

# The Invar Tensor Package

J.M. Martín-García<sup>a</sup>

<sup>a</sup>*Instituto de Estructura de la Materia, CSIC,  
C/ Serrano 123, Madrid 28006, Spain*

R. Portugal<sup>b</sup>, and L.R.U. Manssur<sup>b</sup>

<sup>b</sup>*Laboratório Nacional de Computação Científica (LNCC),  
Av. Getúlio Vargas 333, Petrópolis, RJ, CEP 25651-075, Brazil*

---

## Abstract

The *Invar* package is introduced, a fast manipulator of generic scalar polynomial expressions formed from the Riemann tensor of a four-dimensional metric-compatible connection. The package can maximally simplify any polynomial containing tensor products of up to seven Riemann tensors within seconds. It has been implemented both in Mathematica and Maple algebraic systems.

*Key words:* Riemann tensor, tensor calculus, Mathematica, Maple, computer algebra

*PACS:* 02.70.Wz, 04.20.-q, 02.40.Ky

---

## Program summary

*Title of program:* Invar Tensor Package

*Catalogue identifier:*

*Program obtainable from:* CPC Program Library, Queens University of Belfast, N. Ireland

*Reference in CPC to previous version:* Computer Physics Communications 157 (2004) 173-180

*Catalogue identifier of previous version:* ADSP

*Does the new version supersede the original program?:* No. The original version runs only in Maple and its purpose is to be a kernel for tensor manipulators. The present versions run in Maple and Mathematica and are full tensor expression manipulators.

*Computers:* Any computer running Mathematica versions 5.0 to 5.2 or Maple versions 9 and 10

*Operating systems under which the new version has been tested:* Linux, Unix, Windows XP

*Programming language:* Mathematica and Maple

*Memory required to execute with typical data:* 30 Mb

*No. of bits in a word:* 64 or 32

*No. of processors used:* 1

*No. of bytes in distributed program, including test data, etc.:* Around 14 Mb

*Distribution format:* Uuencoded compressed tar file

*Nature of physical problem:* Manipulation and simplification of tensor expressions. Special attention on simplifying scalar polynomial expressions formed from the Riemann tensor on a four-dimensional metric-compatible manifold.

*Method of solution:* Algorithms of computational group theory to simplify expressions with tensors that obey permutation symmetries. Tables of syzygies of the scalar invariants of the Riemann tensor.

*Restrictions on the complexity of the problem:* The present versions do not fully address the problem of reducing differential invariants or monomials of the Riemann tensor with free indices.

*Typical running time:* Less than a second to fully reduce a monomial of the Riemann tensor of degree 7 in terms of independent invariants.

## 1 Introduction

The Riemann tensor plays the most essential role in the geometrical description of curved manifolds. Therefore it is a central object of study in General Relativity and Cosmology, containing all invariant (i.e. coordinate independent) and local information about the spacetime. However, it is often very difficult to manipulate expressions containing the Riemann tensor, due to its nontrivial algebraic properties. In the last decade new mathematical ideas have drawn much renewed attention to the subproblem of the algebraic scalars of the Riemann tensors, but we are still far from having a simple and efficient algorithm to manipulate generic expressions. Such an algorithm would be very useful in several areas of Mathematics and Physics, for example in the following particular cases: the generic manipulation of Riemann-related tensors, like Lanczos-type potentials for the Weyl tensor in General Relativity [1]; the quantization of Lagrangians for gravity, which requires the computations of scalar counterterms formed by perturbation of the metric and the Riemann tensor [2]; the classification of metrics using the Cartan-Karlhede algorithm, involving a number (7 at most) of derivatives of the components of the Riemann tensor in some basis [3]. See also the Introduction of Ref. [4] for a list of applications of the invariants of the Riemann tensor.

In this article we present for the first time the *Invar* tensor package, a fast manipulator of expressions containing the Riemann tensor (or its associates

Ricci, Weyl, etc.), focusing on the mentioned subproblem of scalar expressions. Further generalizations will be reported in the future. The package has been implemented for both Mathematica and Maple algebraic systems.

## 2 The problem

A metric  $g_{ab}$  on a  $d$ -dimensional manifold has a Riemann tensor with  $d^2(d^2 - 1)/12$  independent components (assume  $d \geq 3$ ). Of these,  $d(d - 1)/2$  change under rotations of the frame used to define the components and hence have no invariant meaning. The other  $N_d \equiv (d + 3)d(d - 1)(d - 2)/12$  components have invariant meaning and can be combined into an infinite number of scalar polynomials of the Riemann tensor (the so called *invariants* of the Riemann tensor) [5]. In general it is possible to define *differential* invariants, in which covariant derivatives (with respect to the Levi-Civita connection associated to the metric  $g$ ) of Riemann are considered, but we shall restrict ourselves to the *algebraic* invariants of the Riemann tensor in this article.

We shall be primarily concerned with invariants formed by products of  $n$  Riemann tensors with all indices contracted (which we will call *invariants*), or products of  $n$  Riemann tensors and a single  $\epsilon$  tensor (to be called *dual invariants*), also with all indices contracted. In both cases we shall refer to  $n$  as the *degree* of the invariant.

The main question is whether it is possible to find a basis of invariants in terms of which all other invariants can be expressed. Ideally, that basis should have a minimal number of members and those members should have minimal degree. For dimension 4, Narlikar and Karmarkar [6] have given a set of  $N_4 = 14$  invariants such that, for generic Lorentzian metrics, it could be possible to give any invariant in terms of the objects of the basis, but possibly involving roots of high degree. For special metrics the expressions could become singular. A basis of 17 objects have been given, using spinor theory, by Carminati and Zakhary [7] which is guaranteed to stay regular for all combinations of the 15 Petrov and 6 Segre types. Using the properties of  $3 \times 3$  symmetric matrices, Sneddon [8] has constructed a basis of 38 invariants such that any other invariant can be given as a polynomial in that basis. For none of those three sets it is known how to generate the expressions (called syzygies) for given invariants in terms of the corresponding bases, though a partial solution has been given recently by Carminati and Lim [9]. Those three sets are highly specialized for 4d Lorentzian metrics, and it seems impossible to generalize them to other cases.

In this article we develop completely new methods to deal with the same problem. We shall construct a basis of invariants up to degree 7 in invariants

and degree 5 in dual invariants, and implement very efficient algorithms to express any other invariant within those limits in terms of that basis. We shall see that our basis contains the 25 invariants of Sneddon's basis with lower degrees than those limits. Our method is purely tensorial, not based on spinors, and hence the results are valid (or can be easily generalized) to non-Lorentzian metrics and other dimensions. They can also be generalized to differential invariants, or even to expressions with free indices. The method is purely based on the Riemann tensor, and does not require its decomposition into Ricci and Weyl parts, a dimensionally dependent decomposition. However, to facilitate the comparison with the previous literature, mostly based on that decomposition, we also provide tools to translate from Weyl invariants, as shown in the appendix. We stress the fact that the basis is constructed simultaneously with all syzygies.

### 3 Algorithms

A general polynomial expression in the Riemann tensor is first expanded into a sum of monomial products of tensors, each one canonicalized independently. Unfortunately, there is no known efficient (i.e. fast and with polynomial scaling in degree) algorithm to manipulate in real time products of Riemann tensors. Previous tensor computer algebra systems use a number of special rules for restricted cases (c.f. the nearly 40 rules of *MathTensor* [10] for monomials of degree three or less), or slow though rather general methods, for example those of *Tools of Tensor Calculus* [11] (based on the construction of large linear systems of equations [12]) and *cadabra* [13] (using Young tableaux decompositions). In any case, it is not possible to go beyond degree 4 in computations taking less than a minute with these systems.

Here we employ a combined approach which uses different algorithms at four different steps (called A, B, C, and D), depending on which of four sets of properties of the Riemann tensor we use: We shall use efficient permutation-group algorithms to canonicalize a given monomial in real time, with respect to the permutation symmetries, into one of several thousand possible canonical monomials (step A). Then we shall use a database of solutions in which we have previously stored the canonical form of any of those monomials with respect to harder symmetries (steps B, C and D), which cannot be handled efficiently in real time. The construction of the database with all those canonical forms is carried out using essentially a brute force approach based on intensive computer power to solve large systems of linear equations. At each of the steps B, C and D new equations will be generated, decreasing the number of independent invariants.

We now give some details and examples of the procedures followed at each

step:

**Step A: Permutation symmetries.** These are the permutation of indices that maintain the tensor invariant modulo a minus sign. For the Riemann tensor,

$$R_{bacd} = -R_{abcd}, \quad R_{cdab} = R_{abcd} \quad (1)$$

are examples of permutation symmetries. The Riemann tensor obeys the cyclic symmetries which are not permutation symmetries, but multiterm symmetries. The symmetries of the metric tensor  $g_{ab}$  and the totally anti-symmetric tensor  $\epsilon_{abcd}$  can also be fully described in terms of permutation symmetries.

Computational group theory has the tools that one needs to address this type of symmetries [14]. For example, the permutation symmetries of the Riemann tensor are described by a subgroup of the symmetric group acting on 4 points, because Riemann is a rank-4 tensor. The permutation symmetries of a product of  $n$  Riemann tensors is a permutation group acting on  $4n$  points, which takes into account not only the permutation symmetries of each tensor but also the permutation of factors. Let us call  $S$  the permutation group of the Riemann invariants of a fixed degree. If there are dummy indices in a tensor expression, there is a new kind of permutation symmetries that comes from dummy index relabelling. Those are described by another permutation group which we call  $D$ . All equivalent index configurations of a Riemann invariant of degree  $n$  are described by a double coset of  $S$  and  $D$  in the symmetric group  $S_{4n}$ . A list of canonical invariants is given by a transversal set of the double cosets of  $S$  and  $D$  in  $S_{4n}$ . There is no efficient way to generate a transversal set. We simply generated randomly invariants until getting one representative for each double coset. By using GAP [15], we confirmed that the transversal set is complete. This strategy must be repeated for each degree. We succeeded in generating the Riemann invariants up to degree 7 using a few GBytes of RAM memory.

We addressed two kinds of Riemann invariants: (1) a product of  $n$  Riemann tensors, which we denote by  $I_{n,r}$  and (2) a product of  $n$  Riemann tensors and a 4-index  $\epsilon$  tensor, denoted by  $D_{n,r}$ , where  $n$  is the degree and the index  $r$  gives the position of the invariant in the list sorted according to a predefined order of permutations. The set  $I_{n,r}$  for all values of  $r$  for a fixed  $n$  is a canonical transversal set with respect to the permutation symmetries involved. The same holds for  $D_{n,r}$ . Those are the invariants that we need to manipulate for the rest of the article. The number of invariants per degree is shown in Table 1. Note for example for the degree 7 invariants the enormous decrease from the complete set of  $28! \sim 3 \cdot 10^{29}$  invariants to 19610 canonical ones.

As an example, the monomial of degree 6

$$R^{bca}{}_c R_{gdh}{}^e R^{gfh}{}_f R_{kba}{}^i R^{kmd}{}_l R_{mei}{}^l$$

Degree	Invariants	Dual invariants
1	1	1
2	4	5
3	13	35
4	57	288
5	288	3031
6	2070	—
7	19610	—

Table 1

Number of independent invariants and dual-invariants after imposing the permutation symmetries.

is converted after step A into

$$I_{6,246} \equiv R^{ab} R^{cd} R_a^e R_b^f R_c^g R_d^h R_e^i R_f^j R_{ghij} \quad (2)$$

If we remove those invariants which are explicit products of invariants of smaller degree, then the numbers decrease even more, and are those shown in columns A of Table 2.

**Step B: The cyclic symmetry.** This is a multiterm symmetry involving three instances of index configurations of the Riemann tensor:

$$R_{abcd} + R_{acdb} + R_{adbc} = 0, \quad (3)$$

which expresses the fact that the Riemann tensor has vanishing totally antisymmetric part; that is, (3) is equivalent to  $R_{[abcd]} = 0$  applying (1). Using this symmetry on the tensors of each of the canonical invariants obtained after step A we generate many new equations relating three invariants of the same degree, hence reducing the number of independent invariants to those in columns B of Table 2. A simple example, applying the cyclic symmetry on the last Riemann tensor of  $I_{6,246}$ , defined in (2), is

$$I_{6,246} - I_{6,243} - I_{6,245} = 0,$$

so that the invariant  $I_{6,246}$  is not independent after step B. Note that so far our results concerning the non-dual invariants are valid for the Riemann tensor of any metric in any dimension. Those for the dual invariants are only valid in 4 dimensions.

**Step C: Dimensionally dependent identities.** Antisymmetrization over more than  $d$  indices in dimension  $d$  gives zero. This simple fact can be used to generate new relations (Lovelock type [16]), which will be therefore valid only for dimensions smaller or equal to  $d$ . From now on we restrict ourselves to dimension 4, and hence generate new equations among the invariants by antisymmetrizing over five indices the independent invariants

that were output by steps A and B. For example, antisymmetrizing over the contravariant indices  $b, c, e, f, h$  of (2) gives a linear relation among 12 of the 270 independent invariants of degree 6 after step B.

This step requires solving large systems of linear equations. For example, for the degree 7 invariants, we generated about 4 equations for each independent invariant, producing a system of more than 5000 equations with integer coefficients for 1639 unknowns. Though the initial integers and the final rational numbers had only a few digits, some intermediate integer numbers had more than 900 digits. One may reduce the number of equations and still get the correct answer, but the number of equations cannot be too close to the number of independent invariants (1639 for degree 7).

**Step D: Signature dependent identities.** Now we have two collections of invariants: those containing the totally antisymmetric tensor (the “duals”) and those not containing it (the “non-duals”). We can relate them by expansion of the product of two  $\epsilon$  tensors using the formula

$$\epsilon^{a_1 \dots a_n} \epsilon_{b_1 \dots b_n} = \sigma \begin{vmatrix} \delta_{b_1}^{a_1} & \dots & \delta_{b_1}^{a_n} \\ \vdots & & \vdots \\ \delta_{b_n}^{a_1} & \dots & \delta_{b_n}^{a_n} \end{vmatrix}, \quad (4)$$

where  $\sigma$  represents the sign of the determinant  $g$  of the metric. That means now that the result will depend on that sign. We follow the convention  $\epsilon_{0123} = \sqrt{|g|}$ , and hence  $\epsilon^{0123} = \sigma/\sqrt{|g|}$ .

This allows us to generate new equations by multiplying pairs of dual invariants with degrees  $n_1$  and  $n_2$ . This generates a polynomial in non-dual invariants of total degree  $n_1 + n_2$ . In general this polynomial contains an invariant of degree  $n_1 + n_2$  and hence a high degree invariant can be expressed in terms of the product of two lower degree dual invariants (the latter play the role of “square roots” of the former). In this way the degree of the elements of the basis can be reduced (see an application of this in appendix B for the Narlikar and Karmarkar basis). However, this method highly depends on the dimension of the manifold, and on the signature of the metric (through the sign  $\sigma$ ). As an example, starting from  $D_{3,2} \equiv -R^{ab}R^{cd}R_{ac}{}^{ef}\epsilon_{bdef}$  we find

$$\sigma D_{3,2}^2 = 4I_{6,11} + 16I_{6,19} + 4I_{6,23}.$$

For definitions of all canonical invariants see the package files.

Degree	A	A*	B	B*	C	C*	D	D*
1	1	1	1	0	1	0	1	0
2	3	4	2	1	2	1	2	1
3	9	27	5	6	3	2	3	2
4	38	232	15	40	4	1	3	1
5	204	2582	54	330	5	2	3	2
6	1613	–	270	–	8	–	4	–
7	16532	–	1639	–	7	–	3	–

Table 2

Number of independent invariants or dual-invariants (denoted with a star) after imposing the different types of relations: A) permutation symmetries, B) cyclic symmetry, C) dimensionally dependent relations (for dimension 4), and D) signature dependent relations (for a metric of negative determinant).

## 4 Implementation

We have implemented the Invar tensor package on Mathematica and Maple computer algebra systems. The Mathematica implementation is on top of the xTensor package [17] and the Maple implementation on top of the Canon tensor package [18]. The underlying commands for tensor manipulation of these packages are fully available, such as metric contraction, conversion of Riemann to Weyl and vice-versa. Both versions share a database with the syzygies up to degree 7 (degree 5 for duals). The invariants are stored as permutations in the disjoint cyclic notation.

The main command of the Invar package, `RiemannSimplify`, canonicalizes tensor expressions with Riemann invariants up to degree 7 and dual invariants up to degree 5. The canonicalization process follows the steps described in Table 2. Each symmetry kind is used one at a time: first the permutation symmetries, second the cyclic symmetries, third the dimension dependent identities, and finally signature dependent identities. There is an option in `RiemannSimplify` that prevents the use of the dimension dependent identities, for users that wish to work with manifolds of generic dimension.

The user has access to all tables of independent invariants and reducing equations. Besides, there are commands to convert Riemann invariants from tensor notation to  $I_{n,r}$  and  $D_{n,r}$  notation (`RInv[n,r]` and `DualRInv[n,r]` in the package) and vice-versa. It is also possible to relate the Narlikar and Karmarkar basis with these invariants (see appendix B).

Tensors are indexed objects such as `Riemann[a,-b,-c,-d]`, which stands for  $R^a{}_{bcd}$ . Contravariant indices are positive and covariant are negative. For ex-



ample, the expression

$$\epsilon^{abcd} R_{ab}{}^{ef} R_{cef}{}^g R_d{}^{hij} R_{gi} R_{hj} + \frac{(\epsilon^{abcd} R_{ab}{}^{ef} R_{cef}{}^g) (R^{ghij} R_{gi} R_{hj})}{8}$$

is identically zero in dimension 4 as can be verified by employing the command

```
expr = epsilon[a,b,c,d]*
      R[-a,-b,e,f]*R[-c,-e,-f,g]*R[-d,h,i,j]*R[-g,-i]*R[-h,-j]+
      1/8*(epsilon[a,b,c,d]*R[-a,-b,e,f]*R[-c,-d,-e,-f])*
      (R[g,h,i,j]*R[-g,-i]*R[-h,-j]);
RiemannSimplify[ expr ]
```

in the Mathematica implementation of the Invar package, or by

```
RiemannSimplify( expr );
```

0

in the Maple implementation.

The list of independent dual invariants of degree 3 and the respective tensor expressions can be generated by the following command in Mathematica

```
Do [ If [ InvSimplify[ DualRInv[3,i] ] == DualRInv[3,i] ,
        Print[ DualRInv[3,i] -> InvToRiemann[ DualRInv[3,i] ] ]
      ],
    { i, MaxDualIndex[ 3 ] } ]
```

and in Maple

```
for i to MaxDualIndex( 3 ) do
  if RiemannSimplify( DualRInv[3,i] ) = DualRInv[3,i] then
    print( DualRInv[3,i] = InvToRiemann( DualRInv[3,i] ) )
  end if
end do;
```

$$DualRInv_{3,7} = R^{a1 a2}{}_{a1}{}^{a3} R_{a2}{}^{a4 a5 a6} R_{a4}{}^{a7 a8} \epsilon_{a3 a5 a6 a8}$$

$$DualRInv_{3,20} = R^{a1 a2 a3 a4} R_{a1 a2}{}^{a5 a6} R_{a3 a4}{}^{a7 a8} \epsilon_{a5 a6 a7 a8}$$

The complete list of commands is described in the appendix C for the Mathematica implementation and in the appendix D for the Maple implementation.

## 5 Conclusions

We have described a systematic procedure to obtain all relations (called syzygies) among the scalar invariants of the Riemann tensor. The procedure relies on a fast algorithm that canonicalizes Riemann monomials using the permutation symmetries of the Riemann tensor. The cyclic relations and the dimension and signature dependent identities are calculated using antisymmetrization methods. The resulting system of equations is solved and stored in a database using the disjoint cyclic notation for permutations. Once the database has been produced and optimized, this approach allows much faster simplifications than real time methods and requires much less memory. Both approaches use exponential algorithms, but our method does it only once, and including the dimensionally-dependent identities (not included in the previously mentioned real-time systems). After loading the database, expressions with scalar invariants are quickly simplified in terms of independent invariants of minimum degree.

The *Invar* tensor package, implemented both in Mathematica and Maple on top of the *xTensor* and *Canon* tensor packages respectively, allows the user to manipulate the database of invariants and syzygies. The database is the same for both implementations.

The method used to build the database can be generalized straightforwardly to find syzygies among differential invariants and monomials of the Riemann tensor with free indices. We are currently analyzing such extensions.

## Acknowledgements

JMM thanks the LNCC for support and hospitality. JMM was supported by the Spanish MEC under the research project FIS2005-05736-C03-02. Part of the computations were performed at the *Centro de SuperComputación de Galicia* (CESGA).

## A Independent invariants

The final list of 25 independent invariants is given in table A.1. The invariants are internally handled in the *Invar* package as permutations of the indices of a product of Riemann tensors (and a 4-index  $\epsilon$  tensor for duals), as given in the ‘Only Riemann’ column. When the contracted Riemann tensors are

Invariant	Only Riemann	Riemann and Ricci
$I_{1,1}$	$R^{ab}{}_{ab}$	$R$
$I_{2,1}$	$R^{ab}{}^c{}_d R_b{}^d{}_{cd}$	$R^{ab} R_{ab},$
$I_{2,2}$	$R^{abcd} R_{abcd}$	$R^{abcd} R_{abcd}$
$D_{2,2}$	$R^{abcd} R_{ab}{}^{ef} \epsilon_{cdef}$	$R^{abcd} R_{ab}{}^{ef} \epsilon_{cdef}$
$I_{3,1}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_c{}^f{}_{ef}$	$-R^{ab} R_a{}^c R_{bc},$
$I_{3,2}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_d{}^f{}_{ef}$	$R^{ab} R^{cd} R_{acbd},$
$D_{3,2}$	$R^{ab}{}^c{}_d R_b{}^{def} R_d{}^g{}_g{}^h{}_{efgh}$	$-R^{ab} R^{cd} R_{ac}{}^{ef} \epsilon_{bdef}$
$I_{3,5}$	$R^{abcd} R_{ab}{}^{ef} R_{cdef}$	$R^{abcd} R_{ab}{}^{ef} R_{cdef}$
$D_{3,13}$	$R^{abcd} R_{ab}{}^{ef} R_{cd}{}^{gh} \epsilon_{efgh}$	$R^{abcd} R_{ab}{}^{ef} R_{cd}{}^{gh} \epsilon_{efgh}$
$I_{4,1}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_c{}^f{}_f{}^g{}_{gh}$	$R^{ab} R_a{}^c R_b{}^d R_{cd},$
$I_{4,5}$	$R^{ab}{}^c{}_d R_b{}^{def} R_c{}^g{}_{ef} R_d{}^h{}_{gh}$	$R^{ab} R^{cd} R_{ac}{}^{ef} R_{bdef}$
$I_{4,7}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_d{}^f{}_e{}^g{}_{gh}$	$R^{ab} R^{cd} R_a{}^e{}_b{}^f{}_{cedf}$
$D_{4,7}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_d{}^{fgh} R_f{}^i{}_i{}^j{}_{eghj}$	$-R^{ab} R^{cd} R_a{}^e{}_b{}^f{}_{ce}{}^{gh} \epsilon_{dfgh}$
$I_{5,2}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_d{}^f{}_f{}^g{}_{gh} R_e{}^h{}_h{}^i{}_{gij}$	$R^{ab} R_a{}^c R_b{}^d R^{ef} R_{cedf}$
$D_{5,2}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_c{}^f{}_f{}^g{}_{gh} R_e{}^{hij} R_h{}^k{}_k{}^l{}_{gijl}$	$-R^{ab} R_a{}^c R_b{}^d R^{ef} R_{ce}{}^{gh} \epsilon_{dfgh}$
$I_{5,8}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_d{}^f{}_e{}^g{}_{gh} R_f{}^h{}_h{}^i{}_{gij}$	$-R^{ab} R_a{}^c R^{de} R_b{}^f{}_c{}^g{}_{dfeg}$
$I_{5,33}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_d{}^{fgh} R_e{}^i{}_{gh} R_f{}^j{}_{ij}$	$R^{ab} R^{cd} R_a{}^e{}_b{}^f{}_{ce}{}^{gh} R_{dfgh}$
$D_{5,76}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_d{}^f{}_e{}^g{}_{gh} R_f{}^{hij} R_h{}^k{}_k{}^l{}_{gijl}$	$-R^{ab} R^{cd} R_a{}^e{}_b{}^f{}_{ce}{}^{ghi} R_{egf}{}^j{}_{cdhij}$
$I_{6,6}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_c{}^f{}_f{}^g{}_{gh} R_e{}^{hij} R_g{}^k{}_{ij} R_h{}^l{}_{kl}$	$R^{ab} R_a{}^c R_b{}^d R^{ef} R_{ce}{}^{gh} R_{dfgh}$
$I_{6,8}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_d{}^f{}_e{}^g{}_{gh} R_f{}^h{}_h{}^i{}_{gij} R_g{}^j{}_{jk} R_i{}^l{}_{kl}$	$R^{ab} R_a{}^c R_b{}^d R^{ef} R_c{}^g{}_d{}^h{}_{egfgh}$
$I_{6,47}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_c{}^f{}_e{}^g{}_{gh} R_f{}^{hij} R_g{}^k{}_{ij} R_h{}^l{}_{kl}$	$-R^{ab} R_a{}^c R^{de} R_b{}^f{}_c{}^g{}_{df}{}^{hi} R_{eghi}$
$I_{6,242}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_d{}^{fgh} R_e{}^i{}_{gh} R_f{}^j{}_{ij} R_j{}^k{}_{kl}$	$R^{ab} R^{cd} R_a{}^e{}_b{}^f{}_{ce}{}^g{}_d{}^h{}_{eg}{}^{ij} R_{fghij}$
$I_{7,14}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_c{}^f{}_e{}^g{}_{gh} R_f{}^h{}_g{}^i{}_{gh} R_h{}^j{}_{jk} R_i{}^l{}_{lm} R_k{}^n{}_{mn}$	$-R^{ab} R_a{}^c R_b{}^d R^{ef} R_e{}^g{}_c{}^h{}_d{}^i{}_{fghij}$
$I_{7,55}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_c{}^f{}_f{}^g{}_{gh} R_e{}^h{}_g{}^i{}_{gh} R_h{}^{jkl} R_i{}^m{}_{kl} R_j{}^n{}_{mn}$	$R^{ab} R_a{}^c R_b{}^d R^{ef} R_c{}^g{}_d{}^h{}_{eg}{}^{ij} R_{fghij}$
$I_{7,391}$	$R^{ab}{}^c{}_d R_b{}^d{}^e{}_f R_d{}^{fgh} R_e{}^i{}_{gh} R_f{}^j{}_{ij} R_j{}^k{}_{kl} R_k{}^l{}_{lm} R_l{}^m{}_{mn}$	$-R^{ab} R_a{}^c R^{de} R_b{}^f{}_c{}^g{}_d{}^h{}_e{}^i{}_{fgh}{}^{jk} R_{gijk}$

Table A.1

All independent invariants for dimension 4, up to degrees 7 (non-duals) and 5 (duals).

transformed into Ricci the canonicalized expressions give a relative sign in several cases ('Riemann and Ricci' column).

## B The Narlikar and Karmarkar basis

This basis does not contain dual invariants, which increases the degree of the elements of the basis from 6 to 7. It is equivalent to the invariants  $I_{1,1}$ ,  $I_{2,1}$ ,  $I_{2,2}$ ,  $D_{2,2}$ ,  $I_{3,1}$ ,  $I_{3,2}$ ,  $D_{3,2}$ ,  $I_{3,5}$ ,  $D_{3,13}$ ,  $I_{4,1}$ ,  $I_{4,5}$ ,  $I_{5,2}$ ,  $D_{5,2}$  and  $I_{6,6}$ , if  $D_{2,2} \neq 0$ . As shown below,  $D_{2,2}$  is a pure Weyl invariant, and hence it does not vanish for non-flat spacetimes (not even for curved *vacuum* spacetimes).

Pure Ricci invariants:

$$I_1 \equiv R = I_{1,1}, \quad (\text{B.1})$$

$$I_2 \equiv R^{ab}R_{ab} = I_{2,1}, \quad (\text{B.2})$$

$$I_3 \equiv R^{ab}R_a{}^cR_{bc} = -I_{3,1}, \quad (\text{B.3})$$

$$I_4 \equiv R^{ab}R_a{}^cR_b{}^dR_{cd} = I_{4,1}, \quad (\text{B.4})$$

Pure Weyl invariants:

$$J_1 \equiv W^{abcd}W_{abcd} = I_{2,2} - 2I_{2,1} + \frac{1}{3}I_{1,1}^2, \quad (\text{B.5})$$

$$\begin{aligned} J_2 &\equiv W^{abcd}W_{ab}{}^{ef}W_{cdef} \\ &= I_{3,5} - 6I_{3,2} + 6I_{3,1} - \frac{1}{2}I_{1,1}I_{2,1} + 7I_{1,1}I_{2,1} - \frac{17}{18}I_{1,1}^3, \end{aligned} \quad (\text{B.6})$$

$$J_3 \equiv W^{abcd}W_{ab}{}^{ef}W_{cd}{}^{gh}W_{efgh} - \frac{1}{4}J_1^2 = \frac{\sigma}{16}D_{2,2}^2, \quad (\text{B.7})$$

$$\begin{aligned} J_4 &\equiv W^{abcd}W^{efgh}W_{abef}W_{cd}{}^{ij}W_{ghij} - \frac{5}{12}J_1J_2 \\ &= -\frac{5\sigma}{96}D_{2,2}(-2D_{3,13} - 4D_{3,2} + D_{2,2}I_{1,1}), \end{aligned} \quad (\text{B.8})$$

Mixed invariants:

$$K_1 \equiv R^{ab}R^{cd}W_{acbd} = I_{3,2} - I_{3,1} - \frac{7}{6}I_{1,1}I_{2,1} + \frac{1}{6}I_{1,1}^3, \quad (\text{B.9})$$

$$\begin{aligned} K_2 &\equiv R^{ab}R^{cd}W_{ac}{}^{ef}W_{bdef} \\ &= I_{4,5} + 2I_{4,1} - \frac{4}{3}I_{1,1}I_{3,2} - 3I_{1,1}I_{3,1} - I_{2,1}^2 + \frac{41}{18}I_{1,1}^2I_{2,1} - \frac{5}{18}I_{1,1}^4, \end{aligned} \quad (\text{B.10})$$

$$\begin{aligned} K_3 &\equiv R^{ab}R^{cd}W_{ac}{}^{ef}W_{bd}{}^{gh}W_{efgh} - \frac{1}{4}J_1K_1 + \frac{1}{12}(I_2 - I_1^2)J_2 \\ &= -\frac{\sigma}{16}D_{2,2}D_{3,2}, \end{aligned} \quad (\text{B.11})$$

$$\begin{aligned} K_4 &\equiv R^{ab}R_a{}^cR^{de}R_d{}^fW_{becf} \\ &= -2I_{5,2} - \frac{7}{6}I_{1,1}I_{4,1} + \frac{1}{2}(I_{2,1} + I_{1,1}^2)I_{3,2} - \frac{1}{6}(I_{2,1} - 11I_{1,1}^2)I_{3,1} \end{aligned}$$

$$+\frac{2}{3}I_{1,1}I_{2,1}^2-\frac{4}{3}I_{1,1}^3I_{2,1}+\frac{1}{6}I_{1,1}^5, \quad (\text{B.12})$$

$$\begin{aligned} K_5 &\equiv R^{ab}R_a{}^cR^{de}R_d{}^fW_{be}{}^{gh}W_{cfgh} = \\ &= -2I_{6,6} - \frac{4}{3}I_{1,1}I_{5,2} + \frac{1}{2}(I_{2,1} + I_{1,1}^2)I_{4,5} + \frac{1}{4}(2I_{2,1} - I_{2,2})I_{4,1} \\ &\quad - \frac{13}{18}I_{1,1}^2(I_{4,1} + I_{1,1}I_{3,1}) + \frac{1}{3}(I_{1,1}I_{2,1} - I_{1,1}^3 - 2I_{3,1})I_{3,2} + \frac{1}{3}I_{3,1}^2 \\ &\quad - \frac{1}{6}I_{1,1}I_{2,2}I_{3,1} + \frac{7}{9}I_{1,1}I_{2,1}I_{3,1} + \frac{1}{24}(3I_{2,1}^2 - I_{1,1}^4)I_{2,2} - \frac{1}{4}I_{2,1}^3 \\ &\quad + \frac{7}{18}I_{1,1}^2I_{2,1}^2 + \frac{1}{36}I_{1,1}^4(I_{1,1}^2 - 8I_{2,1}), \end{aligned} \quad (\text{B.13})$$

$$\begin{aligned} K_6 &\equiv R^{ab}R_a{}^cR^{de}R_d{}^fW_{be}{}^{gh}W_{cf}{}^{ij}W_{ghij} - \frac{1}{4}J_1K_4 + \frac{1}{12}(I_4 - I_2^2)J_2 \\ &= \frac{\sigma}{32}D_{2,2}(4D_{5,2} - D_{3,2}(I_{1,1}^2 + I_{2,1})). \end{aligned} \quad (\text{B.14})$$

## C Main commands in the Mathematica implementation

This appendix lists the main commands of the Invar tensor package in the Mathematica implementation version. This version is written on top of the xTensor package [17].

`InvSimplify[expr, sl]` is the general simplifier of scalar invariants in `expr` when expressed with heads `RInv` and `DualRInv`. The optional argument `sl` is an integer specifying the simplification level: 1 (only permutation symmetries), 2 (also cyclic symmetries), 3 (also dimensionally-dependent identities) and 4 (also signature-dependent identities), which correspond to columns A, B, C and D of table 2, respectively. Its default value is 4.

`RiemannToInv[expr]` converts the algebraic expression `expr` containing Riemann and Ricci tensors into an expression with objects with head `RInv` and `DualRInv`. Conversely, `InvToRiemann[expr]` transforms those objects in `expr` into their corresponding Riemann expressions.

`RiemannSimplify[expr]` is equivalent to the consecutive action on `expr` of `RiemannToInv`, then `InvSimplify`, and then `InvToRiemann`.

`Invs[sl, d]` gives a list off all independent invariants of degree `d` after simplification level `sl`. `DualInvs[sl, d]` does the same for dual invariants.

`MaxIndex[d]` and `MaxDualIndex[d]` return the maximum value that the index `n` can assume in `RInv[d, n]` and `DualRInv[d, n]` respectively.

These are functions of xTensor which can be useful in the Invar package:

`DefManifold[M, dim, {a, b, c, ...}, s]` defines a manifold  $M$  of dimension  $\text{dim}$  and associates the abstract indices  $\{a, b, c, \dots, s\}$  to it.

`DefMetric[ $\sigma$ , g[-a, -b], CD, {";", "\nabla"}]` defines the metric  $g$  on the manifold  $M$ , such that its determinant has sign  $\sigma$ . Its Levi-Civita connection  $CD$  is also defined, as well as the associated curvature tensors `RiemannCD[-a, -b, -c, d]`, `RicciCD[-a, -b]`, `RicciScalarCD[]` and `WeylCD[-a, -b, -c, -d]`, and the totally antisymmetric tensor `epsilong[-a, -b, -c, -d]` (in dimension 4).

`ContractMetric[expr, g]` simplifies contractions with the metric  $g$  in the expression `expr`.

`RiemannToWeyl[expr]` expands Riemann tensors into Weyl, Ricci and RicciScalar parts. `WeylToRiemann[expr]` performs the opposite task.

`RiemannToRicci[expr]` converts contracted Riemann tensors into Ricci tensors, and contracted Ricci into RicciScalar.

## D Main commands in the Maple implementation

This appendix lists the main commands of the Invar tensor package in the Maple implementation version. This version is written on top of the Canon package [18]. The arguments between square brackets are optional.

`RiemannSimplify(expr, [opt])` is a general simplifier of scalar invariants. It contracts any metric tensor in `expr`, simplifies contractions of the Weyl tensor, converts products of Riemann tensors into `RInvs` and `DualRInvs` whenever possible, reduces them using cyclic and dimension dependent identities, and simplifies contractions the Riemann tensor into Ricci and Ricci scalars. The option `opt` controls the simplifying level. The default is `DDI`, which means that the dimension dependent identities are employed. The option `Cyclic` tells `RiemannSimplify` to use only the cyclic identities.

`RiemannToInv(expr)` converts the algebraic expression `expr` that has Riemann tensors into an expression with `RInvs` and `DualRInvs`. `InvToRiemann(expr)` converts the algebraic expression `expr` that has `RInvs` and `DualRInvs` into an expression with Riemann tensors.

`RiemannToRicci(expr)` simplifies contractions of Riemann tensors into Ricci tensors or Ricci scalars. `RicciToRiemann` converts Ricci tensors into Riemann tensors.

`WeylToRiemann(expr)` converts Weyl tensors into Riemann tensors. `RiemannToWeyl(expr)` converts Riemann tensors into Weyl tensors.

`MaxIndex(degree)` and `MaxDualIndex(degree)` return the maximum value that the index `n` can assume in `RInv[degree,n]` and `DualRInv[degree,n]` respectively.

It follows a short description of commands of the Canon package useful in the context.

`TensorDefine(T, n, S, [B])` defines the symmetries of a tensor. `T` is the tensor name, `n` is the number of indices, `S` is a set of permutations, and `B` is the base. For example, the tensor `A` with 3 indices and antisymmetric in the last two indices is defined in the following way: `TensorDefine(A,3,[-1,[[2,3]]],[1,2,3])`. Totally symmetric or antisymmetric tensors can be defined by using the option `Symmetric` or `AntiSymmetric` as the third argument. The list of built-in tensors is: Riemann tensor `R[a,b,c,d]`, Ricci tensor `R[a,b]`, Ricci scalar `R[ ]`, totally antisymmetric tensor `epsilon[a,b,c,d]`, metric tensor `g[a,b]`, Weyl tensor `C[a,b,c,d]`.

`Canonical(expr)` uses the permutation symmetries of the built-in tensors (or tensors defined by the user) to canonicalize the tensor expression `expr`. No further simplification is performed. Most commands of the package uses `Canonical` internally.

`AbsorbMetric(expr)` simplifies contractions with the metric tensor.

## References

- [1] S. B. Edgar and J. M. M. Senovilla, *Class. Quantum Grav.* **21**, L133 (2004).
- [2] M. H. Goroff and A. Sagnotti, *Nucl. Phys. B* **266**, 709 (1986).
- [3] D. Pollney, J. F. Skea and R. d’Inverno, *Class. Quantum Grav.* **17**, 643 (2000).
- [4] C. Cherubini, D. Bini, S. Capozziello and R. Ruffini, *Int. J. Mod. Phys. D* **11**, 827 (2002).
- [5] *Gravitation and Cosmology*, Steven Weinberg, John Wiley & Sons Inc., 1972, Section 6.7.
- [6] V. V. Narlikar and K. R. Karmarkar, *Proc. Indian Acad. Sci.* **129**, 91 (1947).
- [7] J. Carminati and E. Zakhary, *J. Math. Phys.* **42**, 1474 (2001).
- [8] G. E. Sneddon, *J. Math. Phys.* **40**, 5905 (1999).
- [9] J. Carminati and A. E. K. Lim, *J. Math. Phys.* **47**, 052504 (2006).
- [10] L. Parker and S. M. Christensen, *MathTensor: A system for doing tensor analysis by computer*, Addison-Wesley Publishing Company (1994).

- [11] *Tools of Tensor Calculus*, A. Balfagón, P. Castellví and X. Jaén.  
(<http://baldufa.upc.es/xjaen/ttc/>).
- [12] V. A. Ilyin and A. P. Kryukov, *Comp. Phys. Commun.* **96**, 36 (1996).
- [13] *Cadabra*, A field-theory motivated approach to symbolic computer algebra.  
Kasper Peeters 2006. (<http://www.aei.mpg.de/~peekas/cadabra/>).
- [14] L. R. U. Manssur, R. Portugal, B. F. Svaiter, *Group-theoretic Approach for Symbolic Tensor Manipulation*, *Internat. J. Modern Phys. C* **13**, 859 (2002).
- [15] The GAP Group, GAP – Groups, Algorithms, and Programming, Version 4.4;  
2006. (<http://www.gap-system.org>)
- [16] S. B. Edgar and A. Hoglund, *J. Math. Phys.* **43**, 659 (2002).
- [17] *xTensor*, A fast manipulator of tensor expressions, J. M. Martín-García 2002–  
2007, (<http://metric.iem.csic.es/Martin-Garcia/xAct/>)
- [18] L. R. U. Manssur and R. Portugal, *Comp. Phys. Commun.* **157**, 173 (2004).  
(<http://www.lncc.br/~portugal/Canon.html>)