SageNP_Tutorial

November 9, 2024

1 SageNP: Newman-Penrose calculations for SageMath.

The package **SageNP** includes functions for some calculations defined in the Newman-Penrose formalism. The code is based on SageManifolds.

1.1 Coded by:

- Tolga Birkandan (Corr.: birkandant@itu.edu.tr)
- Onur Arman
- Emir Baysazan
- Selinay Sude Binici
- Pelin Ozturk

- **Special thanks to Eric Gourgoulhon**

1.2 Reference:

The reference for all definitions and calculations:

H. Stephani, D. Kramer, M. MacCallum, C. Hoenselaers, and E. Herlt, "Exact Solutions of Einstein's Field Equations", 2nd ed. Cambridge: Cambridge University Press, 2003.

2 BASIC DEFINITIONS AND NOTATION:

- We will use the metric signature: (- + + +)
- For the null-tetrad vector names, the ref. book uses $(k, l, m, \overline{m})$. However, in the code we will use $(l, n, m, \overline{m})$ like the rest of the literature. **Therefore one should set $k \to l, l \to n$ in the ref. book.**
- Products of the vectors are given by: $l^a n_a = -1, m^a \overline{m}_a = 1$, all others zero.
- The metric is found using the covariant null-tetrad vectors as:

  $ds^2 = g_{\mu\nu} dx^\mu \otimes dx^\nu = -l \otimes n - n \otimes l + m \otimes \overline{m} + \overline{m} \otimes m$

  where

  $g_{ab} = -l_a n_b - n_a l_b + m_a \overline{m}_b + \overline{m}_a m_b$

  and,

$$g=\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&0&-1\\0&0&-1&0\end{bmatrix}$$

- Please check the reference book for the details and further definitions.

3 EXAMPLE: Schwarzschild spacetime

The Schwarzschild metric is given by

$ds^2=-\frac{\Delta}{r^2}dt^2+\frac{r^2}{\Delta}dr^2+r^2d\theta^2+r^2sin^2\theta d\phi^2$

where

$\Delta=r^2-2Mr$

Import the SageNP package:

If you have not installed it yet, run

`pip install SageNP`

[1]:
```
try:
    from SageNP import NewmanPenrose
except:
    %pip install SageNP
    from SageNP import NewmanPenrose
```

Define the four dimensional manifold MyManifold:

[2]:
```
MyManifold = Manifold(4 , 'MyManifold', r'\mathcal{Man}')
```

Define the coordinates (t,r,θ,ϕ)**:**

[3]:
```
MyCoordinates.<t,r,th,ph> = MyManifold.chart(r't r th:\theta ph:\phi')
```

Define the metric functions and variables (if needed):

We need the variable M and the function Δ for the Schwarzschild metric

[4]:
```
var('M')

# The function Delta:
# You can either define Delta as a real function of coordinate r as
# Delta=function('Delta',imag_part_func=0)(r)
# and work with this general function,

# or give its exact expression (let us continue with this choice):
Delta=r^2-2*M*r
```

Enter null tetrad elements:

Covariant or contravariant null-tetrad vectors are needed. In this example, we will use covariant vectors for the Schwarzschild metric, namely,

$l_\mu = [1, -\frac{r^2}{\Delta}, 0, 0]$

$n_\mu = [\frac{\Delta}{2r^2}, \frac{1}{2}, 0, 0]$

$m_\mu = [0, 0, -\frac{r}{\sqrt{2}}, -i\frac{r}{\sqrt{2}}sin(\theta)]$

$\overline{m}_\mu = [0, 0, -\frac{r}{\sqrt{2}}, i\frac{r}{\sqrt{2}}sin(\theta)]$

Here, the element ordering is the same as the coordinate ordering. (The first element is the t element, the second is the r element, etc.)

[5]:
```
lvec=[1,-(r^2)/Delta,0,0]

nvec=[Delta/(2*r^2),1/2,0,0]

mvec=[0,0,(-r/sqrt(2)),(-I*r/sqrt(2))*sin(th)]

mbarvec=[0,0,(-r/sqrt(2)),(I*r/sqrt(2))*sin(th)]
```

Define an object of the class:

Here, our null-tetrad vectors lvec, nvec, mvec and mbarvec are covariant. Thus we will use the keyword 'covariant'.

(If they were contravariant, then we should use the keyword 'contravariant'.)

Once the object is defined, the code calculates the metric and displays it on the screen. It is recommended that you check your metric.

[6]:
```
schw=NewmanPenrose(MyManifold,MyCoordinates,lvec,nvec,mvec,mbarvec,'covariant')
```

```
The metric:
```

$$g = \left(\frac{2\,M - r}{r}\right) \mathrm{d}t \otimes \mathrm{d}t + \left(-\frac{r}{2\,M - r}\right) \mathrm{d}r \otimes \mathrm{d}r + r^2 \mathrm{d}\theta \otimes \mathrm{d}\theta + r^2 \sin\left(\theta\right)^2 \mathrm{d}\phi \otimes \mathrm{d}\phi$$

Let us **test the null-tetrad with the product rules** ($l^a n_a = -1, m^a \overline{m}_a = 1$, all others zero.)

[7]:
```
schw.test_nulltetrad()
```

```
Testing null tetrad…
PASSED
```

Calculate and display the **spin coefficients** as given in (Page 75-76, Eq.(7.2)):

(**NOTE**: All page and equation numbers belong to the reference book.)

[8]:
```
schw.calculate_spincoefficients()
schw.show_spincoefficients()
```

```
Calculating spin coefficients…
```

kappaNP=0

tauNP=0

sigmaNP=0

rhoNP=$\frac{1}{r}$

piNP=0

nuNP=0

muNP=$-\frac{2M-r}{2r^2}$

lambdaNP=0

epsilonNP=0

gammaNP=$-\frac{M}{2r^2}$

betaNP=$-\frac{\sqrt{2}\cos(\theta)}{4r\sin(\theta)}$

alphaNP=$\frac{\sqrt{2}\cos(\theta)}{4r\sin(\theta)}$

All spin coefficients are available under their names: **kappaNP, kappabarNP, tauNP, taubarNP, sigmaNP, sigmabarNP, rhoNP, rhobarNP, piNP, pibarNP, nuNP, nubarNP, muNP, mubarNP, lambdaNP, lambdabarNP, epsilonNP, epsilonbarNP, gammaNP, gammabarNP, betaNP, betabarNP, alphaNP, alphabarNP**

Display a **single** spin coefficient:

[9]:
```
show(schw.gammaNP.expr())
```

$-\frac{M}{2r^2}$

Calculate and display the **Weyl tensor components** as given in (Page 38, Eq.(3.59)):

[10]:
```
schw.calculate_Weyl()
schw.show_Weyl()
```

Calculating Weyl components…

Psi0NP=0

Psi1NP=0

Psi2NP=$-\frac{M}{r^3}$

Psi3NP=0

Psi4NP=0

All Weyl tensor components are available under their names: **Psi0NP, Psi1NP, Psi2NP, Psi3NP, Psi4NP**

Display a **single** Weyl tensor component:

[11]:
```
show(schw.Psi2NP.expr())
```

$$-\frac{M}{r^3}$$

Calculate and display the **Ricci tensor components** as given in (Page 78, Eq.(7.10-7.15)):

[12]:
```
schw.calculate_Ricci()
schw.show_Ricci()
```

```
Calculating Ricci components…
```

Phi00NP=0

Phi01NP=0

Phi10NP=0

Phi02NP=0

Phi20NP=0

Phi11NP=0

Phi12NP=0

Phi21NP=0

Phi22NP=0

LambdaNP=0

All Ricci tensor components are available under their names:**Phi00NP, Phi01NP, Phi10NP, Phi02NP, Phi20NP, Phi11NP, Phi12NP, Phi21NP, Phi22NP, LambdaNP**

Display a **single** Ricci tensor component:

[13]:
```
show(schw.Phi00NP.expr())
```

$$0$$

Calculate and display the **Newman-Penrose equations** as given in (Page 79, Eq.(7.21)):

All Newman-Penrose equations are defined as 0 = -(left hand side)+(right hand side) of the equations.

[14]:
```
schw.calculate_NPeq()
schw.show_NPeq()
```

```
Calculating NP equations…
```

NPeq1=0

NPeq2=0

NPeq3=0

NPeq4=0

NPeq5=0

NPeq6=0

NPeq7=0

NPeq8=0

NPeq9=0

NPeq10=0

NPeq11=0

NPeq12=0

NPeq13=0

NPeq14=0

NPeq15=0

NPeq16=0

NPeq17=0

NPeq18=0

All Newman-Penrose equations are available under their names *in the order they are given in the reference*: **NPeq1, NPeq2, NPeq3, NPeq4, NPeq5, NPeq6, NPeq7, NPeq8, NPeq9, NPeq10, NPeq11, NPeq12, NPeq13, NPeq14, NPeq15, NPeq16, NPeq17, NPeq18**

Display a **single** Newman-Penrose equation:

[15]:
```
show(schw.NPeq8.expr())
```

0

Calculate and display the **Bianchi identities** as given in (Page 81, Eq.(7.32)):

All Bianchi identities are defined as 0 = -(left hand side)+(right hand side) of the equations.

[16]:
```
schw.calculate_Bianchi()
schw.show_Bianchi()
```

Calculating Bianchi identities…

BI1=0

BI2=0

BI3=0

BI4=0

BI5=0

BI6=0

BI7=0

BI8=0

BI9=0

BI10=0

BI11=0

All Bianchi identities are available under their names *in the order they are given in the reference*: **BI1, BI2, BI3, BI4, BI5, BI6, BI7, BI8, BI9, BI10, BI11**

Display a **single** Bianchi identity:

[17]:
```
show(schw.BI7.expr())
```

0

Find the **Petrov type** using the **Petrov invariants**:

Calculates the Petrov type using the invariants I, J, K, L, N.

[18]:
```
schw.Petrov_frominvariants()
```

```
Calculating Petrov Type…
Petrov Type D
Attention: This procedure depends on the simplification of the structures.
Therefore the Petrov type can be simpler.
```

The Petrov invariants can be calculated independently if needed:

All Petrov invariants are available under their names: **PetrovinvINP, PetrovinvJNP, PetrovinvKNP, PetrovinvLNP, PetrovinvNNP**

[19]:
```
schw.calculate_PetrovinvINP()
schw.calculate_PetrovinvJNP()
schw.calculate_PetrovinvKNP()
schw.calculate_PetrovinvLNP()
schw.calculate_PetrovinvNNP()

show(schw.PetrovinvINP.expr())
show(schw.PetrovinvJNP.expr())
show(schw.PetrovinvKNP.expr())
show(schw.PetrovinvLNP.expr())
show(schw.PetrovinvNNP.expr())
```

$\frac{3\,M^2}{r^6}$

$\frac{M^3}{r^9}$

0

0

0

Find the **Petrov type** using the *Weyl tensor components**:

[20]:
```
schw.Petrov_fromWeyl()
```

```
Calculating Petrov Type…
Petrov Type D
Attention: This procedure depends on the simplification of the structures.
Therefore the Petrov type can be simpler.
```

Directional derivatives can be calculated as given in (Page 43, Eq.(3.82)):

- **DlNP(X)**: Given X, calculates the D derivative (l direction).
- **DeltanNP(X)**: Given X, calculates the Δ derivative (n direction)
- **deltamNP(X)**: Given X, calculates the δ derivative (m direction)
- **deltambarNP(X)**: Given X, calculates the $\bar{\delta}$ derivative (mbar direction)

Calculate and display the **directional derivatives** of the **spin coefficient** γ **(gammaNP)**:

[21]:
```
show(schw.DlNP(schw.gammaNP).expr())
show(schw.DeltanNP(schw.gammaNP).expr())
show(schw.deltamNP(schw.gammaNP).expr())
show(schw.deltambarNP(schw.gammaNP).expr())
```

$$-\frac{M}{r^3}$$

$$-\frac{2\,M^2 - Mr}{2\,r^4}$$

0

0

Commutators can be calculated as given in (Page 77, Eq.(7.6)):

- *The right-hand sides of the commutation relations are calculated.*
- **Deltan_Dl_commNP(X)**: Given X, calculates the $[\Delta, D]$ commutator.
- **deltam_Dl_commNP(X)**: Given X, calculates the $[\delta, D]$ commutator.
- **deltam_Deltan_commNP(X)**: Given X, calculates the $[\delta, \Delta]$ commutator.
- **deltambar_deltam_commNP(X)**: Given X, calculates the $[\bar{\delta}, \delta]$ commutator.

Calculate and display the **commutators** for the **spin coefficient** ρ **(rhoNP)**:

[22]:
```
show(schw.Deltan_Dl_commNP(schw.rhoNP).expr())
show(schw.deltam_Dl_commNP(schw.rhoNP).expr())
show(schw.deltam_Deltan_commNP(schw.rhoNP).expr())
show(schw.deltambar_deltam_commNP(schw.rhoNP).expr())
```

$-\frac{M}{r^4}$

0

0

0

These are the **right-hand sides** of the commutation relations.

Let us check if $[\Delta, D]$ (Deltan_Dl_commNP) commutation runs correctly **by calculating the left-hand side of the equation using the directional derivatives**: $\Delta D\rho - D\Delta\rho$.

Check the difference to see if it is zero:

[23]:
```
fromcommutatorfunction=schw.Deltan_Dl_commNP(schw.rhoNP).expr()
directcommutation=schw.DeltanNP(schw.DlNP(schw.rhoNP)).expr()-schw.DlNP(schw.
 ↪DeltanNP(schw.rhoNP)).expr()

show((fromcommutatorfunction-directcommutation).simplify_full())
```

0

calculate_allNP() runs the following functions: calculate_spincoefficients(), calculate_Weyl(), calculate_Ricci(), calculate_NPeq(), calculate_Bianchi(), Petrov_frominvariants(), Petrov_fromWeyl()

[24]:
```
schw.calculate_allNP()
```

```
Calculating spin coefficients…
Calculating Weyl components…
Calculating Ricci components…
Calculating NP equations…
Calculating Bianchi identities…
Calculating Petrov Type…
Petrov Type D
Attention: This procedure depends on the simplification of the structures.
Therefore the Petrov type can be simpler.
Calculating Petrov Type…
Petrov Type D
Attention: This procedure depends on the simplification of the structures.
Therefore the Petrov type can be simpler.
```

show_allNP() runs the following functions: show_spincoefficients(), show_Weyl(), show_Ricci(), show_NPeq(), show_Bianchi(), Petrov_frominvariants(), Petrov_fromWeyl()

[25]:
```
schw.show_allNP()
```

kappaNP=0

tauNP=0

sigmaNP=0

rhoNP=$\frac{1}{r}$

piNP=0

nuNP=0

muNP=$-\frac{2\,M-r}{2\,r^2}$

lambdaNP=0

epsilonNP=0

gammaNP=$-\frac{M}{2\,r^2}$

betaNP=$-\frac{\sqrt{2}\cos(\theta)}{4\,r\sin(\theta)}$

alphaNP=$\frac{\sqrt{2}\cos(\theta)}{4\,r\sin(\theta)}$

Psi0NP=0

Psi1NP=0

Psi2NP=$-\frac{M}{r^3}$

Psi3NP=0

Psi4NP=0

Phi00NP=0

Phi01NP=0

Phi10NP=0

Phi02NP=0

Phi20NP=0

Phi11NP=0

Phi12NP=0

Phi21NP=0

Phi22NP=0

LambdaNP=0

NPeq1=0

NPeq2=0

NPeq3=0

NPeq4=0

NPeq5=0

```
NPeq6=0

NPeq7=0

NPeq8=0

NPeq9=0

NPeq10=0

NPeq11=0

NPeq12=0

NPeq13=0

NPeq14=0

NPeq15=0

NPeq16=0

NPeq17=0

NPeq18=0

BI1=0

BI2=0

BI3=0

BI4=0

BI5=0

BI6=0

BI7=0

BI8=0

BI9=0

BI10=0

BI11=0

Calculating Petrov Type…
Petrov Type D
Attention: This procedure depends on the simplification of the structures.
Therefore the Petrov type can be simpler.
Calculating Petrov Type…
Petrov Type D
Attention: This procedure depends on the simplification of the structures.
Therefore the Petrov type can be simpler.
```

Klein-Gordon equation for a massive scalar field can be calculated as defined in G. Silva-Ortigoza, Rev. Mex. Fis. 4, 543 (1996).

```
[26]: var('M2') # mass of the scalar field
Phi = MyManifold.scalar_field(function('Phi')(t,r,th,ph))
schw.kleingordon(Phi,M2)
show(schw.kgNP.expr())
```

$$(2\,M - r)\cos(\theta)\sin(\theta)\,\frac{\partial}{\partial\theta}\Phi(t,r,\theta,\phi) + \left(r^3\frac{\partial^2}{(\partial t)^2}\Phi(t,r,\theta,\phi) - (2\,M M_2^2 r^2 - M_2^2 r^3)\Phi(t,r,\theta,\phi) - 2\,(2\,M^2 - 3\,Mr +$$
$$\overline{\qquad\qquad\qquad\qquad (2\,M}$$

Massive Dirac equation components can be calculated as defined in S. Chandrasekhar, “Mathematical Theory of Black Holes”, Oxford Univ. Press, New York (1983), p.544.

```
[27]: var('M2') # mass
f1 = MyManifold.scalar_field(function('f1')(t,r,th,ph))
f2 = MyManifold.scalar_field(function('f2')(t,r,th,ph))
g1 = MyManifold.scalar_field(function('g1')(t,r,th,ph))
g2 = MyManifold.scalar_field(function('g2')(t,r,th,ph))
schw.dirac(f1,f2,g1,g2,M2)
show(schw.diracNP)
show(schw.diracNP[0].expr())
show(schw.diracNP[1].expr())
show(schw.diracNP[2].expr())
show(schw.diracNP[3].expr())
```

[Scalar field on the 4-dimensional differentiable manifold MyManifold, Scalar field on the 4-dimensional differentia

$$(2\,M - r)\cos(\theta)\,f_2(t,r,\theta,\phi) - 2\left(\sqrt{2}r^2\frac{\partial}{\partial t}f_1(t,r,\theta,\phi) - \left(2\sqrt{2}M - \sqrt{2}r\right)f_1(t,r,\theta,\phi) - \left(-2i\sqrt{2}MM_2r + i\sqrt{2}M_2r\right.\right.$$
$$\overline{\qquad\qquad\qquad\qquad 2\left(2\sqrt{2}Mr\right.}$$

$$\sqrt{2}\Big(r\cos(\theta)\,f_1(t,r,\theta,\phi) + \left(-2i\sqrt{2}M_2r^2g_2(t,r,\theta,\phi) + \sqrt{2}r^2\frac{\partial}{\partial t}f_2(t,r,\theta,\phi) + \left(\sqrt{2}M - \sqrt{2}r\right)f_2(t,r,\theta,\phi) + 2\,r\frac{\partial}{\partial\theta}f_1\right.$$
$$\overline{\qquad\qquad\qquad\qquad 4\,r^2\sin(\theta)}$$

$$-\frac{(2\,M - r)\cos(\theta)\,g_1(t,r,\theta,\phi) + 2\left(\sqrt{2}r^2\frac{\partial}{\partial t}g_2(t,r,\theta,\phi) - \left(-2i\sqrt{2}MM_2r + i\sqrt{2}M_2r^2\right)f_2(t,r,\theta,\phi) - \left(2\sqrt{2}M - \right.\right.}{2\left(2\sqrt{2}Mr\right.}$$

$$-\frac{\sqrt{2}\Big(r\cos(\theta)\,g_2(t,r,\theta,\phi) + \left(2i\sqrt{2}M_2r^2f_1(t,r,\theta,\phi) - \sqrt{2}r^2\frac{\partial}{\partial t}g_1(t,r,\theta,\phi) - \left(\sqrt{2}M - \sqrt{2}r\right)g_1(t,r,\theta,\phi) - \left(2\sqrt{2}M\right.\right.}{4\,r^2\sin(\theta)}$$

SL(2,C) Transformations (Type A, Type B and Type C) defined in Carmeli and Kaye, Annals of Physics 99, 188 (1976) can be calculated.

```
[28]: # Type A:
var('z')
schw.type_A_transformation(z)
schw.show_type_A_transformation()

# Type B:
schw.type_B_transformation(z)
schw.show_type_B_transformation()
```

```
# Type C:
schw.type_C_transformation(z)
schw.show_type_C_transformation()
```

Results of the Type A Transformations:

lNP_trA= $\left[1, \frac{r^2}{2\,Mr - r^2}, 0, 0\right]$

nNP_trA= $\left[\frac{2\,rz^2 - 2\,M + r}{2\,r}, \frac{2\,rz^2 + 2\,M - r}{2\,(2\,M - r)}, -\sqrt{2}rz, 0\right]$

mNP_trA= $\left[z, \frac{rz}{2\,M - r}, -\frac{1}{2}\,\sqrt{2}r, -\frac{1}{2}i\,\sqrt{2}r\sin\left(\theta\right)\right]$

mbarNP_trA= $\left[z, \frac{rz}{2\,M - r}, -\frac{1}{2}\,\sqrt{2}r, \frac{1}{2}i\,\sqrt{2}r\sin\left(\theta\right)\right]$

kappaNP_trA=0

tauNP_trA=0

sigmaNP_trA=0

rhoNP_trA=$\frac{1}{r}$

piNP_trA=0

nuNP_trA=$\frac{z^2}{r}$

muNP_trA=$\frac{2\,z}{r} - \frac{2\,M - r}{2\,r^2}$

lambdaNP_trA=0

epsilonNP_trA=0

gammaNP_trA=$-\frac{\sqrt{2}z\cos\left(\theta\right)}{4\,r\sin\left(\theta\right)} + \frac{\bar{z}}{r} - \frac{M}{2\,r^2}$

betaNP_trA=$-\frac{\sqrt{2}\cos\left(\theta\right)}{4\,r\sin\left(\theta\right)}$

alphaNP_trA=$\frac{z}{r} + \frac{\sqrt{2}\cos\left(\theta\right)}{4\,r\sin\left(\theta\right)}$

Psi0NP_trA=0

Psi1NP_trA=0

Psi2NP_trA=$-\frac{M}{r^3}$

Psi3NP_trA=$-\frac{3\,Mz}{r^3}$

Psi4NP_trA=$-\frac{6\,Mz^2}{r^3}$

Phi00NP_trA=0

Phi01NP_trA=0

Phi02NP_trA=0

Phi10NP_trA=0

Phi11NP_trA=0

Phi12NP_trA=0

Phi20NP_trA=0

Phi21NP_trA=0

Phi22NP_trA=0

LambdaNP_trA=0

Results of the Type B Transformations:

lNP_trB= $\left[|z|^2, \frac{r^2|z|^2}{2\,Mr - r^2}, 0, 0\right]$

nNP_trB= $\left[-\frac{2\,Mr - r^2}{2\,r^2|z|^2}, \frac{1}{2\,|z|^2}, 0, 0\right]$

mNP_trB= $\left[0, 0, -\frac{\sqrt{2}r\overline{z}}{2\,|z|}, -\frac{i\,\sqrt{2}r\overline{z}\sin(\theta)}{2\,|z|}\right]$

mbarNP_trB= $\left[0, 0, -\frac{\sqrt{2}rz}{2\,|z|}, \frac{i\,\sqrt{2}rz\sin(\theta)}{2\,|z|}\right]$

kappaNP_trB=0

tauNP_trB=0

sigmaNP_trB=0

rhoNP_trB=$\frac{z\overline{z}}{r}$

piNP_trB=0

nuNP_trB=0

muNP_trB=$-\frac{2\,M - r}{2\,r^2 z\overline{z}}$

lambdaNP_trB=0

epsilonNP_trB=0

gammaNP_trB=$-\frac{M}{2\,r^2 z\overline{z}}$

betaNP_trB=$-\frac{\sqrt{2}\overline{z}\cos(\theta)}{4\,rz\sin(\theta)}$

alphaNP_trB=$\frac{\sqrt{2}\overline{z}\cos(\theta)}{4\,rz\sin(\theta)}$

Psi0NP_trB=0

Psi1NP_trB=0

Psi2NP_trB=$-\frac{Mz^2}{r^3}$

Psi3NP_trB=0

Psi4NP_trB=0

Phi00NP_trB=0

Phi01NP_trB=0

Phi02NP_trB=0

Phi10NP_trB=0

Phi11NP_trB=0

Phi12NP_trB=0

Phi20NP_trB=0

Phi21NP_trB=0

Phi22NP_trB=0

LambdaNP_trB=0

Results of the Type C Transformations:

lNP_trC=$\left[-\frac{(2\,M-r)z^2-2\,r}{2\,r}, \frac{(2\,M-r)z^2+2\,r}{2\,(2\,M-r)}, -\sqrt{2}rz, 0\right]$

nNP_trC=$\left[-\frac{2\,Mr-r^2}{2\,r^2}, \frac{1}{2}, 0, 0\right]$

mNP_trC=$\left[-\frac{(2\,M-r)z}{2\,r}, \frac{1}{2}\,z, -\frac{1}{2}\,\sqrt{2}r, -\frac{1}{2}i\,\sqrt{2}r\sin(\theta)\right]$

mbarNP_trC=$\left[-\frac{(2\,M-r)z}{2\,r}, \frac{1}{2}\,z, -\frac{1}{2}\,\sqrt{2}r, \frac{1}{2}i\,\sqrt{2}r\sin(\theta)\right]$

kappaNP_trC=$-\frac{1}{2}\,z^2\left(\frac{2\,M-r}{r^2}+\frac{2\,M}{r^2}\right)\overline{z}+\frac{\sqrt{2}z^2\cos(\theta)}{2\,r\sin(\theta)}-\frac{\sqrt{2}z\overline{z}\cos(\theta)}{4\,r\sin(\theta)}$

tauNP_trC=$-\frac{Mz}{r^2}$

sigmaNP_trC=$-\frac{1}{2}\,z^2\left(\frac{2\,M-r}{r^2}+\frac{2\,M}{r^2}\right)-\frac{\sqrt{2}z\cos(\theta)}{2\,r\sin(\theta)}$

rhoNP_trC=$-\frac{(2\,M-r)z}{2\,r^2}-\frac{(2\,M-r)\overline{z}}{2\,r^2}+\frac{1}{r}$

piNP_trC=0

`nuNP_trC=`0

`muNP_trC=`$-\frac{Mz\overline{z}}{r^2}+\frac{\sqrt{2}z\cos(\theta)}{2\,r\sin(\theta)}-\frac{2\,M-r}{2\,r^2}$

`lambdaNP_trC=`0

`epsilonNP_trC=`$-\frac{1}{2}\,z\left(\frac{2\,M-r}{r^2}+\frac{M}{r^2}\right)\overline{z}-\frac{\sqrt{2}z\overline{z}\cos(\theta)}{4\,r\sin(\theta)}+\frac{\sqrt{2}z\cos(\theta)}{4\,r\sin(\theta)}$

`gammaNP_trC=`$-\frac{M}{2\,r^2}$

`betaNP_trC=`$-\frac{(2\,M-r)z}{2\,r^2}-\frac{Mz}{2\,r^2}-\frac{\sqrt{2}\cos(\theta)}{4\,r\sin(\theta)}$

`alphaNP_trC=`$-\frac{Mz}{2\,r^2}+\frac{\sqrt{2}\cos(\theta)}{4\,r\sin(\theta)}$

`Psi0NP_trC=`$-\frac{6\,Mz^2}{r^3}$

`Psi1NP_trC=`$-\frac{3\,Mz}{r^3}$

`Psi2NP_trC=`$-\frac{M}{r^3}$

`Psi3NP_trC=`0

`Psi4NP_trC=`0

`Phi00NP_trC=`0

`Phi01NP_trC=`0

`Phi02NP_trC=`0

`Phi10NP_trC=`0

`Phi11NP_trC=`0

`Phi12NP_trC=`0

`Phi20NP_trC=`0

`Phi21NP_trC=`0

`Phi22NP_trC=`0

`LambdaNP_trC=`0

4 EXAMPLE: Reissner-Nordstrom spacetime

In this example, the null-tetrad will be given by **contravariant** elements, namely,

$l^\mu = [\frac{r^2}{\Delta}, 1, 0, 0]$

$n^\mu = [\frac{1}{2}, -\frac{\Delta}{2r^2}, 0, 0]$

$m^\mu = [0, 0, \frac{1}{r\sqrt{2}}, i\frac{csc(\theta)}{r\sqrt{2}}]$

$\overline{m}^\mu = [0, 0, \frac{1}{r\sqrt{2}}, -i\frac{csc(\theta)}{r\sqrt{2}}]$

Thus, the keyword **'contravariant'** should be given while defining the object as in the following cell.

[29]:
```
reset()

from SageNP import NewmanPenrose

###########################################################################
# Define 4-dim. the manifold:
MyManifold = Manifold(4 , 'MyManifold', r'\mathcal{Man}')
###########################################################################
MyCoordinates.<t,r,th,ph> = MyManifold.chart(r't r th:\theta ph:\phi')
####################################################
# Define the metric functions
var('M,Q')
Delta=r^2-2*M*r+Q^2
####################################################
# Enter null tetrad elements
# These vectors define the Reissner-Nordstrom spacetime
lveccont=[(r^2)/Delta,1,0,0]
nveccont=[1/2,-Delta/(2*r^2),0,0]
mveccont=[0,0,1/(r*sqrt(2)),I*csc(th)/(r*sqrt(2))]
mbarveccont=[0,0,1/(r*sqrt(2)),-I*csc(th)/(r*sqrt(2))]
###########################################################################

# Define the object "reisnor" of the class "SageNP":
reisnor=NewmanPenrose(MyManifold, MyCoordinates, lveccont, nveccont, mveccont,␣
↪mbarveccont, 'contravariant')
```

`Inverting tetrad…`

`The metric:`

$$g = \left(-\frac{Q^2 - 2\,Mr + r^2}{r^2}\right) \mathrm{d}t \otimes \mathrm{d}t + \left(\frac{r^2}{Q^2 - 2\,Mr + r^2}\right) \mathrm{d}r \otimes \mathrm{d}r + r^2 \mathrm{d}\theta \otimes \mathrm{d}\theta + r^2 \sin\left(\theta\right)^2 \mathrm{d}\phi \otimes \mathrm{d}\phi$$

Let us run **show_allNP()** command to **calculate and display** some NP expressions:

[30]:
```
reisnor.show_allNP()
```

`Calculating spin coefficients…`

kappaNP=0

tauNP=0

sigmaNP=0

rhoNP=$-\frac{1}{r}$

piNP=0

nuNP=0

muNP=$-\frac{Q^2-2\,Mr+r^2}{2\,r^3}$

lambdaNP=0

epsilonNP=0

gammaNP=$-\frac{Q^2-Mr}{2\,r^3}$

betaNP=$\frac{\sqrt{2}\cos(\theta)}{4\,r\sin(\theta)}$

alphaNP=$-\frac{\sqrt{2}\cos(\theta)}{4\,r\sin(\theta)}$

Calculating Weyl components…

Psi0NP=0

Psi1NP=0

Psi2NP=$\frac{Q^2-Mr}{r^4}$

Psi3NP=0

Psi4NP=0

Calculating Ricci components…

Phi00NP=0

Phi01NP=0

Phi10NP=0

Phi02NP=0

Phi20NP=0

Phi11NP=$\frac{Q^2}{2\,r^4}$

Phi12NP=0

Phi21NP=0

Phi22NP=0

LambdaNP=0

Calculating NP equations…

NPeq1=0

NPeq2=0

```
NPeq3=0

NPeq4=0

NPeq5=0

NPeq6=0

NPeq7=0

NPeq8=0

NPeq9=0

NPeq10=0

NPeq11=0

NPeq12=0

NPeq13=0

NPeq14=0

NPeq15=0

NPeq16=0

NPeq17=0

NPeq18=0

Calculating Bianchi identities…

BI1=0

BI2=0

BI3=0

BI4=0

BI5=0

BI6=0

BI7=0

BI8=0

BI9=0

BI10=0

BI11=0

Calculating Petrov Type…
Petrov Type D
Attention: This procedure depends on the simplification of the structures.
Therefore the Petrov type can be simpler.
Calculating Petrov Type…
Petrov Type D
```

```
Attention: This procedure depends on the simplification of the structures.
Therefore the Petrov type can be simpler.
```

[ ]:

```

```