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 the Maple package GRTensor (more properly GRTensorII), which I discuss first. Other available packages include a *Mathematica* version called GRTensorM (which is not quite as good), an older Maple package `tensor`, which comes with Maple, and the more specialized packages SHEEP and CLASSI, based on PSL (and hence closely related to REDUCE). (Recent versions of Maple also include a new differential geometry package, which you may wish to explore separately.)

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

These instructions should work on any campus PCs, such as those in the MLC, which have Maple installed, (requires ONID account), on the COSINe UNIX computers (requires COSINe account).

2. STARTING MAPLE

On UNIX machines, start Maple by typing `maple` for a shell interface, or `xmaple &` for the worksheet front end under XWindows. On PCs, there should be a Maple icon on the desktop. All Maple commands must end with either a semicolon or a colon; a colon suppresses output. All definitions must use `:=` , not just `=` . Exit Command Line Maple with the command `quit;` .

On COSINe machines, login to `app.science.oregonstate.edu`, not `frontend`.

Recent Maple installations start the Java interface by default, which does *not* work well with GRTensor. On UNIX machines, you can try typing `maple -cx` for the Classic interface, but this is not available on all machines; if you have trouble, use the shell interface. On PCs, use the Start menu to navigate to Programs and then Maple, where you should find shortcuts to both the Classic and Command Line interfaces.

3. GRTensor

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 campus PCs, you will first need to add `\\poole\ClassFolders` as a network share (this step may be optional), then in Maple type the following command (note the many backslashes):

```
libname:=libname,"\\\\poole\\ClassFolders\\Math-Dray\\GRtensor\\lib":
```

On COSINe machines, instead type the command:

```
libname:=libname,"/home/math/tevian/GRtensor/lib":
```

Then type `readlib(grii):` and `grtensor();` to start GRTensor.

On UNIX machines, you can add the appropriate command to the file `.mapleinit` in your home directory. On PCs, there is a similar file called `maple.ini` , but it's not as straightforward to set up — seek help online or from me.

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! (To use an orthonormal basis, choose "non-holonomic basis", then "covariant components", after which you will need to enter the orthonormal basis of 1-forms, followed by the metric, which is typically the diagonal matrix with entries $(-1\ 1\ 1\ 1)$.)

WARNING: `makeg` does *not* work with the (default) Java interface; either use another interface, or create and load a metric file, as described below.

Some standard metrics can be loaded by first telling GRTensor where to find them with the command (on COSINe machines)

```
grOptionMetricPath:="/home/math/tevian/GRtensor/metrics":
```

after which the file you want can be loaded using `qload` . (On PCs replace the argument in quotes above by `\\\\poole\\ClassFolders\\Math-Dray\\GRtensor\\metrics` .)

You can put this command in a file called `grtensor.ini` in your home directory (or the same directory as your `maple.ini` file), but *not* in your `.mapleinit` file.

For example, the Schwarzschild black hole metric (in an orthonormal basis) can be loaded with `qload(schwb)` . Pay attention to the signature, which may not what you expect!

You can also create metric files of your own by using the canned files as a model, 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 `eta11_` , etc, then finally the basis elements as `bd11_` , etc.; note the trailing underscore in each variable name. For the sphere, the file would be

```
Ndim_:=2:
x1_:=theta:x2_:=phi:
eta11_:=1:eta22_:=1:
bd11_:=r:bd22_:=r*sin(theta):
```

Unspecified metric components are assumed to be zero, so you do not need to explicitly set off-diagonal elements like `eta12_` to zero. See the documentation for `qload` or `grload` for help in determining how to load the file once created.

You can calculate various objects using `grcalc` , and see the result of the calculation using `grdisplay` . For instance, to find the metric, type

```
grcalc(metric);
grdisplay(metric);
```

Many standard tensors are predefined in terms of the metric you have entered or loaded; a complete list can be obtained with the command `?grt_objects` . Be warned however that most named objects assume the use of a coordinate basis. When working with an orthonormal basis, it is necessary to refer to e.g. the components of the curvature 2-form explicitly `R(bup,bdn,bdn,bdn)` ; the short name "`Riemann`" is an alias for the coordinate components `R(dn,dn,dn,dn)` . The connection components can be obtained as `rot(bup,bdn,bdn)` .

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

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

2. SHEEP AND CLASSI

SHEEP and CLASSI are available by logging in to ONID (`shell.onid.oregonstate.edu`) or COSINe (`frontend.science.oregonstate.edu`) shellservers.

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` (for ONID) or `source ~tevian/sheep/.shprc` (for COSINe). (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)` .

> The above initialization files assume you are running `tcsh` . If they give error messages, you are probably running `bash` ; try replacing `.shprc` with `.shprc.sh` .

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) 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 useful to set the same switches discussed below 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 in this context. 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)` .

b) 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.

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

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

```
(load "schwar.crd")
```