Tolman-Oppenheimer-Volkoff equations

This Jupyter/SageMath worksheet is relative to the lectures General relativity computations with SageManifolds given at the NewCompStar School 2016 (Coimbra, Portugal).

These computations are based on SageManifolds (version 1.0, as included in SageMath 7.5).

Click here to download the worksheet file (ipynb format). To run it, you must start SageMath with the Jupyter notebook, with the command `sage -n jupyter`

This worksheet is divided in two parts:

1. Deriving the TOV system from the Einstein equation
2. Solving the TOV system to get stellar models

NB: a version of SageMath at least equal to 7.5 is required to run this worksheet:

In [1]:
```
version()
```

Out[1]:
```
'SageMath version 7.5.1, Release Date: 2017-01-15'
```

First we set up the notebook to display mathematical objects using LaTeX rendering:

In [2]:
```
%display latex
```

1. Deriving the TOV system from the Einstein equation

Spacetime

We declare the spacetime manifold M:

In [3]:
```
M = Manifold(4, 'M')
print(M)
```

```
4-dimensional differentiable manifold M
```

To get some information about the object `M`, we use the question mark:

In [4]:
```
M?
```

Using a double question mark, we get the Python source code (SageMath is **open source**, isn't it?):

In [5]:
```
M??
```

We declare the chart of spherical coordinates (t, r, θ, ϕ), via the method `chart` acting on `M`; to see how to use it, we again use the question mark:

In [6]:
```
M.chart?
```

In [7]:
```
X.<t,r,th,ph> = M.chart(r't r:(0,+oo) th:(0,pi):\theta ph:(0,2*pi):\phi
')
X
```

Out[7]: $(M, (t, r, \theta, \phi))$

Metric tensor

The static and spherically symmetric metric ansatz, with the unknown functions $\nu(r)$ and $m(r)$:

In [8]:
```
g = M.lorentzian_metric('g')
nu = function('nu')
m = function('m')
g[0,0] = -exp(2*nu(r))
g[1,1] = 1/(1-2*m(r)/r)
g[2,2] = r^2
g[3,3] = (r*sin(th))^2
g.display()
```

Out[8]:
$$g = -e^{(2\,\nu(r))}\mathrm{d}t\otimes\mathrm{d}t + \left(-\frac{1}{\frac{2\,m(r)}{r}-1}\right)\mathrm{d}r\otimes\mathrm{d}r + r^2\mathrm{d}\theta\otimes\mathrm{d}\theta + r^2\sin(\theta)^2\mathrm{d}\phi\otimes\mathrm{d}\phi$$

One can display the metric components as a list:

In [9]:
```
g.display_comp()
```

Out[9]:
$$\begin{aligned} g_{t\,t} &= -e^{(2\,\nu(r))} \\ g_{r\,r} &= -\frac{1}{\frac{2\,m(r)}{r}-1} \\ g_{\theta\,\theta} &= r^2 \\ g_{\phi\,\phi} &= r^2\sin(\theta)^2 \end{aligned}$$

By default, only the nonzero components are shown; to get all the components, set the option `only_nonzero` to `False`:

In [10]:
```
g.display_comp(only_nonzero=False)
```

Out[10]:

$$
\begin{aligned}
g_{t\,t} &= -e^{(2\,\nu(r))} \\
g_{t\,r} &= 0 \\
g_{t\,\theta} &= 0 \\
g_{t\,\phi} &= 0 \\
g_{r\,t} &= 0 \\
g_{r\,r} &= -\frac{1}{\frac{2\,m(r)}{r}-1} \\
g_{r\,\theta} &= 0 \\
g_{r\,\phi} &= 0 \\
g_{\theta\,t} &= 0 \\
g_{\theta\,r} &= 0 \\
g_{\theta\,\theta} &= r^2 \\
g_{\theta\,\phi} &= 0 \\
g_{\phi\,t} &= 0 \\
g_{\phi\,r} &= 0 \\
g_{\phi\,\theta} &= 0 \\
g_{\phi\,\phi} &= r^2 \sin(\theta)^2
\end{aligned}
$$

We can also display the metric components as a matrix, via the [ ] operator:

In [11]:
```
g[:]
```

Out[11]:

$$
\begin{pmatrix}
-e^{(2\,\nu(r))} & 0 & 0 & 0 \\
0 & -\frac{1}{\frac{2\,m(r)}{r}-1} & 0 & 0 \\
0 & 0 & r^2 & 0 \\
0 & 0 & 0 & r^2 \sin(\theta)^2
\end{pmatrix}
$$

The [ ] operator can also be used to access to individual elements:

In [12]:
```
g[0,0]
```

Out[12]:

$$-e^{(2\,\nu(r))}$$

Einstein equation

Let us start by evaluating the Ricci tensor of g:

In [13]:
```
Ric = g.ricci()
print(Ric)
```

```
Field of symmetric bilinear forms Ric(g) on the 4-dimensional different
iable manifold M
```

In [14]:
```
Ric.display()
```

Out[14]:

$$\mathrm{Ric}\,(g) = \left(\frac{\left(r^2 e^{(2\,\nu(r))} - 2\,r e^{(2\,\nu(r))} m\,(r)\right)\left(\frac{\partial\,\nu}{\partial r}\right)^2 - \left(r e^{(2\,\nu(r))}\frac{\partial\,m}{\partial r} - 2\,r e^{(2\,\nu(r))} + 3\,e^{(2\,\nu(r))} m\,(r)\right)\frac{\partial\,\nu}{\partial r} + \left(r^2 e^{(2\,\nu(r))} - 2\,r e^{(2\,\nu(r))} m\,(r)\right)\frac{\partial^2\,\nu}{\partial r^2}}{r^2} \right) \mathrm{d}t \otimes \mathrm{d}t$$

$$+ \left(-\frac{\left(r^3 - 2\,r^2 m\,(r)\right)\left(\frac{\partial\,\nu}{\partial r}\right)^2 - 2\,r\frac{\partial\,m}{\partial r} - \left(r^2\frac{\partial\,m}{\partial r} - r m\,(r)\right)\frac{\partial\,\nu}{\partial r} + \left(r^3 - 2\,r^2 m\,(r)\right)\frac{\partial^2\,\nu}{\partial r^2} + 2\,m\,(r)}{r^3 - 2\,r^2 m\,(r)} \right)$$

$$\otimes \mathrm{d}r + \left(\frac{r\frac{\partial\,m}{\partial r} - \left(r^2 - 2\,r m\,(r)\right)\frac{\partial\,\nu}{\partial r} + m\,(r)}{r} \right) \mathrm{d}\theta \otimes \mathrm{d}\theta$$

$$+ \frac{\left(r\frac{\partial\,m}{\partial r} - \left(r^2 - 2\,r m\,(r)\right)\frac{\partial\,\nu}{\partial r} + m\,(r)\right)\sin(\theta)^2}{r} \mathrm{d}\phi \otimes \mathrm{d}\phi$$

The Ricci scalar is naturally obtained by the method `ricci_scalar()`:

In [15]:
```
g.ricci_scalar?
```

In [16]:
```
g.ricci_scalar().display()
```

Out[16]:

$$\mathrm{r}\,(g):\ M \longrightarrow \mathbb{R}$$

$$(t, r, \theta, \phi) \longmapsto -\frac{2\left(\left(r^2 - 2\,r m(r)\right)\left(\frac{\partial\,\nu}{\partial r}\right)^2 - \left(r\frac{\partial\,m}{\partial r} - 2\,r + 3\,m(r)\right)\frac{\partial\,\nu}{\partial r} + \left(r^2 - 2\,r m(r)\right)\frac{\partial^2\,\nu}{\partial r^2} - 2\,\frac{\partial\,m}{\partial r}\right)}{r^2}$$

As a check we can also compute it by taking the trace of the Ricci tensor with respect to g:

In [17]:
```
g.ricci_scalar() == g.inverse()['^{ab}']*Ric['_{ab}']
```

Out[17]: True

The Einstein tensor

$$G_{ab} := R_{ab} - \frac{1}{2} R\, g_{ab},$$

or in index-free notation:

$$G := \mathrm{Ric}(g) - \frac{1}{2} r(g)\, g$$

In [18]:
```
G = Ric - 1/2*g.ricci_scalar() * g
G.set_name('G')
print(G)
```

```
Field of symmetric bilinear forms G on the 4-dimensional differentiable
 manifold M
```

In [19]:
```
G.display()
```

Out[19]:

$$G = \frac{2\,e^{(2\,\nu(r))}\,\frac{\partial\,m}{\partial r}}{r^2}\mathrm{d}t\otimes\mathrm{d}t + \left(\frac{2\left(\left(r^2-2\,rm\left(r\right)\right)\frac{\partial\,\nu}{\partial r}-m\left(r\right)\right)}{r^3-2\,r^2m\left(r\right)}\right)\mathrm{d}r\otimes\mathrm{d}r + \left(\frac{\left(r^3-2\,r^2m\left(r\right)\right)\left(\frac{\partial\,\nu}{\partial r}\right)^2-r\frac{\partial\,m}{\partial r}-\left(r^2\frac{\partial\,m}{\partial r}-r^2+rm\left(r\right)\right)\frac{\partial\,\nu}{\partial r}+\left(r^3-2\,r^2m\left(r\right)\right)\frac{\partial^2\,\nu}{\partial r^2}+m\left(r\right)}{r}\right)\mathrm{d}\theta\otimes\mathrm{d}\theta + \left(\left(r^3-2\,r^2m\left(r\right)\right)\left(\frac{\partial\,\nu}{\partial r}\right)^2-r\frac{\partial\,m}{\partial r}-\left(r^2\frac{\partial\,m}{\partial r}-r^2+rm\left(r\right)\right)\frac{\partial\,\nu}{\partial r}\sin\left(\theta\right)^2+\left(r^3-2\,r^2m\left(r\right)\right)\frac{\partial^2\,\nu}{\partial r^2}+m\left(r\right)\right)}{r}\mathrm{d}\phi\otimes\mathrm{d}\phi$$

The energy-momentum tensor

We consider a perfect fluid matter model. Let us first defined the fluid 4-velocity u:

In [20]:
```
u = M.vector_field('u')
u[0] = exp(-nu(r))
u.display()
```

Out[20]:

$$u = e^{(-\nu(r))}\,\frac{\partial}{\partial t}$$

In [21]:
```
u[:]
```

Out[21]:

$$\left[e^{(-\nu(r))}, 0, 0, 0\right]$$

In [22]:
```
print(u.parent())
```

```
Free module X(M) of vector fields on the 4-dimensional differentiable m
anifold M
```

Let us check that u is a normalized timelike vector, i.e. that $g_{ab}u^a u^b = -1$, or, in index-free notation, $g(u,u) = -1$:

In [23]:
```
g(u,u)
```

Out[23]:

$$g\left(u, u\right)$$

In [24]:
```
print(g(u,u))
```

```
Scalar field g(u,u) on the 4-dimensional differentiable manifold M
```

In [25]:
```
g(u,u).display()
```

Out[25]:

$$g\left(u, u\right): \begin{array}{ccc} M & \longrightarrow & \mathbb{R} \\ (t, r, \theta, \phi) & \longmapsto & -1 \end{array}$$

In [26]:
```
g(u,u).parent()
```

Out[26]:

$$C^{\infty}\left(M\right)$$

In [27]:
```
print(g(u,u).parent())
```

```
Algebra of differentiable scalar fields on the 4-dimensional differenti
able manifold M
```

In [28]:
```
g(u,u) == -1
```

Out[28]: True

To form the energy-momentum tensor, we need the 1-form $\underline{u}$ that is metric-dual to the vector u, i.e. $u_a = g_{ab}u^b$:

In [29]:
```
u_form = u.down(g)
print(u_form)
```

```
1-form on the 4-dimensional differentiable manifold M
```

In [30]:
```
u_form.display()
```

Out[30]: $-e^{\nu(r)}\mathrm{d}t$

The energy-momentum tensor is then

$$T_{ab} = (\rho + p)u_a u_b + p\, g_{ab},$$

or in index-free notation:

$$T = (\rho + p)\underline{u} \otimes \underline{u} + p\, g$$

Since the tensor product $\otimes$ is taken with the * operator, we write:

In [31]:
```
rho = function('rho')
p = function('p')
T = (rho(r)+p(r))* (u_form * u_form) + p(r) * g
T.set_name('T')
print(T)
```

```
Field of symmetric bilinear forms T on the 4-dimensional differentiable
 manifold M
```

In [32]:
```
T.display()
```

Out[32]:
$$T = e^{(2\,\nu(r))}\rho\,(r)\,\mathrm{d}t \otimes \mathrm{d}t + \left(\frac{rp\,(r)}{r - 2\,m\,(r)}\right)\mathrm{d}r \otimes \mathrm{d}r + r^2 p\,(r)\,\mathrm{d}\theta \otimes \mathrm{d}\theta + r^2 p\,(r)\sin(\theta)^2\mathrm{d}\phi \otimes \mathrm{d}\phi$$

In [33]:
```
T(u,u)
```

Out[33]: $T\,(u, u)$

In [34]:
```
print(T(u,u))
```

```
Scalar field T(u,u) on the 4-dimensional differentiable manifold M
```

In [35]:
```
T(u,u).display()
```

Out[35]:
$$\begin{array}{rccc} T\,(u,u): & M & \longrightarrow & \mathbb{R} \\ & (t, r, \theta, \phi) & \longmapsto & \rho\,(r) \end{array}$$

The Einstein equation

The Einstein equation is

$$G = 8\pi T$$

We rewrite it as $E = 0$ with $E := G - 8\pi T$:

In [36]:
```
E = G - 8*pi*T
E.set_name('E')
print(E)
```

```
Field of symmetric bilinear forms E on the 4-dimensional differentiable
 manifold M
```

In [37]:
```
E.display()
```

Out[37]:

$$E = -\frac{2\left(4\,\pi r^2 e^{(2\,\nu(r))}\rho\left(r\right) - e^{(2\,\nu(r))}\frac{\partial\,m}{\partial r}\right)}{r^2}\mathrm{d}t\otimes\mathrm{d}t + \left(-\frac{2\left(4\,\pi r^3 p\left(r\right) - \left(r^2 - 2\,rm\left(r\right)\right)\frac{\partial\,\nu}{\partial r} + m\left(r\right)\right)}{r^3 - 2\,r^2 m\left(r\right)}\right)\mathrm{d}r\otimes\mathrm{d}r + \left(-\frac{\begin{array}{l}8\,\pi r^3 p\left(r\right) - \left(r^3 - 2\,r^2 m\left(r\right)\right)\left(\frac{\partial\,\nu}{\partial r}\right)^2 + r\frac{\partial\,m}{\partial r} + \left(r^2\frac{\partial\,m}{\partial r} - r^2 + rm\left(r\right)\right)\frac{\partial\,\nu}{\partial r}\\ -\left(r^3 - 2\,r^2 m\left(r\right)\right)\frac{\partial^2\,\nu}{\partial r^2} - m\left(r\right)\end{array}}{r}\right)\mathrm{d}\theta\otimes\mathrm{d}\theta + \left(8\,\pi r^3 p\left(r\right) - \left(r^3 - 2\,r^2 m\left(r\right)\right)\left(\frac{\partial\,\nu}{\partial r}\right)^2 + r\frac{\partial\,m}{\partial r} + \left(r^2\frac{\partial\,m}{\partial r} - r^2 + rm\left(r\right)\right)\frac{\partial\,\nu}{\partial r}\right)\sin\left(\theta\right)^2 - \frac{\left(r^3 - 2\,r^2 m\left(r\right)\right)\frac{\partial^2\,\nu}{\partial r^2} - m\left(r\right)}{r}\mathrm{d}\phi\otimes\mathrm{d}\phi$$

In [38]:
```
E.display_comp()
```

Out[38]:

$$\begin{aligned}
E_{t\,t} &= -\frac{2\left(4\,\pi r^2 e^{(2\,\nu(r))}\rho(r) - e^{(2\,\nu(r))}\frac{\partial\,m}{\partial r}\right)}{r^2}\\
E_{r\,r} &= -\frac{2\left(4\,\pi r^3 p(r) - \left(r^2 - 2\,rm(r)\right)\frac{\partial\,\nu}{\partial r} + m(r)\right)}{r^3 - 2\,r^2 m(r)}\\
E_{\theta\,\theta} &= -\frac{8\,\pi r^3 p(r) - \left(r^3 - 2\,r^2 m(r)\right)\left(\frac{\partial\,\nu}{\partial r}\right)^2 + r\frac{\partial\,m}{\partial r} + \left(r^2\frac{\partial\,m}{\partial r} - r^2 + rm(r)\right)\frac{\partial\,\nu}{\partial r} - \left(r^3 - 2\,r^2 m(r)\right)\frac{\partial^2\,\nu}{\partial r^2} - m(r)}{r}\\
E_{\phi\,\phi} &= -\frac{\left(8\,\pi r^3 p(r) - \left(r^3 - 2\,r^2 m(r)\right)\left(\frac{\partial\,\nu}{\partial r}\right)^2 + r\frac{\partial\,m}{\partial r} + \left(r^2\frac{\partial\,m}{\partial r} - r^2 + rm(r)\right)\frac{\partial\,\nu}{\partial r} - \left(r^3 - 2\,r^2 m(r)\right)\frac{\partial^2\,\nu}{\partial r^2} - m(r)\right)\sin(}{r}
\end{aligned}$$

In [39]:
```
E[0,0]
```

Out[39]:

$$-\frac{2\left(4\,\pi r^2 e^{(2\,\nu(r))}\rho\left(r\right) - e^{(2\,\nu(r))}\frac{\partial\,m}{\partial r}\right)}{r^2}$$

In [40]:
```
EE0_sol = solve(E[0,0].expr()==0, diff(m(r),r))
EE0_sol
```

Out[40]:

$$\left[\frac{\partial}{\partial r}m\left(r\right) = 4\,\pi r^2\rho\left(r\right)\right]$$

In [41]:
```
EE0 = EE0_sol[0]
EE0
```

Out[41]:
$$\frac{\partial}{\partial r} m\left(r\right) = 4\,\pi r^{2} \rho\left(r\right)$$

In [42]:
```
E[1,1]
```

Out[42]:
$$-\frac{2\left(4\,\pi r^{3} p\left(r\right) - \left(r^{2} - 2\,rm\left(r\right)\right)\frac{\partial\,\nu}{\partial r} + m\left(r\right)\right)}{r^{3} - 2\,r^{2} m\left(r\right)}$$

In [43]:
```
EE1_sol = solve(E[1,1].expr()==0, diff(nu(r),r))
EE1 = EE1_sol[0]
EE1
```

Out[43]:
$$\frac{\partial}{\partial r}\nu\left(r\right) = \frac{4\,\pi r^{3} p\left(r\right) + m\left(r\right)}{r^{2} - 2\,rm\left(r\right)}$$

In [44]:
```
E[3,3] == E[2,2]*sin(th)^2
```

Out[44]:
True

In [45]:
```
E[2,2]
```

Out[45]:
$$-\frac{8\,\pi r^{3} p\left(r\right) - \left(r^{3} - 2\,r^{2} m\left(r\right)\right)\left(\frac{\partial\,\nu}{\partial r}\right)^{2} + r\frac{\partial\,m}{\partial r} + \left(r^{2}\frac{\partial\,m}{\partial r} - r^{2} + rm\left(r\right)\right)\frac{\partial\,\nu}{\partial r} - \left(r^{3} - 2\,r^{2} m\left(r\right)\right)\frac{\partial^{2}\,\nu}{\partial r^{2}} - m\left(r\right)}{r}$$

The energy-momentum conservation equation

The energy-momentum tensor must obey

$$\nabla_b T^b{}_a = 0$$

We first form the tensor $T^b{}_a$ by raising the first index of T_{ab}:

In [46]:
```
Tu = T.up(g, 0)
print(Tu)
```

```
Tensor field of type (1,1) on the 4-dimensional differentiable manifold
 M
```

We get the Levi-Civita connection ∇ associated with the metric g:

In [47]:
```
nabla = g.connection()
print(nabla)
```

```
Levi-Civita connection nabla_g associated with the Lorentzian metric g
on the 4-dimensional differentiable manifold M
```

In [48]:
```
nabla.display()
```

Out[48]:

$$\Gamma^{t}{}_{t\,r} = \frac{\partial\,\nu}{\partial r}$$
$$\Gamma^{t}{}_{r\,t} = \frac{\partial\,\nu}{\partial r}$$
$$\Gamma^{r}{}_{t\,t} = \frac{\left(re^{(2\,\nu(r))} - 2\,e^{(2\,\nu(r))}m(r)\right)\frac{\partial\,\nu}{\partial r}}{r}$$
$$\Gamma^{r}{}_{r\,r} = \frac{r\frac{\partial\,m}{\partial r} - m(r)}{r^2 - 2\,rm(r)}$$
$$\Gamma^{r}{}_{\theta\,\theta} = -r + 2\,m\,(r)$$
$$\Gamma^{r}{}_{\phi\,\phi} = -(r - 2\,m\,(r))\sin\,(\theta)^2$$
$$\Gamma^{\theta}{}_{r\,\theta} = \frac{1}{r}$$
$$\Gamma^{\theta}{}_{\theta\,r} = \frac{1}{r}$$
$$\Gamma^{\theta}{}_{\phi\,\phi} = -\cos(\theta)\sin(\theta)$$
$$\Gamma^{\phi}{}_{r\,\phi} = \frac{1}{r}$$
$$\Gamma^{\phi}{}_{\theta\,\phi} = \frac{\cos(\theta)}{\sin(\theta)}$$
$$\Gamma^{\phi}{}_{\phi\,r} = \frac{1}{r}$$
$$\Gamma^{\phi}{}_{\phi\,\theta} = \frac{\cos(\theta)}{\sin(\theta)}$$

We apply `nabla` to `Tu` to get the tensor $(\nabla T)^{b}{}_{ac} = \nabla_c T^{b}{}_{a}$ (MTW index convention):

In [49]:
```
dTu = nabla(Tu)
print(dTu)
```

```
Tensor field of type (1,2) on the 4-dimensional differentiable manifold
 M
```

The divergence $\nabla_b T^{b}{}_{a}$ is then computed as the trace of the tensor $(\nabla T)^{b}{}_{ac}$ on the first index (b, position `0`) and last index (c, position 2):

In [50]:
```
divT = dTu.trace(0,2)
print(divT)
```

```
1-form on the 4-dimensional differentiable manifold M
```

We can also take the trace by using the index notation:

In [51]:
```
divT == dTu['^b_{ab}']
```

Out[51]: True

In [52]:
```
print(divT)
```

```
1-form on the 4-dimensional differentiable manifold M
```

In [53]:
```
divT.display()
```

Out[53]:

$$\left((p\,(r) + \rho\,(r))\frac{\partial\,\nu}{\partial r} + \frac{\partial\,p}{\partial r}\right)\mathrm{d}r$$

In [54]:
```
divT[:]
```

Out[54]:
$$\left[0, (p\,(r) + \rho\,(r))\frac{\partial\,\nu}{\partial r} + \frac{\partial\,p}{\partial r}, 0, 0\right]$$

The only non trivially vanishing components is thus

In [55]:
```
divT[1]
```

Out[55]:
$$(p\,(r) + \rho\,(r))\frac{\partial\,\nu}{\partial r} + \frac{\partial\,p}{\partial r}$$

Hence the energy-momentum conservation equation $\nabla_b T^b{}_a = 0$ reduces to

In [56]:
```
EE2_sol = solve(divT[1].expr()==0, diff(p(r),r))
EE2 = EE2_sol[0]
EE2
```

Out[56]:
$$\frac{\partial}{\partial r}p\,(r) = -(p\,(r) + \rho\,(r))\frac{\partial}{\partial r}\nu\,(r)$$

The TOV system

Let us collect all the independent equations obtained so far:

In [57]:
```
for eq in [EE0, EE1, EE2]:
    show(eq)
```

$$\frac{\partial}{\partial r}m\,(r) = 4\,\pi r^2\rho\,(r)$$

$$\frac{\partial}{\partial r}\nu\,(r) = \frac{4\,\pi r^3 p\,(r) + m\,(r)}{r^2 - 2\,rm\,(r)}$$

$$\frac{\partial}{\partial r}p\,(r) = -(p\,(r) + \rho\,(r))\frac{\partial}{\partial r}\nu\,(r)$$

2. Solving the TOV system

In order to solve the TOV system, we need to specify a fluid equation of state. For simplicity, we select a polytropic one:

$$p = k\rho^2$$

In [58]:
```
var('k', domain='real')
p_eos(r) = k*rho(r)^2
p_eos(r)
```

Out[58]:
$$k\rho(r)^2$$

We substitute this expression for p in the TOV equations:

In [59]:
```
EE1_rho = EE1.substitute_function(p, p_eos)
EE1_rho
```

Out[59]:
$$\frac{\partial}{\partial r}\nu\,(r) = \frac{4\,\pi k r^3\rho(r)^2 + m\,(r)}{r^2 - 2\,rm\,(r)}$$

In [60]:
```
EE2_rho = EE2.substitute_function(p, p_eos)
EE2_rho
```

Out[60]:
$$2\,k\rho\left(r\right)\frac{\partial}{\partial r}\rho\left(r\right) = -\left(k\rho(r)^2 + \rho\left(r\right)\right)\frac{\partial}{\partial r}\nu\left(r\right)$$

In [61]:
```
EE2_rho = (EE2_rho / (2*k*rho(r))).simplify_full()
EE2_rho
```

Out[61]:
$$\frac{\partial}{\partial r}\rho\left(r\right) = -\frac{\left(k\rho\left(r\right)+1\right)\frac{\partial}{\partial r}\nu\left(r\right)}{2\,k}$$

We subsitute the expression of equation `EE1_rho` for $\partial\nu/\partial r$, in order to get rid of any derivative in the right-hand sides:

In [62]:
```
EE2_rho = EE2_rho.subs({diff(nu(r),r): EE1_rho.rhs()}).simplify_full()
EE2_rho
```

Out[62]:
$$\frac{\partial}{\partial r}\rho\left(r\right) = -\frac{4\,\pi k^2 r^3\rho(r)^3 + 4\,\pi k r^3\rho(r)^2 + k m\left(r\right)\rho\left(r\right) + m\left(r\right)}{2\left(kr^2 - 2\,krm\left(r\right)\right)}$$

The system to solve for $(m(r), \nu(r), \rho(r))$ is thus

In [63]:
```
for eq in [EE0, EE1_rho, EE2_rho]:
    show(eq)
```

$$\frac{\partial}{\partial r}m\left(r\right) = 4\,\pi r^2\rho\left(r\right)$$

$$\frac{\partial}{\partial r}\nu\left(r\right) = \frac{4\,\pi k r^3\rho(r)^2 + m\left(r\right)}{r^2 - 2\,rm\left(r\right)}$$

$$\frac{\partial}{\partial r}\rho\left(r\right) = -\frac{4\,\pi k^2 r^3\rho(r)^3 + 4\,\pi k r^3\rho(r)^2 + k m\left(r\right)\rho\left(r\right) + m\left(r\right)}{2\left(kr^2 - 2\,krm\left(r\right)\right)}$$

Numerical resolution

Let us use a standard 4th-order Runge-Kutta method:

In [64]:
```
desolve_system_rk4?
```

We gather all equations in a list for the ease of manipulation:

In [65]:
```
eqs = [EE0, EE1_rho, EE2_rho]
```

To get a numerical solution, we have of course to specify some numerical value for the EOS constant k; let us choose $k = 1/4$:

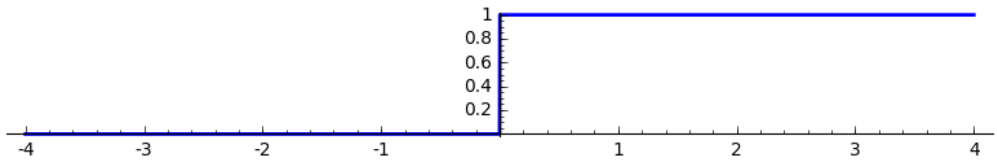
In [66]:
```
k0 = 1/4
rhs = [eq.rhs().subs(k=k0) for eq in eqs]
rhs
```

Out[66]:
$$\left[4\,\pi r^2\rho\left(r\right), \frac{\pi r^3\rho(r)^2 + m\left(r\right)}{r^2 - 2\,rm\left(r\right)}, -\frac{\pi r^3\rho(r)^3 + 4\,\pi r^3\rho(r)^2 + m\left(r\right)\rho\left(r\right) + 4\,m\left(r\right)}{2\left(r^2 - 2\,rm\left(r\right)\right)}\right]$$

The integration for $m(r)$ and $\rho(r)$ has to stop as soon as $\rho(r)$ become negative. An easy way to ensure this is to use the Heaviside function (actually SageMath's `unit_step`; for some reason, `heaviside` does not work for the RK4 numerical resolution):

In [67]:
```
plot(unit_step(x), (x,-4,4), thickness=2, aspect_ratio=1)
```

Out[67]:



and to multiply the r.h.s. of the dm/dr and $d\rho/dr$ equations by $h(\rho)$:

In [68]:
```
rhs[0] = rhs[0] * unit_step(rho(r))
rhs[2] = rhs[2] * unit_step(rho(r))
rhs
```

Out[68]:

$$\left[4\,\pi r^2\rho\left(r\right)\mathrm{u}\left(\rho\left(r\right)\right), \frac{\pi r^3\rho(r)^2+m\left(r\right)}{r^2-2\,rm\left(r\right)}, -\frac{\left(\pi r^3\rho(r)^3+4\,\pi r^3\rho(r)^2+m\left(r\right)\rho\left(r\right)+4\,m\left(r\right)\right)\mathrm{u}\left(\rho\left(r\right)\right)}{2\left(r^2-2\,rm\left(r\right)\right)}\right]$$

Let us add an extra equation, for the purpose of getting the star's radius as an output of the integration, via the equation $dR/dr = 1$, again multiplied by the Heaviside function of ρ:

In [69]:
```
rhs.append(1 * unit_step(rho(r)))
rhs
```

Out[69]:

$$\left[4\,\pi r^2\rho\left(r\right)\mathrm{u}\left(\rho\left(r\right)\right), \frac{\pi r^3\rho(r)^2+m\left(r\right)}{r^2-2\,rm\left(r\right)}, -\frac{\left(\pi r^3\rho(r)^3+4\,\pi r^3\rho(r)^2+m\left(r\right)\rho\left(r\right)+4\,m\left(r\right)\right)\mathrm{u}\left(\rho\left(r\right)\right)}{2\left(r^2-2\,rm\left(r\right)\right)}, \mathrm{u}\left(\rho\left(r\right)\right)\right]$$

For the purpose of the numerical integration via `desolve_system_rk4`, we have to replace the symbolic functions $m(r)$, $\nu(r)$ and $\rho(r)$ by some symbolic variables, m_1, ν_1 and ρ_1, say:

In [70]:
```
var('m_1 nu_1 rho_1 r_1')
rhs = [y.subs({m(r): m_1, nu(r): nu_1, rho(r): rho_1}) for y in rhs]
rhs
```

Out[70]:

$$\left[4\,\pi r^2\rho_1\mathrm{u}\left(\rho_1\right), -\frac{\pi r^3\rho_1^2+m_1}{2\,m_1r-r^2}, \frac{\left(\pi r^3\rho_1^3+4\,\pi r^3\rho_1^2+m_1\rho_1+4\,m_1\right)\mathrm{u}\left(\rho_1\right)}{2\left(2\,m_1r-r^2\right)}, \mathrm{u}\left(\rho_1\right)\right]$$

The integration parameters:

In [71]:
```
rho_c = 1
r_min = 1e-8
r_max = 1
np = 200
delta_r = (r_max - r_min) / (np-1)
```

The numerical resolution, with the initial conditions

$$(r_{\min}, m(r_{\min}), \nu(r_{\min}), \rho(r_{\min}), R(r_{\min})) = (r_{\min}, 0, 0, \rho_{\rm c}, r_{\min})$$

set in the parameter `ics`:

In [72]:
```
sol = desolve_system_rk4(rhs, vars=(m_1, nu_1, rho_1, r_1), ivar=r,
                         ics=[r_min, 0, 0, rho_c, r_min],
                         end_points=(r_min, r_max), step=delta_r)
```

The solution is returned as a list, the first 10 elements of which being:

In [73]:
```
sol[:10]
```

Out[73]:

$$\left[\left[1.00000000000000 \times 10^{-8}, 0, 0, 1, 1.00000000000000 \times 10^{-8}\right],\right.$$
$$\left[0.00502513557789, 5.31396735484 \times 10^{-07}, 0.000105774675722,\right.$$
$$\left.0.999735571504, 0.00502513557789\right],$$
$$\left[0.0100502611558, 4.24965314706 \times 10^{-06}, 0.000384231107483,\right.$$
$$\left.0.999039516928, 0.0100502611558\right],$$
$$\left[0.0150753867337, 1.43327909521 \times 10^{-05}, 0.000846989113836,\right.$$
$$\left.0.997882978864, 0.0150753867337\right],$$
$$\left[0.0201005123116, 3.39411395241 \times 10^{-05}, 0.00149437530532,\right.$$
$$\left.0.996265461165, 0.0201005123116\right],$$
$$\left[0.0251256378894, 6.62085388025 \times 10^{-05}, 0.00232621990806,\right.$$
$$\left.0.99418783561, 0.0251256378894\right],$$
$$[0.0301507634673, 0.000114233579607, 0.00334221530611, 0.991651444462,$$
$$0.0301507634673],$$
$$[0.0351758890452, 0.000181070903418, 0.00454196301924, 0.988657981618,$$
$$0.0351758890452],$$
$$[0.0402010146231, 0.000269722576805, 0.0059249834935, 0.985209466401,$$
$$0.0402010146231],$$
$$[0.045226140201, 0.000383129555491, 0.00749071832026, 0.98130823596,$$
$$\left.0.045226140201]\right]$$

Each element is of the form $[r, m(r), \nu(r), \rho(r), R(r)]$. So to get the list of $(r, \rho(r))$ values, we write

In [74]:
```
rho_sol = [(s[0], s[3]) for s in sol]
rho_sol[:10]
```

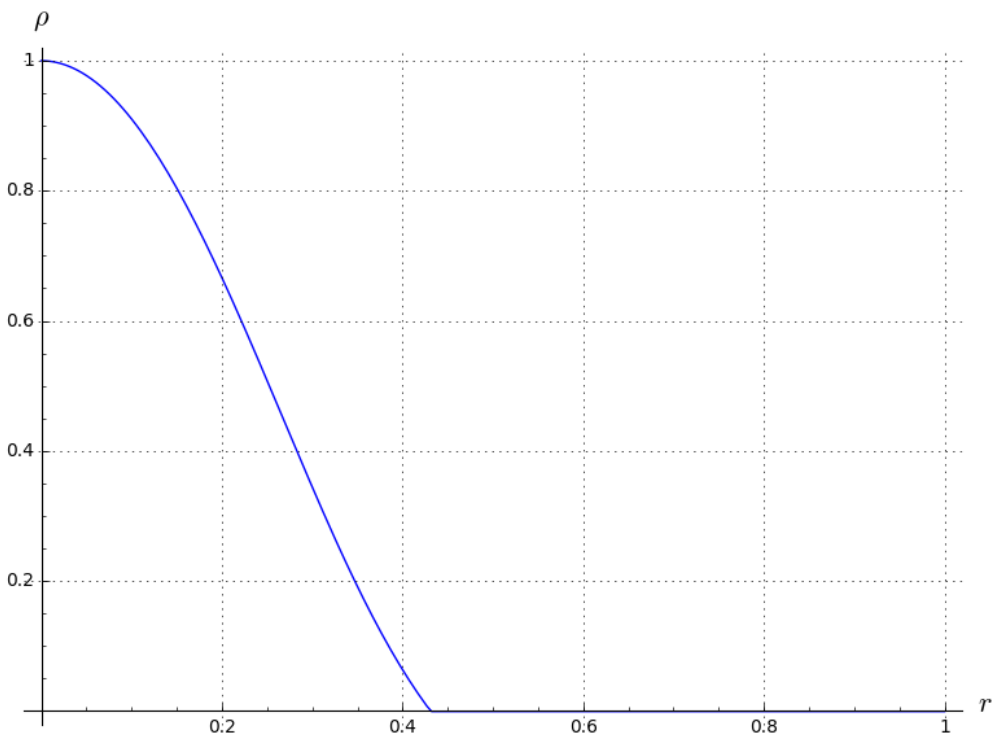
Out[74]:

$$\left[\left(1.00000000000000 \times 10^{-8}, 1\right), (0.00502513557789, 0.999735571504), (0.0100502611558, 0.999039516928), (0.0150753867337, 0.997882978864), (0.0201005123116, 0.996265461165), (0.0251256378894, 0.99418783561), (0.0301507634673, 0.991651444462), (0.0351758890452, 0.988657981618), (0.0402010146231, 0.985209466401), (0.045226140201, 0.98130823596)\right]$$

We may then use this list to have some plot of $\rho(r)$, thanks to the function `line`, which transforms a list into a graphical object:

In [75]:
```
graph = line(rho_sol, axes_labels=[r'$r$', r'$\rho$'], gridlines=True)
graph
```

Out[75]:



The solution for $m(r)$:

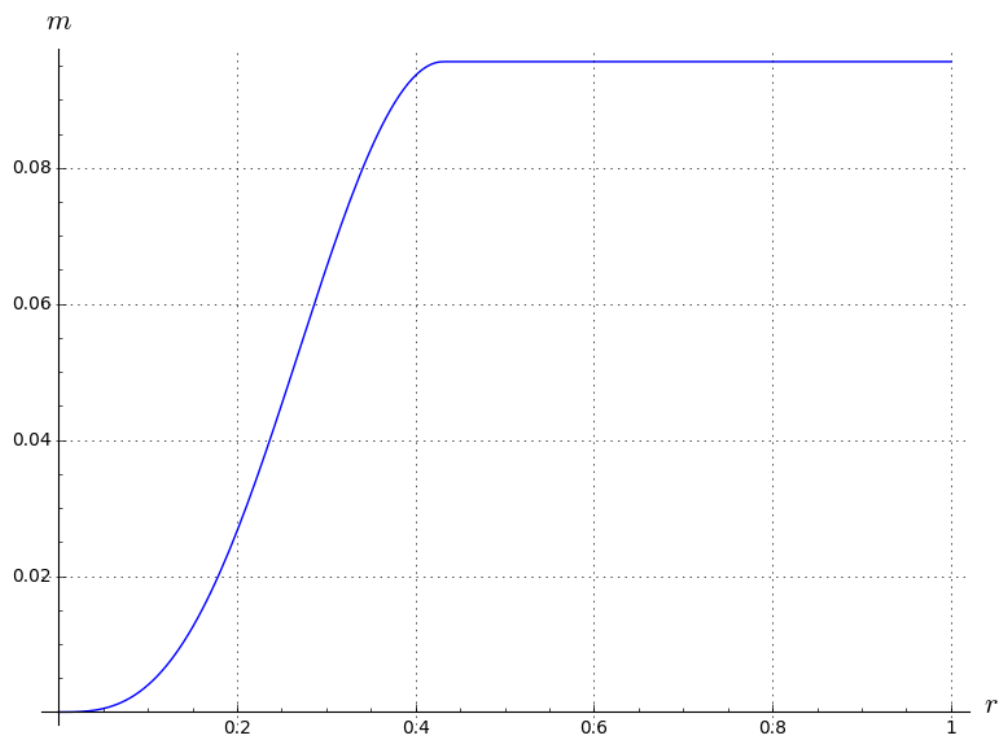
In [76]:
```
m_sol = [(s[0], s[1]) for s in sol]
m_sol[:10]
```

Out[76]:

$$\left[\left(1.00000000000000 \times 10^{-8}, 0\right),\right.$$
$$\left(0.00502513557789, 5.31396735484 \times 10^{-07}\right),$$
$$\left(0.0100502611558, 4.24965314706 \times 10^{-06}\right),$$
$$\left(0.0150753867337, 1.43327909521 \times 10^{-05}\right),$$
$$\left(0.0201005123116, 3.39411395241 \times 10^{-05}\right),$$
$$\left(0.0251256378894, 6.62085388025 \times 10^{-05}\right),$$
$$(0.0301507634673, 0.000114233579607),$$
$$(0.0351758890452, 0.000181070903418),$$
$$(0.0402010146231, 0.000269722576805),$$
$$\left.(0.045226140201, 0.000383129555491)\right]$$

In [77]:
```
graph = line(m_sol, axes_labels=[r'$r$', r'$m$'], gridlines=True)
graph
```

Out[77]:



The solution for $\nu(r)$ (has to be rescaled by adding a constant to ensure $\nu(+\infty) = 1$):

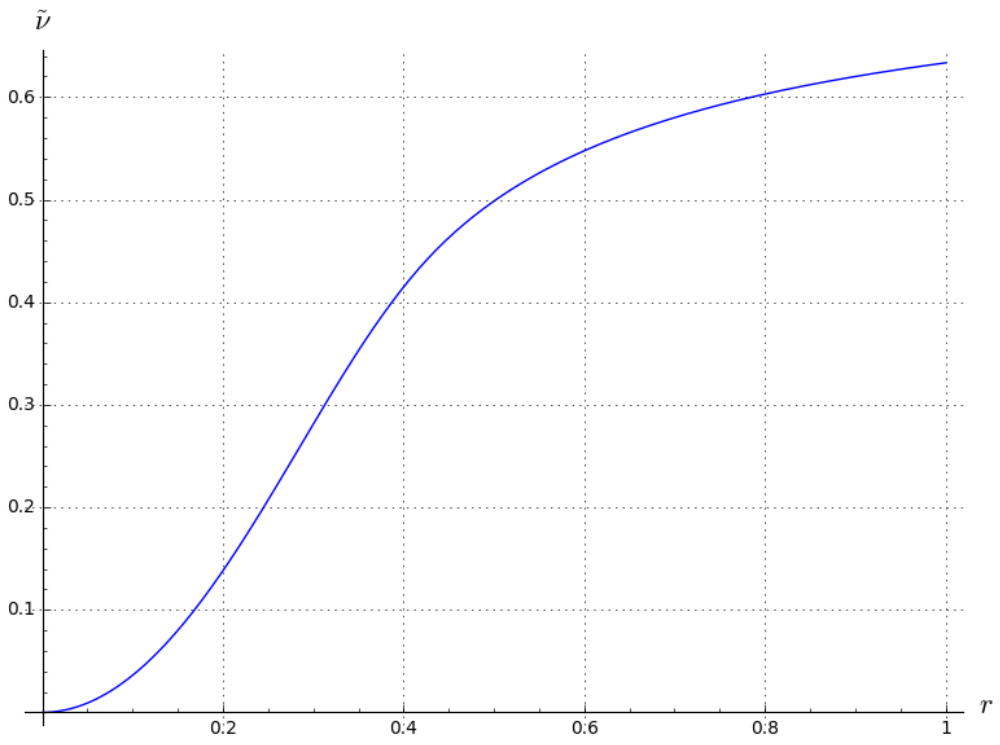
In [78]:
```
nu_sol = [(s[0], s[2]) for s in sol]
nu_sol[:10]
```

Out[78]:

$$\left[\left(1.00000000000000 \times 10^{-8}, 0\right), (0.00502513557789, 0.000105774675722), (0.0100502611558, 0.000384231107483), (0.0150753867337, 0.000846989113836), (0.0201005123116, 0.00149437530532), (0.0251256378894, 0.00232621990806), (0.0301507634673, 0.00334221530611), (0.0351758890452, 0.00454196301924), (0.0402010146231, 0.0059249834935), (0.045226140201, 0.00749071832026)\right]$$

In [79]:
```
graph = line(nu_sol, axes_labels=[r'$r$', r'$\tilde{\nu}$'], gridlines=
True)
graph
```

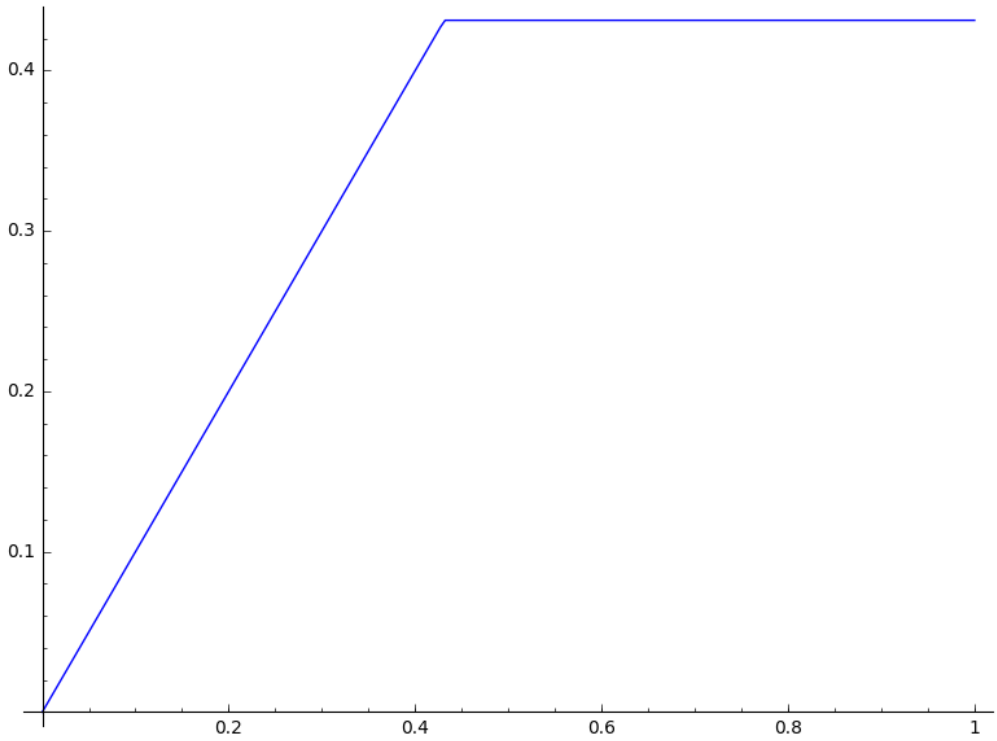
Out[79]:



The solution for $R(r)$:

In [80]:
```
r_sol = [(s[0], s[4]) for s in sol]
line(r_sol)
```

Out[80]:



The total gravitational mass of the star is obtained via the last element (index: -1) of the list of $(r, m(r))$ values:

In [81]:
```
M_grav = m_sol[-1][1]
M_grav
```

Out[81]: 0.095595655983

Similarly, the stellar radius is obtained through the last element of the list of $(r, R(r))$ values:

In [82]:
```
R = r_sol[-1][1]
R
```

Out[82]: 0.431323288769

The star's compactness:

In [83]:
```
M_grav/R
```

Out[83]: 0.221633420852

Sequence of stellar models

Let us perform a loop on the central density. First we set up a list of values for ρ_c:

In [84]:
```
rho_c_min = 0.01
rho_c_max = 3
n_conf = 40
rho_c_list = [rho_c_min + i * (rho_c_max-rho_c_min)/(n_conf-1) for i in
 range(n_conf)]
rho_c_list
```

Out[84]:
$$[0.0100000000000000, 0.0866666666666667, 0.163333333333333, 0.240000000000000, 0.316666666666667, 0.393333333333333, 0.470000000000000, 0.546666666666667, 0.623333333333333, 0.700000000000000, 0.776666666666667, 0.853333333333333, 0.930000000000000, 1.00666666666667, 1.08333333333333, 1.16000000000000, 1.23666666666667, 1.31333333333333, 1.39000000000000, 1.46666666666667, 1.54333333333333, 1.62000000000000, 1.69666666666667, 1.77333333333333, 1.85000000000000, 1.92666666666667, 2.00333333333333, 2.08000000000000, 2.15666666666667, 2.23333333333333, 2.31000000000000, 2.38666666666667, 2.46333333333333, 2.54000000000000, 2.61666666666667, 2.69333333333333, 2.77000000000000, 2.84666666666667, 2.92333333333333, 3.00000000000000]$$

The loop:

In [85]:
```
M_list = list()
R_list = list()
for rho_c in rho_c_list:
    sol = desolve_system_rk4(rhs, vars=(m_1, nu_1, rho_1, r_1), ivar=r,

                             ics=[r_min, 0, 0, rho_c, r_min],
                             end_points=(r_min, r_max), step=delta_r)
    M_list.append( sol[-1][1] )
    R_list.append( sol[-1][4] )
```

The mass along the sequence:

In [86]:
```
M_list
```

Out[86]:
$$[0.00307931536045, 0.0234854698535, 0.0393069279916, 0.0516981286546, 0.0614830416515, 0.0692798437746, 0.0755155592693, 0.0805202557445, 0.0845711377823, 0.0878316691012, 0.0904777978438, 0.0926071710064, 0.0943341571919, 0.0957016482746, 0.096805301455, 0.0976659103806, 0.0983512152837, 0.0988514527311, 0.0992275619866, 0.0994878873502, 0.0996539700497, 0.0997408164201, 0.0997609123123, 0.099725494781, 0.099637243027, 0.0995219231466, 0.0993676470084, 0.0991625186115, 0.09895164244, 0.0987339306628, 0.0984694840218, 0.0982152325099, 0.0979297032307, 0.0976523950536, 0.0973803611296, 0.0970791116622, 0.0967626566092, 0.0964703442622, 0.0961617903134, 0.0958704811036]$$

The radius along the sequence:

In [87]:
```
R_list
```

Out[87]:
$$[0.624790623518, 0.599664995628, 0.575376888668, 0.555276386357, 0.536850925905, 0.520100507312, 0.505025130578, 0.491624795704, 0.479899502688, 0.469849251533, 0.458961479447, 0.449748749221, 0.439698498065, 0.431323288769, 0.424623121332, 0.416247912035, 0.409547744598, 0.40452261902, 0.398659972513, 0.391122284146, 0.386097158568, 0.38107203299, 0.376046907412, 0.373534344623, 0.369346739975, 0.364321614397, 0.359296488819, 0.355946405101, 0.354271363241, 0.349246237663, 0.345896153945, 0.344221112085, 0.340871028367, 0.339195986508, 0.336683423719, 0.33333334, 0.330820777211, 0.329145735352, 0.325795651633, 0.324120609774]$$

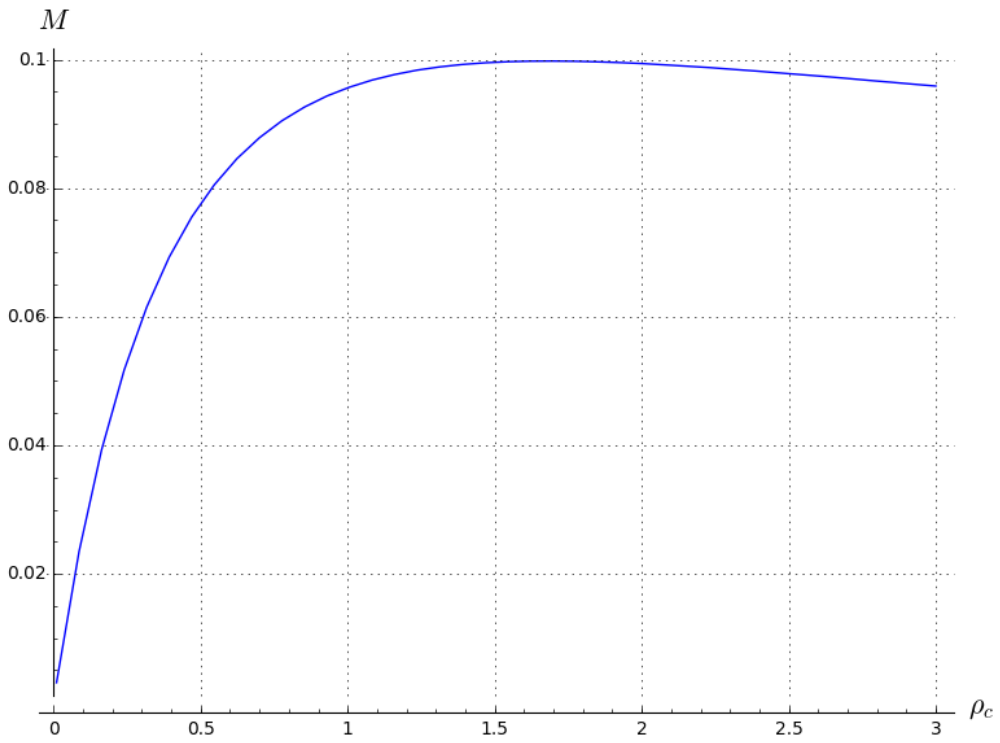
To draw M as a function of ρ_c, we use the Python function `zip` to construct a list of (ρ_c, M) values:

In [88]:
```
zip(rho_c_list, M_list)
```

Out[88]:
[(0.0100000000000000, 0.00307931536045),
(0.0866666666666667, 0.0234854698535),
(0.163333333333333, 0.0393069279916),
(0.240000000000000, 0.0516981286546),
(0.316666666666667, 0.0614830416515),
(0.393333333333333, 0.0692798437746),
(0.470000000000000, 0.0755155592693),
(0.546666666666667, 0.0805202557445),
(0.623333333333333, 0.0845711377823),
(0.700000000000000, 0.0878316691012),
(0.776666666666667, 0.0904777978438),
(0.853333333333333, 0.0926071710064),
(0.930000000000000, 0.0943341571919),
(1.00666666666667, 0.0957016482746),
(1.08333333333333, 0.096805301455),
(1.16000000000000, 0.0976659103806),
(1.23666666666667, 0.0983512152837),
(1.31333333333333, 0.0988514527311),
(1.39000000000000, 0.0992275619866),
(1.46666666666667, 0.0994878873502),
(1.54333333333333, 0.0996539700497),
(1.62000000000000, 0.0997408164201),
(1.69666666666667, 0.0997609123123),
(1.77333333333333, 0.099725494781), (1.85000000000000, 0.099637243027),
(1.92666666666667, 0.0995219231466),
(2.00333333333333, 0.0993676470084),
(2.08000000000000, 0.0991625186115), (2.15666666666667, 0.09895164244),
(2.23333333333333, 0.0987339306628),
(2.31000000000000, 0.0984694840218),
(2.38666666666667, 0.0982152325099),
(2.46333333333333, 0.0979297032307),
(2.54000000000000, 0.0976523950536),
(2.61666666666667, 0.0973803611296),
(2.69333333333333, 0.0970791116622),
(2.77000000000000, 0.0967626566092),
(2.84666666666667, 0.0964703442622),
(2.92333333333333, 0.0961617903134),
(3.00000000000000, 0.0958704811036)]

In [89]:
```
graph = line(zip(rho_c_list, M_list), axes_labels=[r'$\rho_c$', r'$M$']
, gridlines=True)
graph
```

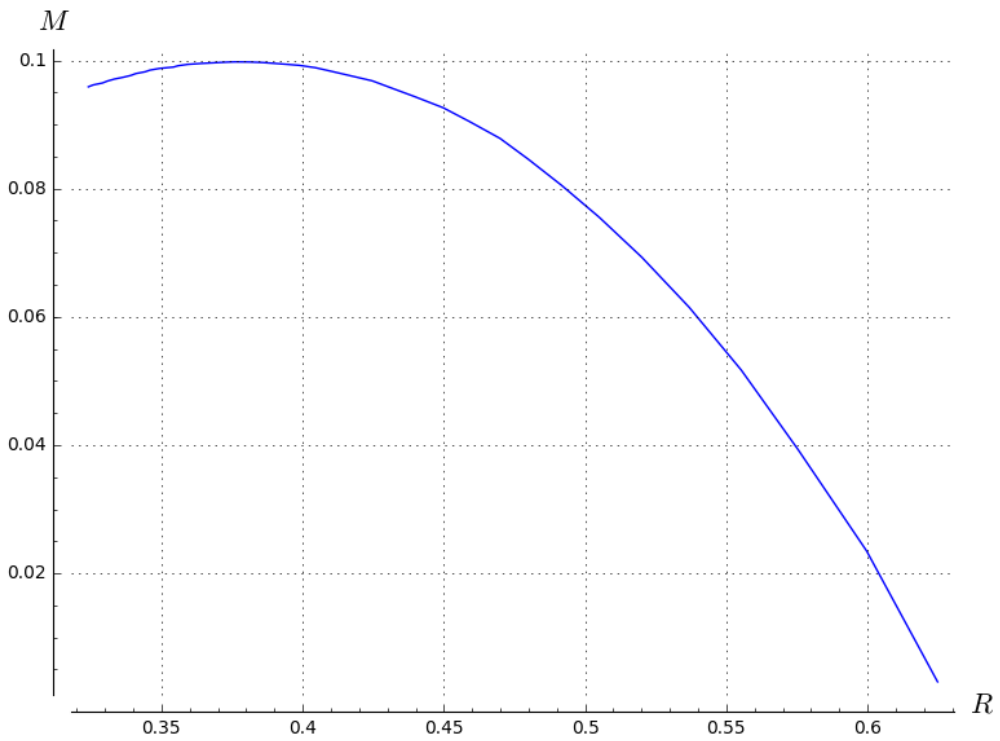
Out[89]:



Similarly, we draw M as a function of R:

In [90]:
```
graph = line(zip(R_list, M_list), axes_labels=[r'$R$', r'$M$'], gridlin
es=True)
graph
```

Out[90]:



and we save the plot in a pdf file to use it in our next publication ;-)

In [91]:
```
graph.save('plot_M_R.pdf')
```