COMPUTER ALGEBRA IN RELATIVITY

1. INTRODUCTION

Below is an introduction to the use of some computer algebra packages in relativity. The most easily accessible is probably the Maple package GRTensor (more properly GRTensorII), which I discuss first; there is also a *Mathematica* version called GRTensorM (which is not quite as good). I also discuss the Maple package `tensor`, which comes with Maple, as well as the more specialized packages SHEEP and CLASSI, based on PSL (and hence closely related to REDUCE).

Some of these packages, such as `tensor` and SHEEP, do coordinate computations, whereas CLASSI does frame computations; GRTensor and GRTensorM can do either.

The instructions are aimed at users of the UNIX computers in the Math and Physics departments, but should work with minor changes on other platforms.

2. PACKAGES BASED ON MAPLE

Start Maple by typing `maple` for a shell interface, or `xmaple &` for the worksheet front end under XWindows. All Maple commands must end with either a semicolon or a colon; a colon suppresses output. All definitions must use `:=` , not just `=` . Exit Maple with the command `quit;` .

a) GRTensor

GRTensor should be currently available on both Math and Physics Dept computers.

GRTensor is freeware; if you have access to Maple elsewhere and would like to obtain a copy of GRTensor, let me know.

To enable Maple to find the GRTensor files, on math machines type (note the backquotes!)

```
libname:=libname,`/tawa/c1t3d0s4/tevian/GRtensor/lib8`:
```

and on physics machines type

```
libname:=libname,`/home/tevian/GRtensor/lib8`:
```

(You can add the appropriate command to the file `.mapleinit` in your home directory.) Then type `readlib(grii):` and `grtensor();` to start GRTensor.

GRTensor contains an interactive program for constructing your own metric: Simply type `makeg(`*metricname*`);` , where *metricname* is the name you wish to call the metric. You will be prompted to enter the metric in one of several forms, and guided through the necessary steps; don't forget the semi-colon after each input!

Some pre-defined standard metrics can be loaded by first telling GRTensor where to find them with the command `grOptionMetricPath:=`/home/tevian/GRtensor/metrics8`:` or `grOptionMetricPath:=`/tawa/c1t3d0s4/tevian/GRtensor/metrics8`:` on physics/math machines, respectively, then loading the file you want using `qload` . (You can put this command in a file called `grtensor.ini` in your home directory, but *not* in your `.mapleinit` file.) For example, the Schwarzschild black hole metric can be loaded with `qload(schw)` .

You can also create metric files of your own by using the canned files as an analogy, then loading your file with `qload` or `grload` . This is not as hard as it sounds, as the file only needs to define the number of coordinates `Ndim_` , list the coordinates `x1_` , etc, and enter the metric components `g11_` , etc; note the trailing underscore in each variable name. For the sphere, the file would be

```
Ndim_:=2:
x1_:=theta:x2_:=phi:
g11_:=r^2:g22_:=r^2*sin(theta)^2:
```

Unspecified metric components are assumed to be zero, so you do not need to explicitly set off-diagonal elements like `g12_` to zero.

Many standard tensors are predefined in terms of the metric you have entered or loaded, including the inverse metric `invmetric`, the Christoffel symbols `Chr2`, the *covariant* (all indices down!) Riemann tensor `Riemann`, the Ricci tensor `Ricci`, the Ricci scalar `Ricciscalar`, and the Einstein tensor `Einstein`; the metric itself is called `metric`.

You can calculate any of these tensors using `grcalc`, and see the result of the calculation using `grdisplay`. For instance, to find the Ricci scalar, type

```
grcalc(Ricciscalar);
grdisplay(Ricciscalar);
```

Further documentation can be obtained by typing `?grtensor`.

You may need to use the `gralter` and/or `grmap` commands to fully simplify your results.

b) Tensor

The package `tensor` is distributed with Maple; if you have Maple, you have `tensor`.

The information in this section is a summary of the online documentation which can be obtained by typing `?tensor` or `help(tensor);`.

Type the command `with(tensor):` to load the `tensor` package, then choose coordinates with a command like `coord:=[theta,phi]:`. Tensors are created using the somewhat messy `create` command, in which you must first explicitly specify which indices are "up" (+1) and "down" (−1), then give a matrix of components. For instance, the (covariant, that is, both indices "down") metric for the sphere can be entered as

```
g:=create([-1,-1],array(symmetric, [[r^2,0],[0,r^2*sin(theta)^2]]));
```

It is essential that the metric be defined as a *symmetric* array. The Ricci scalar `R` is now computed by first successively computing the inverse metric `ginv`, the partial derivatives of the metric `dg` and `ddg`, the Christoffel symbols `Gam`, the Riemann tensor `Rie`, and the Ricci tensor `Ric`, as follows; any names can be used for the intermediate steps.

```
ginv := invert(g,´detg´):
dg := d1metric(g,coord):  ddg := d2metric(dg,coord):
Gam := Christoffel1(dg):
Rie := Riemann(ginv,ddg,Gam):
Ric := Ricci(ginv,Rie):
R := Ricciscalar(ginv,Ric):
```

Finally, to display the result of this computation, use the command `get_compts(R);`.

WARNING: `tensor` defines the Ricci tensor and scalar to be −1 times our definitions.

3. PACKAGES BASED ON MATHEMATICA

Start *Mathematica* by typing `math` for a shell interface, or `mathematica &` for the notebook front end under XWindows. In a notebook, all commands must be entered by holding down the shift key will typing the return key. Note that all *Mathematica* commands are written with square brackets, not parenthesis. An optional semicolon at the end of a command suppresses output. Exit *Mathematica* with the command `Quit` .

4. GRTensorM

GRTensorM is a translation of GRTensor into *Mathematica*. It should be currently available on both Math and Physics Dept computers.

GRTensorM is freeware; if you have access to *Mathematica* elsewhere and would like to obtain a copy of GRTensorM, let me know.

To enable *Mathematica* to find the GRTensorM files, type

```
$Path=AppendTo[$Path,"/tawa/c1t3d0s4/tevian/GRtensor"]
```

on math machines and

```
$Path=AppendTo[$Path,"/home/tevian/GRtensor"]
```

on physics machines. (You can add the appropriate command to the file `init.m` in your home directory.) Then type `<<grii/grt.m` to start GRTensorM.

Most commands are the same as in GRTensor, so long as you remember to change parentheses to square brackets. However, I do not know how to enter the metric by hand, so you will need to use the `makeg` command. If you are using the notebook interface, this will pop up several new windows with questions for you to answer, each of which requires you to first click on the empty box, then enter your answer, then click on OK.

WARNING: Unlike the newer GRTensor, in GRTensorM you must save your metric before using it. To do this, *before* running `makeg` , type `SetOptions[grii,MetricPath->"~"]` , which tells GRTensorM to put such files in your home directory; you may enter any other directory, such as `"/tmp"` instead of your home directory `"~"` . If you then run the command `grmake[test]` , after answering the questions a file named `test.g` will be created in the appropriate directory. Don't forget to delete unwanted files when done!

In order to use system metric files, you must (re)set `MetricPath` by typing the command `SetOptions[grii,MetricPath->"~tevian/GRtensor/grii/metrics/"]`; you could now load the Schwarzschild metric with the command `qload[schw]` .

You may need to replace `~tevian` with `/tawa/c1t3d0s4/tevian` or `/home/tevian` on math or physics machines, respectively.

The `SetOptions` commands can *not* go in your `init.m` file!

To calculate a tensor, use `grcalc` and `grdisplay` as before; you may also need to use `grmap` to simplify your result. For instance, to calculate the Ricci scalar type `grcalc[Ricciscalar]` , and to see the result type `grdisplay[Ricciscalar]` ; you may need to try things like `grmap[Simplify,Ricciscalar]` (followed by another `grdisplay` command) in order to get the final answer.

Some additional help is available by typing `?grtensor` .

5. GRTensorJ

A recent addition to the GRTensor family is the JAVA-based GRTensorJ, which should run in any recent browser. Instructions are available online, but it's pretty straightforward. Go to `http://halo.grtensor.org/~jtensor/local_gr.html` .

6. SHEEP AND CLASSI

SHEEP and CLASSI are currently available on `shell.onid.oregonstate.edu`.

To start the LISP-based packages SHEEP and CLASSI, first define aliases (short names) for the needed commands by typing the command `source ~drayt/sheep/.shprc` . (You can add this command to your `.cshrc` file if you like.) Now type `sheep` to start SHEEP, or `classi` to start CLASSI. To end either program, type `(quit)` .

SHEEP and CLASSI were written specifically for relativity; one of their biggest advantages is that there is a very large library of source files containing various spacetime metrics.

a) SHEEP

SHEEP is designed to do tensor computations using components in a coordinate basis. It is very good at the sort of tensor manipulations needed to, say, compute the curvature tensor of a given metric, but very bad at doing algebra.

After starting SHEEP, it is useful to set some switches by typing `(pon ptevar)` , `(pon nozero)` , and `(on diagonal)` .

> The first two of these control how output is printed, for instance the second of these specifies that only nonzero tensor components should be printed. The last specifies that metrics are assumed to be diagonal. You can avoid typing these repeatedly by putting them in a file called `sheep.ini` in your home directory.

For example, try the following commands:

```
(dimension 2)
(vars h p)
(load cord)
(rpl gdd)      (Type first  r^2$  and then  r^2*sin(h)^2$  when prompted.)
(funs (r))
(wmake gamu)
(wmake ric)
(wmake rscl)
```

After setting the dimension (default is 4) and the names of the variables (optional; note the use of h for θ and p for ϕ), this loads the coordinate package `cord`. Tensor components are assigned with the "replace" command `rpl` , after which it is necessary to specify the functional dependence of r. (You could make r a function of θ by typing `(funs (r h))` .) Tensors are computed with the `make` command and printed to the screen with the `write` command, which can be combined as `wmake` ; all of these commands must be enclosed within parentheses and given an appropriate argument. A list of predefined tensors can be obtained with the command `(help tensors)` .

Here is a more complicated example, the Schwarzschild black hole, which illustrates the tricks which must be used to avoid polynomial division. (This particular example is therefore much easier in, say, GRTensor; try it!)

```
(vars t r h p)
(load cord)
(rpl a)        (Then type  1-2m/r$ .)
(rpl gdd)      (Then type  -a$ ,  1/a$ ,  r^2$  and  r^2*sin(h)^2$ .)
(funs (m) a)
(newsul ricsul)      (Then type  a$ ,  :a$ .)
(usesul ricsul ricc riemc)
(wmake rie)
(wmake ein)
```

These commands define the metric in terms of an explicitly given function $a = 1 - 2m/r$, but do the computation in terms of a and its symbolic derivatives, only substituting for a in the final expressions (This allows division by a to replace the much harder division by $1 - 2m/r$.) The Schwarzschild metric can also be loaded with the command

`(load "schwar.crd")`

b) CLASSI

CLASSI is a macro package written in SHEEP designed to do tensor computations using components in a (generalized) "orthonormal" frame, i.e. with respect to a basis of vector fields whose dot products are constant, typically 1's and 0's. (This framework includes bases in which one or more vectors are null.) The Riemann tensor has many fewer independent components in an orthonormal frame than in a coordinate basis! (One way to see this is to compare the quite small group of orthogonal transformations with the quite large group of arbitrary coordinate transformations.)

Since CLASSI is essentially a souped up version of SHEEP, it is again useful to set the above switches by typing `(pon ptevar)` , `(pon nozero)` , and `(on diagonal)` . These commands can be inserted instead into a file `classi.ini`.

For example, try the following commands:

```
(dimension 2)
(vars h p)
(rpl izud)      (Then type  r$ ,  0$ ,  0$  and  r*sin(h)$ .)
(cartesian iframe)
(funs (r))
(wmake ric)
(wmake rscl)
```

The tensor `izud` (don't ask...) contains the components of the (dual) orthonormal frame with respect to the coordinate dual basis $\{d\theta, d\phi\}$; there is no diagonal switch. The next command indicates that this is a Euclidean orthonormal frame; a Lorentzian orthonormal frame would be indicated by `(lorentz iframe)` . Finally, note that the components of, say, the Ricci tensor are different from before — because the basis is different — but that a scalar, such as the Ricci scalar, is basis independent.

The main reason CLASSI was written was to solve the *equivalence problem* of determining when two metrics are really the same, i.e. are merely coordinate transformations of each other. For instance, CLASSI is able to determine whether a spacetime is spherically symmetric even when the metric is given in bizarre coordinates! For further information, see me.

The original problem tackled in the 1960's by the precursor of SHEEP and CLASSI was to calculate the Einstein tensor for an important spacetime with gravitational radiation, known as the Bondi metric. The original calculation by hand had taken 6 months; the computer took only a few minutes — and found 4 mistakes in the published computation. This program and computation formed the heart of the PhD thesis of none other than textbook author Ray d'Inverno! You can reproduce this computation — in a matter of seconds — by loading the Bondi metric into CLASSI with the command `(load "bondi.lor")` , and typing `(wmake ein)` .