Dexy EinsteinPy Examples*

(Dated: March 26, 2020)

Dexy makes it easy to write reproducible and recreatable documents in any document format, using any combination of programming languages. Dexy integrates prose, code, and data.

Website: https://dexy.it

Package Source: https://github.com/dexy/dexy

Examples: https://github.com/dexy/dexy-examples

I. PREDEFINED METRICS IN SYMBOLIC MODULE EXAMPLES

The `einsteinpy.symbolic.prefedined` module contains many functions with predefined metrics.

BarriolaVilekin: Barriola-Vilekin monopol metric Phys. Rev. Lett. 63, 341 Manuel Barriola and Alexander Vilenkin Published 24 July 1989

BertottiKasner: Birkhoff's theorem with Λ-term and Bertotti-Kasner space Phys. Lett. A, 245:363–365, 1998 W. Rindler

BesselGravitationalWave: Exact gravitational wave solution without diffraction. Class. Quantum Grav., 16:L75–78, 1999. D. Kramer.

An exact solution describing an axisymmetric gravitational wave propagating in the z-direction in closed form. This solution to Einstein's vacuum field equations has the remarkable property that the curvature invariants decrease monotonically with increasing radial distance from the axis and vanish at infinity. The solution is regular at the symmetry axis.

CMetric: The C-metric Stephani (Table 16.2) p188

Davidson: Davidson's cylindrically symmetric radiation perfect fluid universe Davidson, J. Math. Phys., v32, p1560, (1991)

AntiDeSitter: Anti-de Sitter space

Hawking and Ellis (5.9) p131

AntiDeSitterStatic: Static form of Anti-de Sitter space

Hawking and Ellis (5.9) p131

DeSitter: de Sitter space

Hawking and Ellis p125

Ernst: Black holes in a magnetic universe. J. Math. Phys., 17:54–56, 1976. Frederick J. Ernst.

find: Performs a find operation on available functions.

Godel: Godel metric Rev. Mod. Phys., v21, p447, (1949) Stephani (10.25) 122

JanisNewmanWinicour: Reality of the Schwarzschild singularity. Phys. Rev. Lett., 20:878–880, 1968. A. I. Janis, E. T. Newman, and J. Winicour.

Minkowski: Minkowski(flat) space-time in Cartesian coordinates. Space-time without any curvature or matter.

MinkowskiCartesian: Minkowski(flat) space-time in Cartesian coordinates. Space-time without any curvature or matter.

MinkowskiPolar: Minkowski(flat) space-time in Polar coordinates. Space-time without any curvature or matter.

Kerr: Kerr Metric in Boyer Lindquist coordinates.

KerrNewman: Kerr-Newman Metric in Boyer Lindquist coordinates.

ReissnerNordstorm: " The Reissner–Nordström metric in spherical coordinates A static solution to the Einstein–Maxwell field equations, which corresponds to the gravitational field of a charged, nonrotating, spherically symmetric body of mass M.

Schwarzschild: Schwarzschild exterior metric in curvature coordinates Schwarzschild, Sitz. Preuss. Akad. Wiss., p189, (1916) Stephani (13.19) p157

A. Printing the metrics for visualization

All the functions return instances of `einsteinpy.symbolic.metric.MetricTensor`

Here are four examples:

1. *Schwarzschild*

```
sch = Schwarzschild()
t = sch.tensor()
pprint(t)
```

$$\begin{bmatrix} 1-\frac{r_s}{r} & 0 & 0 & 0 \\ 0 & -\frac{1}{c^2\left(1-\frac{r_s}{r}\right)} & 0 & 0 \\ 0 & 0 & -\frac{r^2}{c^2} & 0 \\ 0 & 0 & 0 & -\frac{r^2\sin^2(\theta)}{c^2} \end{bmatrix}$$

* EinsteinPy examples based upon EinsteinPy documentation (c) 2020 EinsteinPy Development Team, used under MIT license.

2. Minkowski

```
mink = Minkowski(c=1)
t = mink.tensor()
pprint(t)
```

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

3. DeSitter

```
mink = DeSitter()
t = mink.tensor()
pprint(t)
```

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & e^{\frac{2x}{\alpha}} & 0 & 0 \\ 0 & 0 & e^{\frac{2x}{\alpha}} & 0 \\ 0 & 0 & 0 & e^{\frac{2x}{\alpha}} \end{bmatrix}$$

4. AntiDeSitter

```
mink = AntiDeSitter()
t = mink.tensor()
pprint(t)
```

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & \cos^2(t) & 0 & 0 \\ 0 & 0 & \cos^2(t)\sinh^2(\chi) & 0 \\ 0 & 0 & 0 & \sin^2(\theta)\cos^2(t)\sinh^2(\chi) \end{bmatrix}$$

B. Calculating the scalar (Ricci) curavtures

They should be constant for De-Sitter: $-\frac{2e^{-\frac{2x}{\alpha}}}{\alpha^2}$ and Anti-De-Sitter spacetimes:

$$\frac{\left(\sin^2(t)-1\right)\sinh^2(\chi)-2\cos^2(t)\sinh^2(\chi)}{\cos^2(t)\sinh^2(\chi)} + \frac{\left(\sin^2(t)-1\right)\sin^2(\theta)\sinh^2(\chi)-2\sin^2(\theta)\cos^2(t)\sinh^2(\chi)}{\sin^2(\theta)\cos^2(t)\sinh^2(\chi)} - 6$$

Which simplifies to -12.

II. USING DEXY WITH EINSTEINPY

For a full introduction to Dexy, please look at the Getting Started Tutorial.

Dexy runs by applying filters to files. Filters can transform file content in arbitrary ways, including executing code and formatting text or data.

Here are some brief introductions to common filters we apply to source code files.

A. pyg & idio

The pyg filter applies pygments syntax highlighting to source code. At the end of this document you can see the complete `predefined_metrics.py` file with syntax highlighting applied.

The idio filter uses the pygments system as well, but it first divides code up into sections using specially formatted comments. This allows you present and discuss code in manageable chunks, and also to leave out irrelevant parts (perhaps include them in an appendix).

For example, here is just the custom pprint function definition:

```
def pprint(expr):
    # unicode doesn't always play nicely with LaTeX
    sympy.pprint(expr, use_unicode=False)
```

B. Running Code

Next, we probably want to be able to actually run the code and see output it generates.

Depending on what we want to do with the output, we can take a variety of approaches.

1. Run With py Filter

The py filter runs python code in the python compiler and saves just the output.

Here is what gets written to stdout by the `predefined_metrics.py` script (long lines truncated for sanity):

```
[    r_s                                          ]
[1 - ---       0         0           0           ]
[     r                                           ]
[                                                 ]
[             -1                                  ]
[   0     ------------    0           0           ]
[          2 /    r_s\                           ]
[         c *|1 - ---|                           ]
[            \     r /                           ]
[                                                 ]
[                          2                      ]
[                        -r                       ]
[   0          0         ----        0           ]
[                          2                      ]
[                         c                       ]
[                                                 ]
[                                  2    2         ]
[                               -r *sin (theta)   ]
[   0          0         0     ----------------- ]
[                                        2        ]
[                                       c         ]
[-1   0    0    0 ]
[                 ]
[0   1.0   0    0 ]
[                 ]
[0    0   1.0   0 ]
[                 ]
[0    0    0   1.0]
```

```
[-1    0       0       0   ]
[                          ]
[      2*x                 ]
[     -----                ]
[     alpha                ]
[0   e         0       0   ]
[                          ]
[              2*x         ]
[             -----        ]
[             alpha        ]
[0     0     e         0   ]
[                          ]
[                      2*x ]
[                     -----]
[                     alpha]
[0     0       0     e     ]
[-1     0             0                          0
[
[       2
[0   cos (t)          0                          0
[
[                2        2
[0      0     cos (t)*sinh (chi)                  0
[
[                                      2           2        2
[0      0             0           sin (theta)*cos (t)*sinh (
```

2. *Run With pycon Filter*

An alternative, the pycon filter, runs Python more like a console or repl, and shows you both the input and output. It's great for tutorials. It also respects sections, so you can include just a section at a time.

For example, here is the pycon output of just the Minkowski section (syntax highlighting is applied by subsequently passing it throug the pyg filter):

```
>>> mink = Minkowski(c=1)
>>> t = mink.tensor()
>>> pprint(t)
[-1   0    0    0 ]
[                  ]
[0   1.0   0    0 ]
[                  ]
[0    0   1.0   0 ]
[                  ]
[0    0    0   1.0]
```

3. *Generating Artifacts*

While including the results of stdout or a REPL-like view can be useful some of the time, especially for code documentation/tutorials, for research papers or work that's more about results, we probably want to have the code running behind the scenes to generate artifacts which we can incorporate into documents in a more flexible way. With Dexy, any data files you generate as side effects will be available to your final document, and you can include them flexibly.

For example, here is a LaTeX rendering of the Schwarzschild tensor:

$$\begin{bmatrix} 1-\frac{r_s}{r} & 0 & 0 & 0 \\ 0 & -\frac{1}{c^2\left(1-\frac{r_s}{r}\right)} & 0 & 0 \\ 0 & 0 & -\frac{r^2}{c^2} & 0 \\ 0 & 0 & 0 & -\frac{r^2\sin^2(\theta)}{c^2} \end{bmatrix}$$

This was generated by saving latex output to a dictionary, and then accessing the desired key from this document:

```
<% set snippets = d['snippets.json'].from_json() %>

$<< snippets['Schwarzschild'] >>$
```

4. *Full Python Source Code*

```
# Example adapted from:
# https://docs.einsteinpy.org/en/latest/examples/Predefined%20Metrics%20in%20Symbolic%20Module.html

### "imports"
from einsteinpy.symbolic import RicciScalar
from einsteinpy.symbolic.predefined import AntiDeSitter
from einsteinpy.symbolic.predefined import DeSitter
from einsteinpy.symbolic.predefined import Minkowski
from einsteinpy.symbolic.predefined import Schwarzschild
from sympy import latex
from sympy import simplify
import einsteinpy.symbolic.predefined
import json
import sympy

snippets = {}

### "metrics-list"
snippets['predefined'] = {}
for k, v in einsteinpy.symbolic.predefined.__dict__.items():
    if callable(v):
        snippets['predefined'][k] = {
                "name" : v.__name__,
                "qualname" : v.__qualname__,
                "docs" : v.__doc__.split("Parameters")[0].strip()
                }
```

```
### "custom-pprint"
def pprint(expr):
    # unicode doesn't always play nicely with LaTeX
    sympy.pprint(expr, use_unicode=False)

### "Schwarzschild"
sch = Schwarzschild()
t = sch.tensor()
pprint(t)
### @end
assert isinstance(sch, einsteinpy.symbolic.metric.MetricTensor)
snippets['Schwarzschild'] = latex(t)

### "Minkowski"
mink = Minkowski(c=1)
t = mink.tensor()
pprint(t)
### @end
assert isinstance(sch, einsteinpy.symbolic.metric.MetricTensor)
snippets['Minkowski'] = latex(t)

### "DeSitter"
mink = DeSitter()
t = mink.tensor()
pprint(t)
### @end
assert isinstance(sch, einsteinpy.symbolic.metric.MetricTensor)
snippets['DeSitter'] = latex(t)

### "AntiDeSitter"
mink = AntiDeSitter()
t = mink.tensor()
pprint(t)
### @end
assert isinstance(sch, einsteinpy.symbolic.metric.MetricTensor)
snippets['AntiDeSitter'] = latex(t)

### "Ricci-curvatures"
scalar_curvature_de_sitter = RicciScalar.from_metric(DeSitter())
scalar_curvature_anti_de_sitter = RicciScalar.from_metric(AntiDeSitter())
### @end
snippets['DeSitter-curvature'] = latex(scalar_curvature_de_sitter.expr)
snippets['AntiDeSitter-curvature'] = latex(scalar_curvature_anti_de_sitter.expr)
snippets['AntiDeSitter-simplified'] = latex(simplify(scalar_curvature_anti_de_sitter.expr))

### "save-latex"
with open("snippets.json", 'w') as f:
    json.dump(snippets, f)
```