CAPABILITIES OF THE *MATHEMATICA* PACKAGE 'RIEMANNIAN GEOMETRY AND TENSOR CALCULUS'

S. BONANOS
Institute of Nuclear Physics, NCSR Demokritos,
15310 Aghia Paraskevi, Greece
E-mail: sbonano@inp.demokritos.gr

A brief description of the basic features of the *Mathematica* package 'Riemannian Geometry and Tensor Calculus' (RGTC) is given. Two examples, illustrating the use of the functions defined in the package in non-trivial calculations, are presented.

1. Introduction

RGTC developed out of the author's attempts to use *Mathematica* for explicit tensor computations in General Relativity. One would think that there would be several packages available for this purpose. Surprisingly, this is not the case: apart from GRTensorM[a], other packages are either intended for symbolic manipulation of tensor expressions (Ricci, EinS, TTC), or do not run under *Mathematica* (GRTensorII, Sheep, Redten), or are not free (MathTensor, Cartan). MathTensor, in particular, is essentially a new progamming language — it has over 250 new commands with rather unintuitive notation.

The basic idea of RGTC is to provide definitions for the most common Riemannian Geometry tensors and tensor operations, so that the user can carry out calculations on tensors using these functions along with any other applicable *Mathematica* function. The package is, therefore, most useful to those who already have some experience with symbolic manipulations in *Mathematica*. In the following we will assume that this is the case.

[a]This is a less developed version, written for *Mathematica* 2.0, of the package GRTensorII that runs under Maple.

2. Representation and Naming of Tensors

The requirement that *Mathematica* functions like **Collect**, **Factor**, **Coefficient**, **Replace**, **Solve**, etc., should be able to act, without modification, on the tensors computed by RGTC dictated the form of the internal representation used for tensors — nested lists. Thus an nth-rank tensor is stored as an n-fold nested list of its components[b]. This representation implies that different forms of the same tensor (e.g., R_{abcd}, $R^a{}_{bcd}$, etc.) must be stored as different nested lists. This is not unexpected if one remembers that the emphasis here is on computing explicit expressions of tensor components in a particular frame, rather than symbolically manipulating the indices of tensor expressions.

One drawback of the nested list representation is that it is redundant: tensors with symmetries have the same components stored in more than one place in the list, while operations on such tensors perform identical calculations more than once. This drawback, resulting in slower performance and higher memory requirements, is not very important with modern desktop computers[c]. And, we believe, it is overshadowed by the advantage of allowing the user to manipulate these lists directly.

To be able to access nested lists representing tensors one must give them names. Although any name will do the job, it helps to have a naming convention for tensors that meets the following criteria

- it contains information on the number and kind (co- or contavariant) of indices
- it resembles the traditional notation for tensors
- it is reasonably short and easy to type

A naming convention that meets these criteria is to use, as part of the name, the letters 'U' (=Up) and 'd' (=down) to denote each free tensor index. Thus, for example, the nested lists representing g_{ab}, g^{ab}, R_{abcd}, $R^a{}_{bcd}$, R_{ab} are stored under the names `gdd, gUU, Rdddd, RUddd, Rdd`, respectively. Contractions can be denoted by replacing the contracted indices by the same letter (Upper and lower case) while the symbol '$' can be used to separate different tensors. For example, the Ricci tensor could also be named `RAdad` or, more awkwardly, `gAC$Radcd`, these names indicating, in addition, which indices of the Riemann tensor have been contracted.

[b]This representation for tensors is also suggested in the *Mathematica* Book, Sec. 3.7.11.
[c]Our definition of 'modern' is quite modest: any computer running at 500 Mhz and allowing 30 MB of RAM to the kernel will perform adequately for most problems.

Finally, covariant differentiations can be denoted by preceding the differentiation index with the letter 'j' (resembling ';'). For example, `RAdddja` and `Rddjd` are appropriate names for the 3rd rank covariant tensors $R^a{}_{bcd;a}$ and $R_{ab;c}$, respectively. Note that this convention requires that the letters 'U' and 'd' be used exclusively for free indices.

The tensors calculated by RGTC are stored under names conforming to this convention. Of course, the user can devise his own convention for naming nested lists representing tensors. One must remember, however, that these are just names for accessing nested lists; the functions which operate on these lists cannot distinguish co- from contra-variant indices from their names!

Specific tensor components can be accessed by taking the appropriate 'part' of the relevant list. For example, `Rdddd[[2,3,2,4]]` is the component R_{2324} while `RUddd[[2,3,2,4]]` gives $R^2{}_{324}$. One can also take subparts: `RUddd[[2,4]]` is the rank-2 tensor with components $R^2{}_{4ab}$.

3. New functions defined by RGTC

RGTC introduces a minimum of new functions. To begin calculations one must first specify a geometry. This is done by[d]:

(1) defining a list of symbols for the coordinates, e.g.,

```
xIN = {t, r, th, phi},
```

(2) defining a nested list of functions of the coordinates — the metric tensor, e.g.,

```
gIN = {{1-2M/r,0,0,0},{0,-r/(r-2M),0,0},
{0,0,-r^2,0},{0,0,0,-r^2 Sin[th]^2}},
```

(3) (optional) assigning a list of simplification rules to the global variable **simpRules**, e.g.,

```
simpRules = {y_. Sin[x_]^2 + y_. Cos[x_]^2 -> y}
```

One then evaluates the main routine of RGTC, **RGtensors**, with `gIN` and `xIN` as arguments:

```
RGtensors[gIN, xIN]
```

[d]Although we will give examples in 4 dimensions, the functions in RGTC work in any dimension.

When **RGtensors** is finished, the following tensors (= nested lists of components in the default coordinate frame) have been calculated and stored under the names in parentheses:

(i) metric (`gdd, gUU`)
(ii) Christoffel symbols of the 2nd kind (`GUdd`)
(iii) Riemann (`Rdddd, RUddd`)
(iv) Ricci, Ricci Scalar, Einstein (`Rdd, R, EUd`)
(v) Weyl (`Wdddd`)

RGtensors prints messages after each tensor is computed. It also tests if the specified metric is *Flat*, *Ricci Flat*, *Conformally Flat*, *Einstein Space* or *Space of Constant Curvature* and, if yes, prints appropriate messages.

The simplification rules defined in **simpRules** are used repeatedly by **RGtensors** to simplify expressions, and the components of all tensors are stored in factored form. Although this slows down **RGtensors** considerably, it ensures that all cancellations have been performed.

If one wishes to do calculations in a frame other than the coordinate frame, one must specify this (co-)frame as a list of 1-forms[e] in the coordinate differentials, say,

```
eIN = {Sqrt[1-2M/r]d[t], d[r]/Sqrt[1-2M/r],
     r d[th], r Sin[th]d[phi]},
```

and evaluate **RGtensors** with `eIN` as a third argument:

```
RGtensors[gINon, xIN, eIN]
```

Of course now the metric, `gINon`, must be given with respect to the frame `eIN`, `gINon = {{1,0,0,0},{0,-1,0,0},{0,0,-1,0},{0,0,0,-1}}`, and the components of all tensors (`Rdddd, EUd`, etc.) are calculated with respect to this frame. Also `GUdd` now stores the appropriate connection coefficients for the specified metric and frame.

RGTC can also be used to do approximate calculations (perturbations of a given metric, asymptotic expansions). For approximate calculations all one has to do is give the metric and/or the frame as a series expansion, to the required order, using the *Mathematica* function **Series**[f].

[e]For calculations with differential forms RGTC uses the author's package EDC, which is available at www.inp.demokritos.gr/~ sbonano/EDC/.
[f]After taking precautions to ensure that **Series** will not expand the coordinate differential $d[r]$, if r is the expansion parameter!

Finally, when `gIN` equals {{0,1,0,0}, {1,0,0,0}, {0,0,0,-1}, {0,0,-1,0}}, **RGtensors** assumes that the frame given in `eIN` is the Newman-Penrose null frame $\{l, n, m, \bar{m}\}$, and defines the functions `PH[i,j], PS[i],` Λ giving the NP quantities $\Phi_{ij}, \Psi_i, \Lambda$ in this frame.

After the basic Riemannian Geometry tensors have been defined by **RGtensors**, the following functions can be used to do further calculations:

- ***Raise, Lower***. As their names imply, these functions compute the components of tensors in different index configurations. They take as arguments a tensor and the position(s) of the indices to be moved (Up or down). For example, evaluating the commands `WUUdd = Raise[Wdddd,1,2];` and `Edd = Lower[EUd,1];` computes the tensors with components $W^{ab}{}_{cd}$ and E_{ab}, respectively.
- The built-in *Mathematica* functions **Outer** and **Transpose** can be used to define new tensors. For example, the commands `ggdddd = Outer[Times, gdd, gdd];` and `gxgdddd = ggdddd - Transpose[ggdddd, {1,3,2,4}];` define 4th rank covariant tensors whose [[a, b, c, d]] components are $g_{ab}g_{cd}$ and $g_{ab}g_{cd} - g_{ac}g_{bd}$, respectively.
- ***Contract, multiDot***. These functions are used for contractions. **Contract** takes as arguments a tensor and one or more lists of two numbers indicating the positions of the indices to be contracted, while **multiDot** takes *two* tensors and a similar list of indices. For example, `Contract[RUddd, {1,3}];` will contract the first index with the third giving the Ricci tensor, while `W2UUdd = multiDot[WUUdd, WUUdd, {3,1}, {4,2}];` contracts the 3rd and 4th indices of the first tensor with the 1st and 2nd indices of the second tensor, giving the square of the Weyl tensor $W^{ab}{}_{mn}W^{mn}{}_{cd}$.

  In both **Contract** and **multiDot**, one must be careful that each pair of numbers {i,j} refers to one 'U' and one 'd' index (in either order). Otherwise, meaningless results will be obtained.
- ***covD, covDiv, Bianchi***. **covD** takes as arguments an nth-rank tensor and a list of numbers (the positions of its contravariant indices) and returns a tensor of rank $n+1$ — the covariant derivative of the tensor. The last index is the differentiation index. For example, `Rddjd = covD[Rdd]` has components $R_{ab;c}$, while `WUUddjd = covD[WUUdd,{1,2}];` is the tensor with components $W^{ab}{}_{cd;e}$.
  covDiv calculates the covariant divergence of a tensor with respect to a contravariant index. If there are more than one contravariant

indices, the index to be contracted is placed in an extra set of curly brackets {}. Examples: `covDiv[RUddd, {1}];` is the tensor with components $R^a{}_{bcd;a}$, while `covDiv[WUUdd,{1,{2}}];` is the 3rd-rank tensor with components $W^{ab}{}_{cd;b}$.
Using **covD** and **covDiv** the function **Bianchi[n]** calculates the full ($n = 0$), contracted once ($n = 1$) and contracted twice ($n = 2$) Bianchi Identities (tensors of rank 5, 3, 1 respectively) and returns a list of any non-vanishing terms. This is normally the empty list.

- ***eta[], HStar***. **eta[i__]** is a function that generates the totally antisymmetric tensor η with contravariant indices at the positions specified by i. Thus, in four dimensions, `eta[]` is the totally covariant tensor η_{abcd}, `eta[1]` is the tensor $\eta^a{}_{bcd}$, and `eta[3,4]` the tensor $\eta_{ab}{}^{cd}$. **HStar[x]** is a function that computes the Hodge-dual of x. It requires that a coframe basis has been given in **RGtensors**.

New tensors can be defined either by operations on other tensors, as in the above examples, or by explicitly giving a list of their components as functions of the coordinates.

In addition, the following auxiliary functions are used:

- ***FacSimp***. **FacSimp[x]** applies the simplification rules defined in the list **simpRules** and factors every component of the tensor x. The tensors computed by **RGtensors**, as well as those returned by the functions **Raise**, **Lower**, **Contract**, **multiDot**, **covD**, **covDiv**, **Bianchi** have all been simplified using **FacSimp**.
- ***FuncRepRules*** ("Function Replacement Rules").
FuncRepRules[f[x__], g_, n_] generates a list of replacement rules for replacing $f[x]$ with g and the derivatives of $f[x]$ (up to order n) with the corresponding derivatives of g, where g is any expression. The last argument, n, is optional; if it is omitted, rules up to second order are generated. f can have any number of arguments. Thus, while `Rdd/.f[u,v]` $\rightarrow$ `P[u]*Q[v]` only replaces $f[u, v]$ but leaves the derivatives of $f[u, v]$ unchanged, the command `Rdd/.FuncRepRules[f[u,v], P[u]*Q[v]]` replaces both $f[u, v]$ and its derivatives up to 2nd order.
- ***zeroQ, nonZeroN, nonZeroL, indepTerms***. These functions are useful in examining tensors.
zeroQ[x] tests if x is the zero tensor and returns *True* or *False*.
nonZeroN[x] returns the *Number* of non-zero components of the tensor x.

nonZeroL[x] returns a *List* of the distinct non-zero components of the tensor x (if Z is any expression, Z, $-Z$, $2Z$ are considered distinct).
indepTerms[x] returns a list of the 'independent' components of the tensor x. Independent here means not equal to $\pm$ a numerical multiple of another component.

Finally, RGTC uses the following global variables:
simpRules — a list of simplification rules.
TrigRules — a predefined list of simplification rules for handling factors of $\sin^2\theta$ in the metric.
Dim — an integer = dimension of space.
coordList — a list of the coordinate names.
deList, **dxRuleList** — these lists are required for implicit frame calculations — see Sec. 5.

4. Sample calculations: Going beyond the Ricci tensor

The package RGTC contains numerous examples indicating the use of the functions defined above. Here we will give the commands necessary for two more complicated calculations.

4.1. *Bach and Bell-Robinson Tensors*

The first example computes the Bach ($= W_a{}^r{}_b{}^s{}_{;s;r} + \frac{1}{2}W_a{}^r{}_b{}^s R_{rs}$) and Bell-Robinson ($= \frac{1}{4}(W_a{}^r{}_b{}^s W_{crds} + {*W}_a{}^r{}_b{}^s * W_{crds})$, where ${*W}_{abcd} = \frac{1}{2}\eta_{ab}{}^{rs}W_{rscd}$) tensors for a metric conformal to Schwarzschild with M=M[t] and cosmological constant, and verifies some of their properties.

```
xIN = {t, r, th, phi}; g00 = 1 - 2 M[t]/r -Lambda r^2/3;
gIN = Omega[t,r]^2 DiagonalMatrix[{g00, -1/g00,
-r^2, -r^2 Sin[th]^2}];
simpRules = TrigRules;
RGtensors[gIN, xIN]
WdUdU = Raise[Wdddd,2,4];
WdUdSjs = covDiv[WdUdU,{2,{4}}];
BachTdd = FacSimp[covDiv[WdUdSjs,{2}] +
multiDot[WdUdU, Rdd, {2,1},{4,2}]/2];
(*Bach Tensor vanishes when conformal to Einstein space*)
zeroQ[FacSimp[BachTdd/.FuncRepRules[M[t],M0,4]]]
```

```
starWdddd = multiDot[eta[3,4], Wdddd/2, {3,1},{4,2}];
starWdUdU = Raise[starWdddd,2,4];
BRTdddd = FacSimp[multiDot[WdUdU, Wdddd, {2,2},{4,4}]/4 +
multiDot[starWdUdU, starWdddd, {2,2},{4,4}]/4];
(*check symmetric*)
zeroQ[BRTdddd-Transpose[BRTdddd,{1,3,2,4}]]
BRTUUdd = Raise[BRTdddd,1,2];
(*check traceless*)
zeroQ[Contract[BRTUUdd,{1,3}]]
(*Bell-Robinson Tensor divergenceless if Einstein space*)
BRTUSddjs = covDiv[BRTUUdd,{1,{2}}];
zeroQ[BRTUSddjs/.Join[FuncRepRules[Omega[t,r],1],
FuncRepRules[M[t],M0,4]]]
```

4.2. *Asymptotic behavior of the Bondi metric near null infinity.*

First define the coordinates, metric and null frame:

```
xIN = {u, r, th, phi};
gNPnull = {{0,1,0,0},{1,0,0,0},{0,0,0,-1},{0,0,-1,0}};
m3 = r Exp[gamma[u,r,th]](d[th]-U[u,r,th]d[u]);
m4 = r Exp[-gamma[u,r,th]]Sin[th]d[phi];
nullFrame ={d[u], Exp[2*beta[u,r,th]](d[r]+V[u,r,th]/2 d[u]),
(m3 + I m4)/Sqrt[2], (m3 - I m4)/Sqrt[2]};
simpRules = TrigRules;
```

One can now, EITHER compute first the exact expressions for all tensors, and then substitute in `Rdd` series expansions for the unknown functions using **FuncRepRules**:

```
RGtensors[gNPnull, xIN, nullFrame]
gammaSer = FuncRepRules[gamma[u,r,th], Series[c1[u,th]/r +
c2[u,th]/r^2 + c3[u,th]/r^3, {r,Infinity,3}]];
(*and similarly for the other functions --- beta, U, V*)
serRdd = FacSimp[Rdd/.Join[gammaSer,betaSer,...]];
```

OR, one can expand the unknown functions in the null frame *before* calling **RGtensors**, in which case all tensors will be computed as series expansions[g]

[g]Assuming betaSer, etc., have been defined; see also footnote f.

```
serFrame = FacSimp[nullFrame/.Join[gammaSer,betaSer,...]];
RGtensors[gNPnull, xIN, serFrame]
```

Both calculations result in the Ricci tensor being expressed in the form of a series in inverse powers of r. One can then proceed to determine the solution by equating to zero different terms in the expansion.

5. Implicit Frame Calculations

For some problems, giving the explicit expressions for the coframe is not the most efficient way of doing the calculations. Suppose one wants to do calculations in a Bianchi-Type IX cosmological model. Referred to an appropriate frame, the metric is a function of the time coordinate only. If the frame is written explicitly in terms of the differentials of the coordinates, however, it has a rather complicated form which will cause **RGtensors** to proceed very slowly. Yet, the exterior derivatives of the frame vectors are simple expressions not involving the coordinates. In such cases one can leave the frame undefined — in the form {`e[1], e[2], e[3], e[4]`} — and define two global variables: **deList** to contain the exterior derivatives of the frame, say {`e[2]` ∧ `e[3]`, `e[3]` ∧ `e[1]`, `e[1]` ∧ `e[2]`, `0`} and **dxRuleList** to give rules for replacing the differentials of any coordinates appearing in the metric (or in **deList**) in terms of the `e[i]` basis, `dxRuleList=`{`d[t]`→`e[4]`}. In such cases **RGtensors** is called with an implicit frame {`e[_]`} as third argument,

```
xIN = {x1, x2, x3, t};
gIN = DiagonalMatrix[{-A[t], -B[t], -C[t], T[t]}];
deList = {e[2] ∧ e[3], e[3] ∧ e[1], e[1] ∧ e[2], 0};
dxRuleList = {d[t]→e[4]};
RGtensors[gIN, xIN, {e[_]}]
```

and the calculations proceed much faster.

More detailed instructions are included in the package. To make best use of RGTC, the user must be familiar with the basics of *Mathematica*, especially operations with lists and the use of patterns and replacement rules. RGTC requires *Mathematica* v3.0 or higher. It is available for free at http//:www.inp.demokritos.gr/˜ sbonano/ RGTC/. *Mathematica* notebooks containing the full calculations outlined in Secs. 4.1, 4.2 can also be found there.