Neutron Star TOV equation

using xAct package in Mathematica

Physics of Compact Objects, Jan-Apr 2021

Md Arif Shaikh, Postdoctoral Fellow, ICTS-TIFR

Feb 27, 2021

Load the Necessary packages

In the current tutorial we want to compute the components of the Einstein tensor in a particular basis. **xTensor** does not do this. For this we need **xCoba** which performs component calculations. However, **xCoba** needs **xTensor** for it's internal calculations. So we need these two packages which are bundled inside the **xAct** package.

In[1]:=

```
<< xAct`xTensor`
<< xAct`xCoba`
```

```
------------------------------------------------------------

Package xAct`xPerm`  version 1.2.3, {2015, 8, 23}

CopyRight (C) 2003-2020, Jose M. Martin-Garcia, under the General Public License.

Connecting to external linux executable...

Connection established.

------------------------------------------------------------

Package xAct`xTensor`  version 1.1.4, {2020, 2, 16}

CopyRight (C) 2002-2020, Jose M. Martin-Garcia, under the General Public License.

------------------------------------------------------------

These packages come with ABSOLUTELY NO WARRANTY; for details type
  Disclaimer[]. This is free software, and you are welcome to redistribute
  it under certain conditions. See the General Public License for details.

------------------------------------------------------------

------------------------------------------------------------

Package xAct`xCoba`  version 0.8.5, {2020, 11, 29}

CopyRight (C) 2005-2020, David Yllanes and
  Jose M. Martin-Garcia, under the General Public License.

------------------------------------------------------------

These packages come with ABSOLUTELY NO WARRANTY; for details type
  Disclaimer[]. This is free software, and you are welcome to redistribute
  it under certain conditions. See the General Public License for details.

------------------------------------------------------------
```

Define the Manifold

The first step is to define our manifold. We give a name to our manifold, dimension of the manifold and a set of abstract indices for the tensor on this manifold. The manifold is defined using the function **DefManifold**.

```
In[3]:= DefManifold[M4, 4, {α, β, γ, μ, ν, λ, σ, η}]
```

```
** DefManifold: Defining manifold M4.

** DefVBundle: Defining vbundle TangentM4.
```

M4 is the manifold we are defining, **4** is the dimension of this manifold and the greek symbols are abstract indices for the tensors on this manifold.

Define the metric

Now that we have defined our manifold, we will now define a metric on this manifold. We do this using the **DefMetric** function.

```
In[4]:= DefMetric[-1, metric[-α, -β], CD, {";", "∇"}]
```

```
** DefTensor: Defining symmetric metric tensor metric[-α, -β].

** DefTensor: Defining antisymmetric tensor epsilonmetric[-α, -β, -γ, -η].

** DefTensor: Defining tetrametric Tetrametric[-α, -β, -γ, -η].

** DefTensor: Defining tetrametric Tetrametric†[-α, -β, -γ, -η].

** DefCovD: Defining covariant derivative CD[-α].

** DefTensor: Defining vanishing torsion tensor TorsionCD[α, -β, -γ].

** DefTensor: Defining symmetric Christoffel tensor ChristoffelCD[α, -β, -γ].

** DefTensor: Defining Riemann tensor RiemannCD[-α, -β, -γ, -η].

** DefTensor: Defining symmetric Ricci tensor RicciCD[-α, -β].

** DefCovD:  Contractions of Riemann automatically replaced by Ricci.

** DefTensor: Defining Ricci scalar RicciScalarCD[].

** DefCovD:  Contractions of Ricci automatically replaced by RicciScalar.

** DefTensor: Defining symmetric Einstein tensor EinsteinCD[-α, -β].

** DefTensor: Defining Weyl tensor WeylCD[-α, -β, -γ, -η].

** DefTensor: Defining symmetric TFRicci tensor TFRicciCD[-α, -β].

** DefTensor: Defining Kretschmann scalar KretschmannCD[].

** DefCovD:  Computing RiemannToWeylRules for dim 4

** DefCovD:  Computing RicciToTFRicci for dim 4

** DefCovD:  Computing RicciToEinsteinRules for dim 4

** DefTensor: Defining weight +2 density Detmetric[]. Determinant.
```

-1 is the sign of the metric determinant.

Define the coordinate chart

We want to work in a particular coordinate basis. We will work in the spherical polar coordinate basis. For this we need to define our chart. This is done using the function **DefChart**

In[5]:=
```
DefChart[cb, M4, {0, 1, 2, 3}, {t[], r[], θ[], φ[]}]
```

```
** DefChart: Defining chart cb.
** DefTensor: Defining coordinate scalar t[].
** DefTensor: Defining coordinate scalar r[].
** DefTensor: Defining coordinate scalar θ[].
** DefTensor: Defining coordinate scalar φ[].
** DefMapping: Defining mapping cb.
** DefMapping: Defining inverse mapping icb.
** DefTensor: Defining mapping differential tensor dicb[-ạ̊, icbα].
** DefTensor: Defining mapping differential tensor dcb[-α, cbạ̊].
** DefBasis: Defining basis cb. Coordinated basis.
** DefCovD: Defining parallel derivative PDcb[-α].
** DefTensor: Defining vanishing torsion tensor TorsionPDcb[α, -β, -γ].
** DefTensor: Defining symmetric Christoffel tensor ChristoffelPDcb[α, -β, -γ].
** DefTensor: Defining vanishing Riemann tensor RiemannPDcb[-α, -β, -γ, η].
** DefTensor: Defining vanishing Ricci tensor RicciPDcb[-α, -β].
** DefTensor: Defining antisymmetric +1 density etaUpcb[α, β, γ, η].
** DefTensor: Defining antisymmetric -1 density etaDowncb[-α, -β, -γ, -η].
```

The arguments are the name for our chart **cb**, the manifold **M4** on which we are defining this chart, indices for the coordinate and coordinates.

Define additional scalar functions

We need to define any scalar functions we will need later to describe our metric. For spherically symmetric we need to define the two scalar functions Λ and Φ.

In[6]:=
```
DefScalarFunction[Λ]
DefScalarFunction[Φ]
```

```
** DefScalarFunction: Defining scalar function Λ.
** DefScalarFunction: Defining scalar function Φ.
```

Define the metric in spherical coordinates

Now we define our metric components in the spherical polar coordinates. We can do this by first defining a matrix for the metric and the define a **CTensor** by giving the metric as input and finally setting the metric for the vector bundles of the basis **cb**

```
In[8]:= met = DiagonalMatrix[{-Exp[2 Φ[r[]]], Exp[2 Λ[r[]]], r[]^2, r[]^2 Sin[θ[]]^2}]
Out[8]= {{-e^(2 Φ[r]), 0, 0, 0}, {0, e^(2 Λ[r]), 0, 0}, {0, 0, r^2, 0}, {0, 0, 0, r^2 Sin[θ]^2}}
```

```
In[9]:= met // MatrixForm
Out[9]//MatrixForm=
```

$$\begin{pmatrix} -e^{2\Phi[r]} & 0 & 0 & 0 \\ 0 & e^{2\Lambda[r]} & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2\,\mathrm{Sin}[\theta]^2 \end{pmatrix}$$

```
In[10]:= g = CTensor[met, {-cb, -cb}]
Out[10]= CTensor[{{-e^(2 Φ[r]), 0, 0, 0}, {0, e^(2 Λ[r]), 0, 0},
           {0, 0, r^2, 0}, {0, 0, 0, r^2 Sin[θ]^2}}, {-cb, -cb}, 0]
```

```
In[11]:= SetCMetric[g, -cb]
Out[11]= SetCMetric[CTensor[{{-e^(2 Φ[r]), 0, 0, 0}, {0, e^(2 Λ[r]), 0, 0},
            {0, 0, r^2, 0}, {0, 0, 0, r^2 Sin[θ]^2}}, {-cb, -cb}, 0], -cb]
```

we can access the metric components in the following way

```
In[12]:= g[{0, -cb}, {0, -cb}]
Out[12]= -e^(2 Φ[r])
```

Covariant derivative associated to the metric

```
In[13]:= cd = CovDOfMetric[g]
Out[13]= CCovD[PDcb,
          CTensor[{{{0, Φ'[r], 0, 0}, {Φ'[r], 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}},
             {{e^(-2 Λ[r]+2 Φ[r]) Φ'[r], 0, 0, 0}, {0, Λ'[r], 0, 0},
              {0, 0, -e^(-2 Λ[r]) r, 0}, {0, 0, 0, -e^(-2 Λ[r]) r Sin[θ]^2}},
             {{0, 0, 0, 0}, {0, 0, 1/r, 0}, {0, 1/r, 0, 0}, {0, 0, 0, -Cos[θ] Sin[θ]}},
             {{0, 0, 0, 0}, {0, 0, 0, 1/r}, {0, 0, 0, Cot[θ]}, {0, 1/r, Cot[θ], 0}}},
           {cb, -cb, -cb}, 0], CTensor[{{-e^(2 Φ[r]), 0, 0, 0}, {0, e^(2 Λ[r]), 0, 0},
            {0, 0, r^2, 0}, {0, 0, 0, r^2 Sin[θ]^2}}, {-cb, -cb}, 0]]
```

```
? CovDOfMetric
```

CovDOfMetric[metric] gives the covariant derivative operator associated
to metric at definition time. CovDOfMetric[metric, torsion] gives the covariant
derivative operator associated to metric and having the given torsion tensor.
Specifying Zero torsion returns the Levi–Civita connection of the metric.

From this we can get the Riemann tensor, Ricci Tensor, Ricci Scalar and Einstein Tensor.

```
riemann = Riemann[cd]
```

```
CTensor[{{{{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}},
    {{0, 𝕖^(-2 Λ[r]+2 Φ[r]) (-Λ′[r] Φ′[r] + Φ′[r]^2 + Φ″[r]), 0, 0},
     {-Λ′[r] Φ′[r] + Φ′[r]^2 + Φ″[r], 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}},
    {{0, 0, (𝕖^(-2 Λ[r]+2 Φ[r]) Φ′[r])/r, 0}, {0, 0, 0, 0}, {𝕖^(-2 Λ[r]) r Φ′[r], 0, 0, 0},
     {0, 0, 0, 0}}, {{0, 0, 0, (𝕖^(-2 Λ[r]+2 Φ[r]) Φ′[r])/r}, {0, 0, 0, 0},
     {0, 0, 0, 0}, {𝕖^(-2 Λ[r]) r Sin[θ]^2 Φ′[r], 0, 0, 0}}},
   {{{0, 𝕖^(-2 Λ[r]+2 Φ[r]) (Λ′[r] Φ′[r] - Φ′[r]^2 - Φ″[r]), 0, 0},
     {Λ′[r] Φ′[r] - Φ′[r]^2 - Φ″[r], 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}},
    {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}},
    {{0, 0, 0, 0}, {0, 0, Λ′[r]/r, 0}, {0, -𝕖^(-2 Λ[r]) r Λ′[r], 0, 0}, {0, 0, 0, 0}},
    {{0, 0, 0, 0}, {0, 0, 0, Λ′[r]/r}, {0, 0, 0, 0},
     {0, -𝕖^(-2 Λ[r]) r Sin[θ]^2 Λ′[r], 0, 0}}}, {{{0, 0, -(𝕖^(-2 Λ[r]+2 Φ[r]) Φ′[r])/r, 0},
     {0, 0, 0, 0}, {-𝕖^(-2 Λ[r]) r Φ′[r], 0, 0, 0}, {0, 0, 0, 0}},
    {{0, 0, 0, 0}, {0, 0, -Λ′[r]/r, 0}, {0, 𝕖^(-2 Λ[r]) r Λ′[r], 0, 0}, {0, 0, 0, 0}},
    {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}}, {{0, 0, 0, 0},
     {0, 0, 0, 0}, {0, 0, 0, 1 - 𝕖^(-2 Λ[r])}, {0, 0, (-1 + 𝕖^(-2 Λ[r])) Sin[θ]^2, 0}}},
   {{{0, 0, 0, -(𝕖^(-2 Λ[r]+2 Φ[r]) Φ′[r])/r}, {0, 0, 0, 0}, {0, 0, 0, 0},
     {-𝕖^(-2 Λ[r]) r Sin[θ]^2 Φ′[r], 0, 0, 0}}, {{0, 0, 0, 0}, {0, 0, 0, -Λ′[r]/r},
     {0, 0, 0, 0}, {0, 𝕖^(-2 Λ[r]) r Sin[θ]^2 Λ′[r], 0, 0}}, {{0, 0, 0, 0},
     {0, 0, 0, 0}, {0, 0, 0, -1 + 𝕖^(-2 Λ[r])}, {0, 0, (1 - 𝕖^(-2 Λ[r])) Sin[θ]^2, 0}},
    {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}}}},
  {-cb, -cb, -cb, cb}, 0]
```

In[16]:=
```
ricci = Ricci[cd]
```

Out[16]=

$$\text{CTensor}\Big[\Big\{\Big\{\frac{1}{r}e^{-2\Lambda[r]+2\Phi[r]}\left(\left(2-r\,\Lambda'[r]\right)\Phi'[r]+r\,\Phi'[r]^2+r\,\Phi''[r]\right),0,0,0\Big\},\\ \Big\{0,\frac{\Lambda'[r]\left(2+r\,\Phi'[r]\right)-r\left(\Phi'[r]^2+\Phi''[r]\right)}{r},0,0\Big\},\\ \Big\{0,0,e^{-2\Lambda[r]}\left(-1+e^{2\Lambda[r]}+r\,\Lambda'[r]-r\,\Phi'[r]\right),0\Big\},\\ \Big\{0,0,0,e^{-2\Lambda[r]}\,\text{Sin}[\theta]^2\left(-1+e^{2\Lambda[r]}+r\,\Lambda'[r]-r\,\Phi'[r]\right)\Big\}\Big\},\{-cb,-cb\},0\Big]$$

In[17]:=
```
R = RicciScalar[cd]
```

Out[17]=

$$\text{CTensor}\Big[\frac{1}{r^2}\\ 2\,e^{-2\Lambda[r]}\left(-1+e^{2\Lambda[r]}-2\,r\,\Phi'[r]-r^2\,\Phi'[r]^2+r\,\Lambda'[r]\left(2+r\,\Phi'[r]\right)-r^2\,\Phi''[r]\right),\{\},0\Big]$$

Get the Einstein Tensor

For the Einstein equation we need the Einstein tensor. This is easy to obtain from the covariant derivative operator we obtained earlier.

In[18]:=
```
G = Einstein[cd]
```

Out[18]=

$$\text{CTensor}\Big[\\ \Big\{\Big\{\frac{e^{-2\Lambda[r]+2\Phi[r]}\left(-1+e^{2\Lambda[r]}+2\,r\,\Lambda'[r]\right)}{r^2},0,0,0\Big\},\Big\{0,\frac{1-e^{2\Lambda[r]}+2\,r\,\Phi'[r]}{r^2},0,0\Big\},\\ \Big\{0,0,e^{-2\Lambda[r]}\,r\left(\Phi'[r]+r\,\Phi'[r]^2-\Lambda'[r]\,(1+r\,\Phi'[r])+r\,\Phi''[r]\right),0\Big\},\Big\{0,0,0,\\ e^{-2\Lambda[r]}\,r\,\text{Sin}[\theta]^2\left(\Phi'[r]+r\,\Phi'[r]^2-\Lambda'[r]\,(1+r\,\Phi'[r])+r\,\Phi''[r]\right)\Big\}\Big\},\{-cb,-cb\},0\Big]$$

As earlier we can access the components of the Einstein tensor in the following way

In[19]:=
```
Gtt = G[{0, -cb}, {0, -cb}]
```

Out[19]=

$$\frac{e^{-2\Lambda[r]+2\Phi[r]}\left(-1+e^{2\Lambda[r]}+2\,r\,\Lambda'[r]\right)}{r^2}$$

In[20]:=
```
Grr = G[{1, -cb}, {1, -cb}]
```

Out[20]=

$$\frac{1-e^{2\Lambda[r]}+2\,r\,\Phi'[r]}{r^2}$$

In[21]:=
```
Gθθ = G[{2, -cb}, {2, -cb}]
```

Out[21]=

$$e^{-2\Lambda[r]}\,r\left(\Phi'[r]+r\,\Phi'[r]^2-\Lambda'[r]\,(1+r\,\Phi'[r])+r\,\Phi''[r]\right)$$

In[22]:=
```
Gϕϕ = G[{3, -cb}, {3, -cb}]
```

Out[22]=

$$e^{-2\Lambda[r]}\,r\,\text{Sin}[\theta]^2\left(\Phi'[r]+r\,\Phi'[r]^2-\Lambda'[r]\,(1+r\,\Phi'[r])+r\,\Phi''[r]\right)$$

Stress-Energy Tensor

Now that we have got the Einstein tensor, to set up the Einstein equation we need the energy momentum tensor. We need to define the pressure filed, the density field and the four-velocity

In[23]:=
```
DefScalarFunction[p]
DefScalarFunction[ρ]
```

```
** DefScalarFunction: Defining scalar function p.
** DefScalarFunction: Defining scalar function ρ.
```

In[25]:=
```
u = CTensor[{Exp[Φ[r[]]], 0, 0, 0}, {-cb}]
```

Out[25]= $\text{CTensor}\left[\left\{e^{\Phi[r]}, 0, 0, 0\right\}, \{-\text{cb}\}, 0\right]$

In[26]:=
```
T[α_, β_] := (p[r[]] + ρ[r[]]) u[α] u[β] + p[r[]] g[α, β]
```

In[27]:=
```
T[-α, -β] // Simplify
```

Out[27]=
$$\boxed{\begin{array}{cccc} e^{2\,\Phi[r]}\,\rho[r] & 0 & 0 & 0 \\ 0 & e^{2\,\Lambda[r]}\,p[r] & 0 & 0 \\ 0 & 0 & p[r]\,r^2 & 0 \\ 0 & 0 & 0 & p[r]\,r^2\,\text{Sin}[\theta]^2 \end{array}}_{\alpha\beta}$$

In[28]:=
```
T[{0, -cb}, {0, -cb}] // Simplify
```

Out[28]= $e^{2\,\Phi[r]}\,\rho[r]$

Setup Einstein Equation

We have the Einstein Tensor and the Stress-Energy Tensor. So we can define a function for the components of the Einstein Equation. We set G=1

In[29]:=
```
EinEq[α_, β_] := G[α, β] - 8 π T[α, β]
```

In[30]:=
```
EinEqtt = EinEq[{0, -cb}, {0, -cb}] // Simplify
```

Out[30]= $\frac{1}{r^2} e^{-2\,\Lambda[r]+2\,\Phi[r]} \left(-1 + e^{2\,\Lambda[r]} - 8\,e^{2\,\Lambda[r]}\,\pi\,r^2\,\rho[r] + 2\,r\,\Lambda'[r]\right)$

In[31]:=
```
EinEqrr = EinEq[{1, -cb}, {1, -cb}] // Simplify
```

Out[31]= $-8\,e^{2\,\Lambda[r]}\,\pi\,p[r] + \frac{1 - e^{2\,\Lambda[r]} + 2\,r\,\Phi'[r]}{r^2}$

In[32]:=
```
EinEqθθ = EinEq[{2, -cb}, {2, -cb}] // Simplify
```
Out[32]=
$$r\left(-8\pi p[r]\,r + e^{-2\Lambda[r]}\left(\Phi'[r] + r\,\Phi'[r]^2 - \Lambda'[r]\,(1 + r\,\Phi'[r]) + r\,\Phi''[r]\right)\right)$$

In[33]:=
```
EinEqϕϕ = EinEq[{3, -cb}, {3, -cb}] // Simplify
```
Out[33]=
$$r\,\mathrm{Sin}[\theta]^2\left(-8\pi p[r]\,r + e^{-2\Lambda[r]}\left(\Phi'[r] + r\,\Phi'[r]^2 - \Lambda'[r]\,(1 + r\,\Phi'[r]) + r\,\Phi''[r]\right)\right)$$

TOV Equations

Get Λ’ from tt component

In[34]:=
```
Solve[EinEqtt == 0, D[Λ[r[]], r[]]]
```
Out[34]=
$$\left\{\left\{\Lambda'[r] \to \frac{1 - e^{2\Lambda[r]} + 8\,e^{2\Lambda[r]}\,\pi\,r^2\,\rho[r]}{2\,r}\right\}\right\}$$

In[35]:=
```
DefScalarFunction[m]
```
$$\text{Lambdaval} = \frac{1}{2}\mathrm{Log}\left[\frac{1}{1 - 2\,\frac{m[r[]]}{r[]}}\right]$$

```
** DefScalarFunction: Defining scalar function m.
```
Out[36]=
$$\frac{1}{2}\mathrm{Log}\left[\frac{1}{1 - \frac{2\,m[r]}{r}}\right]$$

In[37]:=
```
Lambdadash =
  (Solve[EinEqtt == 0, D[Λ[r[]], r[]]][[1]] /. {Λ[r[]] → Lambdaval}) // Values
```
Out[37]=
$$\left\{\frac{1 - \frac{1}{1-\frac{2\,m[r]}{r}} + \frac{8\pi r^2 \rho[r]}{1-\frac{2\,m[r]}{r}}}{2\,r}\right\}$$

TOV Equation 1 from rr component

From the rr component we get. First we replace Λ in rr component.

In[38]:=
```
EinEqrr2 = EinEqrr /. {Λ[r[]] → Lambdaval} // Simplify
```
Out[38]=
$$\frac{2\left(m[r] + 4\pi\,p[r]\,r^3 + 2\,m[r]\,r\,\Phi'[r] - r^2\,\Phi'[r]\right)}{\left(2\,m[r] - r\right)\,r^2}$$

9]:=
```
TOVEq1 = Solve[EinEqrr2 == 0, D[Φ[r[]], r[]]] // Simplify
```
9]=
$$\left\{\left\{\Phi'[r] \to -\frac{m[r] + 4\pi p[r]\, r^3}{2\, m[r]\, r - r^2}\right\}\right\}$$

TOV Equation 2 from

First we replace Λ, Λ’, Φ’ and Φ’’ in the $\theta\theta$ component

0]:=
```
Phidash = TOVEq1[[1]] // Values
```
0]=
$$\left\{-\frac{m[r] + 4\pi p[r]\, r^3}{2\, m[r]\, r - r^2}\right\}$$

1]:=
```
Phiddash = D[Phidash, r[]] // Simplify
```
1]=
$$\left\{\frac{1}{r^2\,(-2\, m[r] + r)^2}\left(2\, m[r]^2 + r^2\, m'[r] + 4\pi p[r]\, r^4\,(1 + 2\, m'[r]) + 4\pi r^5\, p'[r] - 2\, m[r]\,\left(r + 8\pi p[r]\, r^3 + 4\pi r^4\, p'[r]\right)\right)\right\}$$

2]:=
```
EinEqθθ2 = EinEqθθ /. { Λ[r[]] → Lambdaval, D[Λ[r[]], r[]] → Lambdadash,
    D[Φ[r[]], r[]] → Phidash, D[D[Φ[r[]], r[]], r[]] → Phiddash} // Simplify
```
2]=
$$\left\{\frac{1}{-2\, m[r] + r} r\left(16\pi^2 p[r]^2\, r^4 - 4\pi r^2 \rho[r] + m'[r] + 8\pi p[r]\, r^2\,\left(-2\pi r^2 \rho[r] + m'[r]\right) + 4\pi r^3\, p'[r] + 4\pi\, m[r]\, r\,\left(p[r] + \rho[r] - 2\, r\, p'[r]\right)\right)\right\}$$

We replace m’ using

3]:=
```
mdash = 4 π r[]^2 ρ[r[]]
```
3]=
$$4\pi r^2 \rho[r]$$

4]:=
```
EinEqθθ3 = EinEqθθ2 /. {D[m[r[]], r[]] → mdash} // Simplify
```
4]=
$$\left\{\frac{4\pi r^2\,\left(r^2\,\left(4\pi p[r]^2\, r + 4\pi p[r]\, r\, \rho[r] + p'[r]\right) + m[r]\,\left(p[r] + \rho[r] - 2\, r\, p'[r]\right)\right)}{-2\, m[r] + r}\right\}$$

5]:=
```
TOVEq2 =
 Solve[EinEqθθ3 == 0, D[p[r[]], r[]]] // FullSimplify // Values // Flatten
```
5]=
$$\left\{\frac{\left(m[r] + 4\pi p[r]\, r^3\right)\,\left(p[r] + \rho[r]\right)}{\left(2\, m[r] - r\right)\, r}\right\}$$