Computing of Curvature Invariants in Arbitrary Dimension

Stanisław Ewert-Krzemieniewski
(Technical University of Szczecin)

December 27, 1999

1 Introduction

The MathTensor software enables to calculate the curvature invariants of differentiable manifolds in the case when the dimension of manifold under consideration is a concrete number. Proving the existence of geometric objects it is important to have examples for all dimensions n, greater or equal to some n_0, i.e. when the dimension is just a symbol.

The aim of the paper is to show how, using MathTensor and Mathematica commands, one can define the covariant and contravariant components of the metric tensor. Then we will show how to define different curvature tensors and how to execute the calculation of their components. The presented method is demonstrated by the use of the local form of the metric tensor

$$ds^2 = Q_{11}(dx^1)^2 + Q_{22}(dx^2)^2 + \sum_{a,b=3}^{n-2} k_{ab}dx^a dx^b + 2(dx^1 dx^n + dx^2 dx^{n-1}) \quad (1)$$

with some assumptions on its components.

The author used Mathematica 2.2.1 MS-DOS and MathTensor 2.2.[1][2]

[1]1991 MSC: 53-04, 53B20, 53B30.

[2]Key words: curvature invariants.

2 Preliminaries

In this section we briefly present what we are going to calculate. As usually, we adopt the Einstein convention, i.e. the repeated lower and upper indices are summed over the indicated range. For example $A_{ar}B_c^{rb} = \sum_{r=1}^n A_{ar}B_c^{rb}$.

Let (M, g) be semi-Riemannian manifold covered by a system of coordinate neighbourhoods (x, U), $U \subset M$, x being smooth mapping from U into R^n and g be a metric tensor, i.e. a smooth, non-degenerated, symmetric tensor field of type $(0, 2)$ defined on M. If $p \in M$ and $\frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j}$ are coordinate vector fields around the point p, then the local components of g at p are given by $g_{ij} = g_{ij}(p) = g(\frac{\partial}{\partial x^i}_{|p}, \frac{\partial}{\partial x^j}_{|p})$. If $[g_{ij}]$ is a matrix with components g_{ij}, then its reciprocal is denoted by $[g^{ij}]$. Let $z = z(t) = (z^1(t), ..., z^n(t))$, $t \in \langle a, b\rangle$, be a smooth curve in M. Then the length s of z is given by $s = \int_a^b \sqrt{g_{ij}(z(t))\frac{dz^i}{dt}\frac{dz^i}{dt}}dt$.

A linear connection on a manifold M is a bilinear mapping ∇ which satisfies the following conditions: for arbitrary vector fields X, Y on M and a smooth function f on M, we have

$$\nabla_{fX}Y = f\nabla_X Y, \qquad \nabla_X(fY) = X(f)Y + f\nabla_X Y.$$

On any semi-Riemannian manifold (M, g) there exist a unique symmetric connection, known as the Levi-Civita connection, such that $X(g(Y, Z)) - g(\nabla_X, Y, Z) - g(X, \nabla_X Z) = 0$ for all vector fields on M. Then we have

$$\nabla_{\frac{\partial}{\partial x^i}}\frac{\partial}{\partial x^j} = \Gamma^r_{ij}\frac{\partial}{\partial x^r},$$

$$\nabla_{\frac{\partial}{\partial x^l}}T_{ab} = T_{ab,l} - \Gamma^r_{al}T_{rb} - \Gamma^r_{bl}T_{ar}$$

where

$$\Gamma^k_{ij} = \Gamma^k_{ji} = \frac{1}{2}g^{ks}(g_{sj,i} + g_{is,j} - g_{ij,s})$$

are called the Christoffel symbols, T being a $(0, 2)$ tensor field with components T_{ab} and comma denotes ordinary differentiation with respect to $k - th$ variable. Then the components of the Riemann-Christoffel curvature tensor R, the Ricci tensor S and the scalar curvature TrS are as follows:

$$R_{hijk} = \frac{1}{2}(g_{ij,hk} - g_{ik,hj} + g_{hk,ij} - g_{hj,ik}) + g_{rs}\Gamma^r_{ij}\Gamma^s_{hk} - g_{rs}\Gamma^r_{ik}\Gamma^s_{hj},$$

$$S_{ij} = S_{ji} = g^{rs} R_{rijs},$$
$$TrS = g^{rs} S_{rs}.$$

Moreover, we have the Weyl conformal curvature tensor C with components

$$C_{hijk} = R_{hijk} - \frac{1}{n-2}(g_{ij}S_{hk} - g_{ik}S_{hj} + g_{hk}S_{ij} - g_{hj}S_{ik}) + \frac{TrS}{(n-1)(n-2)}(g_{ij}g_{hk} - g_{ik}g_{hj})$$

which is invariant of conformal transformations.

Finally, we mention that the Riemann-Christoffel curvature tensor has the following symmetry properties:

$$R_{hijk} = R_{jkhi} = -R_{ihjk}, \qquad R_{hijk} + R_{hjki} + R_{hkij} = 0$$

and that the symmetry properties of the tensor C are analogous.

3 Defining the metric tensor

Let h be the metric given by (1). Then the (symmetric) matrix of its covariant components and its reciprocal are

$$\left[\ h_{ij}\ \right] = \begin{bmatrix} Q_{11} & 0 & 0 & . & . & . & 0 & 0 & 1 \\ 0 & Q_{22} & 0 & . & . & . & 0 & 1 & 0 \\ 0 & 0 & k_{33} & . & . & . & k_{3,n-2} & 0 & 0 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ 0 & 0 & k_{n-2,3} & . & . & . & k_{n-2,n-2} & 0 & 0 \\ 0 & 1 & 0 & . & . & . & 0 & 0 & 0 \\ 1 & 0 & 0 & . & . & . & 0 & 0 & 0 \end{bmatrix}$$

,

$$\left[\ h^{ij}\ \right] = \begin{bmatrix} 0 & 0 & 0 & . & . & . & 0 & 0 & 1 \\ 0 & 0 & 0 & . & . & . & 0 & 1 & 0 \\ 0 & 0 & k^{33} & . & . & . & k^{3,n-2} & 0 & 0 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ 0 & 0 & k^{n-2,3} & . & . & . & k^{n-2,n-2} & 0 & 0 \\ 0 & 1 & 0 & . & . & . & 0 & -Q_{22} & 0 \\ 1 & 0 & 0 & . & . & . & 0 & 0 & -Q_{11} \end{bmatrix}$$

respectively. It is easily seen that each of the matrices above consists of 9 blocks. We will use three types of indices: integers running over the rang $1, 2$; RegularIndices running over the range $3, ..., n-2$ and two aIndices, namely alb, ala taking the values $n-1$, n respectively.

In what follows we begin each subsession with two commands

- In[1]:= ¡¡mathtensor.m;
- In[2]:= AddIndexTypes
- In[3]:= DefineTensor[{h, Q, k}, {{2, 1}, 1}];
- Definition[h, Q, k]¿¿mytensor.m

We define simultaneously covariant and contravariant components of h by blocks:

- In[4]:= (
    - h[a_?NegIntegerQ, b_?NegIntegerQ] := Q[a, b] /; a === b;
    - h[a_?NegIntegerQ, b_?NegIntegerQ] := 0 /; a =!= b;
    - h[a_?PosIntegerQ, b_?PosIntegerQ] := 0;
    - h[a_?IntegerQ, b_?IndexRegQ] := 0;
    - h[a_?NegIntegerQ, b_?LowerIndexaQ] := 1 /; Position[Downuserlista, b] === {{-a}};
    - h[a_?NegIntegerQ, b_?LowerIndexaQ] := 0 /; Position[Downuserlista, b] =!= {{-a}};
    - h[a_?PosIntegerQ, b_?UpperIndexaQ] := 1 /; Position[Upserlista, b] === {{a}};
    - h[a_?PosIntegerQ, b_?UpperIndexaQ] := 0 /; Position[Upuserlista,b] =!= {{a}};
    - h[a_?IndexRegQ, b_?IndexRegQ] := k[a, b];
    - h[a_?IndexaQ, b_?IndexRegQ] := 0;
    - h[a_?LowerIndexaQ, b_?LowerIndexaQ] := 0;

- ∗ h[a_?UpperIndexaQ, b_?UpperIndexaQ] := -Q[-Position[Upperuserlista, a] [[1, 1]], -Position[Upperuserlista, a][[1, 1]]] /; a === b
- ∗ h[a_?UpperIndexaQ, b_?UpperIndexaQ] := 0 /; a =!= b;
)

. For theoretical proposes it is convenient to put $h_{11} = Q_{11}$, $h_{22} = Q_{22}$. However, this may leads to misunderstandings since in further computations expressions of the form $Q_{11,ij}+ k_{ij}$ may appear which MathTensor sees as a sum of $(0, 4)$ and $(0, 2)$ tensors.

It is also possible to use bIndices insted of integers

- ∗ h[a_?LowerIndexbQ, b_?LowerIndexbQ] := Q[a, b] /; a === b
- ∗ h[a_?LowerIndexbQ, b_?LowerIndexbQ] := 0 /; a =!= b
- ∗ h[a_?UpperIndexbQ, b_?UpperIndexbQ] := 0
- ∗ h[a_?IndexbQ, b_?IndexRegQ] := 0
- ∗ h[a_?LowerIndexaQ, b_?LowerIndexbQ] := 1 /; Position[Downuserlista, a] === Position[Downuserlistb, b]
- ∗ h[a_?LowerIndexaQ, b_?LowerIndexbQ] := 0 /; Position[Downuserlista, a] =!= Position[Downuserlistb, b]
- ∗ h[a_?UpperIndexaQ, b_?UpperIndexbQ] := 1 /; Position[Upuserlista, a] === Position[Upserlistb, b]
- ∗ h[a_?UpperIndexaQ, b_?UpperIndexbQ] := 0 /; Position[Upuserlista, a] =!= Position[Upserlistb, b]
- ∗ h[a_?PosIntegerQ, b_?UpperIndexbQ] := 0

We will not continue this approach.

- In[5]:= Definition[Q, k, h]¿¿compmet.m; Exit

4 Connection

We star new subsession. First we define the connection A

- In[3]:= ¡¡mytensor.m;

- In[4]:= DefineTensor[A, {{1,3,2},1}]; Definition[A]¿¿¿mytensor.m

- In[5]:= MakeSumRange[h[ua, uf]* (OD[h[lf, lc], lb] + OD[h[lb, lf], lc] - OD[h[lb,l c], lf])/2,
  - {lf, -2, -1, ala, alb, lf}] // Expand;
- In[6]:= A[ua_, lb_, lc_] := A[ua, lb, lc] = Module[{uf, lf}, UpLo[{uf}, {lf}]; %]
- In[7]:= Definition[h, A]¿¿defaff.m; %5¿¿¿defaff.m

Now with help of the editor we can obtain the appropriate formula for components of the connection. Having done it, we create the file containing subsets of indices of all possible types.

- In[8]:= IU = {2,1,ala,alb,lh}; IL = {-2, -1, ala, alb, lh, li, lj};
- In[9]:= Flatten[Table[{ IU[[r]], IL[[s]], IL[[t]] }, {r,5}, {s,7}, {t, s, 7}], 2]¿¿indaff.m
- In[10]:= Exit

We are ready to calculate the components of A. Start the new subsession.

- In[3]:= ¡¡compmet.m
- In[4]:= ¡¡defaff.m
- In[5]:= ¡¡indaff.m;
- In[6]:= OD[k[a_, b_], c__] := 0
- In[7]:= Apply[A[##]&, %5, {1}]¿¿compaff1.m; Exit

Again, using the editor and the results from compaff1.m we create the final file compaff.m containing formulas for components of the connection A.

- A[1, -2, -2] = OD[Q[-2, -2], ala]/2
- A[1, -1, -1] = -OD[Q[-1, -1], ala]/2
- A[2, -2, -2] = -OD[Q[-2, -2], alb]/2
- A[2, -1, -1] = -OD[Q[-1, -1], alb]/2
- A[aua, -2, -2] = -OD[Q[-2, -2], -1]/2 + (OD[Q[-2, -2], ala]*Q[-1, -1])/2

- A[aua, -2, -1] = -OD[Q[-1, -1], -2]/2
- A[aua, -1, -1] = -OD[Q[-1, -1], -1]/2 + (OD[Q[-1, -1], ala]*Q[-1, -1])/2
- A[aua, -1, f_?LowerIndexAllTypesQ] := A[aua, -1, f] = OD[Q[-1, -1], f]/2
- A[aub, -2, -2] = -OD[Q[-2, -2], -2]/2 + (OD[Q[-2, -2], alb]*Q[-2, -2])/2
- A[aub, -2, -1] = OD[Q[-2, -2], -1]/2
- A[aub, -2, f_?LowerIndexAllTypesQ] := A[aub, -2, f] = OD[Q[-2, -2], f]/2
- A[aub, -1, -1] = -OD[Q[-1, -1], -2]/2 + (OD[Q[-1, -1], alb]*Q[-2, -2])/2
- A[uk_?UpperIndexQ, -1, -1] := A[uk, -1, -1] = Module[{uf,lf}, UpLo[{uf},{lf}]; -(k[uk, uf]*OD[Q[-2, -2], lf])/2]
- A[uk_?UpperIndexQ, -2, -2] := A[uk, -2, -2] = Module[{uf,lf}, UpLo[{uf},{lf}]; -(k[uk, uf]*OD[Q[-1, -1], lf])/2]
- A[a_, b_, c_] := 0

5 Curvature Tensor

We follow the previous section. First we define the curvature tensor R.

- In[3]:= Symmetries[RiemannR[la, lb, lc, ld]];
- In[4]:= DefineTensor[R, %]; Definition[R]¿¿¿mytensor.m

. It may be necessary to add in file mytensor.m the following lines
R[axx_, bxx_, cxx_, dxx_] :=
1*Apply[R, {cxx, dxx, axx, bxx}] /;
SomeQ[axx, cxx] &&
IndicesAndNotOrderedQ[{axx, bxx, cxx, dxx} [[2, 4]]]

- In[5]:= ¡¡mytensor.m
- In[6]:= MakeSumRange[

```
 (OD[ h[li, lj], lh, lk] - OD[ h[li, lk], lh, lj] +
OD[ h[lh, lk], li, lj] - OD[ h[ lh, lj], li, lk] )/2 +
h[le, lf] A[ue, lh, lk] A[uf, li, lj] - h[le, lf] A[ue, lh, lj] A[uf, li, lk],
{le, -2, -1, ala, alb, le}, {lf, -2, -1, ala, alb, lf} ] // Expand;
```

The first expression inside [...] defines the components of the curvature tensor.

- `In[7]:= R[lh_, li_, lj_, lk_] := R[lh, li, lj, lk] = Module[ {ue, uf, le, lf}, UpLo[ {ue, uf}, {le,lf} ];        %]`
- `In[8]:= Definition[h, A, R]¿¿defriem.m; %6¿¿¿defriem.m`

We must go to the editor to place %6 in the right position.

- `In[9]:= {-2, -1, ala, alb, lh, li, lj, lk};`
- `In[10]:= Flatten[ Table[ {%[[r]], %[[s]]}, {r, 8}, {s, r, 8}], 1];`
- `In[11]:= Flatten[ Table[ Flatten[ {%[[r]], %[[s]]} ],  {r, Length[%]}, {s, r, Length[%]} ], 1]¿¿indriem.m`
- `In[12:= Exit`

We start the next session to compute components of R for sets of indices contained in indriem.m.

- `In[3]:= ¡¡compmet.m`
- `In[4]:= ¡¡compaff.m`
- `In[4]:= ¡¡defriem.m`
- `In[5]:= ¡¡indriem.m;`
- `In[6]:= OD[ k[a_, b_ ], c__ ] := 0`
- `In[7]:= Apply[ R[##]&, %5, {1}]¿¿comprie1.m; Exit`

Printing comprie1.m one can define the components of R as follows

```
∗ R[-2, -1, -2, -1] =
  Module[ {ub, uc, lb, lc],    UpLo[{ub, uc},{lb, lc}];
-(OD[ Q[-2, -2], alb]*OD[ Q[-1, -1], -2])/4 +
(OD[ Q[-2, -2], ala]*OD[ Q[-1, -1], -1])/4 -
(OD[ Q[-2, -2], -1]*OD[ Q[-1, -1], ala])/4 +
(OD[ Q[-2, -2], -2]*OD[ Q[-1, -1], alb])/4 -
(k[ub, uc]*OD[ Q[-2, -2], lb]*OD[ Q[-1, -1], lc])/4 -
OD[ Q[-2, -2], -1, -1]/2 - OD[ Q[-1, -1], -2, -2]/2 +
(OD[ Q[-2, -2], alb]*OD[ Q[-1, -1], alb]*Q[-2, -2])/4 +
(OD[ Q[-2, -2], ala]*OD[ Q[-1, -1], ala]*Q[-1, -1])/4 ]
```

* R[-2, b_?LowerIndexAllTypesQ, -2, d_?LowerIndexAllTypesQ] :=
  R[-2, b, -2, d] = -(OD[Q[-2, -2], b, d])/2
* R[-1, b_?LowerIndexAllTypesQ, -1, d_?LowerIndexAllTypesQ] :=
  R[-1, b, -1, d] = -(OD[Q[-1, -1], b, d])/2
* R[-2, -1, -2, d_?LowerIndexAllTypesQ] :=
  R[-2, -1, -2, d] =
  (OD[Q[-2, -2], ala]*OD[Q[-1, -1], d])/4 - OD[Q[-2, -2], -1, d]/2
* R[-2, -1, -1, d_?LowerIndexAllTypesQ] :=
  R[-2, -1, -1, d] =
  -((OD[Q[-2, -2], d]*OD[Q[-1, -1], alb])/4 + OD[Q[-1, -1], -2, d]/2
* R[a_, b_, c_,d_] := 0

and save it in the file compriem.m.

Thus, step by step, we can compute the components of the Ricci tensor, the scalar curvature, the Weyl conformal curvature tensor and their covariant derivatives.

References

[1] S. Gallot, D. Hulin, J. Lafontaine, Riemannian Geometry, Second Edition, Springer-Verlag,1990.

[2] B. O'Neill, Semi-Riemannian Geometry with Applications to Relativity, Academic Press, New York-London, 1983.

Stanisław Ewert-Krzemieniewski
Institute of Mathematics

Technical University of Szczecin
Al. Piastów 17
70-310 Szczecin, Poland
e-mail: ewert@arcadia.tuniv.szczecin.pl