PHYS 514 Assignment 9

A.V. Zwaniga

Due Thursday March 28 2019

1 Problem 1

I have produced code that will calculate

1. the Christoffel symbols;
2. the Riemann tensor;
3. the Ricci tensor;
4. the Ricci scalar;
5. and the Einstein tensor.

You can find my code on my GitHub page at

`https://github.com/andrewzwaniga/CourseWork`

Please navigate to the folder `PHYS514`. There you will find several files, most of which are utility classes, and one of which is a script that should be run to observe the analysis of a given set of metrics. The most important observation one should make about my code is that it relies on the `sympy` library for symbolic calculations in `python`. (https://docs.sympy.org/latest/tutorial/intro.html)

Here is a brief description of my code; the files are documented with comments also.

1. `analyze_metric.py` can be run at the command line with `python` using

   ```
   $ python analyze_metric.py
   ```

   This will print a number of messages to the console explaining what work is being done behind the scenes with the utility classes. Importantly, the components of tensors will be printed to the screen. One can verify here what the Einstein tensor is for a given metric.

2. `christoffel.py, einstein.py, metric.py, ricci.py, riemann.py`, and `scalar.py` are the utility classes needed to complete calculations for a given metric. These classes are very similar in structure, and in fact could likely be implemented as extensions of a more general, abstract class properly called `Tensor`. (However, I did not have time to make this abstraction.) As an example, an instance of the `Metric` class will have two important attributes: `index_dict` and `elements`. If the metric is n-dimensional, `index_dict` is a dictionary of length n for which the keys are integers $0, 1, \dots, n-1$ and the values are strings corresponding to the names of the variables/indices in the metric. `elements` is initialized as a 2-level dictionary and then cast to a single a dictionary of length n^2 for which the keys are concatenated strings representing what the indices would look like symbolically, (e.g. `'tt'`) and the values are `sympy.core.symbols.Symbol` objects.

3. `metric_analysis.py` is where the bulk of my calculations are staged. There are many functions inside this script, and they relate to initializing metrics, calculating the other tensors from a metric, and handling the saving and reading of symbols to and from text files. I resolved to write some things into text files because I found that `sympy` does not handle matrix inversion in reliable manner. (The fact is that some times inverting the metric would be quick and other times the calculation would hang very cryptically.)

4. On this note, please find the inverse of the Kerr metric written in LaTeX in `problem3_latex.txt` and written in `python`-ready format in `problem3_python.txt`.

2 Problem 2

Please find included with my submission a printout of my solution to this problem as obtained using my code. Namely, I solved this by running the script `analyze_metric.py` at the command line. For the Scwharzchild metric, I find that my code produces $G_{\mu\nu} = 0$ as required. For the de Sitter metric, I find that my code produces $g^{\mu\nu} G_{\mu\nu} = -12H^2$ which indicates that in fact the de Sitter metric is a solution to Einstein's vacuum equation with a cosmological constant, namely $G_{\mu\nu} + \Lambda g_{\mu\nu} = 0$, such that $\Lambda = 12H^2$.

3 Problem 3

Please find included with my submission a printout of my solution to this problem as obtained using my code. Again, my solution was obtained by running `analyze_metric.py` at the command line with `python`. Unfortunately, `sympy` was not able to simplify the Einstein tensor for the Kerr metric in a reasonable amount of time. (I ran my code overnight for about 12 hours and it had not finished computing.) As a solution, Prof. Maloney suggested that I evaluate the Einstein tensor that my code produces at a point. I find that the components of the Einstein tensor are very small or identically zero at several points, though my attached solution indicates only one point. Generally speaking the deviations from zero are at the 10^{-15} level and so I am inclined to believe that my code in fact demonstrates that the Kerr metric is a solution to the vacuum Einstein equation, $G_{\mu\nu} = 0$.

4 Problem 4

Let the worldline be labelled by an affine parameter y so that $x^\mu(y) = (t(y), r(y), \theta(y), \phi(y))$.By running `analyze_metric.py`, in particular the function `analysis.solve_geodesic.py`, I obtain the solution for $\left(\frac{dr}{dy}\right)^2$ as

$$\left[-E^2 + \frac{L^2 R}{r^3(y)} - \frac{L^2}{r^2(y)} + \frac{R}{r(y)} - 1.0\right]$$

which is the result we had in class as written with an effective potential $V_{\text{eff}}(r)$.