s` and `t` indices denotes that either of them followed by a number is a valid index. The *delta* argument of the Levi-Civita symbol (`EpsilonTensor`) serves to declare the signature of the metric. In the example above, the value `-\delta` implies that the—undeclared—metric is Lorentzian.

Next, we declare the dependencies and symmetries of the curvature tensors. In addition to the `TableauSymmetry` property, Cadabra has the `Symmetric` and `AntiSymmetric` properties, which endow the symmetry to all the indices of the object.

```
5   R^{s1 s2 s3 s4}::TableauSymmetry( shape={2,2},
        indices={0,2,1,3} ).
6   R^{s1 s2}::Symmetric.
```

In order to simplify the final expressions, we define a set of substitution rules for the contractions of the curvature tensor,

```
7    toR := {R^{s1 s2 s1 s2} = R, R^{s1 s2 s2 s1} = - R};
8    toRic := {R^{s1 s2 s1 s3} = R^{s2 s3},
9      R^{s2 s1 s3 s1} = R^{s2 s3},
10     R^{s1 s2 s3 s1} = - R^{s2 s3},
11     R^{s2 s1 s1 s3} = - R^{s2 s3}};
```

Zeroth order in curvature

From the Eq. (10), it is obvious that the zeroth curvature term is the cosmological constant monomial in the action. Thus, no further work is needed.

Linear curvature Lagrangian

From the Eq. (10), the linear curvature Lagrangian is

$$\mathcal{L}^{(D,1)} = \sqrt{g}\,\frac{1}{2}\,\delta^{b_1 b_2}_{a_1 a_2}\,R^{a_1 a_2}{}_{b_1 b_2}. \tag{11}$$

Since the measure ($\sqrt{g}$) does not carry indices, let us focus on the remaining expression. The general strategy is to expand the *generalised* Kronecker delta, consider all possible contractions, and use the substitution rules in order to present the result in a simpler form. However, we need a few extra manipulations in the middle as shown below.

```
12   LL1 := 2/2 R^{a1 a2 b1 b2} \delta^{a1 b1 a2 b2};
13   expand_delta(LL1)
14   distribute(LL1)
15   eliminate_kronecker(LL1)
16   canonicalise(LL1)
17   rename_dummies(LL1)
18   substitute(LL1, toR)
19   substitute(LL1, toRic)
20   sort_product(LL1)
21   sort_sum(LL1)
22   canonicalise(LL1)
23   rename_dummies(LL1)
24   collect_factors(LL1);
```

$$R^{a_1 a_2 b_1 b_2}\delta^{a_1 b_1 a_2, b_2} \tag{12}$$

$$R. \tag{13}$$

In the definition of the expression `LL1`, the factor 2 in the numerator is to compensate a $(2p)!$ factor in the definition of the generalised delta in Cadabra. The result of Eq. (13) is the scalar curvature, which corresponds to the Einstein–Hilbert Lagrangian.

Before moving toward terms with higher order in curvature, let us introduce the programming capability of Cadabra. The manipulations above, can be turned in to a Python-like function, that contain the set of algorithms to be applied to an input expression. Hence, the above code will be replaced by,

```
12   def LLmanip(ex):
13       expand_delta(ex)
14       distribute(ex)
15       eliminate_kronecker(ex)
16       canonicalise(ex)
17       rename_dummies(ex)
18       substitute(ex, toR)
19       substitute(ex, toRic)
20       sort_product(ex)
21       sort_sum(ex)
22       canonicalise(ex)
23       rename_dummies(ex)
24       collect_factors(ex)
25       return(ex)
```

and finally, we define the linear curvature term of the Lanczos–Lovelock series, and apply the newly define algorithm,

```
26   LL1 := 2/2 R^{a1 a2 b1 b2} \delta^{a1 b1 a2 b2};
27   LLmanip(LL1);
```

$$R^{a_1 a_2 b_1 b_2}\delta^{a_1 b_1 a_2 b_2},$$

$$R. \tag{14}$$

Quadratic curvature Lagrangian

From the Eq. (10), the quadratic curvature Lagrangian is

$$\mathcal{L}^{(D,2)} = \sqrt{g}\,\frac{1}{2^2}\,\delta^{b_1 b_2 b_3 b_4}_{a_1 a_2 a_3 a_4}\,R^{a_1 a_2}{}_{b_1 b_2}R^{a_3 a_4}{}_{b_3 b_4}. \qquad (15)$$

We proceed as in the previous case,

```
28    LL2 := 4*3*2/2/2 R^{a1 a2 b1 b2} R^{a3 a4 b3 b4} \
          delta^{a1 b1 a2 b2 a3 b3 a4 b4};
29    LLmanip(LL2);
```

$$6R^{a_1 a_2 b_1 b_2}R^{a_3 a_4 b_3 b_4}\delta^{a_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4} \qquad (16)$$

$$R^2 - 4R^{a_1 a_2}R^{a_1 a_2} + R^{a_1 a_2 a_3 a_4}R^{a_1 a_2 a_3 a_4} \qquad (17)$$

The result of Eq. (17) is the scalar combination quadratic in the curvature tensor known as Gauß–Bonnet term. Although in four dimensions this is a topological term, in dimensions higher than four it generates dynamical field equations.

Cubic curvature Lagrangian

From the Eq. (10), the cubic curvature Lagrangian is

$$\mathcal{L}^{(D,3)} = \sqrt{g}\,\frac{1}{2^3}\,\delta^{b_1 b_2 b_3 b_4 b_5 b_6}_{a_1 a_2 a_3 a_4 a_5 a_6}$$
$$\times\,R^{a_1 a_2}{}_{b_1 b_2}R^{a_3 a_4}{}_{b_3 b_4}R^{a_5 a_6}{}_{b_5 b_6}. \qquad (18)$$

And repeating the procedure above we get,

```
30    LL3:= 6*5*4*3*2/2/2/2 R^{a1 a2 b1 b2}R^{a3 a4 b3 b4}
          R^{a5 a6 b5 b6} \delta^{a1 b1 a2 b2 a3 b3 a4
          b4 a5 b5 a6 b6};
31    LLmanip(LL3);
```

$$90R^{a_1 a_2 b_1 b_2}R^{a_3 a_4 b_3 b_4}R^{a_5 a_6 b_5 b_6}$$
$$\times\,\delta^{a_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4 a_5 b_5 a_6 b_6}, \qquad (19)$$

$$R^3 - 12RR^{a_1 a_2}R^{a_1 a_2} + 3RR^{a_1 a_2 a_3 a_4}R^{a_1 a_2 a_3 a_4}$$
$$+ 16R^{a_1 a_2}R^{a_1 a_3}R^{a_2 a_3} + 24R^{a_1 a_2}R^{a_3 a_4}R^{a_1 a_3 a_2 a_4}$$
$$- 24R^{a_1 a_2}R^{a_1 a_3 a_4 a_5}R^{a_2 a_3 a_4 a_5}$$
$$+ 2R^{a_1 a_2 a_3 a_4}R^{a_1 a_2 a_5 a_6}R^{a_3 a_4 a_5 a_6}$$
$$- 8R^{a_1 a_2 a_3 a_4}R^{a_1 a_5 a_3 a_6}R^{a_2 a_6 a_4 a_5}. \qquad (20)$$

This Lagrangian is known as the Müller-Hoissen term [17].

In Appendix B we present the Lanczos–Lovelock Lagrangian of order four in curvature.

3.2. Field equations of the Lanczos–Lovelock Lagrangians

The field equations derived from the Lanczos–Lovelock Lagrangians are [17-19]

$$G^{(p)\,n}{}_m = -\frac{1}{2^{p+1}}\delta^{nb_1\cdots b_{2p}}_{ma_1\cdots a_{2p}}\,R^{a_1 a_2}{}_{b_1 b_2}\cdots R^{a_{2p-1}a_{2p}}{}_{b_{2p-1}b_{2p}} =$$
$$-\frac{(2p+1)!}{2^{p+1}}\delta^n_{[m}R^{a_1 a_2}{}_{a_1 a_2}\cdots R^{a_{2p-1}a_{2p}}{}_{a_{2p-1}a_{2p}]}. \qquad (21)$$

We can use the same function `LLmanip` to obtain the expressions for the field equations. Note that in the output below, in order to shrink the expressions, we have written the piece accompanying the Kronecker δ as a Lagrangian itself.

As before, the term independent of the curvature needs no calculation,

$$G^{(0)\,n}{}_m = -\frac{1}{2}\delta^n_m. \qquad (22)$$

While the field equations for the linear, quadratic and cubic are shown below,

```
32    feqLL1 := - 3*2/2 R^{a1 a2 b1 b2} \delta^{m n a1
          b1 a2 b2};
33    LLmanip(feqLL1);
```

$$-\frac{3}{2}R^{a_1 a_2 b_1 b_2}\delta^{mna_1 b_1 a_2 b_2}, \qquad (23)$$

$$R^{mn} - \frac{1}{2}R\delta^{mn}. \qquad (24)$$

```
34    feqLL2 := - 5*4*3*2/2/2/2 R^{a1 a2 b1 b2} R^{a3 a4
          b3 b4} \delta^{m n a1 b1 a2 b2 a3 b3 a4 b4};
35    LLmanip(feqLL2)
36    factor_out(_, $\delta^{m n}$)
37    substitute(_, $@(LL2) -> L^{(D,2)}$);
```

$$-15R^{a_1 a_2 b_1 b_2}R^{a_3 a_4 b_3 b_4}\delta^{mna_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4} \qquad (25)$$

$$2RR^{mn} - 4R^{a_1 a_2}R^{ma_1 na_2} - 4R^{ma_1}R^{na_1}$$
$$+ 2R^{ma_1 a_2 a_3}R^{na_1 a_2 a_3} - \frac{1}{2}\delta^{mn}L^{(D,2)}. \qquad (26)$$

```
38    feqLL3 := - 7*6*5*4*3*2/2/2/2/2 R^{a1 a2 b1 b2} R^{
          a3 a4 b3 b4} R^{a5 a6 b5 b6} \delta^{m n a1 b1
          a2 b2 a3 b3 a4 b4 a5 b5 a6 b6};
39    LLmanip(feqLL3)
40    factor_out(_, $\delta^{m n}$)
41    substitute(_, $@(LL3) -> L^{(D,3)}$);
```

$$-315R^{a_1 a_2 b_1 b_2}R^{a_3 a_4 b_3 b_4}R^{a_5 a_6 b_5 b_6}$$
$$\times\,\delta^{mna_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4 a_5 b_5 a_6 b_6}, \qquad (27)$$

$$3R^2 R^{mn} - 12RR^{a_1 a_2} R^{ma_1 na_2} - 12RR^{ma_1} R^{na_1}$$

$$+ 6RR^{ma_1 a_2 a_3} R^{na_1 a_2 a_3} + 24R^{na_1} R^{a_2 a_3} R^{ma_2 a_1 a_3}$$

$$+ 24R^{a_1 a_2} R^{a_1 a_3} R^{ma_2 na_3} - 12R^{a_1 a_2} R^{ma_1 a_3 a_4} R^{na_2 a_3 a_4}$$

$$- 12R^{mn} R^{a_1 a_2} R^{a_1 a_2} + 24R^{ma_1} R^{na_2} R^{a_1 a_2}$$

$$+ 24R^{ma_1} R^{a_2 a_3} R^{na_2 a_1 a_3} + 24R^{a_1 a_2} R^{ma_3 na_4} R^{a_1 a_3 a_2 a_4}$$

$$- 24R^{a_1 a_2} R^{ma_3 a_1 a_4} R^{na_3 a_2 a_4} - 12R^{ma_1} R^{na_2 a_3 a_4} R^{a_1 a_2 a_3 a_4}$$

$$+ 3R^{mn} R^{a_1 a_2 a_3 a_4} R^{a_1 a_2 a_3 a_4} - 12R^{na_1} R^{ma_2 a_3 a_4} R^{a_1 a_2 a_3 a_4}$$

$$- 12R^{ma_1 na_2} R^{a_1 a_3 a_4 a_5} R^{a_2 a_3 a_4 a_5}$$

$$- 24R^{ma_1 a_2 a_3} R^{na_4 a_2 a_5} R^{a_1 a_5 a_3 a_4}$$

$$+ 6R^{ma_1 a_2 a_3} R^{na_1 a_4 a_5} R^{a_2 a_3 a_4 a_5} - \frac{1}{2}\delta^{mn} L^{(D,3)}. \tag{28}$$

4. Language of differential forms

A very useful calculation tool in physics is the exterior differential calculus, which—unlike tensor calculus—deals only with the set of completely antisymmetric (differentiable) tensors of $(0, p)$-type, for $0 \leq p \leq D = \dim(M)$. For a given value of p, the elements are called *differential forms of degree p* or *p-forms*, and they form an $\mathcal{F}(M)$-module denoted by $\Omega^p(M)^{vi}$. The generalisation of the tensor product to a product that preserves the antisymmetry of the result is dubbed *wedge* product, $\wedge : \Omega^p(M) \times \Omega^q(M) \to \Omega^{p+q}(M)$, and provides structure of algebra to the space of all exterior differential forms,

$$\Omega(M) = \sum_{p=0}^{D} \Omega^p(M), \tag{29}$$

known as exterior algebra. One can endow the exterior algebra with more structure by defining the *exterior derivative*, a smooth map $d : \Omega^p(M) \to \Omega^{p+1}(M)$ satisfying the (super)Leibniz rule,

$$d(\alpha \wedge \beta) = d\alpha \wedge \beta + (-1)^p \alpha \wedge d\beta, \tag{30}$$

for $\alpha \in \Omega^p(M)$ and $\beta \in \Omega^q(M)$, and nilpotency, *i.e.* $d^2 = 0$, and the *Hodge star operator*, a map $\star : \Omega^p(M) \to \Omega^{D-p}(M)$, which allows defining a symmetric scalar product among p-forms,

$$(\alpha, \beta) = \int \alpha \wedge \star\beta \quad \text{for} \quad \alpha, \beta \in \Omega^p(M). \tag{31}$$

A clear advantage of using differential forms over tensors is that there are only a finite set of independent forms. However, in order to encode the information within symmetric or mixed tensors, one has to allow differential forms to be valued on a Lie algebra^{vii}. Hence, in the following, we consider that the Lie algebra behind the gravitational theory is the Lorentz algebra.

In this formalism, the geometric information carried by the metric and affine connection is encoded in the vielbein 1-form (e^a) and the spin connection 1-form (ω^{ab}). From them, one can calculate the curvature and torsion 2-forms using the structural equations of Cartan,

$$de^a + \omega^a{}_b \wedge e^b = \mathrm{T}^a,$$

$$d\omega^a{}_b + \omega^a{}_m \wedge \omega^m{}_b = \mathrm{R}^a{}_b. \tag{32}$$

4.1. Bianchi identities from Cartan structural equations

Definitions

As in previous examples, for this example we won't use the declarations in the `header.cnb` file. Hence, we need to declare the indices, derivative (in this case the exterior derivative), and the geometrical objects—*i.e.* the differential forms.

```
1   {a,b,c,l,m,n}::Indices.
2   d{#}::ExteriorDerivative;.
3   d{#}::LaTeXForm("\mathrm{d}").
4   T{#}::LaTeXForm("\mathrm{T}").
5   R{#}::LaTeXForm("\mathrm{R}").
6   {e^{a}, \omega^{a}_{b}}::DifferentialForm(degree=1);
7   {T^{a}, R^{a}_{b}}::DifferentialForm(degree=2);
```

In the above code, we have introduced the property `DifferentialForm`, which through its argument `degree=p` assigns the property p-form to a given object or set of them, and `ExteriorDerivative` which defines the exterior derivative to the symbol d.

Cartan structural equations

Now, we define the structural equations.

```
8   struc1 := d{e^{a}} + \omega^{a}_{b} ^ e^{b} - T^{a}
        = 0;
9   struc2 := d{\omega^{a}_{b}} + \omega^{a}_{m} ^ \
        omega^{m}_{b} - R^{a}_{b} = 0;
```

$$de^a + \omega^a{}_b \wedge e^b - \mathrm{T}^a = 0,$$

$$d\omega^a{}_b + \omega^a{}_m \wedge \omega^m{}_b - \mathrm{R}^a{}_b = 0. \tag{33}$$

In the following, we will also use the structural equations as definitions of the exterior derivatives of the vielbein and spin connection 1-forms. Therefore, we shall utilise the `isolate` algorithm—from the `cdb.core.manip` library—to define substitution rules.

```
10   from cdb.core.manip import isolate
11   de:= @(struc1):
12   isolate(de, $d{e^{a}}$);
```

$$de^a = -\omega^a{}_b \wedge e^b + \mathrm{T}^a, \tag{34}$$

```
13   domega := @(struc2):
14   isolate(domega, $d{\omega^{a}_{b}}$);
```

$$d\omega^a{}_b = -\omega^a{}_m \wedge \omega^m{}_b + \mathrm{R}^a{}_b. \tag{35}$$

Bianchi identities

The bianchi identities are obtained by applying the exterior derivative to the structural equations.

First Bianchi identity.- We apply the exterior derivative to the expression `struc1`, and then distribute and apply the Leibniz rule (`product_rule`),

```
15    Bianchi1 := d{ @(struc1) };
16    distribute(Bianchi1)
17    product_rule(_);
```

$$d\left(de^a + \omega^a{}_b \wedge e^b - T^a\right) = 0, \quad (36)$$

$$d\omega^a{}_b \wedge e^b - \omega^a{}_b \wedge de^b - dT^a = 0. \quad (37)$$

Now, the expression has the exterior derivative of the vielbein and spin connection. Hence, we substitute the rules `de` and `domega`,

```
18    substitute(Bianchi1, de, repeat=True)
19    substitute(Bianchi1, domega, repeat=True)
20    distribute(_)
21    rename_dummies(_);
```

$$R^a{}_b \wedge e^b - \omega^a{}_b \wedge T^b - dT^a = 0. \quad (38)$$

In tensor form, the above relation is expressed as

$$R^\lambda{}_{[\rho\mu\nu]} - \nabla_{[\mu}T^\lambda{}_{\nu\rho]} - T^\lambda{}_{\sigma[\rho}T^\sigma{}_{\mu\nu]} = 0. \quad (39)$$

The above expression is the *algebraic* Bianchi identity, which in the absence of torsion takes the well-known form,

$$R^\lambda{}_{[\rho\mu\nu]} = R^\lambda{}_{\mu\nu\rho} + R^\lambda{}_{\rho\mu\nu} + R^\lambda{}_{\nu\rho\mu} = 0. \quad (40)$$

Second Bianchi identity.- Applying the exterior derivative to `struc2`, we get

```
22    Bianchi2 := d{ @(struc2) };
23    distribute(Bianchi2)
24    product_rule(_);
```

$$d\left(d\omega^a{}_b + \omega^a{}_m \wedge \omega^m{}_b - R^a{}_b\right) = 0, \quad (41)$$

$$d\omega^a{}_m \wedge \omega^m{}_b - \omega^a{}_m \wedge d\omega^m{}_b - dR^a{}_b = 0. \quad (42)$$

Here, only the exterior derivative of ω appears, then

```
25    substitute(Bianchi2, domega, repeat=True)
26    distribute(_)
27    rename_dummies(_);
```

$$R^a{}_c \wedge \omega^c{}_b - \omega^a{}_c \wedge R^c{}_b - dR^a{}_b = 0. \quad (43)$$

The above result is the *differential* Bianchi identity, that is written in tensor form as

$$R^\lambda{}_{\sigma[\mu\nu;\rho]} + R^\lambda{}_{\sigma\tau[\rho}T^\tau{}_{\mu\nu]} = 0, \quad (44)$$

which in the absence of torsion takes the well-known form,

$$R^\lambda{}_{\sigma[\mu\nu;\rho]} = R^\lambda{}_{\sigma\mu\nu;\rho} + R^\lambda{}_{\sigma\nu\rho;\mu} + R^\lambda{}_{\sigma\rho\mu;\nu} = 0. \quad (45)$$

5. Einstein equations from a variational principle

In this example, we derive the field equations of General Relativity from Einstein–Hilbert action.

5.1. Definitions

For this example, we shall import our `header` file from Sec. 2.2. Note that although the indices are declared as four-dimensional, and the coordinates as spherical, the result is valid for a generic choice of coordinates and dimension.

```
1    from libraries.header import *
2    init_properties(coordinates=$t, r, \theta, \phi$)
```

Then, we declare the LATEX output of a variety of expressions that will show up in the process.

```
3    Lm::LaTeXForm("\mathcal{L}_{\text{mat}}").
4    Dg::LaTeXForm("\sqrt{-g}").
5    dg{#}::LaTeXForm("\delta{g}").
6    dLm::LaTeXForm("\delta{\mathcal{L}_{\text{mat}}}").
7    dDg::LaTeXForm("\delta{\sqrt{-g}}").
8    dCn{#}::LaTeXForm("\delta{\Gamma}").
9    dR{#}::LaTeXForm("\delta{R}").
```

5.2. Variational problem

Our starting point is the Einstein–Hilbert action, amended with a cosmological constant term.

```
10    action := S = \int{Dg [ \frac{1}{2 \kappa} ( sc - 2
          \Lambda) + Lm ]}{x}.
11    distribute(_);
```

$$S = \int \left(\frac{1}{2}\sqrt{-g}\kappa^{-1}R - \sqrt{-g}\kappa^{-1}\Lambda + \sqrt{-g}\mathcal{L}_{\text{mat}}\right)dx. \quad (46)$$

Next, we build the variation of the objects (fields) present in the action.

The variation of the metric determinant is given by

```
12    deltaMetricDeterminant := dDg = - \frac{1}{2} Dg
          g_{\mu\nu} dg^{\mu\nu};
```

$$\delta\sqrt{-g} = -\frac{1}{2}\sqrt{-g}g_{\mu\nu}\delta g^{\mu\nu}. \quad (47)$$

By definition, the variation of the matter Lagrangian with respect to the metric gives the stress tensor,

```
13    matterLagrangianVariation := Dg T_{\mu\nu}
          dg^{\mu\nu} = - 2 d{Dg Lm};
14    product_rule(_)
15    distribute(_)
16    substitute(_, $d{Dg} -> dDg$)
17    substitute(_, deltaMetricDeterminant)
18    manip.swap_sides(_);
```

$$\sqrt{-g}T_{\mu\nu}\delta g^{\mu\nu} = -2\delta\left(\sqrt{-g}\mathcal{L}_{\text{mat}}\right), \quad (48)$$

$$\sqrt{-g}g_{\mu\nu}\delta g^{\mu\nu}\mathcal{L}_{\text{mat}} - 2\sqrt{-g}\delta\mathcal{L}_{\text{mat}} = \sqrt{-g}T_{\mu\nu}\delta g^{\mu\nu}. \quad (49)$$

Next comes the variation of the Christoffel symbols. We then isolate the derivative of the variation, as that object will appear in the variation of the Riemann tensor.

```
19    deltaDerivativeCn := \nabla_{\sigma}{dCn^{\mu}_{\nu\
         tau}} = \partial_{\sigma}{dCn^{\mu}_{\nu\tau}}
20    + ch^{\mu}_{\sigma\gamma} dCn^{\gamma}_{\nu\tau}
21    - ch^{\gamma}_{\sigma\nu} dCn^{\mu}_{\gamma\tau}
22    - ch^{\gamma}_{\sigma\tau} dCn^{\mu}_{\nu\gamma};
23    manip.isolate(_, $\partial_{\sigma}{dCn^{\mu}_{\nu\
         tau}}$);
```

$$\nabla_\sigma\delta\Gamma^\mu{}_{\nu\tau} = \partial_\sigma\delta\Gamma^\mu{}_{\nu\tau} + \Gamma^\mu{}_{\sigma\gamma}\delta\Gamma^\gamma{}_{\nu\tau}$$
$$- \Gamma^\gamma{}_{\sigma\nu}\delta\Gamma^\mu{}_{\gamma\tau} - \Gamma^\gamma{}_{\sigma\tau}\delta\Gamma^\mu{}_{\nu\gamma}, \quad (50)$$

$$\partial_\sigma\delta\Gamma^\mu{}_{\nu\tau} = -\Gamma^\mu{}_{\sigma\gamma}\delta\Gamma^\gamma{}_{\nu\tau} + \Gamma^\gamma{}_{\sigma\nu}\delta\Gamma^\mu{}_{\gamma\tau}$$
$$+ \Gamma^\gamma{}_{\sigma\tau}\delta\Gamma^\mu{}_{\nu\gamma} + \nabla_\sigma\delta\Gamma^\mu{}_{\nu\tau}. \quad (51)$$

The variation of the Riemann tensor involves covariant derivatives of the variation of the Christoffel connection.

```
24    deltaRiemannTensor = vary( rm(), $ch^{\mu}_{\nu\tau}
         ->dCn^{\mu}_{\nu\tau}, rm^{\tau}_{\sigma\mu\nu}
         -> dR^{\tau}_{\sigma\mu\nu}$);
25    sort_product(_)
26    substitute(_,deltaDerivativeCn)
27    meld(_);
```

$$\delta R^\tau{}_{\sigma\mu\nu} = \partial_\mu\delta\Gamma^\tau{}_{\nu\sigma} - \partial_\nu\delta\Gamma^\tau{}_{\mu\sigma} + \delta\Gamma^\tau{}_{\mu\lambda}\Gamma^\lambda{}_{\nu\sigma}$$
$$+ \Gamma^\tau{}_{\mu\lambda}\delta\Gamma^\lambda{}_{\nu\sigma} - \delta\Gamma^\tau{}_{\nu\lambda}\Gamma^\lambda{}_{\mu\sigma} - \Gamma^\tau{}_{\nu\lambda}\delta\Gamma^\lambda{}_{\mu\sigma}, \quad (52)$$

$$\delta R^\tau{}_{\sigma\mu\nu} = \nabla_\mu\delta\Gamma^\tau{}_{\nu\sigma} - \nabla_\nu\delta\Gamma^\tau{}_{\mu\sigma}. \quad (53)$$

From the last expression one obtains the variation of the Ricci tensor, dubbed *Palatini identity*:

```
28    deltaRicciTensor = vary( rc(), $rm^{\tau}_{\sigma\
         tau\nu} -> dR^{\tau}_{\sigma\tau\nu}, rc_{\
         sigma\nu} -> dR_{\sigma\nu}$);
29    substitute(_, deltaRiemannTensor);
```

$$\delta R_{\sigma\nu} = \delta R^\tau{}_{\sigma\tau\nu}, \quad (54)$$

$$\delta R_{\sigma\nu} = \nabla_\tau\delta\Gamma^\tau{}_{\nu\sigma} - \nabla_\nu\delta\Gamma^\tau{}_{\tau\sigma}. \quad (55)$$

Similarly, the variation of the scalar curvature yields

```
30    deltaScalarCurvature = vary( rs(), $sc-> dR, rc_{\
         sigma\nu} -> dR_{\sigma\nu}, g^{\mu\nu}->dg^{\
         mu\nu}$);
31    substitute(_, deltaRicciTensor)
32    distribute(_)
33    substitute(_, $\nabla_{\sigma}{dCn^{\mu}_{\nu\tau}}
         g^{\gamma\lambda} -> \nabla_{\sigma}{dCn^{\mu}_
         {\nu\tau} g^{\gamma\lambda}}$)
34    canonicalise(_);
```

$$\delta R = \nabla_\nu\left(\delta\Gamma^\nu{}_{\tau\sigma}g^{\tau\sigma}\right) - \nabla_\nu\left(\delta\Gamma^\tau{}_{\tau\sigma}g^{\nu\sigma}\right) + R_{\nu\sigma}\delta g^{\nu\sigma}. \quad (56)$$

We can now vary the action,

```
35    deltaAction = vary( $@(action):$, $Dg -> dDg, sc ->
         dR, Lm -> d{Lm}, S -> d{S}$);
36    substitute(_, deltaMetricDeterminant)
37    substitute(_, matterLagrangianVariation)
38    substitute(_, deltaScalarCurvature)
39    distribute(_)
40    rename_dummies(_)
41    factor_out(_, $dg^{\mu\nu}, Dg$);
```

$$\delta S = \int\left(\frac{1}{2}\delta\sqrt{-g}\kappa^{-1}R + \frac{1}{2}\sqrt{-g}\kappa^{-1}\delta R\right.$$
$$\left. - \delta\sqrt{-g}\kappa^{-1}\Lambda + \delta\sqrt{-g}\mathcal{L}_{\text{mat}} + \sqrt{-g}\delta\mathcal{L}_{\text{mat}}\right)\mathrm{d}x, \quad (57)$$

$$\delta S = \int\left(\sqrt{-g}\delta g^{\mu\nu}\left[-\frac{1}{4}g_{\mu\nu}\kappa^{-1}R + \frac{1}{2}\kappa^{-1}R_{\mu\nu}\right.\right.$$
$$\left.+\frac{1}{2}g_{\mu\nu}\kappa^{-1}\Lambda - \frac{1}{2}T_{\mu\nu}\right] + \sqrt{-g}\left[\frac{1}{2}\kappa^{-1}\nabla_\mu\left\{\delta\Gamma^\mu{}_{\nu\tau}g^{\nu\tau}\right\}\right.$$
$$\left.\left. - \frac{1}{2}\kappa^{-1}\nabla_\tau\left\{\delta\Gamma^\mu{}_{\mu\nu}g^{\tau\nu}\right\}\right]\right)\mathrm{d}x. \quad (58)$$

The terms proportional to $\delta g^{\mu\nu}$ are the Einstein equations, the rest is a total derivative. In order to rewrite this term in the familiar form, let us first assign to a variable the term in the first bracket on the right-hand side. To select a piece of an expression one uses the array notation of python: (i) the first level of the expression is an equality, therefore the [0] and [1] components represent the left-hand side and right-hand side respectively, thus we chose the [1] component; (ii) now the top-level is an integration, and the [0] and [1] components represent the argument and integration variable respectively, thus we have to select the [0] component; (iii) the argument of the integral is the sum of two terms, since we are interested in the first, we have to select the [0] component, and finally; (iv) from that term the bracket is the [2] component. Therefore, the expression of interest is deltaAction[1][0][0][2]. Hence,

```
42    t1 = deltaAction[1][0][0][2]
43    eom:= 2 \kappa @(t1) = 0;
44    distribute(_)
45    collect_factors(_)
46    manip.to_rhs(_, $\kappa T_{\mu\nu}$);
```

$$2\kappa\left(-\frac{1}{4}g_{\mu\nu}\kappa^{-1}R + \frac{1}{2}\kappa^{-1}R_{\mu\nu}\right.$$
$$\left.+\frac{1}{2}g_{\mu\nu}\kappa^{-1}\Lambda - \frac{1}{2}T_{\mu\nu}\right) = 0, \quad (59)$$

$$-\frac{1}{2}g_{\mu\nu}R + R_{\mu\nu} + g_{\mu\nu}\Lambda = \kappa T_{\mu\nu}. \quad (60)$$

These are the Einstein equations, which can be written in terms of the Einstein tensor, $G_{\mu\nu}$,

```
47    EinsteinEq := @(eom):
48    substitute(EinsteinEq, manip.swap_sides(ei()));
```

$$G_{\mu\nu} + g_{\mu\nu}\Lambda = \kappa T_{\mu\nu}, \qquad (61)$$

or in the Ricci form, which is obtained after eliminating the scalar curvature from the Einstein equations. Therefore, we first calculate the trace to isolate the scalar curvature

```
49    trEinsteinEq := @(eom):
50    manip.multiply_through(_,$g^{\mu\nu}$);
51    distribute(_)
52    substitute(_, $g^{\mu \nu} T_{\mu \nu} = T$)
53    substitute(_, manip.swap_sides(rs()))
54    eliminate_metric(_)
55    eliminate_kronecker(_)
56    manip.isolate(trEinsteinEq, $sc()$);
```

$$g^{\mu\nu}\left(-\frac{1}{2}g_{\mu\nu}R + R_{\mu\nu} + g_{\mu\nu}\Lambda\right) = g^{\mu\nu}\kappa T_{\mu\nu}, \qquad (62)$$

$$R = -T\kappa + 4\Lambda, \qquad (63)$$

and substitute into the field equations,

```
57    RicciFormEq := @(eom):
58    substitute(_, trEinsteinEq)
59    distribute(_);
60    manip.isolate(_, $rc_{\mu \nu}$)
61    factor_out(_, $\kappa$);
```

$$\frac{1}{2}g_{\mu\nu}T\kappa - g_{\mu\nu}\Lambda + R_{\mu\nu} = \kappa T_{\mu\nu}, \qquad (64)$$

$$R_{\mu\nu} = g_{\mu\nu}\Lambda + \kappa\left(T_{\mu\nu} - \frac{1}{2}g_{\mu\nu}, T\right), \qquad (65)$$

Note that in the above manipulation we have used that $\delta^{\mu}_{\mu} = 4^{viii}$, and consequently the field equations in the form of Ricci in arbitrary dimension gets factor modifications, unlike the standard Einstein's form.

6. Spacetime solutions: Schwarzschild spacetime

In 1916, a few months after the publication of Einstein's equations, the first non-trivial solution was found by Schwarzschild, who intended to model the gravitational field of an isolated spherically symmetric object.

The metric compatible with the three-dimensional spherical symmetry has a line element given by

$$ds^2(g) = -\exp(A(t,r))dt^2$$
$$+ \exp(B(t,r))dr^2 + r^2 d\Omega^2_{(2)}, \qquad (66)$$

where $d\Omega^2_{(2)}$ is the line element of a two-dimensional sphere, *i.e.*,

$$d\Omega^2_{(2)} = d\theta^2 + \sin^2(\theta)d\phi^2. \qquad (67)$$

The above metric will be the starting point of our calculation, with the exception that for simplicity, we shall consider the functions A and B as time-independent. Such condition could be derived from the field equations, and is a corollary of the dubbed *Birkhoff theorem* [20-23].

Definitions

In this section, we solve Einstein field equations for Schwarzschild spacetimes. In this example, we upload the definitions on our `header` file, described in Sec. 2.2.

```
1    from libraries.header import *
2    init_properties(coordinates=$t,r,\theta,\phi$,
          metrics=[$g_{\mu\nu}$, $\eta_{\mu\nu}$])
```

Calculating the field equations

The Einstein equations in vacuum are equivalent to the vanishing Ricci tensor, $R_{\mu\nu} = 0$. Hence, we proceed to calculate the Ricci tensor from the ansatz in Eq. (66).

In `Cadabra`, the metric is defined through a series of substitutions, and the inverse metric is calculated—by `SymPy` algorithms—from the same substitutions. For this end, the algorithm `complete` is used. Notice below the use of the LaTeX notation for the exponential function.

```
3    {A,B,f}::Depends(r).
4    ss := { g_{t t} = - \exp(A),
5    g_{r r} = \exp(B),
6    g_{\theta\theta} = r**2,
7    g_{\phi\phi} = r**2 \sin(\theta)**2 }.
8
9    complete(ss, $g^{\mu\nu}$);
```

$$\left[g_{tt} = -\exp A, \quad g_{rr} = \exp B, \right.$$
$$g_{\theta\theta} = r^2, \quad g_{\phi\phi} = r^2(\sin\theta)^2,$$
$$g^{tt} = -\exp(-A), \quad g^{rr} = \exp(-B),$$
$$\left. g^{\phi\phi} = \left(r^2(\sin\theta)^2\right)^{-1}, \quad g^{\theta\theta} = r^{-2} \right]. \qquad (68)$$

With the metric (and its inverse) defined, one proceeds to calculate the components of the Levi-Civita, Riemann and Ricci tensors, and the scalar curvature. This is performed by the algorithm `evaluate`, which accepts as second argument a series of substitution rules to execute the evaluation. For the Levi-Civita connection the rules are the components of the metric, *i.e.*, `ss`.

```
10    evaluate(ch(), ss, rhsonly=True);
```

```
1    rc_comp = substitute(rc(), rm)
2    evaluate(rc_comp, ss, rhsonly=True);
```

$$\Gamma^{\mu}{}_{\nu\tau} = \Box_{\nu\tau}{}^{\mu} \begin{cases} \Box_{\phi r}{}^{\phi} = r^{-1} \\[4pt] \Box_{\phi\theta}{}^{\phi} = (\tan\theta)^{-1} \\[4pt] \Box_{\theta r}{}^{\theta} = r^{-1} \\[4pt] \Box_{rr}{}^{r} = \dfrac{1}{2}\partial_r B \\[4pt] \Box_{tr}{}^{t} = \dfrac{1}{2}\partial_r A \\[4pt] \Box_{r\phi}{}^{\phi} = r^{-1} \\[4pt] \Box_{\theta\phi}{}^{\phi} = (\tan\theta)^{-1} \\[4pt] \Box_{r\theta}{}^{\theta} = r^{-1} \\[4pt] \Box_{rt}{}^{t} = \dfrac{1}{2}\partial_r A \\[4pt] \Box_{\phi\phi}{}^{r} = -r\exp(-B)(\sin\theta)^2 \\[4pt] \Box_{\phi\phi}{}^{\theta} = -\dfrac{1}{2}\sin(2\theta) \\[4pt] \Box_{\theta\theta}{}^{r} = -r\exp(-B) \\[4pt] \Box_{tt}{}^{r} = \dfrac{1}{2}\exp(A-B)\,\partial_r A \end{cases} \tag{69}$$

Since the Riemannian curvature is expressed in terms of the connection, one should first substitute the algebraic expression of the Levi-Civita connection in terms of the metric (using the `substitute` algorithm), and finally evaluate the curvature by passing the substitution rules of the metric.

```
11    rm_comp = substitute(rm(), ch())
12    evaluate(rm_comp, ss, rhsonly=True);
```

The output of the `evaluate` command is large, and useless for our purpose in this section. Therefore, in the following some output lines are avoided intentionally.

The Ricci tensor is defined from the Riemann tensor, one should substitute its expression (notice that at this stage the Riemann tensor is defined in terms of the metric, its derivative and the metric inverse) before calculating with the algorithm `evaluate`.

$$R_{\sigma\nu} = \Box_{\nu\sigma} \begin{cases} \Box_{tt} = \left(r\left(\dfrac{1}{4}(\partial_r A)^2 - \dfrac{1}{4}\partial_r A\partial_r B + \dfrac{1}{2}\partial_{rr}A \right) + \partial_r A \right)\exp(A-B)\, r^{-1} \\[10pt] \Box_{\theta\theta} = \dfrac{1}{2}\left(-r\partial_r A + r\partial_r B + 2\exp B - 2 \right)\exp(-B) \\[10pt] \Box_{\phi\phi} = \dfrac{1}{2}\left(-r\partial_r A + r\partial_r B + 2\exp B - 2 \right)\exp(-B)(\sin\theta)^2 \\[10pt] \Box_{rr} = \left(r\left(-\dfrac{1}{4}(\partial_r A)^2 + \dfrac{1}{4}\partial_r A\partial_r B - \dfrac{1}{2}\partial_{rr}A \right) + \partial_r B \right) r^{-1} \end{cases} \tag{70}$$

Solving the field equations

At the end of the last section, we manage to calculate the Ricci tensor derived from the Schwarzschild metric anzats. Since the Einstein equations in vacuum (without cosmological constant) are equivalent to the vanishing Ricci tensor, in this section, we shall manipulate the results above to solve the field equations.

The package `cdb.core.component`—imported in the header file—defines functions that allows one to access components of a *geometrical object*. In particular, the function `get_component`, extracts a single component of an expression (aka geometrical object).

Now, we can assign the calculated components to expressions, say $r00$ and $r11$.

```
1    r00 = comp.get_component(rc_comp, $t,t$)[1];
2    r11 = comp.get_component(rc_comp, $r,r$)[1];
```

$$\left(r\left[\frac{1}{4}(\partial_r A)^2 - \frac{1}{4}\partial_{rr}A\partial_r B + \frac{1}{2}\partial_r A\right]\right.$$

$$\left. + \partial_r A\right)\exp\left(A - B\right)r^{-1},\tag{71}$$

$$\left(r\left[-\frac{1}{4}(\partial_r A)^2 + \frac{1}{4}\partial_r A\partial_r B - \frac{1}{2}\partial_{rr}A\right]\right.$$

$$\left. + \partial_r B\right)r^{-1}.\tag{72}$$

The symbol `[1]` at the end of the `get_component` function restricts our selection to the right-hand side of the expression.

By looking at the Eqs. (71) and (72), one notices that their arguments in the inner brackets are opposite. Hence, we consider the combination

$$r\left(\exp(B - A)R_{tt} + R_{rr}\right) = 0.\tag{73}$$

```
1    expr1 := r \exp(B - A) @(r00) + r @(r11);
```

There are several ways of simplifying the above result. Below, we use the function `map_sympy` to expand the expression.

```
1    map_sympy(expr1, "expand");
```

$$\partial_r A + \partial_r B.\tag{74}$$

Then the first expression, *i.e.* `expr1`, requires that $B(r) = -A(r) + C$. For the sake of simplicity, we shall set $C = 0$, nonetheless, in general grounds this choice represents a scaling on the time coordinate.

Now consider the $\theta\theta$-component of the Ricci tensor, after substituting $B(r) = -A(r)$.

```
1    r22 = comp.get_component(rc, $\theta, \theta$)[1];
2    expr2 := @(r22):
3    substitute(expr2, $B -> - A$);
```

The ordinary differential equation looks nicer if one transforms even further $\exp(A) \to f$ or equivalently $A \to \ln(f)$.

```
1    substitute(_ , $A -> \log(f)$)
2    map_sympy(_ , "expand");
```

$$-r\partial_r f - f + 1.\tag{75}$$

The `cdb.sympy.solvers` package provides wrappers to make `SymPy` solvers available from `Cadabra`. It was imported in the `header`, therefore, we use the algorithm `dsolve` to solve the differential equation.

```
1    eq2 := @(expr2) = 0;
2    solA = solv.dsolve(eq2, $f$);
```

$$-r\partial_r f - f + 1 = 0,\tag{76}$$

$$f = (C_1 + r)r^{-1}.\tag{77}$$

Therefore, the (static) spherically symmetric Ricci flat solution of Einstein field equations requires the metric tensor to be

$$ds^2(g) = -\left(1 + \frac{C_1}{r}\right)dt^2 + \frac{dr^2}{\left(1 + \frac{C_1}{r}\right)} + r^2 d\Omega^2_{(2)}.\tag{78}$$

The constant C_1 is fixed to $-2m$ by the Newtonian limit.

7. Discussion and conclusions

Along the paper, we have introduced (through examples) some of the basic functionalities of `Cadabra2`, focusing on applications to the analysis of gravitational models. We believe that once the user surpasses the initial barrier of the philosophy—in which all the properties of the objects have to be declared—, the notation and manipulation of these objects using the built-in `algorithms` is straightforward. Moreover, one can stack a sequence of these algorithms into a custom function or even build algorithms from scratch, to ease the tasks during the process of solving problems.

In Sec. 2.2 we mentioned the advantage of creating a (*per project*) header file, to facilitate the declaration of variables. That file could—in principle–be shared between projects, but we recommend avoiding such practice, since the personal notation might change (depending for example of the collaborators of the projects).

The tensor manipulation exemplified in Sec. 3 shows the simplicity behind the action of the algorithms on expressions. Nonetheless, we should highlight the fact that in order to achieve that simplicity, one has to relax the mathematical formalities. Notice for example that in our example the Einstein sum convention was dropped in favour of the simplicity of the substitution rules.

More refined tensor manipulations were shown in Sec. 5, in which we used widely the substitution of expressions, and the library `cdb.core.manip` to accomplish our goal. It is important to make an allusion to the fact that within the current version of the software, the `substitute` algorithm is unable to distinguish the expression `ch^{\rho}_{\mu \nu}` from `\Gamma^{\rho}_{\mu \nu}` (despite the fact that we specify the `\Gamma` typography for the `ch` expression). This might cause undesired results.

We show the operational and computational capabilities of `Cadabra2` in handling tensor components, and interact with the `Python` library `SymPy`. Through Sec. 6 we managed to find and solve the Einstein field equations for the Schwarzschild ansatz. Although we stick to `SymPy` as computational backend, if installed in the system, one could use `Mathematica`'s kernel as computational backend.

In Sec. 4 we showed a first application of the ability of `Cadabra2` to operate with differential forms. Even though our example was very simple, it is possible to go further with the current competence with the exterior calculus. However, there is room for several improvements, as for example managing algebra valued differential forms.

To conclude, we want to remark that given the extensible potential of `Cadabra2` through `Python`, it is possible to use the software to target research problems. Future work will focus on applying the above concepts to the analysis of gravitational waves [7].

Appendix

A. Benchmarking in Cadabra

Benchmarking is useful for a variety of purposes; locating bottlenecks, optimisation, examining complexity, comparison again other software packages: the list goes on. In perturbative calculations many of these become relevant and so we developed a timing module which is distributed with this paper.

The basic principle is simple: `Timer` objects measure the time taken between reaching various *checkpoints*, and accumulate these into named bins. For example

```
1   from libraries.timing import Timer
2
3   timer = Timer()
4   timer.start()
5   foo()
6   timer.checkpoint("foo")
7   boring()
8   timer.checkpoint()
9   bar()
10  timer.checkpoint("bar")
11  foo()
12  timer.checkpoint("foo")
```

In this snippet we have three functions; `foo` and `bar` which we want to measure, and `boring` which we need to run but we are not very interested in. After calling `start` on our timer, we cycle through some `foo` routines then call `checkpoint` with the "foo" label. We then do our `boring` routine, and to avoid this cluttering up our final timing results once we're done with this we call `checkpoint` with an empty label. We then call and checkpoint `bar`, followed by a second cycle of `foo`s which accumulates to the time the first call to `checkpoint("foo")` registered.
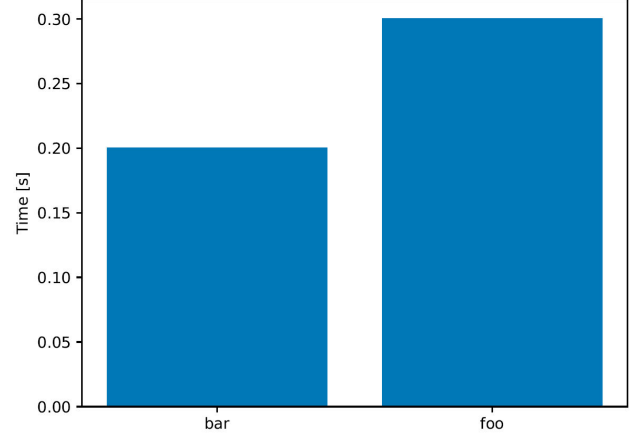

FIGURE 2. Bar chart showing the amount of time spent in executing each routine.

We can have a look at the amount of time spent doing each routine by calling `plot_bar_chart` which uses `matplotlib` to produce a plot of the results (see Fig. 2):

```
13  timer.plot_bar_chart();
```

We can also see how long the entire routine took by calling `total_elapsed`. If we want to include the time spent in our `boring` function, then passing `include_idle=True` adds this to the total:

```
14  print("timer.total_elapsed(False):", timer.
        total_elapsed(False))
15  print("timer.total_elapsed(True):", timer.
        total_elapsed(True))
```

```
timer.total_elapsed(False):
                0.5009605884552002
timer.total_elapsed(True):
                1.5020785331726074
```

B. Fourth-order curvature Lagrangian of the Lanczos–Lovelock series

As a complementary material, we present the term $\mathcal{L}^{(D,4)}$ of the Lanczos–Lovelock series. It is shown in a separated appendix because obtaining it is not relevant for the purpose of the article, however it could be useful for the reader interested in this research topic. The code below follows from the developed in Sec. 3.1.

```
41  LL4 := 8*7*6*5*4*3*2/2/2/2/2/2 R^{a1 a2 b1 b2}
        R^{a3 a4 b3 b4} R^{a5 a6 b5 b6} R^{a7 a8 b7 b8}
        \delta^{a1 b1 a2 b2 a3 b3 a4 b4 a5 b5 a6 b6 a7
        b7 a8 b8};
42  LLmanip(LL4);
```

$$2520 R^{a_1 a_2 b_1 b_2} R^{a_3 a_4 b_3 b_4} R^{a_5 a_6 b_5 b_6} R^{a_7 a_8 b_7 b_8} \delta^{a_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4 a_5 b_5 a_6 b_6 a_7 b_7 a_8 b_8}, \tag{B.1}$$

$$R^4 - 24 R^2 R^{a_1 a_2} R^{a_1 a_2} + 6 R^2 R^{a_1 a_2 a_3 a_4} R^{a_1 a_2 a_3 a_4} + 64 R R^{a_1 a_2} R^{a_1 a_3} R^{a_2 a_3} + 96 R R^{a_1 a_2} R^{a_3 a_4} R^{a_1 a_3 a_2 a_4}$$

$$- 96 R R^{a_1 a_2} R^{a_1 a_3 a_4 a_5} R^{a_2 a_3 a_4 a_5} + 8 R R^{a_1 a_2 a_3 a_4} R^{a_1 a_2 a_5 a_6} R^{a_3 a_4 a_5 a_6} - 32 R R^{a_1 a_2 a_3 a_4} R^{a_1 a_5 a_3 a_6} R^{a_2 a_6 a_4 a_5}$$

$$+ 48R^{a_1a_2}R^{a_1a_2}R^{a_3a_4}R^{a_3a_4} - 24R^{a_1a_2}R^{a_1a_2}R^{a_3a_4a_5a_6}R^{a_3a_4a_5a_6} - 96R^{a_1a_2}R^{a_1a_3}R^{a_2a_4}R^{a_3a_4}$$

$$- 384R^{a_1a_2}R^{a_1a_3}R^{a_4a_5}R^{a_2a_4a_3a_5} + 192R^{a_1a_2}R^{a_1a_3}R^{a_2a_4a_5a_6}R^{a_3a_4a_5a_6} + 96R^{a_1a_2}R^{a_3a_4}R^{a_1a_3a_5a_6}R^{a_2a_4a_5a_6}$$

$$- 192R^{a_1a_2}R^{a_3a_4}R^{a_1a_5a_2a_6}R^{a_3a_5a_4a_6} + 192R^{a_1a_2}R^{a_3a_4}R^{a_1a_5a_3a_6}R^{a_2a_5a_4a_6} + 192R^{a_1a_2}R^{a_1a_3a_2a_4}R^{a_3a_5a_6a_7}R^{a_4a_5a_6a_7}$$

$$- 96R^{a_1a_2}R^{a_1a_3a_4a_5}R^{a_2a_3a_6a_7}R^{a_4a_5a_6a_7} + 384R^{a_1a_2}R^{a_1a_3a_4a_5}R^{a_2a_6a_4a_7}R^{a_3a_7a_5a_6}$$

$$+ 3R^{a_1a_2a_3a_4}R^{a_1a_2a_3a_4}R^{a_5a_6a_7a_8}R^{a_5a_6a_7a_8} - 48R^{a_1a_2a_3a_4}R^{a_1a_2a_3a_5}R^{a_4a_6a_7a_8}R^{a_5a_6a_7a_8}$$

$$+ 6R^{a_1a_2a_3a_4}R^{a_1a_2a_5a_6}R^{a_3a_4a_7a_8}R^{a_5a_6a_7a_8} - 96R^{a_1a_2a_3a_4}R^{a_1a_2a_5a_6}R^{a_3a_7a_5a_8}R^{a_4a_8a_6a_7}$$

$$+ 48R^{a_1a_2a_3a_4}R^{a_1a_5a_3a_6}R^{a_2a_7a_4a_8}R^{a_5a_7a_6a_8} - 96R^{a_1a_2a_3a_4}R^{a_1a_5a_3a_6}R^{a_2a_7a_5a_8}R^{a_4a_7a_6a_8}, \tag{B.2}$$

while the field equations are given by

$$- 11340R^{a_1a_2b_1b_2}R^{a_3a_4b_3b_4}R^{a_5a_6b_5b_6}R^{a_7a_8b_7b_8}\delta^{mna_1b_1a_2b_2a_3b_3a_4b_4a_5b_5a_6b_6a_7b_7a_8b_8}, \tag{B.3}$$

$$G^{(4)mn} = 4R^3R^{mn} - 24R^2R^{a_1a_2}R^{ma_1na_2} - 24R^2R^{ma_1}R^{na_1} + 12R^2R^{ma_1a_2a_3}R^{na_1a_2a_3} + 96RR^{na_1}R^{a_2a_3}R^{ma_2a_1a_3}$$

$$+ 96RR^{a_1a_2}R^{a_1a_3}R^{ma_2na_3} - 48RR^{a_1a_2}R^{ma_1a_3a_4}R^{na_2a_3a_4} - 48RR^{mn}R^{a_1a_2}R^{a_1a_2} + 96RR^{ma_1}R^{na_2}R^{a_1a_2}$$

$$+ 96RR^{ma_1}R^{a_2a_3}R^{na_2a_1a_3} + 96RR^{a_1a_2}R^{ma_3na_4}R^{a_1a_3a_2a_4} - 96RR^{a_1a_2}R^{ma_3a_1a_4}R^{na_3a_2a_4}$$

$$- 48RR^{ma_1}R^{na_2a_3a_4}R^{a_1a_2a_3a_4} + 12RR^{mn}R^{a_1a_2a_3a_4}R^{a_1a_2a_3a_4} - 48RR^{na_1}R^{ma_2a_3a_4}R^{a_1a_2a_3a_4}$$

$$- 48RR^{ma_1na_2}R^{a_1a_3a_4a_5}R^{a_2a_3a_4a_5} - 96RR^{ma_1a_2a_3}R^{na_4a_2a_5}R^{a_1a_5a_3a_4} + 24RR^{ma_1a_2a_3}R^{na_1a_4a_5}R^{a_2a_3a_4a_5}$$

$$- 192R^{na_1}R^{a_1a_2}R^{a_3a_4}R^{ma_3a_2a_4} + 96R^{a_1a_2}R^{a_1a_2}R^{a_3a_4}R^{ma_3na_4} - 96R^{a_1a_2}R^{ma_1a_2a_3}R^{na_4a_5a_6}R^{a_3a_4a_5a_6}$$

$$- 24R^{a_1a_2}R^{ma_1na_2}R^{a_3a_4a_5a_6}R^{a_3a_4a_5a_6} - 192R^{a_1a_2}R^{a_3a_4}R^{ma_1a_2a_5}R^{na_3a_4a_5} - 192R^{a_1a_2}R^{a_1a_3}R^{a_2a_4}R^{ma_3na_4}$$

$$+ 192R^{a_1a_2}R^{a_3a_4}R^{ma_1a_3a_5}R^{na_2a_4a_5} + 192R^{a_1a_2}R^{a_3a_4}R^{ma_1na_5}R^{a_2a_3a_4a_5} - 48R^{a_1a_2}R^{ma_1a_3a_4}R^{na_2a_5a_6}R^{a_3a_4a_5a_6}$$

$$+ 96R^{a_1a_2}R^{ma_1na_3}R^{a_2a_4a_5a_6}R^{a_3a_4a_5a_6} - 192R^{na_1}R^{a_2a_3}R^{a_2a_4}R^{ma_3a_1a_4} + 96R^{a_1a_2}R^{a_1a_3}R^{ma_2a_4a_5}R^{na_3a_4a_5}$$

$$+ 96R^{na_1}R^{a_2a_3}R^{ma_2a_4a_5}R^{a_1a_3a_4a_5} + 192R^{a_1a_2}R^{ma_1a_3a_4}R^{na_5a_3a_6}R^{a_2a_6a_4a_5} + 96R^{ma_1}R^{na_1}R^{a_2a_3}R^{a_2a_3}$$

$$- 48R^{a_1a_2}R^{a_1a_2}R^{ma_3a_4a_5}R^{na_3a_4a_5} + 64R^{mn}R^{a_1a_2}R^{a_1a_3}R^{a_2a_3} - 192R^{ma_1}R^{na_2}R^{a_1a_3}R^{a_2a_3}$$

$$- 192R^{ma_1}R^{a_2a_3}R^{a_2a_4}R^{na_3a_1a_4} - 192R^{a_1a_2}R^{a_1a_3}R^{ma_4na_5}R^{a_2a_4a_3a_5} + 192R^{a_1a_2}R^{a_1a_3}R^{ma_4a_2a_5}R^{na_4a_3a_5}$$

$$- 192R^{ma_1}R^{a_1a_2}R^{a_3a_4}R^{na_3a_2a_4} + 96R^{mn}R^{a_1a_2}R^{a_3a_4}R^{a_1a_3a_2a_4} + 192R^{a_1a_2}R^{a_3a_4}R^{ma_5na_1}R^{a_2a_3a_4a_5}$$

$$+ 96R^{a_1a_2}R^{a_3a_4}R^{ma_5a_1a_3}R^{na_5a_2a_4} + 96R^{ma_1}R^{a_1a_2}R^{na_3a_4a_5}R^{a_2a_3a_4a_5} - 192R^{ma_1}R^{na_2}R^{a_3a_4}R^{a_1a_3a_2a_4}$$

$$- 192R^{ma_1}R^{a_2a_3}R^{na_4a_1a_5}R^{a_2a_4a_3a_5} + 96R^{ma_1}R^{a_2a_3}R^{na_2a_4a_5}R^{a_1a_3a_4a_5} + 192R^{ma_1}R^{a_2a_3}R^{na_4a_2a_5}R^{a_1a_4a_3a_5}$$

$$- 96R^{mn}R^{a_1a_2}R^{a_1a_3a_4a_5}R^{a_2a_3a_4a_5} + 96R^{na_1}R^{a_1a_2}R^{ma_3a_4a_5}R^{a_2a_3a_4a_5} - 192R^{na_1}R^{a_2a_3}R^{ma_4a_1a_5}R^{a_2a_4a_3a_5}$$

$$+ 192R^{na_1}R^{a_2a_3}R^{ma_4a_2a_5}R^{a_1a_4a_3a_5} + 96R^{a_1a_2}R^{ma_3a_4a_5}R^{na_6a_4a_5}R^{a_1a_3a_2a_6} + 96R^{a_1a_2}R^{ma_3na_4}R^{a_1a_3a_5a_6}R^{a_2a_4a_5a_6}$$

$$- 192R^{a_1a_2}R^{ma_3na_4}R^{a_1a_5a_2a_6}R^{a_3a_5a_4a_6} + 192R^{a_1a_2}R^{ma_3a_4a_5}R^{na_3a_4a_6}R^{a_1a_5a_2a_6}$$

$$+ 192R^{a_1a_2}R^{ma_3na_4}R^{a_1a_5a_3a_6}R^{a_2a_5a_4a_6} + 192R^{a_1a_2}R^{ma_3a_4a_5}R^{na_6a_1a_4}R^{a_2a_3a_5a_6}$$

$$+ 96R^{a_1a_2}R^{ma_3na_1}R^{a_2a_4a_5a_6}R^{a_3a_4a_5a_6} - 96R^{a_1a_2}R^{ma_3a_1a_4}R^{na_3a_5a_6}R^{a_2a_4a_5a_6}$$

$$- 192R^{a_1a_2}R^{ma_3a_1a_4}R^{na_5a_4a_6}R^{a_2a_2a_5a_3a_6} - 192R^{a_1a_2}R^{ma_3a_4a_5}R^{na_1a_4a_6}R^{a_2a_5a_3a_6}$$

$$- 96R^{a_1a_2}R^{ma_3a_4a_5}R^{na_3a_1a_6}R^{a_2a_6a_4a_5} + 192R^{a_1a_2}R^{ma_3a_1a_4}R^{na_5a_2a_6}R^{a_3a_6a_4a_5}$$

$$+ 96R^{a_1a_2}R^{ma_3a_4a_5}R^{na_1a_2a_6}R^{a_3a_6a_4a_5} - 24R^{ma_1}R^{na_1}R^{a_2a_3a_4a_5}R^{a_2a_3a_4a_5} + 96R^{ma_1}R^{na_2}R^{a_1a_3a_4a_5}R^{a_2a_3a_4a_5}$$

$$- 48R^{ma_1}R^{na_2a_3a_4}R^{a_1a_2a_5a_6}R^{a_3a_4a_5a_6} + 192R^{ma_1}R^{na_2a_3a_4}R^{a_1a_5a_3a_6}R^{a_2a_6a_4a_5}$$

$$+ 96R^{ma_1}R^{na_2a_1a_3}R^{a_2a_4a_5a_6}R^{a_3a_4a_5a_6} + 8R^{mn}R^{a_1a_2a_3a_4}R^{a_1a_2a_5a_6}R^{a_3a_4a_5a_6} - 32R^{mn}R^{a_1a_2a_3a_4}R^{a_1a_5a_3a_6}R^{a_2a_6a_4a_5}$$

$$+ 96R^{na_1}R^{ma_2a_1a_3}R^{a_2a_4a_5a_6}R^{a_3a_4a_5a_6} - 48R^{na_1}R^{ma_2a_3a_4}R^{a_1a_2a_5a_6}R^{a_3a_4a_5a_6} + 192R^{na_1}R^{ma_2a_3a_4}R^{a_1a_5a_3a_6}R^{a_2a_6a_4a_5}$$

$$+ 96R^{ma_1na_2}R^{a_1a_3a_2a_4}R^{a_3a_5a_6a_7}R^{a_4a_5a_6a_7} - 48R^{ma_1na_2}R^{a_1a_3a_4a_5}R^{a_2a_3a_6a_7}R^{a_4a_5a_6a_7}$$

$$+ 192R^{ma_1na_2}R^{a_1a_3a_4a_5}R^{a_2a_6a_4a_7}R^{a_3a_7a_5a_6} - 192R^{ma_1a_2a_3}R^{na_4a_5a_6}R^{a_1a_5a_2a_7}R^{a_3a_4a_6a_7}$$

$$- 96R^{ma_1a_2a_3}R^{na_4a_5a_6}R^{a_1a_5a_4a_7}R^{a_2a_3a_6a_7} - 96R^{ma_1a_2a_3}R^{na_4a_2a_5}R^{a_1a_5a_6a_7}R^{a_3a_4a_6a_7}$$

$$- 48R^{ma_1a_2a_3}R^{na_4a_2a_3}R^{a_1a_5a_6a_7}R^{a_4a_5a_6a_7} + 192R^{ma_1a_2a_3}R^{na_4a_2a_5}R^{a_1a_6a_3a_7}R^{a_4a_6a_5a_7}$$

$$- 192R^{ma_1a_2a_3}R^{na_4a_2a_5}R^{a_1a_6a_4a_7}R^{a_3a_6a_5a_7} - 48R^{ma_1a_2a_3}R^{na_4a_5a_6}R^{a_1a_7a_2a_3}R^{a_4a_7a_5a_6}$$

$$+ 96R^{ma_1a_2a_3}R^{na_4a_5a_6}R^{a_1a_7a_2a_4}R^{a_3a_7a_5a_6} + 24R^{ma_1a_2a_3}R^{na_1a_4a_5}R^{a_2a_3a_6a_7}R^{a_4a_5a_6a_7}$$

$$- 96R^{ma_1a_2a_3}R^{na_1a_4a_5}R^{a_2a_6a_4a_7}R^{a_3a_7a_5a_6} - 96R^{ma_1a_2a_3}R^{na_1a_2a_4}R^{a_3a_5a_6a_7}R^{a_4a_5a_6a_7}$$

$$+ 12R^{ma_1a_2a_3}R^{na_1a_2a_3}R^{a_4a_5a_6a_7}R^{a_4a_5a_6a_7} - \frac{1}{2}\delta^{mn}L^{(D,4)}. \tag{B.4}$$

Authors declaration

The authors have no conflicts to disclose.

Acknowledgments

---

i. The official site for `Cadabra` is `https://cadabra.science`, and the actual code is hosted at `https://github.com/kpeeters/cadabra2`.

ii. In order to distinguish the output of the code blocks, from the (natural) equations of the paper, we use a light-yellow background for the former.

iii. The content of this section is available on the user contributed notebooks section of the official `Cadabra` webpage.

iv. The minus sign in front of the determinant is necessary because the spacetime has a Lorentzian signature.

v. The project can be downloaded from it official Git-Lab repository `https://gitlab.com/cdbgr/cadabra-gravity-I`.

vi. Note that the zero-forms on M are just functions on the manifold, and therefore $\Omega^0(M) = \mathcal{F}(M)$.

vii. For gauge theories this algebra corresponds to the algebra of the gauge group, while for General Relativity it is the algebra of the Lorentz group. It is worth mentioning that there are modified theories of gravity which consider extended algebras of the gravitational sector, such as the de Sitter, Anti de Sitter or Poincaré algebras.

viii. The values of the indices have been set in the `header` file.

1. M. A. H. MacCallum, Computer algebra in gravity research, *Living Rev. Rel.* **21** (2018) 6, `https://doi.org/10.1007/s41114-018-0015-6`.

2. K. Peeters, Cadabra: a Field-Theory Motivated Symbolic Computer Algebra System, *Comput. Phys. Commun.* **176** (2007) 550, `https://doi.org/10.1016/j.cpc.2007.01.003`.

3. K. Peeters, Introducing Cadabra: A Symbolic Computer Algebra System for Field Theory problems (2007). `https://doi.org/10.48550/arXiv.hep-th/0701238`

4. K. Peeters, Symbolic Field Theory With Cadabra, *Computeralgebra Rundbrief* **41** (2007) 16.

5. K. Peeters, Cadabra2: computer algebra for field theory revisited, *J. Open Source Softw.* **3** (2018) 1118, `https://doi.org/10.21105/joss.01118`.

6. V. V. Baran and D. S. Delion, Analytical approach for the quartet condensation model, *Phys. Rev. C* **99** (2019) 031303, `https://doi.org/10.1103/PhysRevC.99.031303`.

7. O. Castillo-Felisola, D. T. Price, and M. Scomparin, Cadabra and Python algorithms in general relativity and cosmology II: Gravitational waves, *Comput. Phys. Commun.* **289**

(2023) 108748, `https://doi.org/10.1016/j.cpc.2023.108748`.

8. L. Brewin, Using Cadabra for Tensor Computations in General Relativity (2019), `http://arxiv.org/abs/1912.08839v1`.

9. A. Einstein, Zur Allgemeinen Relativitätstheorie, *Sitzungsber. Preuss. Akad. Wiss.* **1** (1915) 778

10. A. Einstein, Die Grundlage Der Allgemeinen Relativitätstheorie, *Ann. Phys.* **49** (1916) 284

11. K. Peeters, Cadabra manual pages, `https://cadabra.science/man.html`.

12. A. Meurer *et al.*, Sympy: Symbolic Computing in Python, *PeerJ Computer Science* **3** (2017) e103, `https://doi.org/10.7717/peerj-cs.103`.

13. C. Lanczos, A Remarkable Property of the Riemann- Christoffel Tensor in Four dimensions, *Annals Math.* (1938) 842

14. D. Lovelock, The Uniqueness of the Einstein Field Equations in a Four-Dimensional Space, *Archive for Rational Mechanics and Analysis* **33** (1969) 54, `https://doi.org/10.1007/BF00248156`.

15. D. Lovelock, The Einstein Tensor and Its generalizations, *J. Math. Phys.* **12** (1971) 498, `https://doi.org/10.1063/1.1665613`.

16. T. Padmanabhan and D. Kothawala, Lanczos-Lovelock Models of Gravity, *Phys. Rep* **531** (2013) 115, `https://doi.org/10.1016/j.physrep.2013.05.007`.

17. F. Müeller-Hoissen, Spontaneous Compactification With Quadratic and Cubic Curvature Terms, *Phys. Lett. B* **163** (1985) 106, `https://doi.org/10.1016/0370-2693(85)90202-3`.

18. F. Müeller-Hoissen, From Chern-Simons To Gauss-Bonnet, *Nucl. Phys. B* **346** (1990) 235, `https://doi.org/10.1016/0550-3213(90)90246-A`.

19. T. Verwimp, On Higher Dimensional Gravity: the Lagrangian, Its Dimensional Reduction and a Cosmological Model, *Class. Quantum Grav.* **6** (1989) 1655, `https://doi.org/10.1088/0264-9381/6/11/018`.

20. J. T. Jebsen, *Ark. Mat. Ast. Fys.* **15** (1921)

21. G. D. Birkhoff, *Relativity and Modern Physics* (Harvard University Press, 1923).

22. W. Alexandrow, *Ann. Der Phys.* **72** (1923) 141

23. J. Eisland, *Trans. Amer. Math. Soc.* **23** (1925) 213

24. L. Brewin, hybrid-latex, (2019).`https://github.com/leo-brewin/hybrid-latex`.