Analysis of White Dwarfs and Neutron Stars

Maria Lucia Marcelli

May 2025

Overview

1. White Dwarfs
2. Pure Neutron Stars
3. Neutron Stars with protons and electrons
4. Neutron Stars with compOSE

Introduction

- Compact stars, i.e. white dwarfs and neutron stars, are the final stages in the evolution of ordinary stars.
- White Dwarfs: stabilized by **electrons**' degeneracy pressure
- Neutron Stars: stabilized by **neutrons**' degeneracy pressure

White Dwarfs

Structure Equations

- Two forces acting on the star: gravitational one and the one related to the degeneracy pressure. The gravitational force is given by:

$$dF = -\frac{Gdm \cdot m(r)}{r^2} \tag{1}$$

with

$$\frac{dm}{dr} = \rho(r) 4\pi r^2 \tag{2}$$

Therefore :

$$\frac{dp}{dr} = -\frac{G\rho(r) \cdot m(r)}{r^2} \tag{3}$$

- Boundary conditions: $m(r = 0) = 0$ and $p(r = 0) = p_c$ with p_c central pressure

White Dwarfs

Equations of State

- One needs to fix the relation between the pressure p and the energy density ϵ.
- Matter in white dwarfs can be treated as an ideal Fermi gas of degenerate electrons: their distributions is described by the Fermi-Dirac statistics

$$f(E) = \frac{1}{exp[(E-\mu)/k_B T] + 1} \tag{4}$$

- number density of the degenerate electron gas:

$$n_e = \int_0^{k_F} \frac{2}{(2\pi\hbar)^3} d^3k = \frac{8\pi}{(2\pi\hbar)^3} \int_0^{k_F} k^2 dk = \frac{k_F^3}{3\pi^2\hbar^3} \tag{5}$$

White Dwarfs

Equations of State

- The star should be electrically neutral: each electron must be neutralised by a proton. To compute the mass density in the white dwarf one can concentrate just on the nucleon mass m_n:

$$\rho = n_e \cdot m_N \cdot \frac{A}{Z} \tag{6}$$

- The energy density can be divided into two components, one term coming from nucleons and the other from electrons:

$$\epsilon \quad = \quad n_e m_N \frac{A}{Z} c^2 + \epsilon_{\rm elec}(k_F) \quad . \tag{7}$$

where $\epsilon_{\rm elec}(k_F)$ is the energy density of electrons.

White Dwarfs

Equations of State

- With the electron energy

$$E(k) = \sqrt{k^2c^2 + m_e^2c^4} \tag{8}$$

one can write

$$\begin{aligned} \epsilon_{\rm elec}(k_F) &= \frac{8\pi}{(2\pi\hbar)^3}\int_0^{k_F} E(k)k^2dk \\ &= \frac{\epsilon_0}{8}\left[(2x^3+x)(1+x^2)^{1/2} - \sinh^{-1}(x)\right] \end{aligned} \tag{9}$$

with

$$\epsilon_0 = \frac{m_e^4c^5}{\pi^2\hbar^3} \tag{10}$$

and

$$x = k_F/m_ec. \tag{11}$$

White Dwarfs

Equations of State

- One can also obtain the electrons' pressure, given by:

$$\begin{aligned} p(k_F) &= \frac{1}{3}\frac{8\pi}{(2\pi\hbar)^3}\int_0^{k_F}\frac{k^2c^2}{E(k)}k^2dk \\ &= \frac{\epsilon_0}{24}\left[(2x^3-3x)(1+x^2)^{1/2}+3\sinh^{-1}(x)\right]. \end{aligned} \quad (12)$$

- The energy density is dominated by the mass density of nucleons while the electrons contribute to most of the pressure.
- To get an equation of the form $p = p(\epsilon)$ one can consider two estreme cases: $x << 1$ (non-relativistic one) and $x >> 1$ (relativistic one).

White Dwarfs

Polytropic Equations of State

- Non relativistic case:

$$p \approx K_{\mathrm{non-rel}} \epsilon^{5/3} \tag{13}$$

with

$$K_{\mathrm{non-rel}} = \frac{\hbar^2}{15\pi^2 m_e} \left(\frac{3\pi^2 Z}{m_N c^2 A} \right)^{5/3} . \tag{14}$$

- Relativistic case:

$$p(\epsilon) \approx K_{\mathrm{rel}}\, \epsilon^{4/3} \tag{15}$$

where

$$K_{\mathrm{rel}} = \frac{\hbar c}{12\pi^2} \left(\frac{3\pi^2 Z}{m_N c^2 A} \right)^{4/3} . \tag{16}$$

White Dwarfs

General Relativity Corrections

- If the stars are very compact one has to take into account effects from General Relativity.
- Using Einstein's equations:

$$G_{\mu\nu} = -\frac{8\pi G}{c^4} T_{\mu\nu} \quad , \tag{17}$$

one arrives at the Tolman–Oppenheimer–Volkoff (TOV) equation:

$$\frac{dp}{dr} = -\frac{G\epsilon(r)m(r)}{c^2 r^2}\left[1+\frac{p(r)}{\epsilon(r)}\right]\left[1+\frac{4\pi r^3 p(r)}{m(r)c^2}\right]\left[1-\frac{2Gm(r)}{c^2 r}\right]^{-1} \quad . \tag{18}$$

Computational Analysis for White Dwarfs

Polytropic EoS

- Case of a polytropic EoS (for both non-relativistic and relativistic case) in a Newtonian regime:

$$p = K\epsilon^{\gamma} \quad , \tag{19}$$

$$\frac{dm}{dr} = \rho(r)4\pi r^2, \tag{20}$$

$$\frac{dp}{dr} = -\frac{G\rho(r)\cdot m(r)}{r^2}. \tag{21}$$

- One can rewrite these equations in terms of dimensionless quantities.

Computational Analysis for White Dwarfs

Polytropic EoS

- Defining ρ_c as the central matter density,

$$R_0 = \frac{c}{(4\pi G \rho_c)^{1/2}} \tag{22}$$

and

$$M_0 = \frac{4\pi c^3 \rho_c}{(4\pi G \rho_c)^{3/2}} \tag{23}$$

One can introduce these dimensionless quantities:

$$\hat{r} = r/R_0 \tag{24}$$

$$\hat{\rho} = \rho/\rho_c \tag{25}$$

$$\hat{m} = m/M_0 \tag{26}$$

$$\hat{p} = p/(\rho_c c^2) \tag{27}$$

Computational Analysis for White Dwarfs

Polytropic EoS

- From these definitions, one can obtain the differential equations for the dimensionless quantities:

$$\hat{p} = w\hat{\rho}^{\gamma} \tag{28}$$

with

$$w = Kc^{2(\gamma-1)}{\rho_c}^{\gamma-1} \tag{29}$$

and

$$\frac{d\hat{m}}{d\hat{r}} = \hat{r}^2\hat{\rho} \tag{30}$$

$$\frac{d\hat{p}}{d\hat{r}} = -\frac{\hat{m}\hat{\rho}}{\hat{r}^2} \tag{31}$$

Computational Analysis for White Dwarfs

NON RELATIVISTIC Polytropic EoS in Newtonian regime

- Define the system of differential equations:

```
#Define the system of dimensionless differential eqns
def white_dwarf(r, y, w ,z):
    p ,m  = y
    if p <= 0:
        return np.array([0, 0])
    dpdr = -m/(r**2)*(p/w)**z
    dmdr = (r**2)*(p/w)**z
    return np.array([dpdr, dmdr])
```

- Define the central pressure and all the needed parameters for the system of dimensionless differential equations:

```
#Parameters for EoS

#pressure in the centre of the white dwarf
p_c=2.33002e21

#non relativistic constant in the eos p=Knr*e**(5/3)
Knr = h_t**2/(15*np.pi**2*m_e)*(3*np.pi**2*((2*m_n*c**2)**(-1)))**(5/3)

#matter density in the centre of the white dwarf
ro_c=(p_c/Knr)**(3/5)*c**(-2)

#dimensionless param for the dimensionless EoS
w = Knr*c**(4/3)*ro_c**(2/3)

#inverse of gamma
z= 3/5

#initial values for the dimensionless system of differential eqns
p0=p_c/ro_c*c**(-2)
m0 = 0.0
y0 = np.array([p0, m0])
r0, rf, dr = 1e-6, 0.04, 0.001
```

Computational Analysis for White Dwarfs

RK4

- To solve the system of differential equations one can use the RK4 method, which is implemented by this code:

```
#Define RK4 method
def rk4(f, y0, r0, rf, dr, w, z):
    r_values = np.arange(r0, rf, dr)
    y_values = np.zeros((len(r_values), len(y0)))
    y_values[0] = y0

    for i in range(1, len(r_values)):
        r = r_values[i - 1]
        y = y_values[i - 1]
        k1 = dr * f(r, y, w, z)
        k2 = dr * f(r + dr/2, y + k1/2, w, z)
        k3 = dr * f(r + dr/2, y + k2/2, w, z)
        k4 = dr * f(r + dr, y + k3, w, z)

        y_values[i] = y + (k1 + 2*k2 + 2*k3 + k4) / 6

    return r_values, y_values
```

Computational Analysis for White Dwarfs

NON RELATIVISTIC Polytropic EoS in Newtonian regime

- Solve the system of differential equations:

```
# Solve using RK4
r_values, y_values = rk4(white_dwarf, y0, r0, rf, dr, w, z)
```

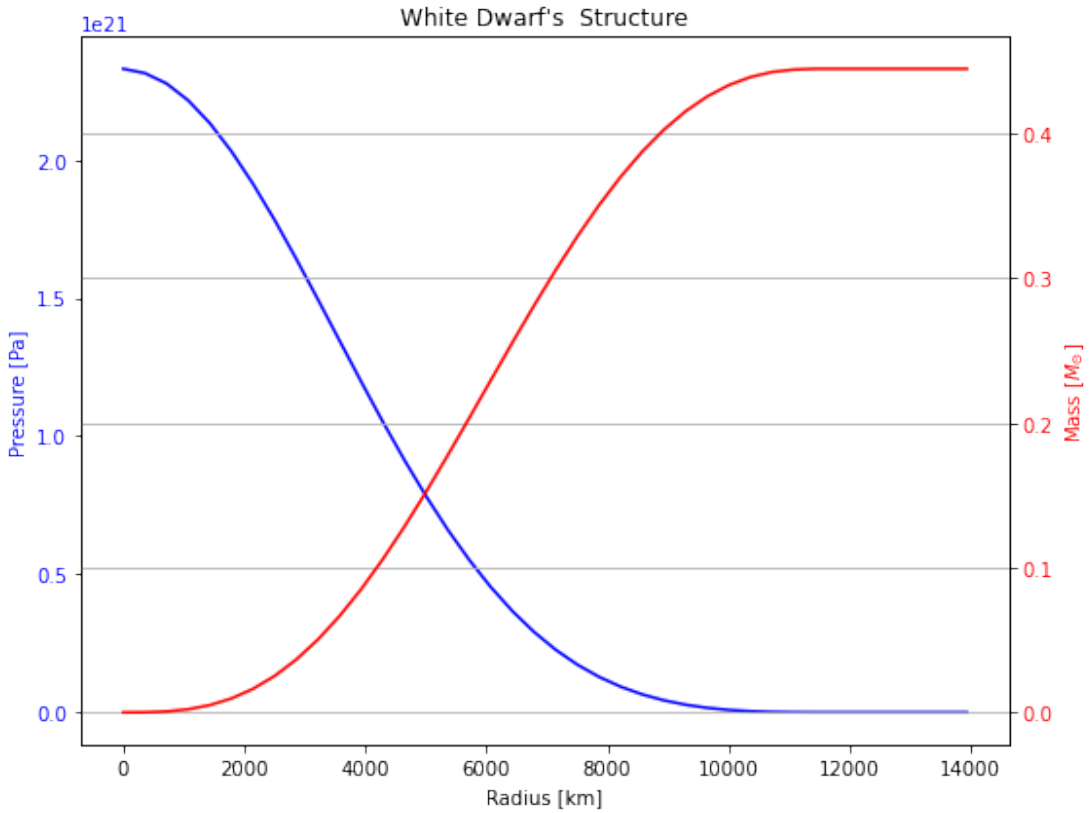
- Return to dimensionful quantities:

```
#Back to dimensionful quantities

m_values= y_values[:, 1]*4*np.pi*ro_c/(ro_c*4*np.pi*G)**(3/2)*c**3/m_sun
p_values= y_values[:, 0]*ro_c*c**2
r_values1=r_values/(ro_c*4*np.pi*G)**(1/2)*c*10**(-3)
```

Computational Analysis for White Dwarfs

NON RELATIVISTIC Polytropic EoS in Newtonian regime

- Plot the behaviour of the pressure and the mass inside the white dwarf:



Computational Analysis for White Dwarfs

NON RELATIVISTIC Polytropic EoS in Newtonian regime

- Find the final radius and mass of the white dwarf:

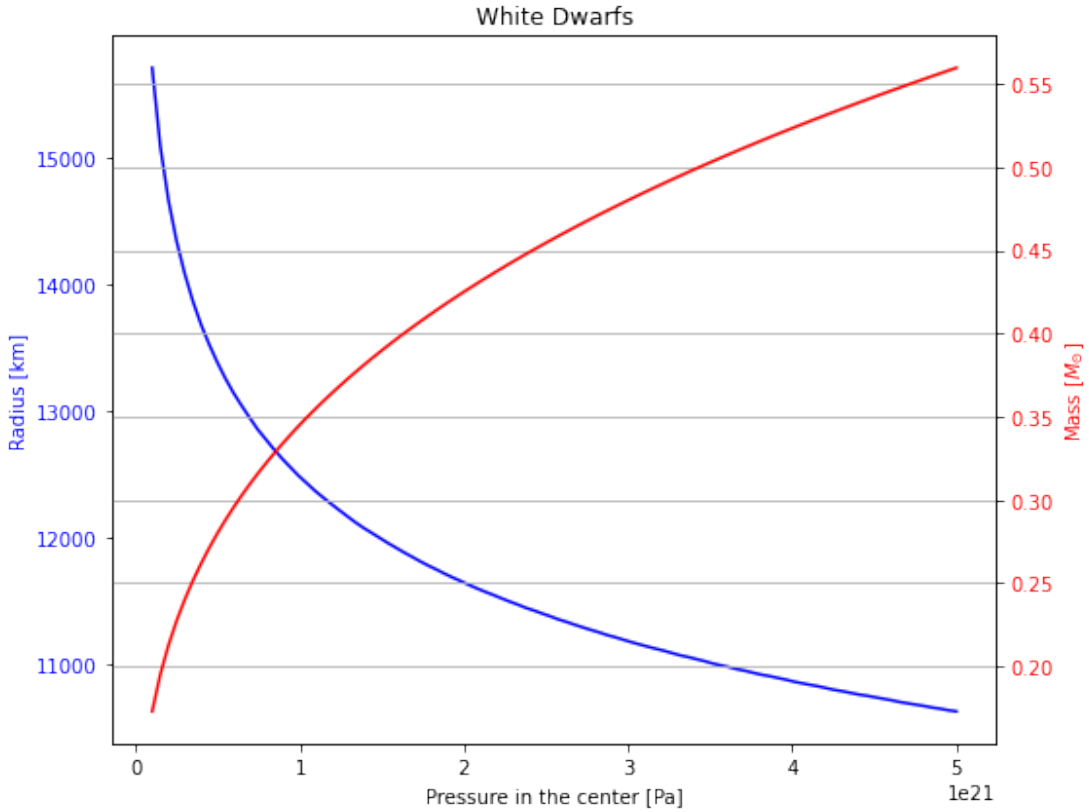
```
#Find mass and radius of the white dwarf(Km and M_sun)
for i in range(len(p_values)):
    if p_values[i] <= 0:
        r_stop = r_values1[i]  # White dwarf's radius
        M_star = m_values[i]  # White dwarf's mass
        break

print(r_stop)
print(M_star)
```

```
11437.953522005835
0.4448101643696684
```

Computational Analysis for White Dwarfs

NON RELATIVISTIC Polytropic EoS in Newtonian regime

- Plot the radii and masses of white dwarfs for various central pressures:



Computational Analysis for White Dwarfs

RELATIVISTIC Polytropic EoS in Newtonian regime

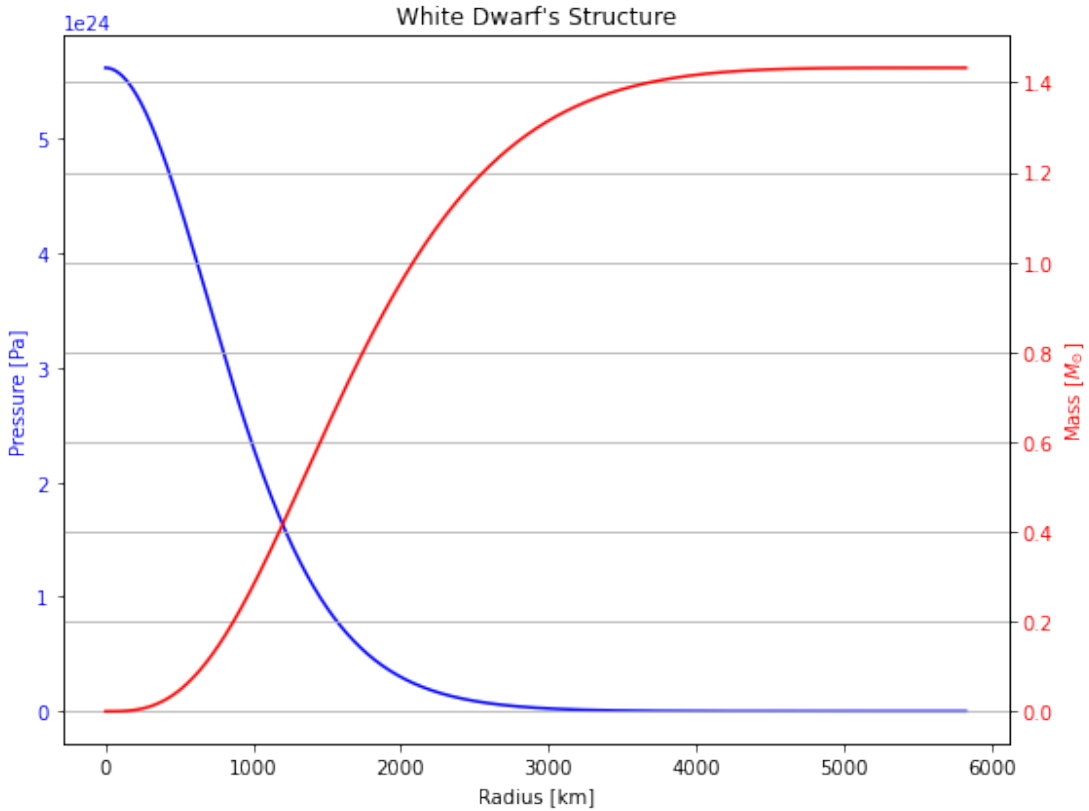
- Same system of differential equations
- Define the central pressure and all the needed parameters for the system of dimensionless differential equations:

```
#Parameters for EoS

#relativistic constant in the eos p=Kr*e**(4/3)
Kr = h_t*c/(12*np.pi**2)*(3*np.pi**2*((2*m_n*c**2)**(-1)))**(4/3)

#pressure in the centre of the white dwarf
p_c=5.62e24

#matter density in the centre of the white dwarf
ro_c=(p_c/Kr)**(3/4)*c**(-2)

#dimensionless param for dimensionless EoS
w = Kr*c**(2/3)*ro_c**(1/3)

#inverse of gamma
z= 3/4

#initial values for the dimensionless system of differential eqns
p0=p_c/ro_c*c**(-2)
m0 = 0.0
y0 = np.array([p0, m0])
r0, rf, dr = 1e-6, 0.25, 0.0001
```

Computational Analysis for White Dwarfs

RELATIVISTIC Polytropic EoS in Newtonian regime

- Solve the system of differential equations
- Return to dimensionful quantities
- Plot the behaviour of pressure and mass inside the white dwarf:



Computational Analysis for White Dwarfs

RELATIVISTIC Polytropic EoS in Newtonian regime

- Find the final radius and mass of the white dwarf:

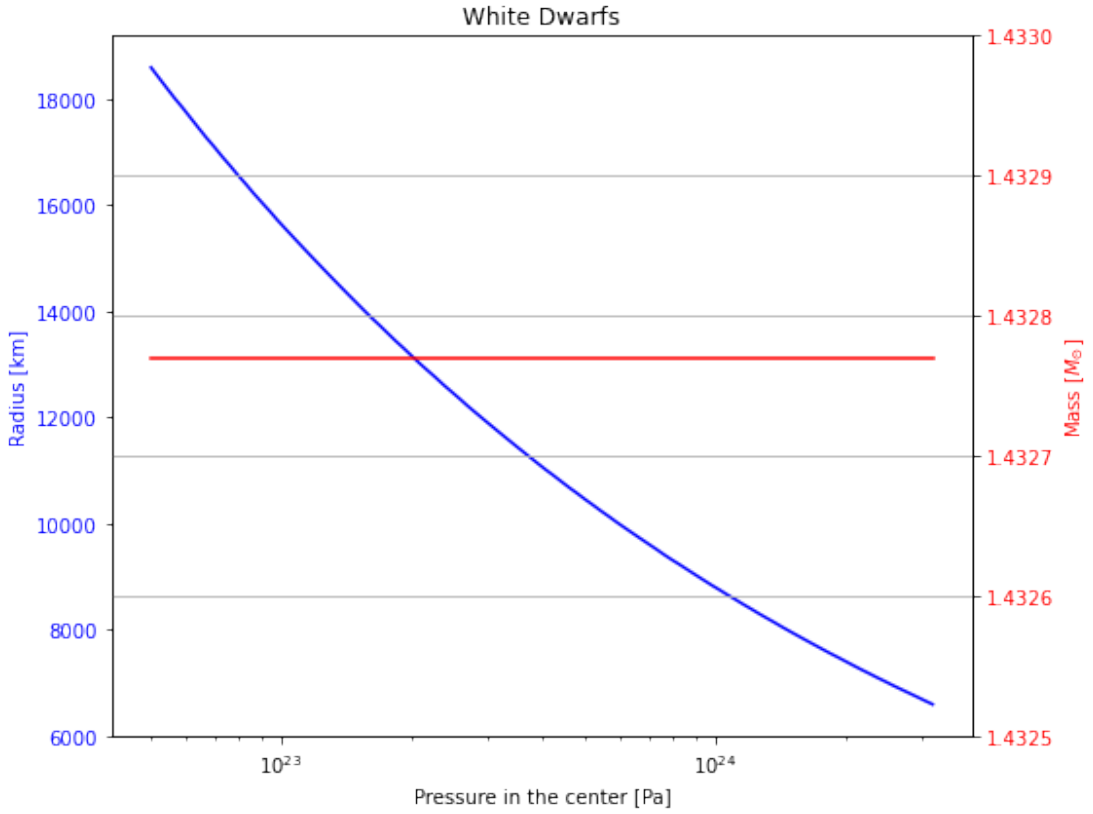
```
#Find mass and radius of the White Dwarf (Km and M_sun)
for i in range(len(p_values)):
    if p_values[i] <= 0:
        r_stop = r_values1[i]  # White Dwarf's radius
        M_star = m_values[i]  # White Dwarf's mass
        break

print(r_stop)
print(M_star)
```

```
5711.1504325725255
1.4327703586476153
```

Computational Analysis for White Dwarfs

RELATIVISTIC Polytropic EoS in Newtonian regime

- Plot the radii and masses of white dwarfs for various central pressures:



Computational Analysis for White Dwarfs

NON RELATIVISTIC Polytropic EoS in GR regime

- Define the system of differential equations and its solver by using the function *solveivp* from *scipy.integrate*

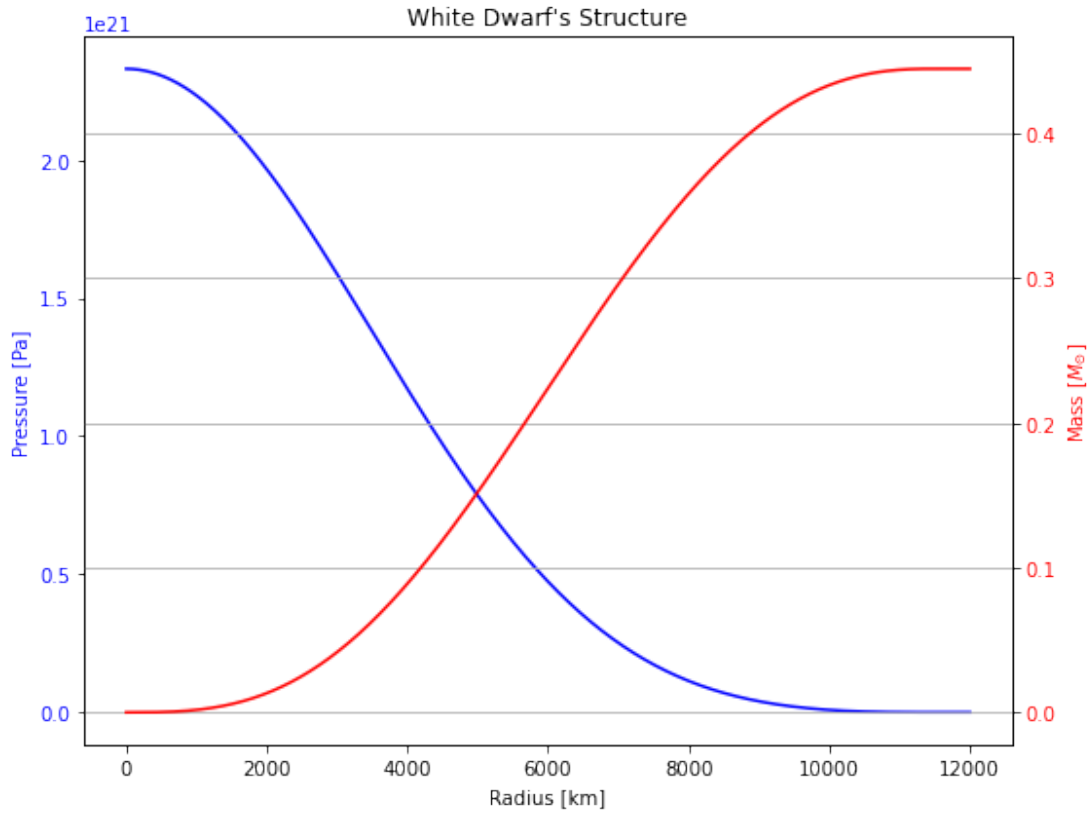
```
def tov_equations(r, y):
    p, m = y
    if p <= 0:
        return np.array([0, 0])
    e = (p / w)**z
    dmdr = 4 * np.pi * r**2 * e / c**2
    dpdr = -G * e * m / (c**2 * r**2) * (1 + p / e) * (1 + 4 * np.pi * r**3 * p / (m * c**2)) / (1 - 2 * G * m / (c**2 * r))
    return [dpdr, dmdr]


def solve_tov(p_c):
    y0 = [p_c, 1e-6]
    r_span = (1e-6, 1.2e7)

    sol = solve_ivp(tov_equations, r_span, y0, method='RK45', max_step=1e3)
    return sol
```

- Define central pressure p_c (same value as the one considered in the NR newtonian regime to compare the two cases)

Computational Analysis for White Dwarfs

NON RELATIVISTIC Polytropic EoS in GR regime

- Plot the behaviour of the pressure and the mass inside the white dwarf:



Computational Analysis for White Dwarfs

NON RELATIVISTIC Polytropic EoS in GR regime

- Find the final radius and mass of the white dwarf:

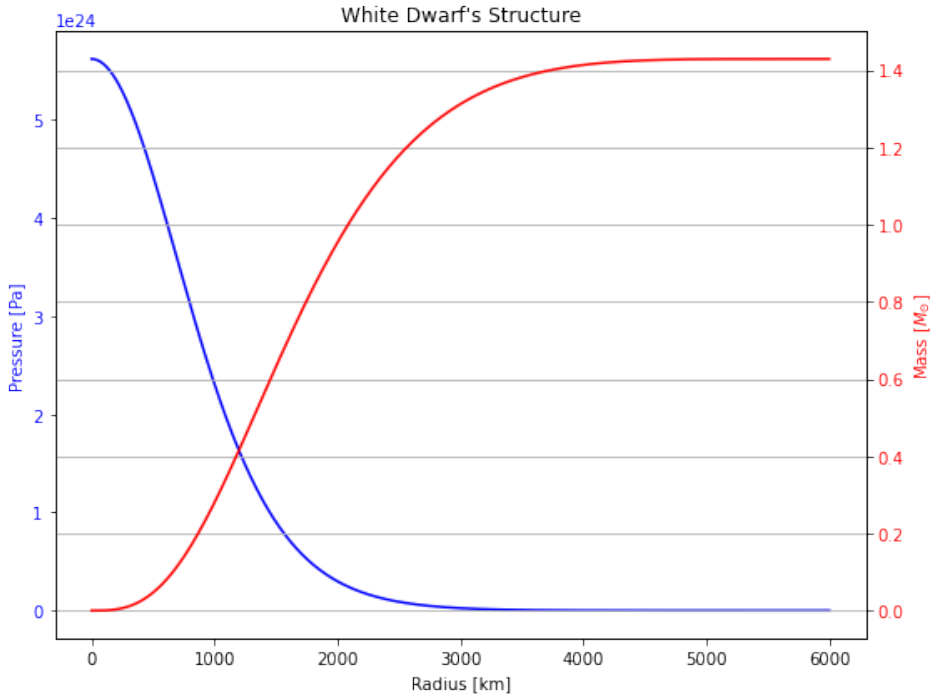
```
#Find radius and mass of the white dwarf (km and M_sun)
for i in range(len(p_values)):
    if p_values[i] <= 0:
        r_stop = r_values1[i]  # White dwarf's radius
        M_star = m_values[i]  # White dwarf's mass
        break

print(r_stop)
print(M_star)
```

```
11462.904188806258
0.4448416920588513
```

Computational Analysis for White Dwarfs

RELATIVISTIC Polytropic EoS in GR regime

- Same system of differential equations and same solving method
- Define central pressure p_c (same value as the one considered in the R newtonian regime to compare the two cases)
- Plot the behaviour of pressure and mass inside the white dwarf:



Computational Analysis for White Dwarfs

RELATIVISTIC Polytropic EoS in GR regime

- Find the final radius and mass of the white dwarf:

```
#Find radius and mass of the white dwarf (km and M_sun)
for i in range(len(p_values)):
    if p_values[i] <= 0:
        r_stop = r_values1[i]  # White dwarf's radius
        M_star = m_values[i]  # White dwarf's mass
        break

print(r_stop)
print(M_star)
```

```
5711.145622209247
1.4292272583081747
```

Computational Analysis for White Dwarfs

Looking for zero-points

- It is necessary to have a relation between the energy density and the pressure to calculate the mass and the radius of the star
- From equations 7-12 one can see that both the pressure and the energy density are functions of the Fermi momentum
- One can use a root-finding routine to find for a given pressure the corresponding Fermi momentum and use it in the energy density's expression.
- In this way one can obtain the energy density ϵ for a given pressure p.
- To get the Fermi-momentum one can apply a root-finding method to the equation

$$p(k_F) - p = 0 \tag{32}$$

where p is given and $p(k_F)$ taken from 12.

Computational Analysis for White Dwarfs

Looking for zero-points: code

- Define the explicit expressions for the pressure and the energy density
- Define a function which, via a root-finding method (*brentq* in this case), gives the energy density associated to a given pressure

```
e0=m_e**4*c**5/(np.pi**2*h_t**3)

#Explicit form of the pressure
def pressure(x, p_g):
    p=e0/24*((2*x**3-3*x)*(1+x**2)**(1/2)+3*np.arcsinh(x))
    p_diff=p-p_g
    return(p_diff)


#Explicit form of the energy density
def energy(x):
    e_elec=e0/8*((2*x**3+x)*(1+x**2)**(1/2)-np.arcsinh(x))
    k=x*m_e*c
    n=k**3/(3*np.pi**2*h_t**3)
    e_tot=2*n*m_n*c**2+e_elec
    return(e_tot)


#Definition of a function which gives me the corresponding e for a given p
def energy_density(p):
    root = brentq(pressure, -100000, 100000, args=(p))
    e=energy(root)
    return e
```

Computational Analysis for White Dwarfs

Looking for zero-points: code

- Use this defined function inside the system of differential equations:

```
#Define TOV equations using the defined function that gives me the energy density for a given pressure

def tov_equations(r, y):
    p, m = y
    if p <= 0:
        return np.array([0, 0])
    e=energy_density(p)
    dmdr = 4 * np.pi * r**2 * e / c**2
    dpdr = -G * e * m / (c**2 * r**2) * (1 + p / e) * (1 + 4 * np.pi * r**3 * p / (m * c**2)) / (1 - 2 * G * m / (c**2 * r))
    return [dpdr, dmdr]
```

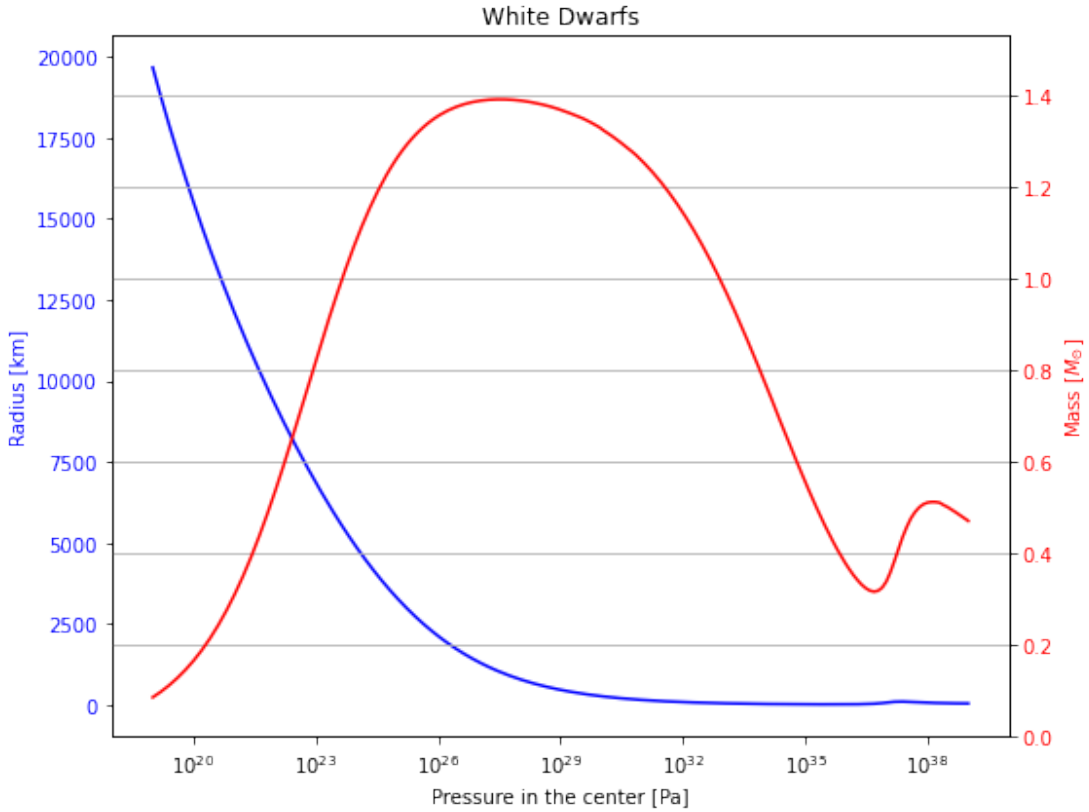
- Define the solver for the differential equations using *solveivp* from *scipy.integrate*

```
def solve_tov(p_c):
    y0 = [p_c, 1e-6]
    r_span = (1e-6, 1e8)

    sol = solve_ivp(tov_equations, r_span, y0, method='LSODA', max_step=1e4)
    return sol
```

Computational Analysis for White Dwarfs

Looking for zero-points: code

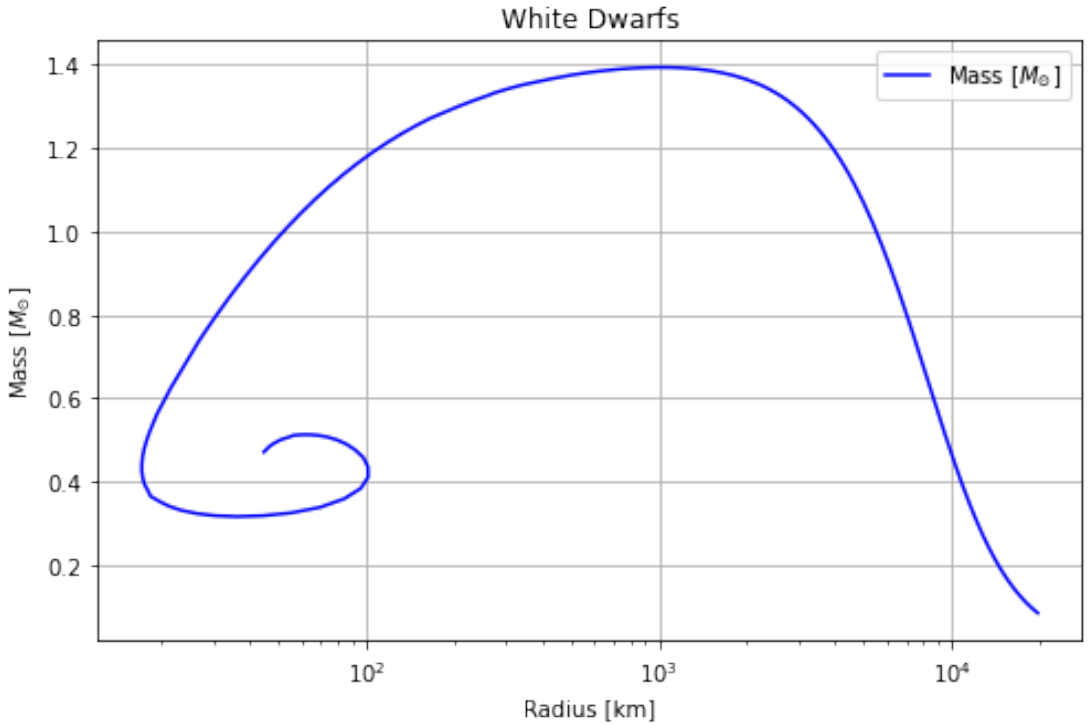
- Solve for many possible central pressures
- Plot the radii and masses of various white dwarfs



Computational Analysis for White Dwarfs

Looking for zero-points: code

- Plot the masses of various white dwarfs with respect to their radii:



Pure Neutron Stars

Structure Equations

- Pure non-relativistic neutron stars are described by an EoS of a Fermi gas of neutrons.
- The neutron number density is given by:

$$n_n = \frac{k_n^3}{3\pi^2\hbar^3}. \tag{33}$$

- In analogy to the Fermi gas for electrons, one has:

$$\epsilon(x) = \frac{\epsilon_0}{8}[(2x^3 + x)(1 + x^2)^{1/2} - \sinh^{-1}(x)] \tag{34}$$

$$p(x) = \frac{\epsilon_0}{24}[(2x^3 - 3x)(1 + x^2)^{1/2} + 3\sinh^{-1}(x)] \tag{35}$$

with

$$x = \frac{k_n}{m_n c}, \quad \epsilon_0 = \frac{m_n^4 c^5}{\pi^2\hbar^3} \quad . \tag{36}$$

Pure Neutron Stars

NON RELATIVISTIC polytropic case

- In the non-relativistic case these equations can be simplified and can lead to a polytropic EoS:

$$p(\epsilon) = K_{nrel}\epsilon^{5/3} \tag{37}$$

with

$$K_{nrel} = \frac{\hbar^2}{15\pi^2 m_n}\left(\frac{3\pi^2}{m_n c^2}\right)^{5/3} \tag{38}$$

- Using the same process (and the same definitions) that was considered to obtain dimensionless differential equations in the case of white dwarfs, one can obtain a system of dimensionless differential equations also for these pure neutron stars.

Computational Analysis for Pure Neutron Stars

NON RELATIVISTIC Polytropic EoS in Newtonian regime

- Define the system of dimensionless differential equations:

```
#Define the dimensionless system of differential eqns

def neutron_star(r, y, w ,z):
    p ,m  = y
    if p <= 0:
        return np.array([0, 0])
    dpdr = -m/(r**2)*(p/w)**z
    dmdr = (r**2)*(p/w)**z
    return np.array([dpdr, dmdr])
```

- Define the central pressure and all the needed parameters for the system of dimensionless differential equations:

```
#Parameters for EoS

#non relativistic constant in the eos p=Knr*e**(5/3)
Knr = h_t**2/(15*np.pi**2*m_n)*(3*np.pi**2/(m_n*c**2))**(5/3)

#pressure in the centre of the neutron star
p_c=1e31

#matter density in the centre of the neutron star
ro_c=(p_c/Knr)**(3/5)*c**(-2)

#dimensionless param for dimensionless eos
w = Knr*c**(4/3)*ro_c**(2/3)

#inverse of gamma
z= 3/5

#initial values for the dimensionless system of differential eqns
p0=p_c/ro_c*c**(-2)
m0 = 0.0
y0 = np.array([p0, m0])
r0, rf, dr = 1e-6, 0.6, 0.0001
```

Computational Analysis for Pure Neutron Stars

NON RELATIVISTIC Polytropic EoS in Newtonian regime

- Solve the system of differential equations:

```
# Solve using RK4
r_values, y_values = rk4(neutron_star, y0, r0, rf, dr, w, z)
```

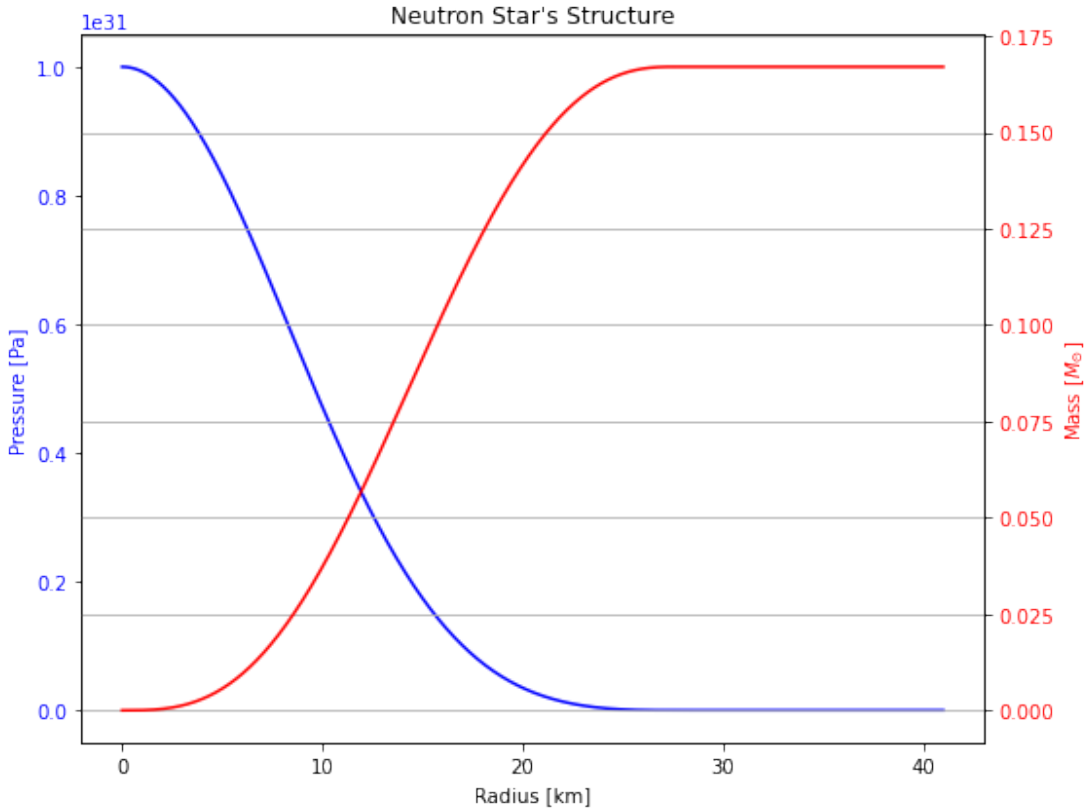
- Return to dimensionful quantities:

```
#Back to the dimensionful quantities

m_values= y_values[:, 1]*4*np.pi*ro_c/(ro_c*4*np.pi*G)**(3/2)*c**3/m_sun
p_values= y_values[:, 0]*ro_c*c**2
r_values1=r_values/(ro_c*4*np.pi*G)**(1/2)*c*10**(-3)
```

Computational Analysis for Pure Neutron Stars

NON RELATIVISTIC Polytropic EoS in Newtonian regime

- Plot the behaviour of the pressure and the mass inside the neutron star:



Computational Analysis for Pure Neutron Stars

NON RELATIVISTIC Polytropic EoS in Newtonian regime

- Find the final radius and mass of the neutron star:

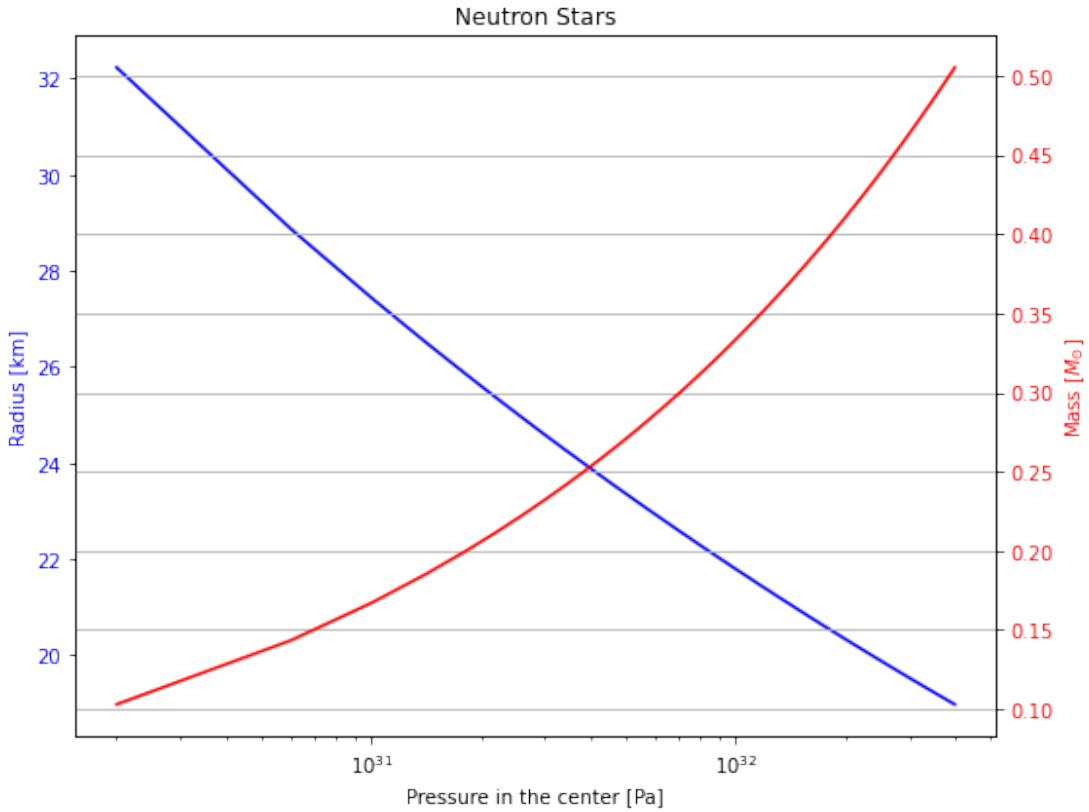
```
#Radius and Mass of the Neutron Star (Km and M_sun)
for i in range(len(p_values)):
    if p_values[i] <= 0:
        r_stop = r_values1[i]
        M_star = m_values[i]
        break

print(r_stop)
print(M_star)
```

```
27.444700095384896
0.16701177616868582
```

Computational Analysis for Pure Neutron Stars

NON RELATIVISTIC Polytropic EoS in Newtonian regime

- Plot the radii and masses of neutron stars for various central pressures:



Computational Analysis for Pure Neutron Stars

NON RELATIVISTIC Polytropic EoS in GR regime

- Define the system of differential equations and its solver by using the function *solveivp* from *scipy.integrate*

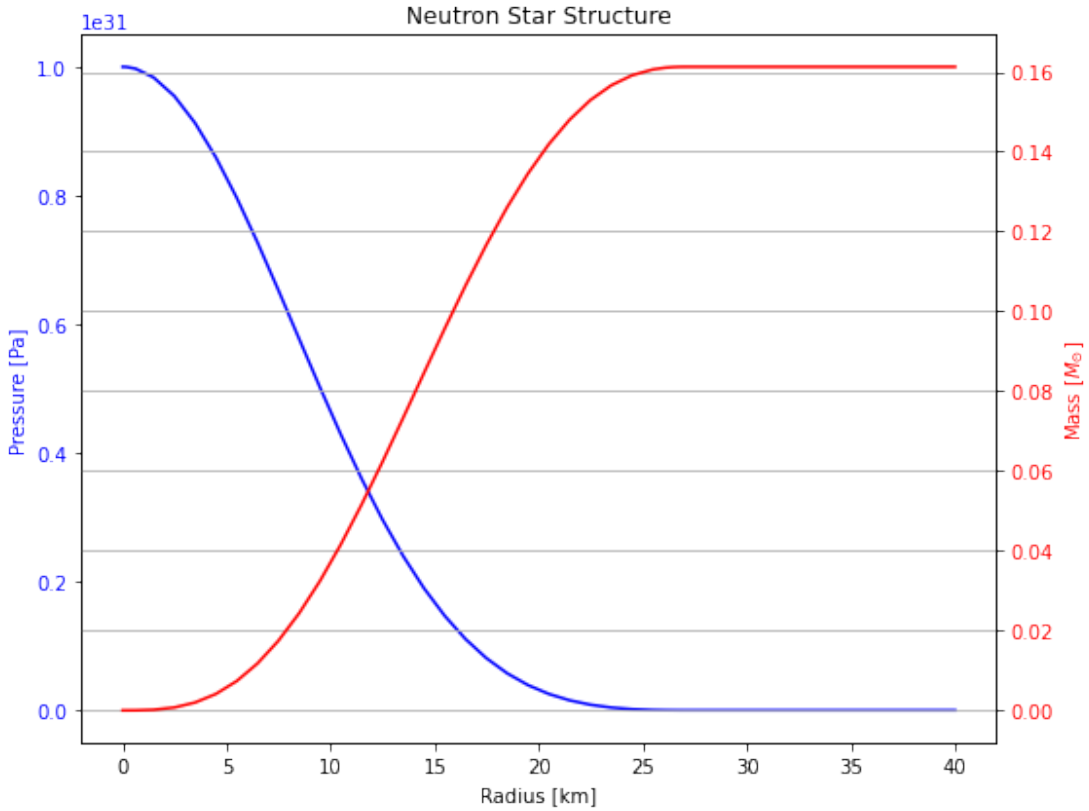
```
def tov_equations(r, y):
    p, m = y
    if p <= 0:
        return np.array([0, 0])

    e = (p / w)**z
    dmdr = 4 * np.pi * r**2 * e / c**2
    dpdr = -G * e * m / (c**2 * r**2) * (1 + p / e) * (1 + 4 * np.pi * r**3 * p / (m * c**2)) / (1 - 2 * G * m / (c**2 * r))
    return [dpdr, dmdr]


def solve_tov(p_c):
    y0 = [p_c, 1e-6]
    r_span = (1e-6, 4e4)

    sol = solve_ivp(tov_equations, r_span, y0, method='RK45', max_step=1e3)
    return sol
```

- Define central pressure p_c (same value as the one considered in the newtonian regime to compare the two cases)

Computational Analysis for Pure Neutron Stars

NON RELATIVISTIC Polytropic EoS in GR regime

- Plot the behaviour of the pressure and the mass inside the neutron star:



Computational Analysis for Pure Neutron Stars

NON RELATIVISTIC Polytropic EoS in GR regime

- Find the final radius and mass of the neutron star:

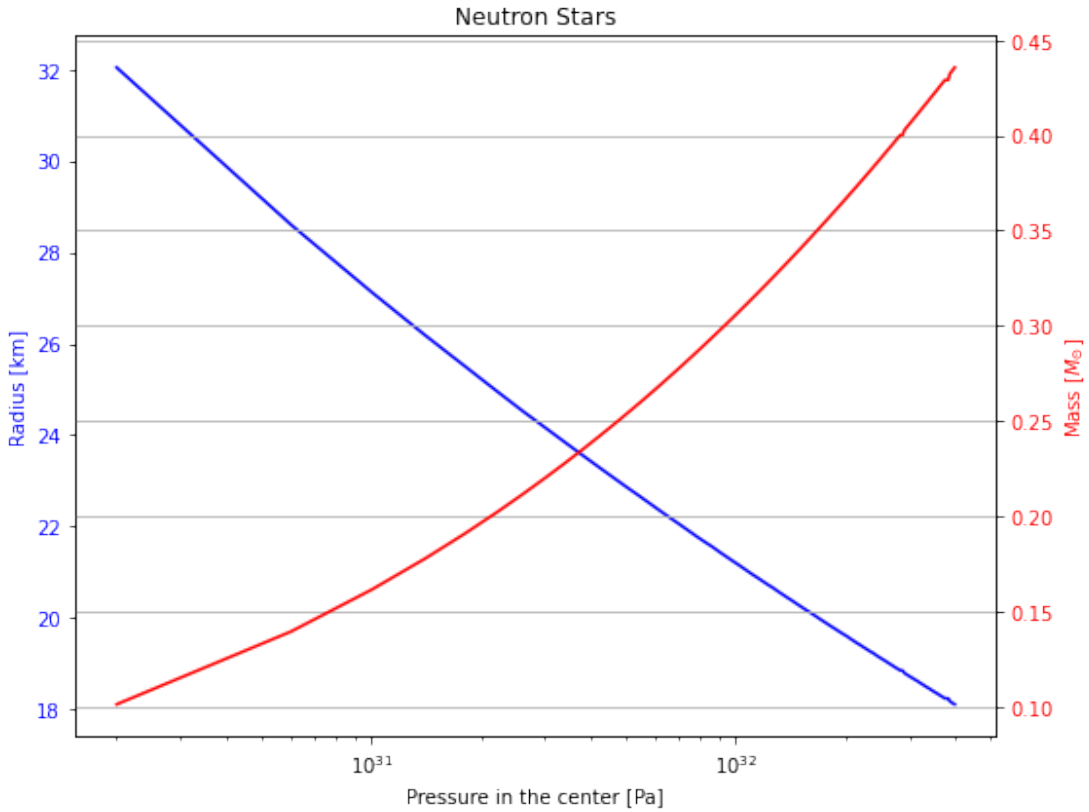
```
#Radius and Mass of the Neutron star (Km and M_sun)
for i in range(len(p_values)):
    if p_values[i] <= 0:
        r_stop = r_values1[i]
        M_star = m_values[i]
        break

print(r_stop)
print(M_star)
```

```
27.135347192907975
0.16126890962034873
```

Computational Analysis for Pure Neutron Stars

NON RELATIVISTIC Polytropic EoS in GR regime

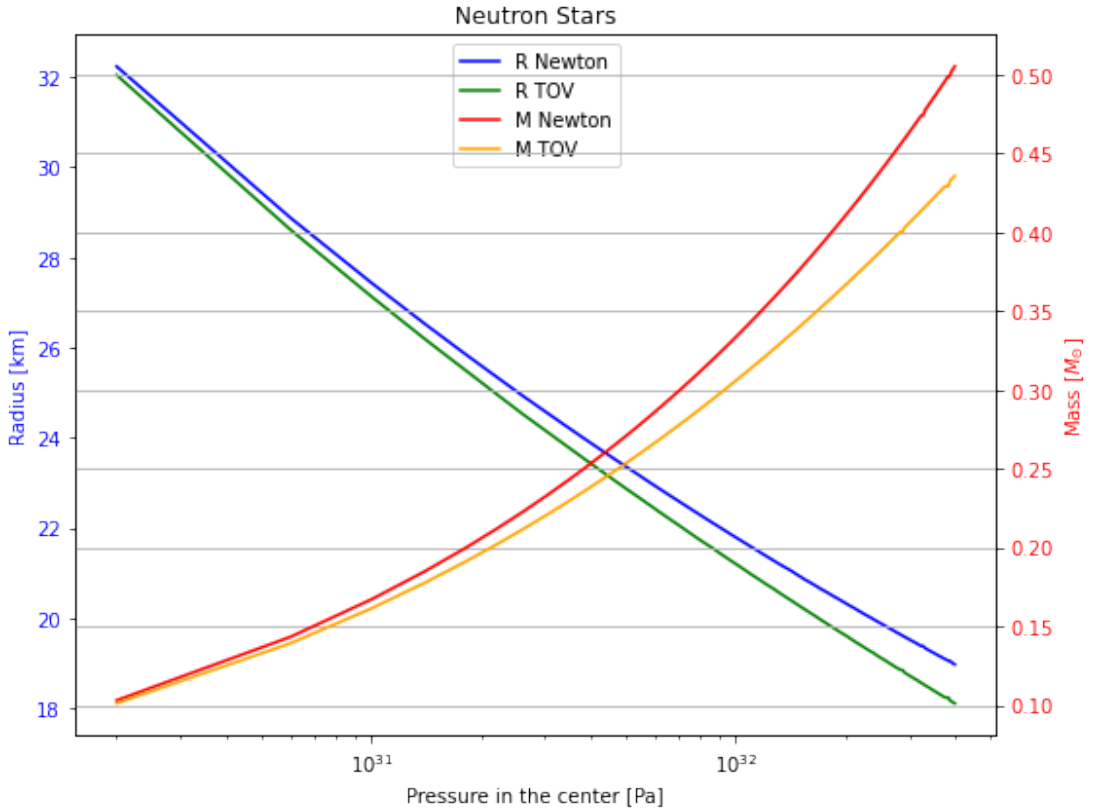
- Plot the radii and masses of neutron stars for various central pressures:



Computational Analysis for Pure Neutron Stars

NON RELATIVISTIC Polytropic EoS: Newtonian Regime VS GR regime

- Compare the results obtained from the Newtonian regime with the ones obtained in the GR one :



Computational Analysis for Pure Neutron Stars

Looking for zero points: code

- Once again one can use a root-finding routine to obtain an equation of state valid over the whole range of pressure and energy density.
- Define the explicit expressions for the pressure and the energy density
- Define a function which, via a root-finding method (brentq in this case), gives the energy density associated to a given pressure

```
e0=m_n**4*c**5/(np.pi**2*h_t**3)

#Explicit form of the pressure

def pressure(x, p_g):
    p=e0/24*((2*x**3-3*x)*(1+x**2)**(1/2)+3*np.arcsinh(x))
    p_diff=p-p_g
    return(p_diff)


#Explicit form of the energy
def energy(x):
    e_n=e0/8*((2*x**3+x)*(1+x**2)**(1/2)-np.arcsinh(x))
    return(e_n)


#Definition of a function which gives me the corresponding e for a given p
def energy_density(p):
    root = brentq(pressure, 0, 10000, args=(p))
    e=energy(root)
    return e
```

Computational Analysis for Pure Neutron Stars

Looking for zero-points: code

- Use this defined function inside the system of differential equations:

```
#Define TOV equations using the defined function that gives me the energy density for a given pressure

def tov_equations(r, y):
    p, m = y
    if p <= 0:
        return np.array([0, 0])
    e=energy_density(p)
    dmdr = 4 * np.pi * r**2 * e / c**2
    dpdr = -G * e * m / (c**2 * r**2) * (1 + p / e) * (1 + 4 * np.pi * r**3 * p / (m * c**2)) / (1 - 2 * G * m / (c**2 * r))
    return [dpdr, dmdr]
```

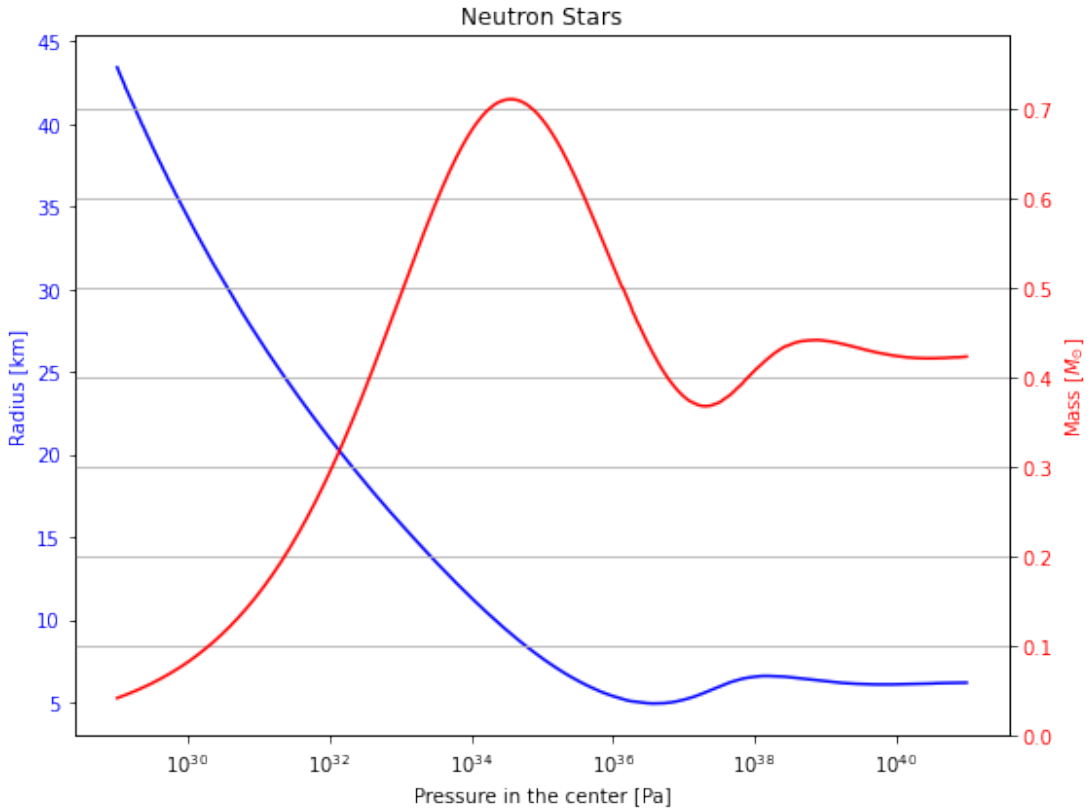
- Define the solver for the differential equations using *solveivp* from *scipy.integrate*

```
def solve_tov(p_c):
    y0 = [p_c, 1e-6]
    r_span = (1e-6, 5e4)

    sol = solve_ivp(tov_equations, r_span, y0, method='LSODA', max_step=1e4)
    return sol
```

Computational Analysis for Pure Neutron Stars

Looking for zero-points: code

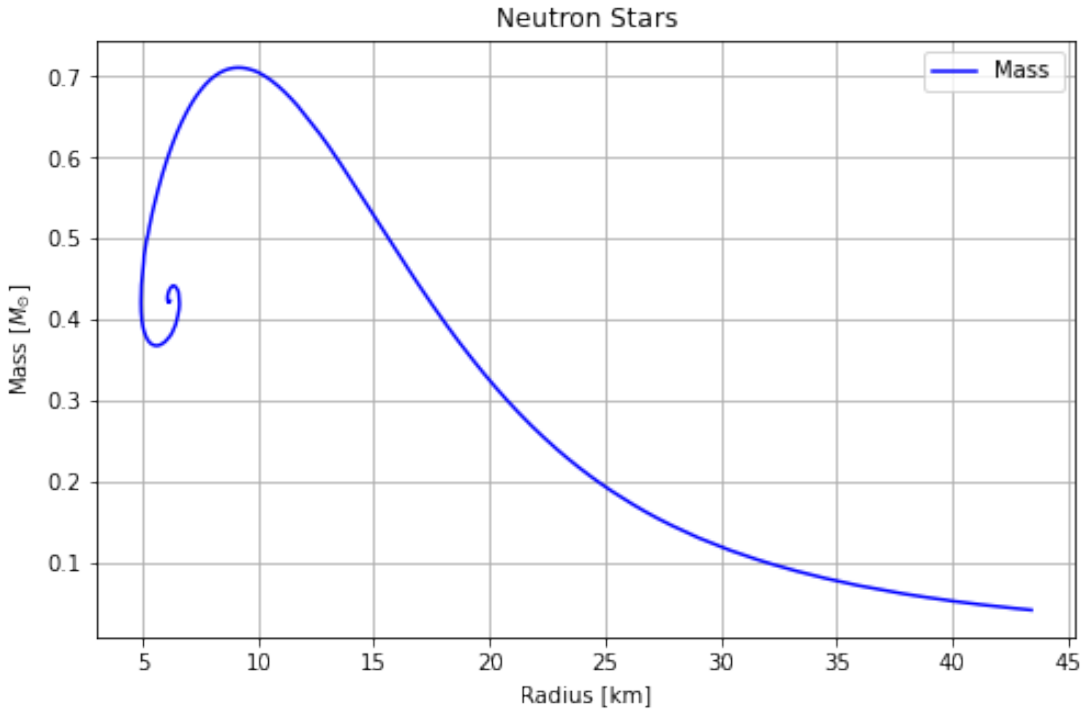
- Solve for many possible central pressures
- Plot the radii and masses of neutron stars for various central pressures:



Computational Analysis for Pure Neutron Stars

Looking for zero-points: code

- Plot the masses of various neutron stars with respect to their radii:



Neutron Stars with protons and electrons

Equations

- Since the neutron is unstable, a neutron star will not consist of neutrons only, but also of protons and electrons
- The electrons and protons are produced by β–decay:

$$n \longrightarrow p + e^- + \bar{\nu}_e. \tag{39}$$

- As all reactions are in chemical equilibrium, for the chemical potentials holds:

$$\mu_n = \mu_p + \mu_e. \tag{40}$$

- Due to stability reasons, it is assumed that the whole star is electrically neutral:

$$n_p = n_e \Leftrightarrow k_p = k_e \tag{41}$$

and the Fermi momenta of protons and electrons must be equal.

Neutron Stars with protons and electrons

Equations

- With the definition of the chemical potential for an ideal Fermi gas with zero temperature,

$$\mu_i(k_i) = \frac{d\epsilon_i}{dn_i} = (k_i^2c^2 + m_i^2c^4)^{1/2}, \quad i = n, p, e, \tag{42}$$

and using eqs. (40) and (41), one obtains:

$$(k_n^2c^2 + m_n^2c^4)^{1/2} - (k_p^2c^2 + m_p^2c^4)^{1/2} - (k_p^2c^2 + m_e^2c^4)^{1/2} = 0. \tag{43}$$

- This equation can be solved for the Fermi momenta of protons k_p as a function of the Fermi momenta of neutrons k_n:

$$\mathrm{k}_p(k_n) = \frac{[(k_n^2c^2+m_n^2c^4-m_e^2c^4)^2-2m_p^2c^4(k_n^2c^2+m_n^2c^4+m_e^2c^4)+m_p^4c^8]^{1/2}}{2c(k_n^2c^2+m_n^2c^4)^{1/2}} \tag{44}$$

Neutron Stars with protons and electrons

Equations

- One should notice that if there are no neutrons present, i.e. $k_n = 0$:

$$k_p(0) = 1.264 \cdot 10^{-3} m_n c. \tag{45}$$

  Therefore equation 44 can only be used if $k_p > 1.264 \cdot 10^{-3} m_n c$, which is fulfilled for $p > p_{\mathrm{crit}} = 3.038 \cdot 10^{23}\ \mathrm{Pa}$.

- Below this pressure, no neutrons are present, so the "neutron star" consists of protons and electrons only.

Neutron Stars with protons and electrons

Equations

- The total energy and the total pressure are the sum of the energy and pressure of electrons, protons and neutrons:

$$\epsilon_{tot} = \sum_{i=n,p,e} \epsilon_i \,, \quad p_{tot} = \sum_{i=n,p,e} p_i \,, \tag{46}$$

where

$$\epsilon_i(k_i) = \frac{8\pi}{(2\pi\hbar)^3} \int_0^{k_i} (k^2c^2 + m_i^2c^4)^{1/2} k^2 dk \tag{47}$$

$$p_i(k_i) = \frac{1}{3}\frac{8\pi}{(2\pi\hbar)^3} \int_0^{k_i} (k^2c^2 + m_i^2c^4)^{-1/2} k^4 dk. \tag{48}$$

Computational Analysis for Neutron Stars with protons and electrons

Equation of State

- One can obtain the equation of state using these previous expressions
- To do this one considers the same root-finding process to get a function which , for a given pressure, returns the associated energy density
- In this case one can define dimensionless quantities such as $x = k_n/m_nc$, $w = k_p/m_pc$ and $z = k_e/m_ec$ that must be related consistently to 41 and 44.
- One should also pay attention to the existence of a critical pressure, under which there are no neutrons.

Computational Analysis for Neutron Stars with protons and electrons

Equation of State

```
#Building the equation of state

e0_p = m_p**4 * c**5 / (np.pi**2 * h_t**3)
e0_n = m_n**4 * c**5 / (np.pi**2 * h_t**3)
e0_e = m_e**4 * c**5 / (np.pi**2 * h_t**3)

#Define dimensionless quantities for the root finding of the momentum

#Dimensionless parameter for the proton w=k_p/(m_p*c) as a function of x=k_n/(m_n*c)
def proton_w(x):
    term1 = (x**2 + 1 - (m_e**2 / m_n**2))**2
    term2 = -2 * (m_p**2 / m_n**2) * (x**2 + 1 + (m_e**2 / m_n**2))
    term3 = (m_p**4 / m_n**4)
    numerator = np.sqrt(m_n**4 * (term1 + term2 + term3))
    denominator = 2 * m_p * m_n * np.sqrt(x**2 + 1)
    return numerator / denominator

#Dimensionless parameter for the electron z=k_e/(m_e*c)=k_p/(m_e*c) as a function of w=k_p/(m_p*c)
def electron_z(w):
    return w * (m_p / m_e)

#Define difference between pressure given by protons, electrons and neutrons and a given pressure
def pressure_tot_func(x, p_g):
    w = proton_w(x)
    z = electron_z(w)
    p_p = e0_p / 24 * ((2 * w**3 - 3 * w) * np.sqrt(1 + w**2) + 3 * np.arcsinh(w))
    p_e = e0_e / 24 * ((2 * z**3 - 3 * z) * np.sqrt(1 + z**2) + 3 * np.arcsinh(z))
    p_n = e0_n / 24 * ((2 * x**3 - 3 * x) * np.sqrt(1 + x**2) + 3 * np.arcsinh(x))
    return p_n + p_e + p_p - p_g

#Define energy density given by protons, electrons and neutrons
def energy_tot_func(x):
    w = proton_w(x)
    z = electron_z(w)
    e_p = e0_p / 8 * ((2 * w**3 + w) * np.sqrt(1 + w**2) - np.arcsinh(w))
    e_e = e0_e / 8 * ((2 * z**3 + z) * np.sqrt(1 + z**2) - np.arcsinh(z))
    e_n = e0_n / 8 * ((2 * x**3 + x) * np.sqrt(1 + x**2) - np.arcsinh(x))
    return e_n + e_e + e_p
```

Computational Analysis for Neutron Stars with protons and electrons

Equation of State

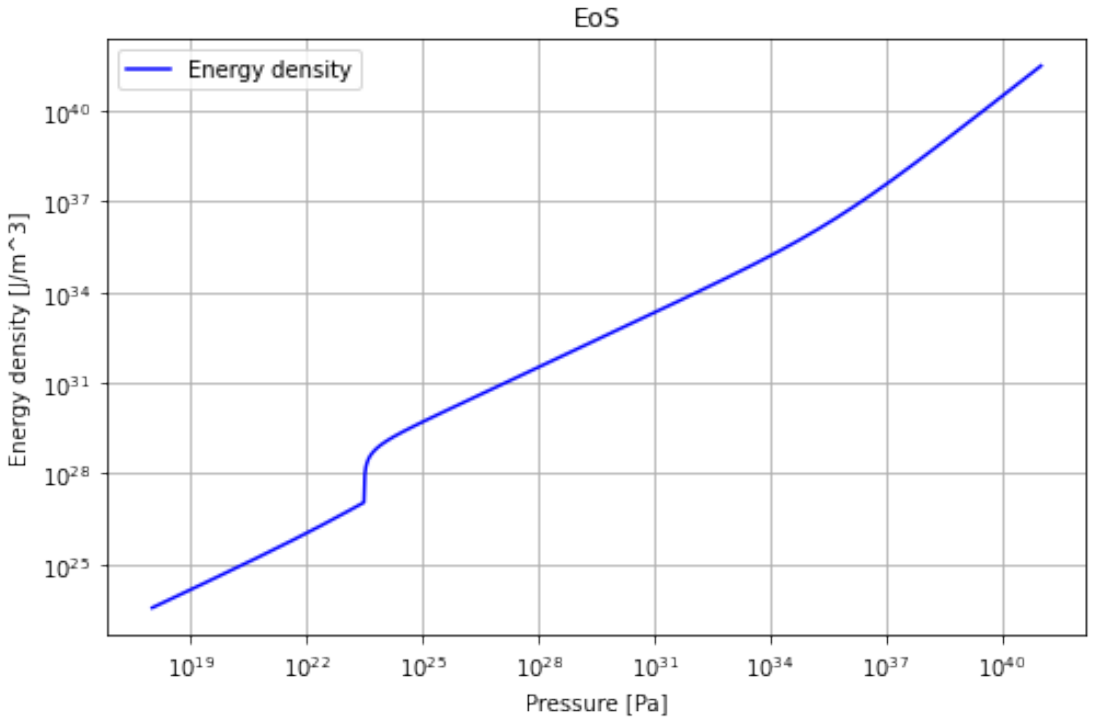
```
#Define difference between pressure given only by protons and electrons and a given pressure (happens under criti
def pressure_1tot_func(u, p_g):
    z = electron_z(u)
    p_p = e0_p / 24 * ((2 * u**3 - 3 * u) * np.sqrt(1 + u**2) + 3 * np.arcsinh(u))
    p_e = e0_e / 24 * ((2 * z**3 - 3 * z) * np.sqrt(1 + z**2) + 3 * np.arcsinh(z))
    return p_p + p_e - p_g

#Define energy density when there are only protons and electrons, which happens under a critical pressure
def energy_1tot_func(u):
    z = electron_z(u)
    e_p = e0_p / 8 * ((2 * u**3 + u) * np.sqrt(1 + u**2) - np.arcsinh(u))
    e_e = e0_e / 8 * ((2 * z**3 + z) * np.sqrt(1 + z**2) - np.arcsinh(z))
    return e_p + e_e
```

```
#Define total energy density: this function gives me the corresponding energy density for a given p
#taking into account that there exists a critical pressure
def energy_density(p):
    p_threshold = 3.054e23

    if p < p_threshold:
        root = brentq(pressure_1tot_func, -1000, 1000, args=(p,))
        return energy_1tot_func(root)
    else:
        root = brentq(pressure_tot_func, -1000, 1000, args=(p,))
        return energy_tot_func(root)
```

Computational Analysis for Neutron Stars with protons and electrons

Equation of State

- One can obtain the plot of the equation of state:



Computational Analysis for Neutron Stars with protons and electrons

Codes

- Use this defined function inside the differential equations:

```
#Define TOV equations using the defined function that gives me the energy density for a given pressure

def tov_equations(r, y):
    p, m = y
    if p <= 0:
        return np.array([0, 0])
    e=energy_density(p)
    dmdr = 4 * np.pi * r**2 * e / c**2
    dpdr = -G * e * m / (c**2 * r**2) * (1 + p / e) * (1 + 4 * np.pi * r**3 * p / (m * c**2)) / (1 - 2 * G * m / (c**2 * r))
    return [dpdr, dmdr]
```

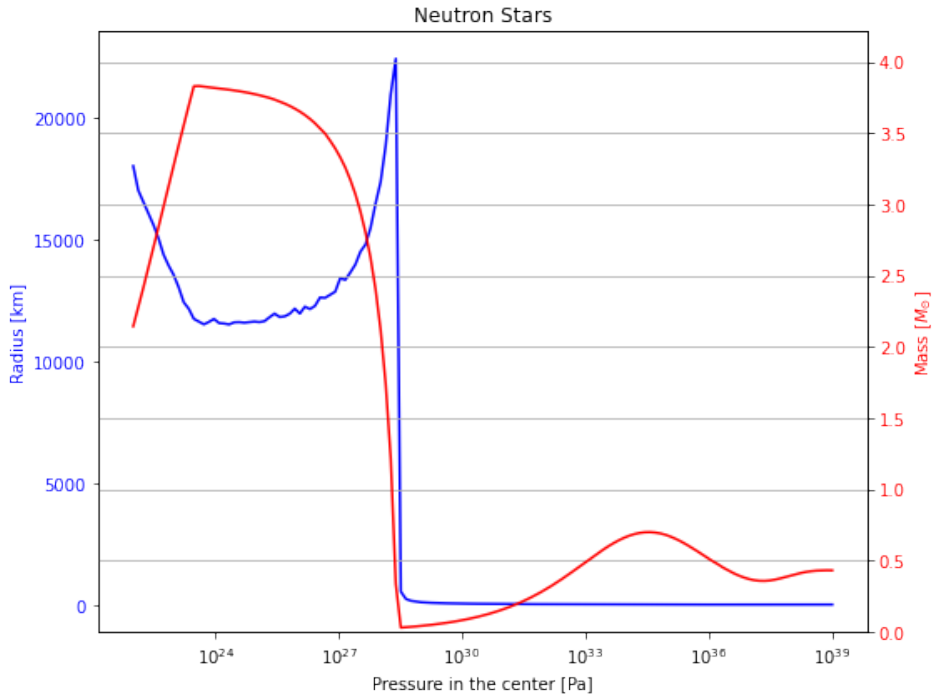
- Define the solver for the differential equations using *solveivp* from *scipy.integrate*

```
def solve_tov(p_c):
    y0 = [p_c, 1e-6]
    r_span = (1e-6, 3e7)

    sol = solve_ivp(tov_equations, r_span, y0, method='LSODA', max_step=5e2)
    return sol
```

Computational Analysis for Neutron Stars with protons and electrons

Codes

- Solve for many possible central pressures
- Plot the radii and masses of the neutron stars for various central pressures



Computational Analysis for Neutron Stars with protons and electrons

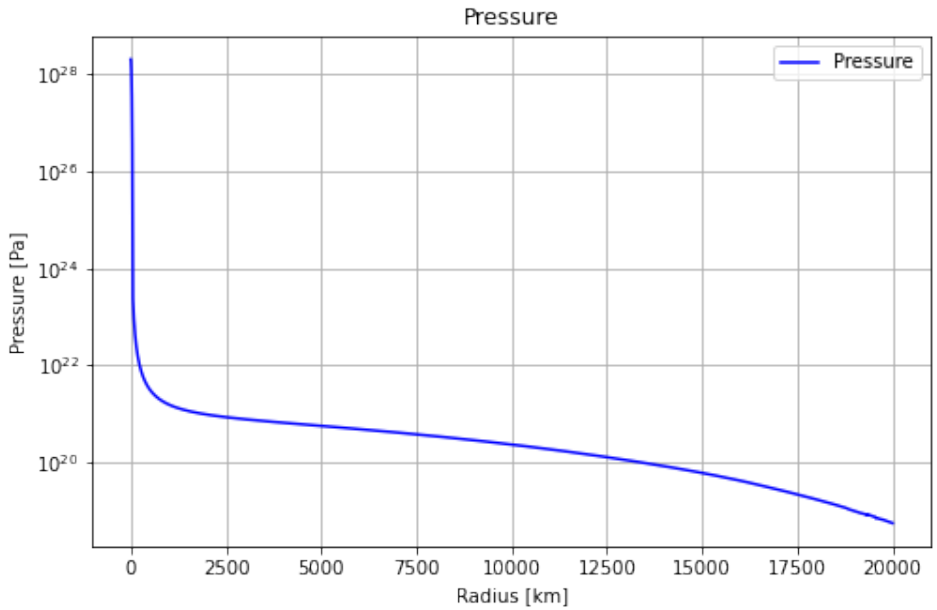
Explanation

- Below the critical pressure, the neutron star consists of protons and electrons. Above this pressure, neutrons appear resulting in a bend of $M(p_c)$ and $R(p_c)$.
- The unusual peak of $R(p_c)$ can be explained by the fact that around $p_c \simeq 10^{28}\ \mathrm{Pa}$ the pressure approaches zero very slowly but never reaches it or falls below.
- Therefore, it is necessary to introduce a cutoff ($p = 10^{18}\ \mathrm{Pa}$), so that the integration terminates as soon as this pressure is reached.
- This kind of neutron star consists of a dense central core and a huge low-density crust.

Computational Analysis for Neutron Stars with protons and electrons

Codes

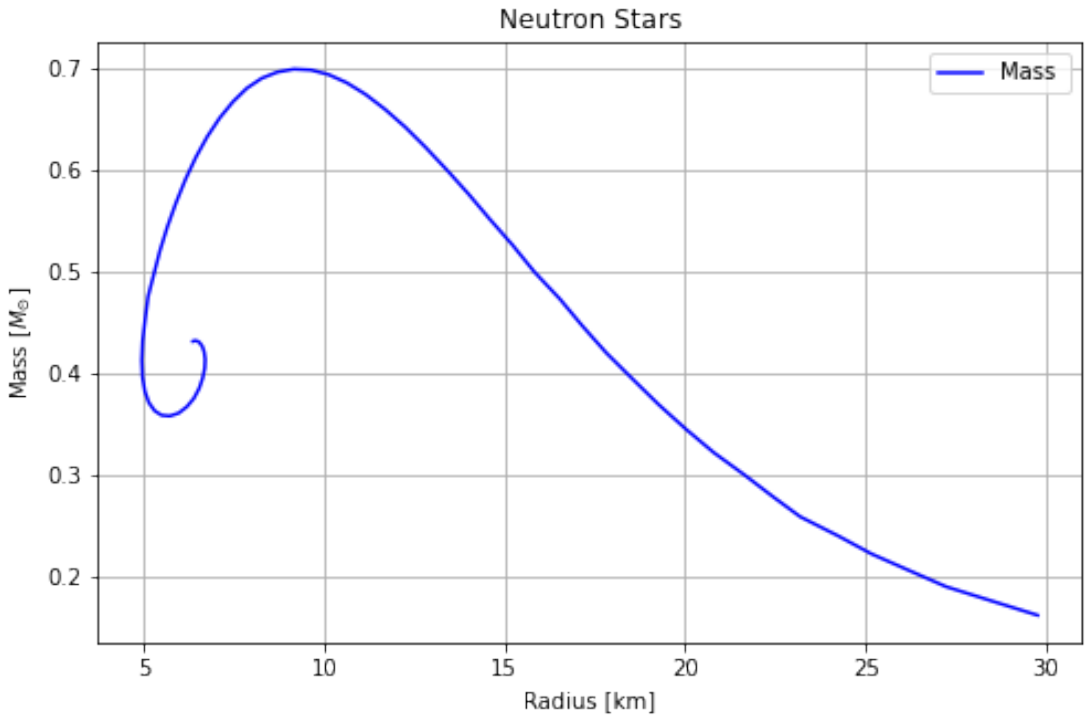
- One can see the pressure's particular behaviour mentioned above by plotting $p(r)$ in an neutron star for a given central pressure
- In this case $p_c = 2 \cdot 10^{28}\ \mathrm{Pa}$ and one considers the same equation of state used before



Computational Analysis for Neutron Stars with protons and electrons

Codes

- Plot the masses of various neutron stars with protons and electrons with respect to their radii:



Neutron Stars compOSE

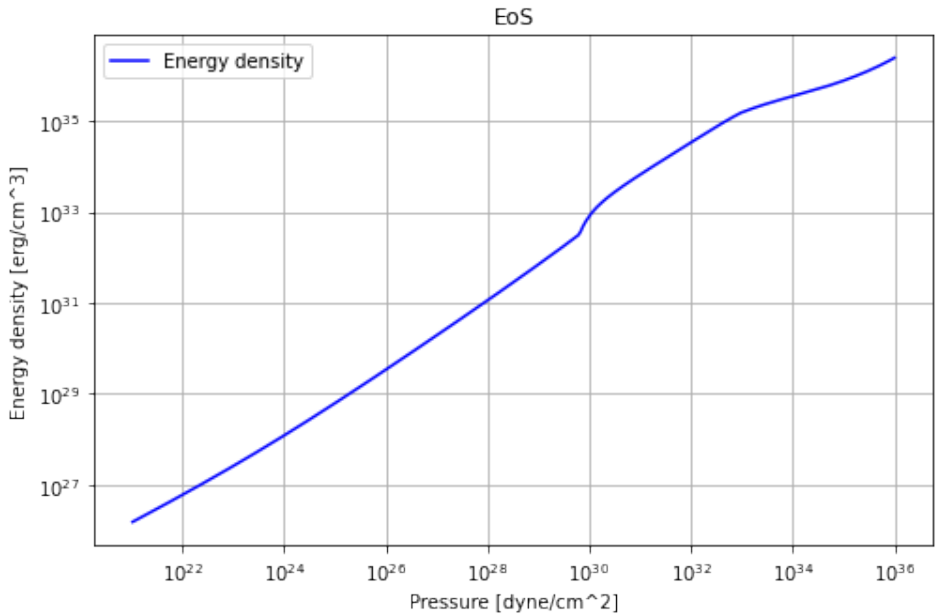
compOSE

- The online service CompOSE provides data tables for different equations of state (EoS).
- One can download their data, plot the equations of state and then study the resulting neutron stars.

Neutron Stars compOSE

RMF1 at T=0

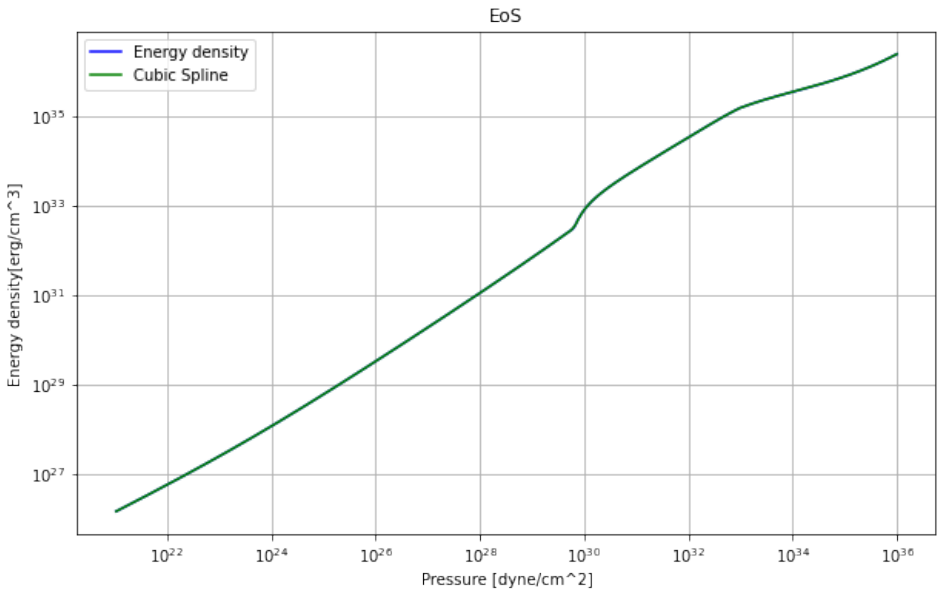
- This EoS describes a neutron star with a core of homogeneous n,p,e matter and a reconstructed crust.
- The EoS can be plotted using the downloaded data.



Neutron Stars compOSE

RMF1 at T=0

- One can use *Cubic spline* to get a function which fits the downloaded data.
- The downloaded data and the obtained function can be plotted together and compared.



Neutron Stars compOSE

RMF1 at T=0

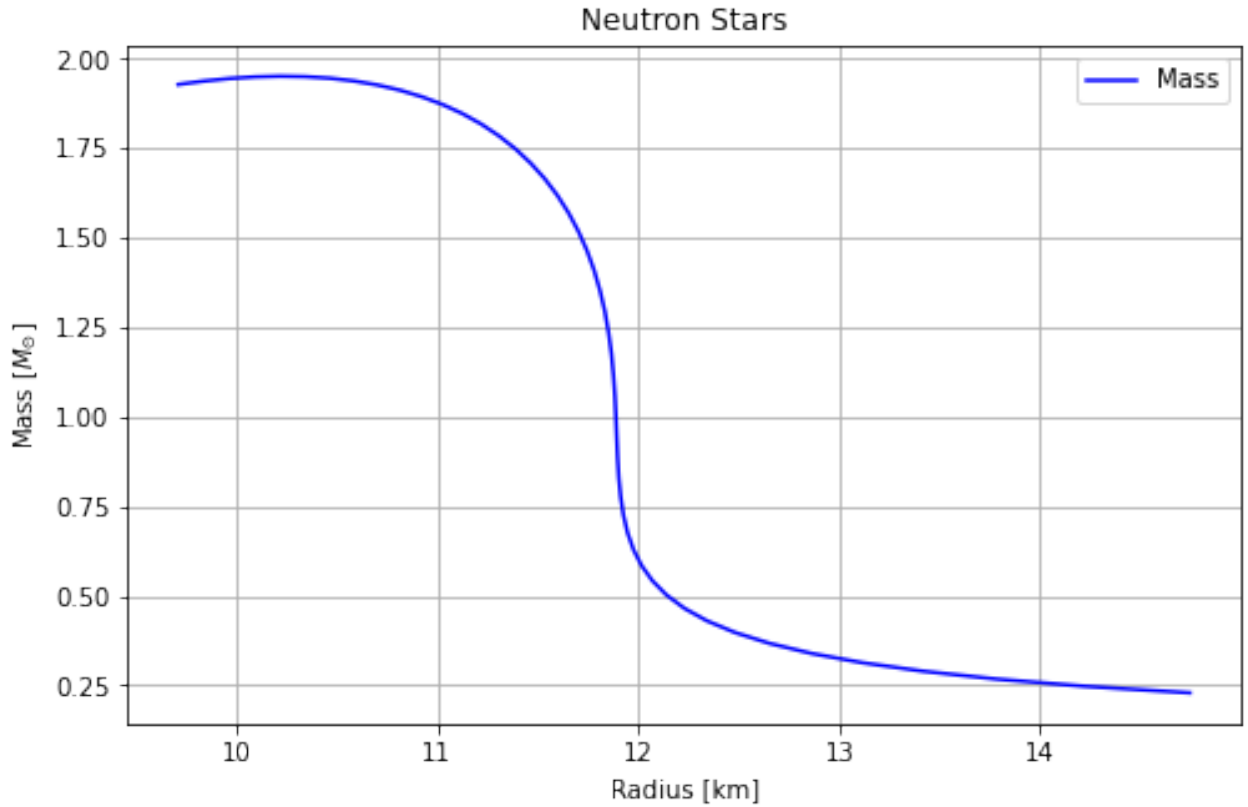
- At this point, one can use this EoS to solve the TOV equations for many possible central pressures.

```
def tov_equations(r, y):
    p, m = y
    e=eos(p)
    dmdr = 4 * np.pi * r**2 * e / c**2
    dpdr = -G * e * m / (c**2 * r**2) * (1 + p / e) * (1 + 4 * np.pi * r**3 * p / (m * c**2)) / (1 - 2 * G * m / (c**2 * r))
    return [dpdr, dmdr]


def solve_tov(p_c):
    y0 = [p_c, 1e-6]
    r_span = (1e-6, 5e8)

    sol = solve_ivp(tov_equations, r_span, y0, method='LSODA', max_step=1e3)

    return sol
```

Neutron Stars compOSE

RMF1 at T=0

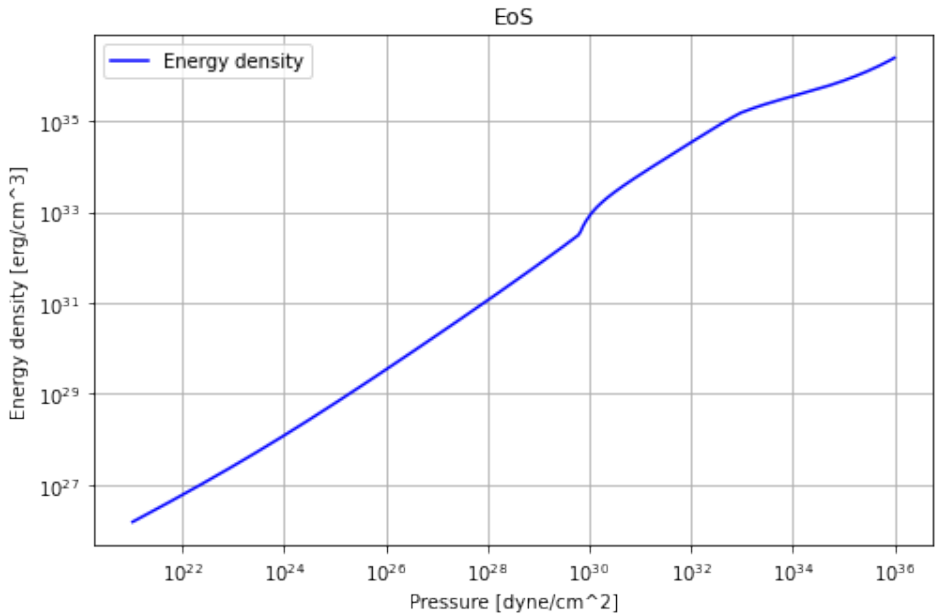
- Plot of the masses of various neutron stars with respect to their radii:



Neutron Stars compOSE

RMF2 at T=0

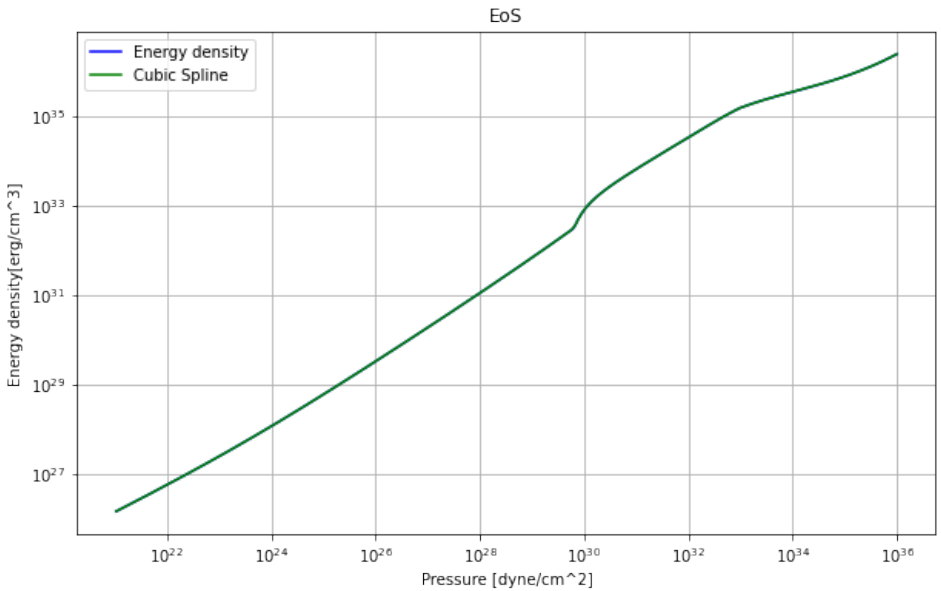
- This EoS is in β-equilibrium at T $= 0$.
- The core is homogeneous n, p, e matter.
- The EoS can be plotted using the downloaded data.



Neutron Stars compOSE

RMF2 at T=0

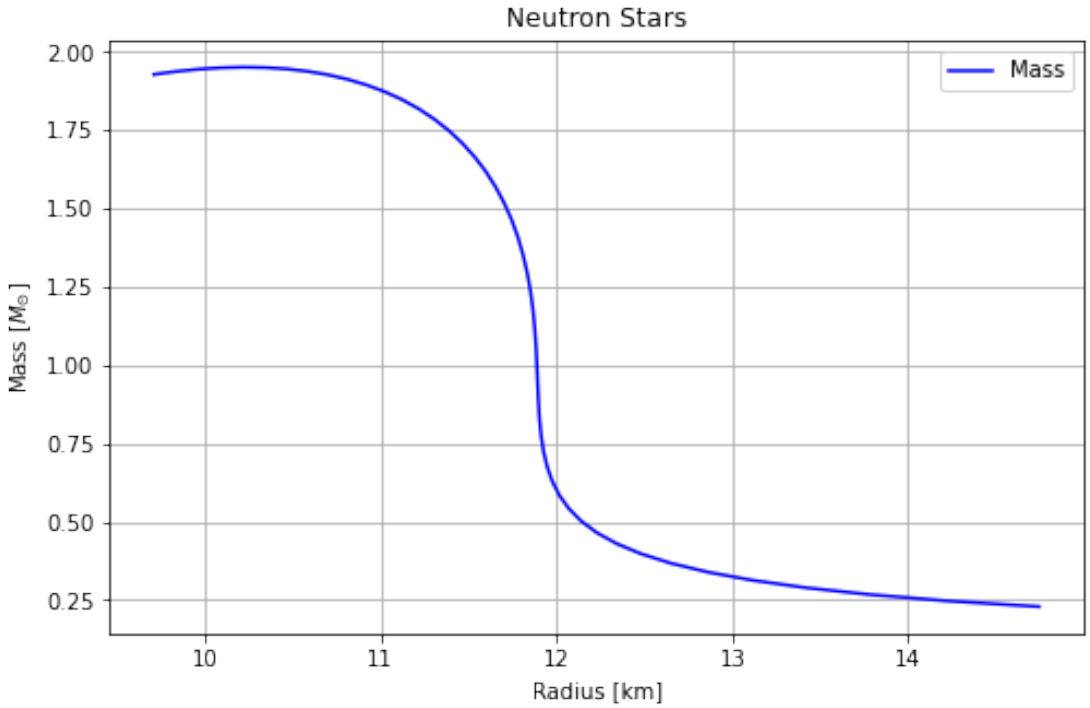
- One can use *Cubic spline* to get a function which fits the downloaded data.
- The downloaded data and the obtained function can be plotted together and compared.



Neutron Stars compOSE

RMF2 at T=0

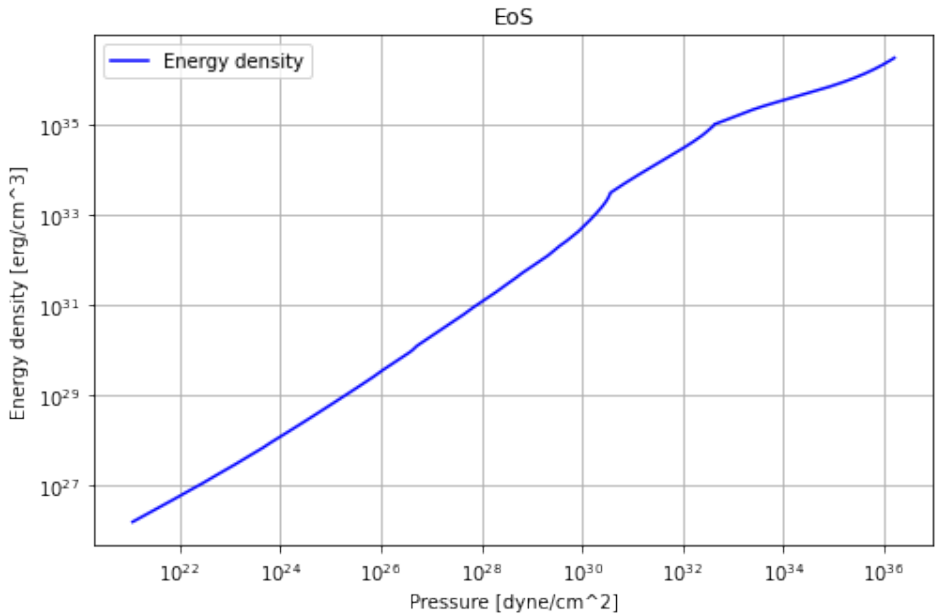
- At this point, one can use this EoS to solve the TOV equations for many possible central pressures.
- Plot of the masses of various neutron stars with respect to their radii:



Neutron Stars compOSE

RMF3 at T=0

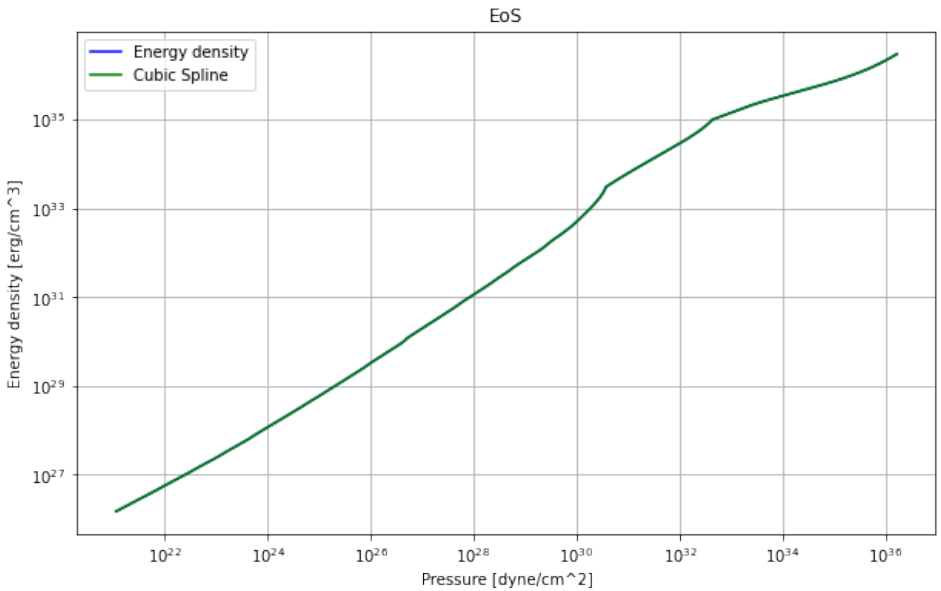
- This EoS is in β-equilibrium at T $= 0$.
- The core is homogeneous n, p, e matter.
- The EoS can be plotted using the downloaded data.



Neutron Stars compOSE

RMF3 at T=0

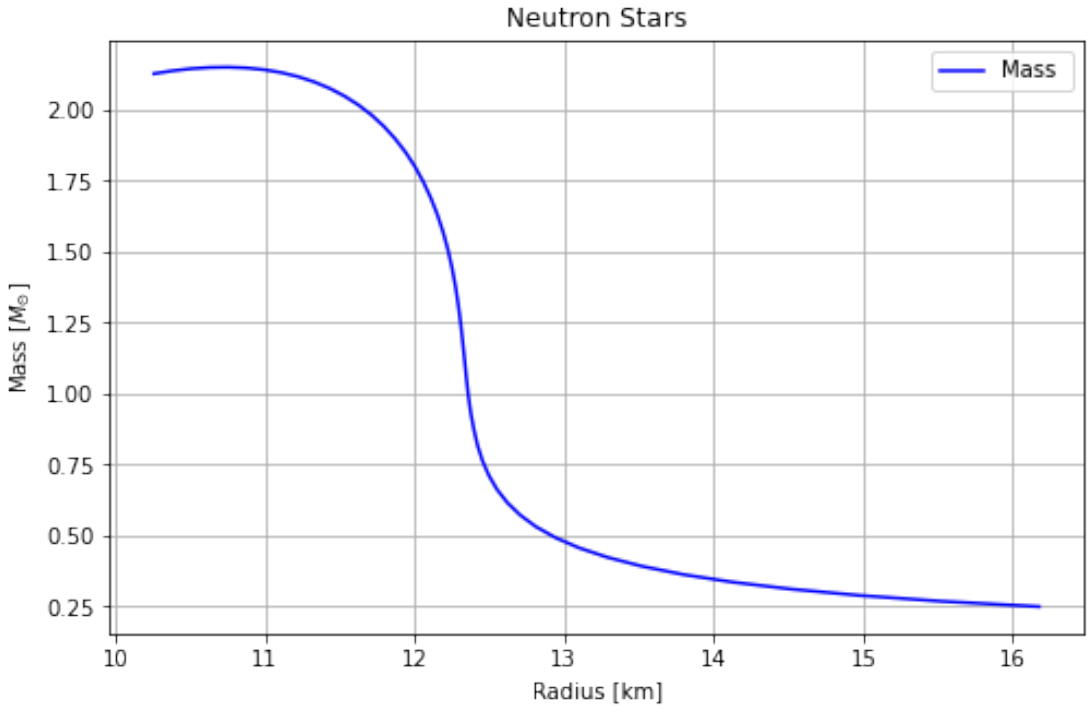
- One can use *Cubic spline* to get a function which fits the downloaded data.
- The downloaded data and the obtained function can be plotted together and compared.



Neutron Stars compOSE

RMF3 at T=0

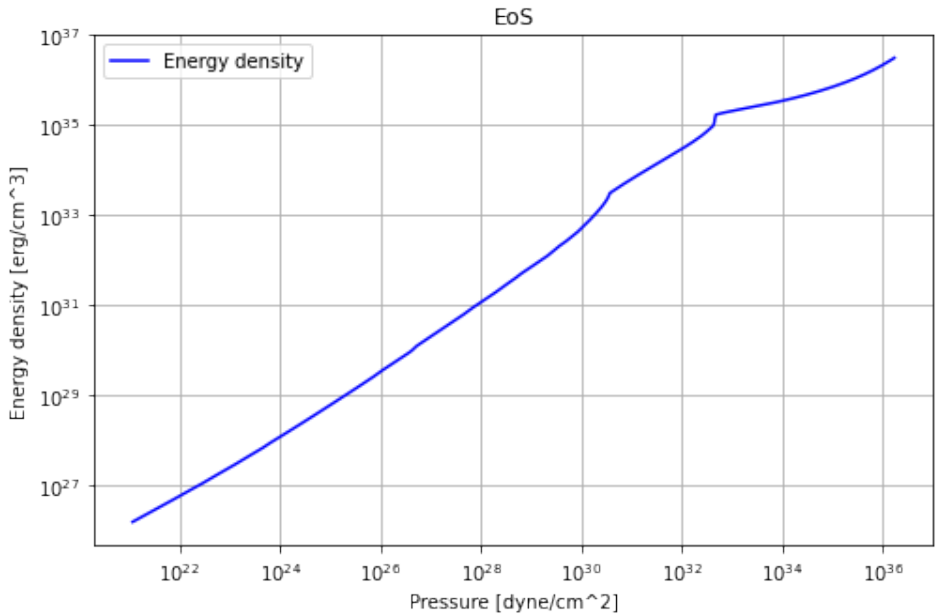
- At this point, one can use this EoS to solve the TOV equations for many possible central pressures.
- Plot of the masses of various neutron stars with respect to their radii:



Neutron Stars compOSE

RMF4 at T=0

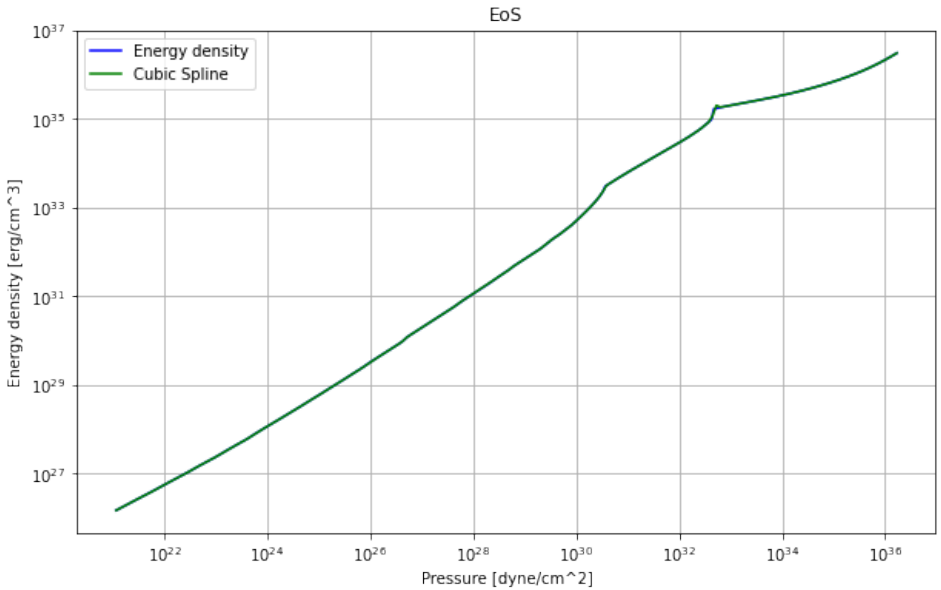
- This EoS is in β-equilibrium at T $= 0$.
- The core is homogeneous n, p, e matter.
- The EoS can be plotted using the downloaded data.



Neutron Stars compOSE

RMF4 at T=0

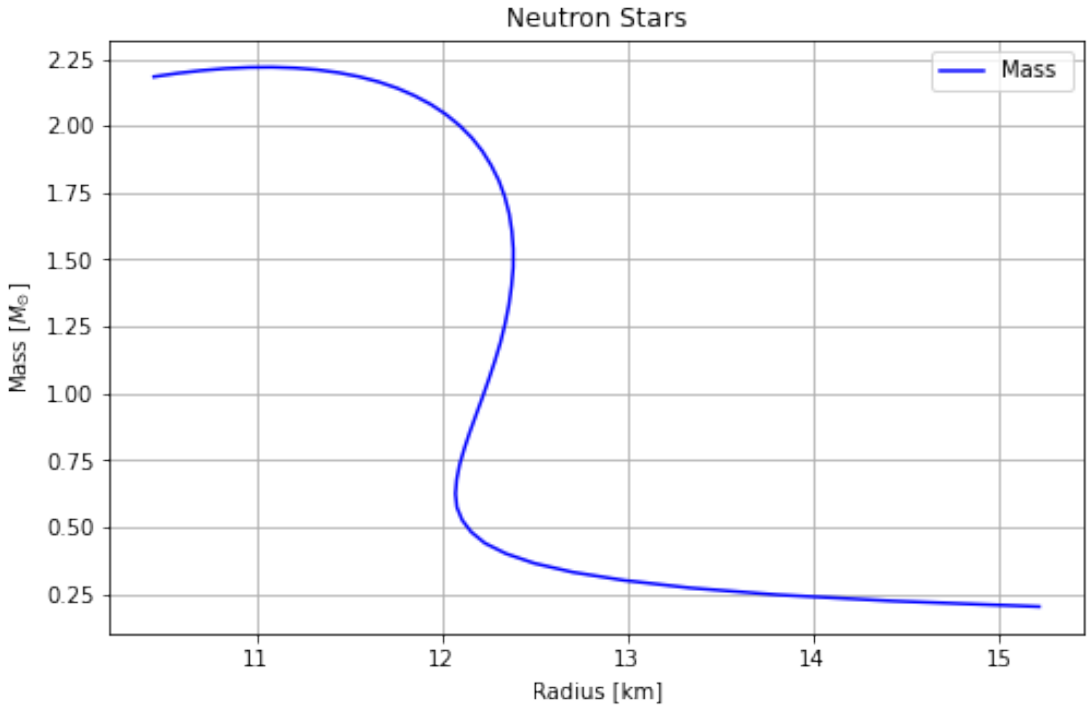
- One can use *Cubic spline* to get a function which fits the downloaded data.
- The downloaded data and the obtained function can be plotted together and compared.



Neutron Stars compOSE

RMF4 at T=0

- At this point, one can use this EoS to solve the TOV equations for many possible central pressures.
- Plot of the masses of various neutron stars with respect to their radii:



References

- *Compact Stars for Undergraduates* , Irina Sagert, Matthias Hempel, Carsten Greiner, and Jurgen Schaffner-Bielich
- *Computational Physics*, *Lecture Notes Fall* 2015, M. Hjorth-Jensen (in particular Part III)

Thank you for the attention!

May 2025