Neutron stars

The Tolman – Oppenheimer – Volkoff equations

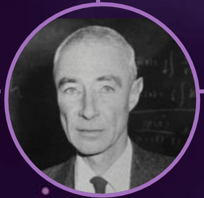
Nicholas Pieretti

Theoretical and Numerical aspects of Nuclear Physics

The discovery

1054: Chinese observation of Crab supernova

1939: First neutron star models by **Tolman**, **Oppenheimer** and **Volkoff**



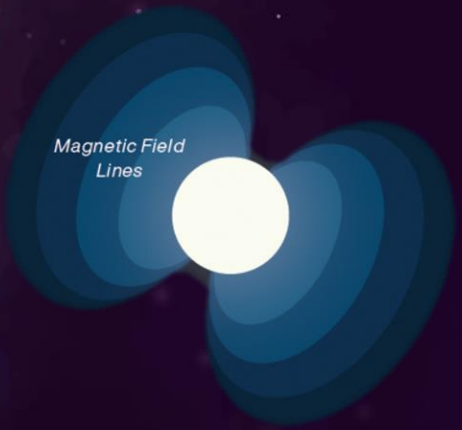
1932: Neutron discovery by **Chadwick** leads to **Landau's** prediction of neutron stars

1967: Bell and **Hewish** discover the first neutron star as a radio pulsar

Neutron star types

Magnetar

Extremely high magnetic fields:
$10^{13} - 10^{15}$ Gauss
Long rotation periods:
$5 - 12$ seconds

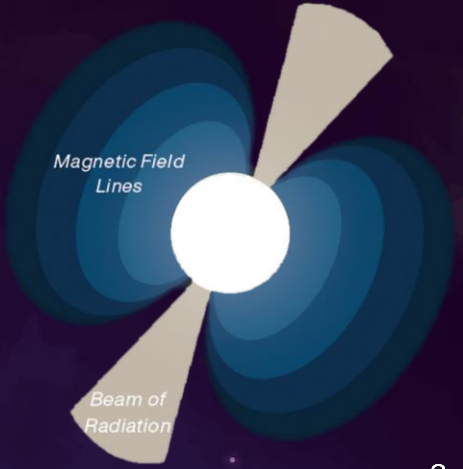


Pulsar

Emit twin beams of radiation from magnetic poles.
Short rotation periods:
down to milliseconds

… and both

Only 6 detected so far.
Could emit bursts at irregular intervals



15 – 30 km

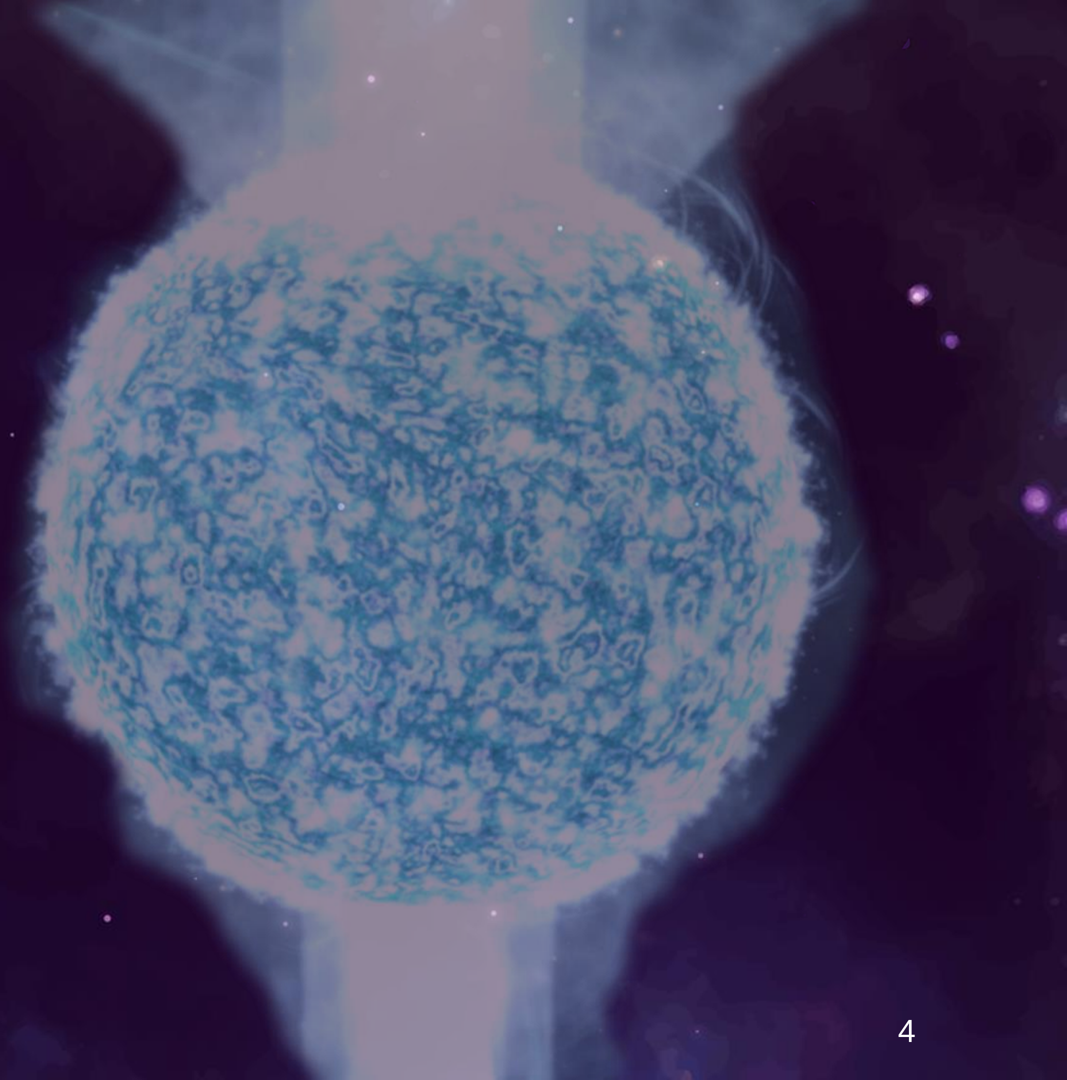
Diameter

$\geq 10\ M_{\odot}$

Original star mass

1-2 $M_{\odot}$

Total mass

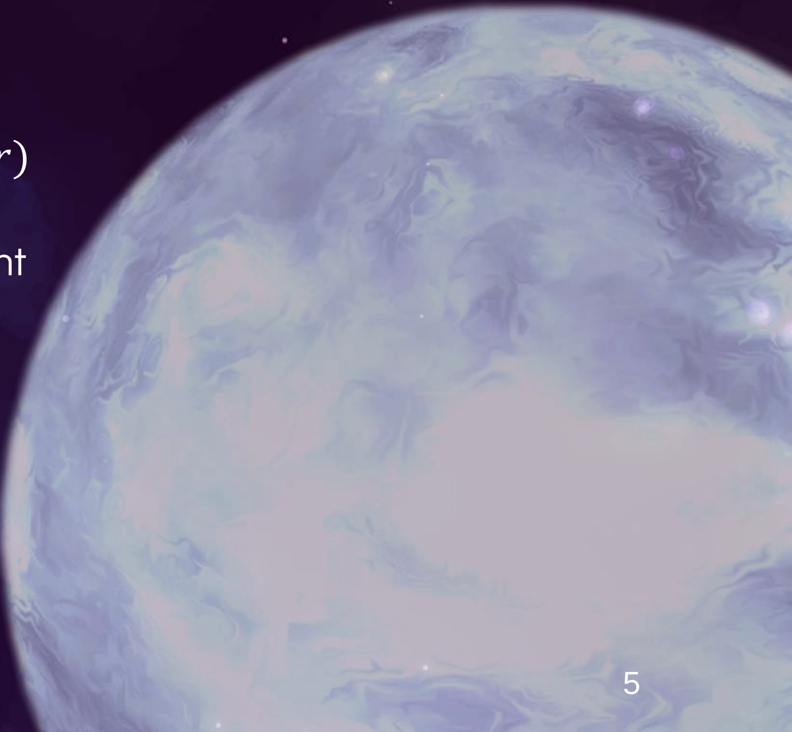


Non-relativistic equilibrium equations

Gravitational force for an element of volume:

$$F_G = -\frac{Gm(r)}{r^2}\,\rho(r)$$

- $G = 6.67 \times 10^{-11}\mathrm{m}^3\ /\ \mathrm{kgs}^2$ gravitational constant
- ρ = mass density
- $m(r) = 4\pi \int_0^r \rho(r')r'^2 dr'$



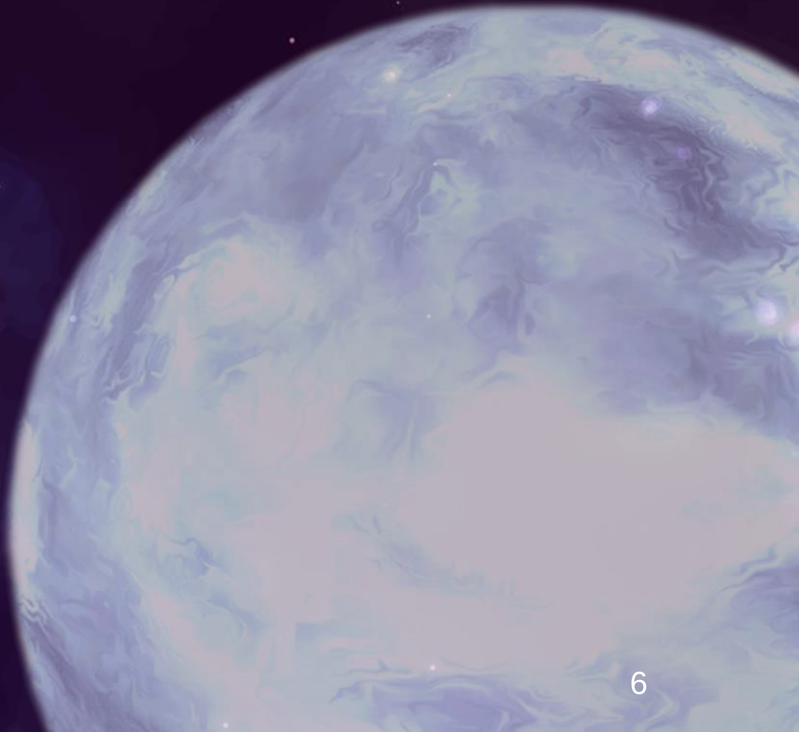
Non-relativistic equilibrium equations

The structure equations are

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

$$\frac{dP}{dr} = -\frac{Gm(r)}{r^2}\rho(r) = -\frac{Gm(r)}{r^2}\frac{\epsilon(r)}{c^2}$$

- $\epsilon = \rho c^2$ energy density
- Valid when $\frac{V(r)}{mc^2} = \frac{Gm}{rc^2} \ll 1$

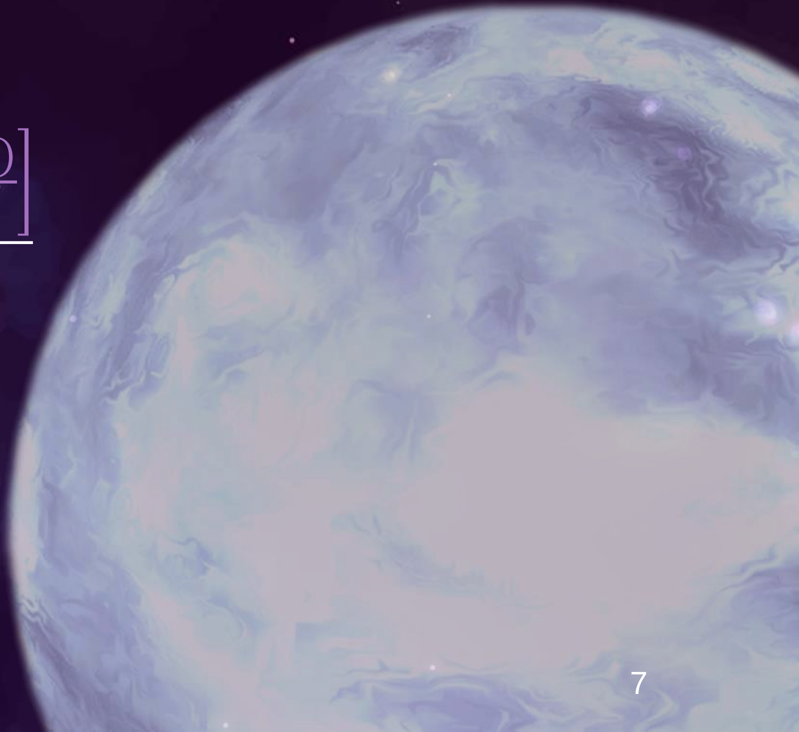


Tolman – Oppenheimer – Volkoff equations

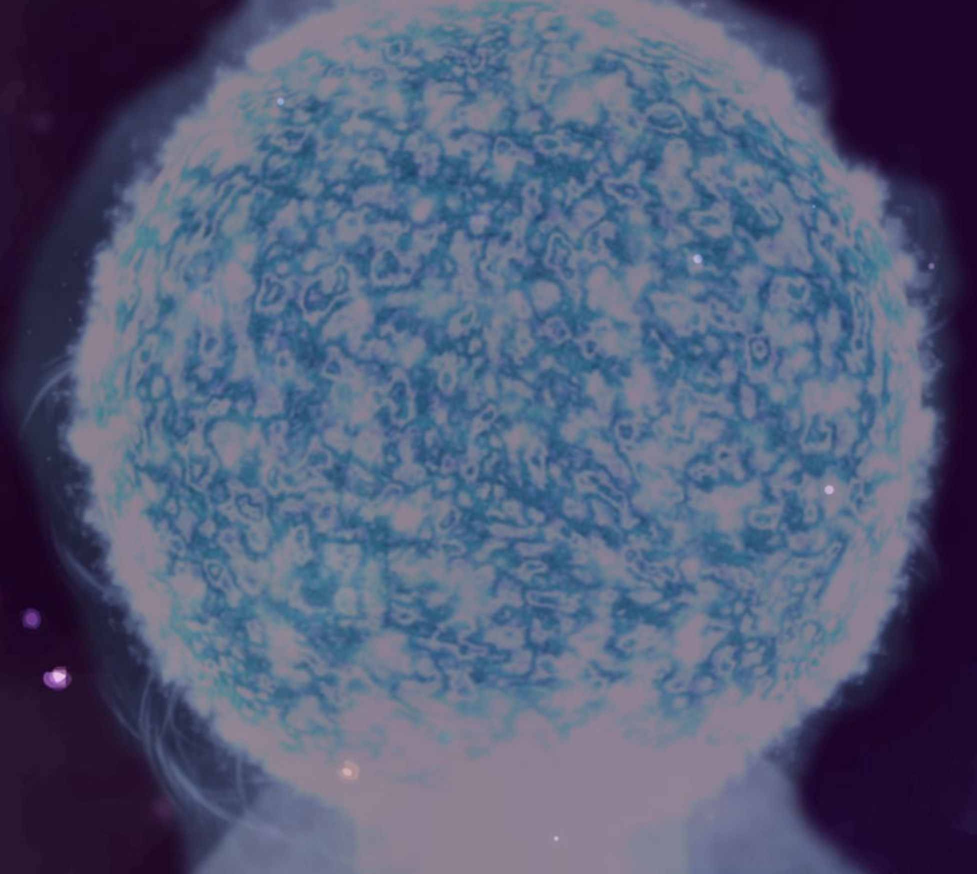
$$\frac{dm}{dr} = 4\pi r^2 \rho(r)$$

$$\frac{dP}{dr} = -\frac{Gm(r)}{r^2}\rho(r)\frac{\left[1+\frac{P(r)}{\rho(r)c^2}\right]\left[1+\frac{4\pi r^3 P(r)}{m(r)c^2}\right]}{\left[1-\frac{2Gm(r)}{rc^2}\right]}$$

- Account for special and general relativity corrections
- Relativity enhances gravity!



Solving the equations



Initial conditions

$$r = 0 \rightarrow m(0) = 0$$

Final conditions

$$\mathrm{r} = R \rightarrow P(R) = 0$$
$$\rightarrow M = m(R)$$

...What about ρ?

Before that, white dwarfs

- Low- or medium-mass star at the end of its lifetime, $R \approx 10^4 \mathbf{km}$
- Not enough thermal pressure → electrons pushed closer together
- Electrons in the lowest energy levels
- Pauli exclusion principle → star is stabilised against gravity

Fermi gas model for electrons

Number of states dn available at momentum k per unit volume

$$dn = \frac{d^3k}{(2\pi\hbar)^3} = \frac{4\pi k^2 dk}{(2\pi\hbar)^3}$$

Integrating

$$n = \frac{8\pi}{(2\pi\hbar)^3}\int_0^{k_F} k^2 dk = \frac{k_F^3}{3\pi^2\hbar^3}$$

Where $k_F c$ is the Fermi energy level, i.e., maximum electron energy

Fermi gas model for electrons

The electron energy density ϵ is found similarly

$$\epsilon_{el}(k_F) = \frac{8\pi}{(2\pi\hbar)^3} \int_0^{k_F} \sqrt{k^2c^2 + m_e^2c^4}\, k^2 dk$$

The total energy density includes the rest mass density (primarily given by nucleons)

$$\epsilon = n m_N \frac{A}{Z} + \epsilon_{el}(k_F)$$

Fermi gas model for electrons

From thermodynamics

$$P = -\left[\frac{\partial U}{\partial V}\right]_{T=0} = n^2 \frac{d\left(\frac{\epsilon}{n}\right)}{dn} = n\frac{d\epsilon}{dn} - \epsilon$$

From the calculations we get:

Relativistic case ($k_F \gg m_e$):

$$P(k_F) \approx K_{rel}\epsilon^{\frac{4}{3}}$$

Non-relativistic case ($k_F \ll m_e$):

$$P(k_F) \approx K_{nonrel}\epsilon^{\frac{5}{3}}$$

Pure neutron star, Fermi gas EoS

Same as for white dwarfs, but with m_n instead of m_e

Relativistic case ($k_F \gg m_n$):

$$P(k_F) \approx K_{rel}\epsilon$$

Non-relativistic case ($k_F \ll m_n$):

$$P(k_F) \approx K_{nonrel}\epsilon^{\frac{5}{3}}$$

For arbitrary relativity, we can fit the energy density with

$$\epsilon(P) = A_{NR}P^{\frac{3}{5}} + A_R P$$

Numerical solutions

Variables rescaling

Introducing:

- $M_s = 1.98892 \times 10^{30}$ kg solar mass
- $R_s = \frac{2GM_s}{c^2}$ sun Schwarzschild radius
- $\gamma = \frac{M_s c^2}{R_s^3}$

The quantities present in the equations can be rescaled as:

$$M = \bar{M} M_s, \qquad P = \bar{P}\,\gamma, \qquad r = \bar{r} R_s, \qquad \rho = \frac{\bar{\rho}\gamma}{c^2}$$

where $\bar{M}, \bar{P}, \bar{r}$ and $\bar{\rho}$ are dimensionless variables

Numerical solutions

Equations to be solved

Non relativistic equations

$$\frac{d\bar{m}}{d\bar{r}} = 4\pi\bar{r}^2\bar{\rho}(\bar{r})$$

$$\frac{d\bar{P}}{d\bar{r}} = -\frac{\bar{m}(\bar{r})}{2\bar{r}^2}\bar{\rho}(\bar{r})$$

Relativistic equations

$$\frac{d\bar{m}}{d\bar{r}} = 4\pi\bar{r}^2\bar{\rho}(\bar{r})$$

$$\frac{d\bar{P}}{d\bar{r}} = -\frac{\bar{m}(\bar{r})}{2}\bar{\rho}(\bar{r})\left[1+\frac{\bar{P}(\bar{r})}{\bar{\rho}(\bar{r})}\right]\left[1+\frac{4\pi\bar{r}^3\bar{P}(\bar{r})}{\bar{m}(\bar{r})}\right]\left[\frac{1}{\bar{r}^2-\bar{m}(\bar{r})\bar{r}}\right]$$

Pure neutron star, Fermi gas EoS

Using the phenomenological EoS

$$\bar{\rho}(\bar{P}) = C_{NR}\bar{P}^{\frac{3}{5}} + C_R\bar{P}$$

The values of C_{NR} and C_R which have been used are the ones suggested in [1]:

$$\bar{\rho}(\bar{P}) = 0.871\bar{P}^{\frac{3}{5}} + 2.867\bar{P}$$

[1] Alex Gezerlis. Numerical Methods in Physics with Python. Cambridge University Press, 2 edition.

Fourth order Runge-Kutta method

- Given an ordinary differential equation with initial conditions

$$\frac{dy(t)}{dt} = f(t, y), \qquad y(t_0) = y_0$$

- General predictor-corrector method:
  - Compute slope at $t_i \rightarrow f(t_i, y_i)$
  - Prediction $y_{i+1} \approx y(t_i) + hf(t_i, y_i)$
  - Compute slope at $t_{i+1} \rightarrow f(t_{i+1}, y_{i+1})$
  - Corrected prediction $y_{i+1} \approx y(t_i) + h/2(f(t_i, y_i) + f(t_{i+1}, y_{i+1}))$
  - …

Fourth order Runge-Kutta method

- Given an ordinary differential equation with initial conditions

$$\frac{dy(t)}{dt} = f(t, y), \qquad y(t_0) = y_0$$

- $y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$, where

  - $k_1 = f(y_i, t_i)$
  - $k_2 = f\left(y_i + \frac{k_1}{2}, t_i + \frac{h}{2}\right)$
  - $k_3 = f\left(y_i + \frac{k_2}{2}, t_i + \frac{h}{2}\right)$
  - $k_4 = f(y_i + k_3, t_i + h)$

- Error of the order $O(h^4)$

RK4 in TOV equations

```
def rk4_gen(f, r0, y0, step, max_step):
    xs = r0 + np.arange(max_step) * step
    ys = []

    yvals = y0.copy()
    for x in xs:
        if yvals[0] < 0:  # Stop when pressure crosses 0
            break
        ys.append(yvals.copy())
        k0 = step * f(x, yvals)
        k1 = step * f(x + step / 2, yvals + k0 / 2)
        k2 = step * f(x + step / 2, yvals + k1 / 2)
        k3 = step * f(x + step, yvals + k2)
        yvals += (k0 + 2 * k1 + 2 * k2 + k3) / 6

    return np.array(xs[:len(ys)]), np.array(ys)
```

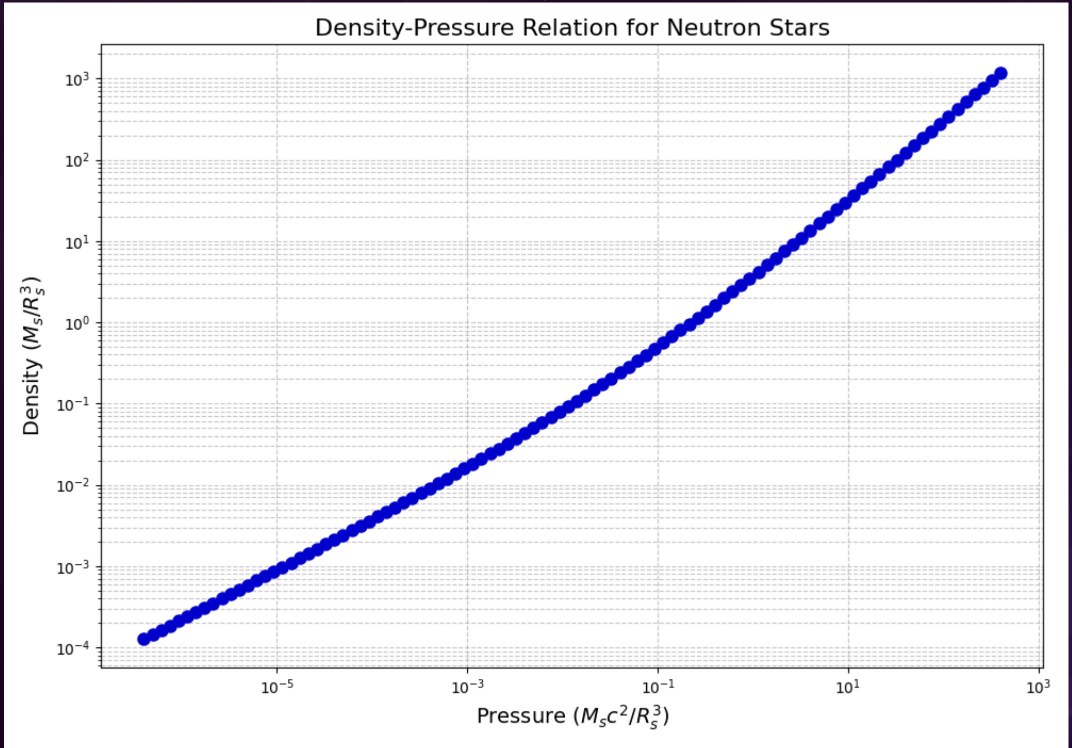
$$ys = [P, M]$$

$$xs = r, \qquad \mathrm{dr} = 5 \times 10^{-3}$$

$$f = \left[\frac{dP}{dr}, \frac{dM}{dr}\right]$$

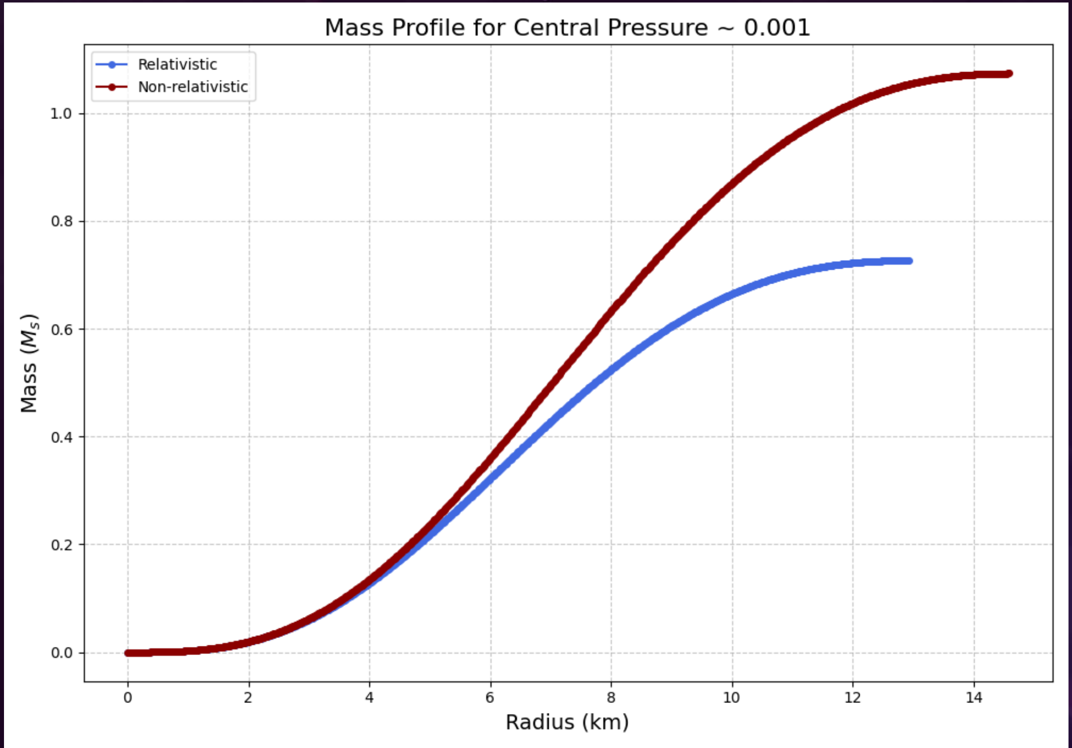
Different values of $\bar{P}_C$ over the range $10^{-7} - 10^{2}$

Results: EOS

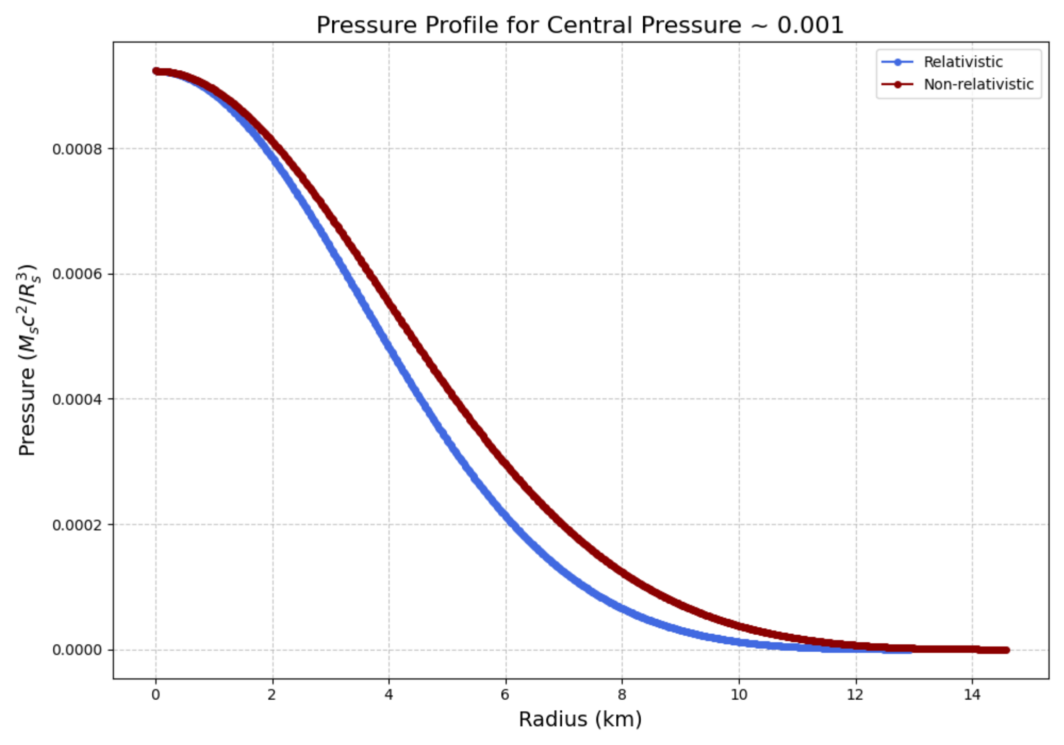


Density-Pressure Relation for Neutron Stars
Density (M_s/R_s^3)
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-3}
10^{-1}
10^1
10^3
Pressure ($M_s c^2/R_s^3$)

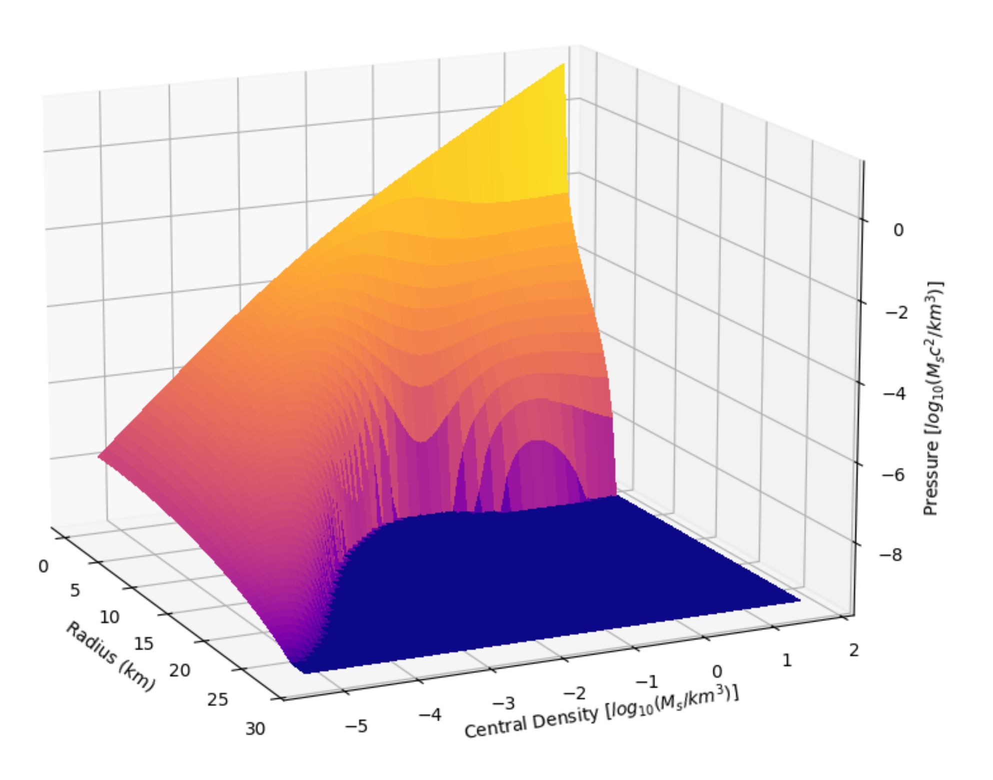
Results: Mass profile



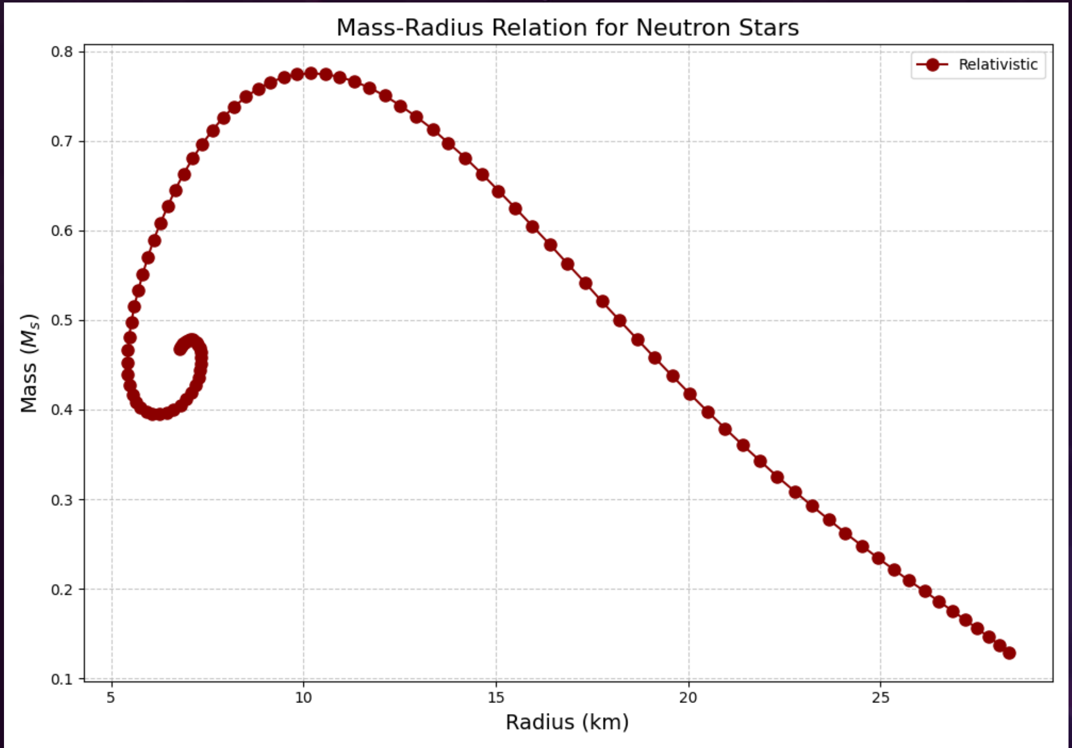
Results: Pressure profile



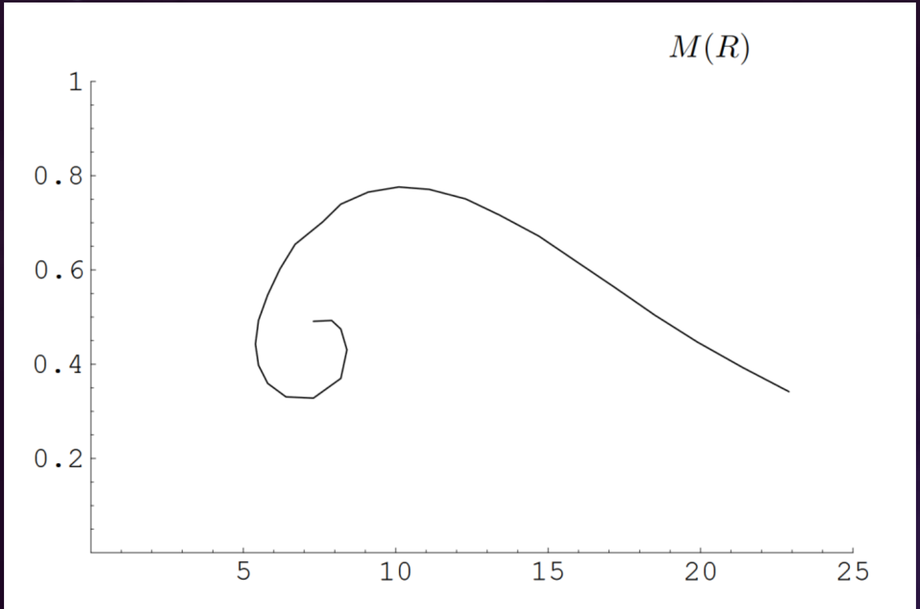
Results: Pressure 3D profile



Results: Mass-Radius relation

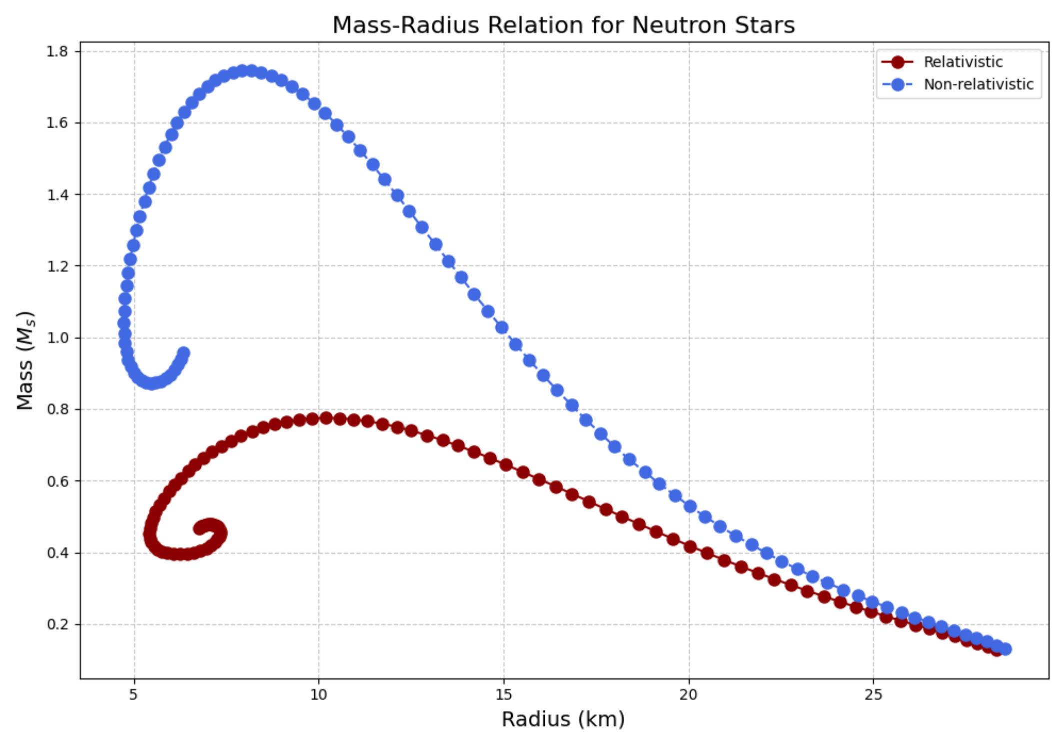


Results: Mass-Radius relation



Richard R. Silbar and Sanjay Reddy. Neutron stars for undergraduates. American Journal of Physics, 72(7):892-905, 2004. doi: 10.1119/1.1703544. URL https://arxiv.org/abs/nucl-th/0309041

Results: Mass-Radius relation, comparison



Execution time test

Time perfromances evaluated for both RK4 method and solve_ivp Python function

For $\mathrm{dr} = 5 \times 10^{-3}$ average time:

- Custom RK4: $170\ \mathrm{ms}$
- solve_ivp: $138\ \mathrm{ms}$, but worse results

For $\mathrm{dr} = 1 \times 10^{-2}$ average time:

- Custom RK4: $138\ \mathrm{ms}$
- solve_ivp: $150\ \mathrm{ms}$

Code structure

- constants.py: Contains all the physical constants and all the parameters of the simulation.
- eos.py: Contains the equation of state employed in the project.
- tov_solver.py: Contains the equations to be solved and the fourth-order Runge-Kutta method.
- tov_calculations.py: Contains the function that manages the computation of the solutions.
- utils.py: Contains utility functions for operations across the project.
- time_performance.py: Contains a function to estimate execution time of the RK4 method
- outputs/: Directory where result plots are saved.

Thank you!

Nicholas Pieretti

Theoretical and Numerical aspects of Nuclear Physics