Neutron Star Modeling

Michael Andaloro[1]

[1] Mathematics Department, University of Dallas, Irving, United States of America

E-mail: mandaloro@udallas.edu

Abstract

As a star enters into supernova, the star will explode with the force of a star that is at least 8 solar masses. In order for a star to become a neutron star, it must follow specific mass and density guidelines or else the supernova will become a quark and eventually collapse into a black hole. When a neutron star is formed, it will be composed of neutrons which came from protons and electrons being pressed so tightly together that they fused into a neutron. As the star lives its' life, the neutrons are burned and radiated off the star. By using a fourth order Runge-Kutta algorithm, it will be possible to numerically solve the coupled differential equations that form the equation of state and structural model of a neutron star. These equations specifically resolve the Bethe & Johnson equations of state relating to the Tolman – Oppenheimer – Volkoff (TOV) equations considering a rotating perturbation.

Keywords: RK4, TOV, Rotating Perturbation

1. Introduction

A star at the mid-point of its lifetime will consist primarily of ionized gasses in hydrostatic equilibrium between its' own gravitational attraction and the pressure of the thermonuclear radiation. A newborn star is composed almost entirely of hydrogen and helium, and the rate of fusion between the two atoms follows a proton-proton chain that forms the helium in an exothermic reaction. An exothermic reaction is a reaction that releases energy through light or heat. As the star continues to react, the hydrogen will consistently be used up which will result in a drop in the pressure of the radiation. This drop causes the star to contract. When the star contracts, the increase of pressure allows for fusion between heavier atoms deeper inside the star and this helps keep the star stable in its exothermic reaction.

As the star continues its' lifespan, the process of burning the hydrogen will continue until the star can no longer sustain the rate of fusion between the atoms. When this lack of fusion happens, the thermonuclear pressure that holds the star up will fall. When the pressure falls, the gravitational pressure becomes greater than the pressure from the reaction and the star will collapse. When this collapse happens, the outer layers of the star will push inwards. This push by the outer layer immediately produces an immense pressure spike at the core. The instant intense pressures overcomes the Pauli electron exclusion energies. With this loss of exclusion, the protons and electrons in the core are permitted to decay into neutrons through electron capture. This decay creates a Bose-Fermi gas. This gas is temperature independent.

This immense pressure coupled with the energy from the decaying electrons causes a spike in energy. This energy rushes to the outer layers bringing along with it atoms and other matter. This rush blasts much of the star's matter out into space and creates the phenomena known as a supernova.

When the supernova ends, the star has become one of two possibilities: a neutron star or a black hole. A black hole occurs when the original star's core is massive enough.

Assuming, a black hole has not formed, the core of the star will be the only remaining matter of the star that was not blown into space. A neutron star is formed entirely of neutrons. These neutrons are held in equilibrium between fizzing into nothingness and further collapsing into a black hole. After all, neutron stars are usually less than 15 miles long, yet they can have a mass greater than that of our own Sun. This is because a neutron star is composed of just neutrons. Almost everything in this universe has electrons which are very light (compared to a neutron) and take up space for their orbits. A neutron star has little electrons and as such, can be much more densely packed with just neutrons.

Neutron stars were first explored by Oppenheimer (co-founder of the TOV function). His work suggested that a neutron star could support itself as long as the total mass of the star is less than 2*M. His work as well as the work of other scientists has concluded that a neutron star can only exist within a boundary of density. The limits are defined as being less than the Schwarzschild Limit and greater than the Chandrasekhar Limit. The Schwarzchild Limit is the lowest a neutron star can be before collapsing into a black hole. A Chandrasekhar Limit is the highest density a neutron star can exist at before it fizzes into nothingness.

The density of the star clearly plays a role in the density of the formation of a star. This project aims to explore in depth the creation of a neutron star using both classical and relativistic models. Although classical models were used, because of the insane gravity these stars exhibit, a relativistic model was necessary to yield effective results. The final goal of this project is to visually display the range of values upon which a star can be formed. In other words, we should be able to see a line that will show the limits to what can form a neutron star. Anything outside of this range will not become a star.

2. Procedures

This project was conducted in two main parts. The first part was a classical (Newtonian) model of the star's mass as well as a relativistic model of the star's mass. Both parts of this project share many of the same parameters regarding the star's basic information as well as the algorithms were shared between the two. One common aspect shared is the assumption that the neutron star is in hydrostatic equilibrium. This means that the gravitational force is directly counter balanced by the pressures within the core of the star. The equilibrium results in a stable structure, thus allowing the star to exist in the first place. To compute this equilibrium, we can use the equation:

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

This equation is Newton's Law of Gravitation. This law states that every atom is attracted to every atom and the strength varies based on size and distance. $\rho(r)$ is the density at each distance r and $m(r)$ is the mass of the sphere. G is the gravitational constant and r is the radius of the star

After establishing the law of gravitation, I then considered the neutron star equations of state. I decided to use an equation of state called the Bethe – Johnson equation of state. The Bethe - Johnson equation was simpler than the other equations of state and yielded sufficient results for my work.

The reason I use the Bethe – Johnson equation is because Newton's equation describes the gravitational attraction of the star, but it does not consider the force that keeps that star from collapsing: the energy from the decaying neutrons. This force is considered in another equation given by Bethe and Johnson.

In order to use these equations successfully, it became necessary to use the Fourth Order Runge – Kutta Algorithm. This algorithm consists of putting numbers through four separate calculations to achieve a numerical solution to differential equations. The four equations are:

$$k_1 = f(t_n, y_n)$$

$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$$

$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$$

$$k_4 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_3)$$

Each of the equations can be combined at the end to form the solution:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

The beauty of the RK4 algorithm is that it can become extremely accurate. The accuracy of the RK4 method increases when smaller step sizes are used. This method, however, becomes ineffective when the parameters become absurd. One instance of this was when the pressure within the core became 0 once the r value became greater than the radius of the star.

After detailing the classical model of the star, I moved on to the Tolman – Oppenheimer – Volkoff equation. This equation models the star as it moves through its' life. The TOV equation is given as:

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

This equation can also yield larger results for big masses:

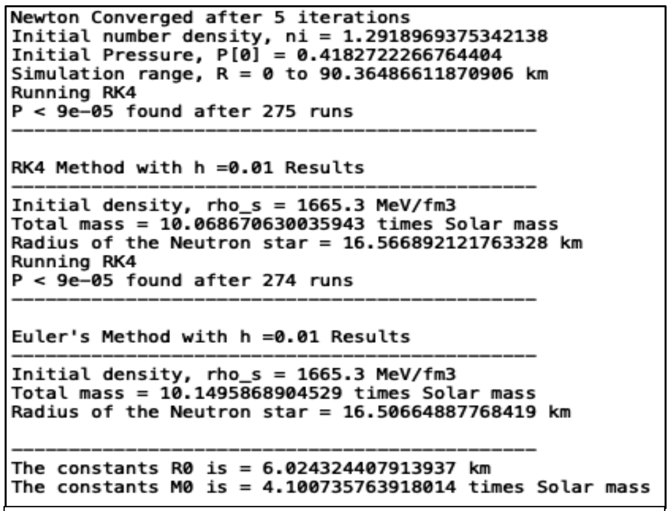
$$M_1 = \int_0^R \frac{4\,\pi\,r^2\,\rho}{\sqrt{1-\frac{2Gm}{rc^2}}}dr$$

Using these two models to demonstrate the star's motion, I was able to come up with several different results for the mass of the stars as the star's size changes.

The equations were all coded using Spyder accessed through an Anaconda Distribution. The language used is Python 3.0.

3. Results

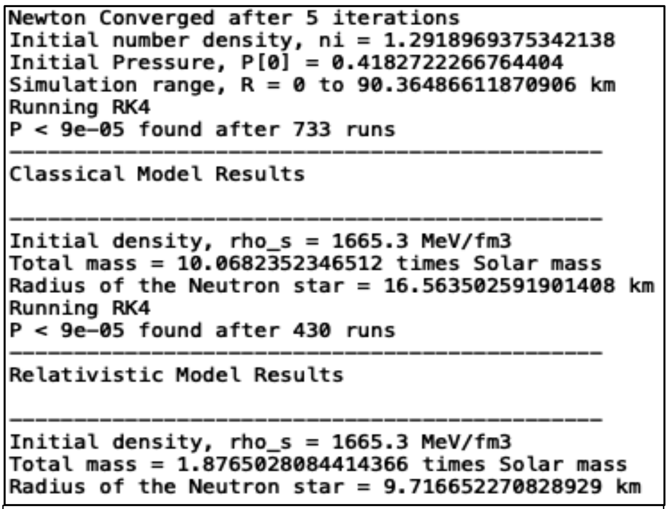
After creating the code using the functions and fine-tuning it to output the correct graphs and improving runtime, I began to output the various results from both the classical and relativistic models. To preserve readability, I have two separate python files, one for the classical model and the other for the relativistic model. Both files print the same result types, but with different numbers. The following screenshot of the Spyder console shows the values of the functions when calculated using the classical values (Newtonian).



```
Newton Converged after 5 iterations
Initial number density, ni = 1.2918969375342138
Initial Pressure, P[0] = 0.4182722266764404
Simulation range, R = 0 to 90.36486611870906 km
Running RK4
P < 9e-05 found after 275 runs
--------------------------------------------

RK4 Method with h =0.01 Results
--------------------------------------------
Initial density, rho_s = 1665.3 MeV/fm3
Total mass = 10.068670630035943 times Solar mass
Radius of the Neutron star = 16.566892121763328 km
Running RK4
P < 9e-05 found after 274 runs
--------------------------------------------

Euler's Method with h =0.01 Results
--------------------------------------------
Initial density, rho_s = 1665.3 MeV/fm3
Total mass = 10.1495868904529 times Solar mass
Radius of the Neutron star = 16.50664887768419 km

--------------------------------------------
The constants R0 is = 6.024324407913937 km
The constants M0 is = 4.100735763918014 times Solar mass
```

Figure 1: Output from classical method

Figure one shows the numerical outputs of the function when used with the classical method. These same inputs were used on the relativistic model but the outputs for the relativistic model were much more different:



```
Newton Converged after 5 iterations
Initial number density, ni = 1.2918969375342138
Initial Pressure, P[0] = 0.4182722266764404
Simulation range, R = 0 to 90.36486611870906 km
Running RK4
P < 9e-05 found after 733 runs
--------------------------------------------
Classical Model Results

--------------------------------------------
Initial density, rho_s = 1665.3 MeV/fm3
Total mass = 10.0682352346512 times Solar mass
Radius of the Neutron star = 16.563502591901408 km
Running RK4
P < 9e-05 found after 430 runs
--------------------------------------------
Relativistic Model Results

--------------------------------------------
Initial density, rho_s = 1665.3 MeV/fm3
Total mass = 1.8765028084414366 times Solar mass
Radius of the Neutron star = 9.716652270828929 km
```

Figure 2: Output from the relativistic method

As figure two shows, the relativistic simulation is able to find the outer edge much quicker than the classical model. The outer edge for both models is found when the pressure becomes less than 0.00009. The reason I did not go to perfectly 0 was to improve runtime. With step sizes of 0.01, the pressure results would sometimes go right into the negatives which would break the entire program. A smaller step size would not finish computing due to the large number of calculations required. After lots of tweaking, I found that I was able to go to 0.00009 and still maintain a level of accuracy that was acceptable to me. These functions also output the numerical results into graphs. The first two graphs are composed using the classical (Newtonian) model.

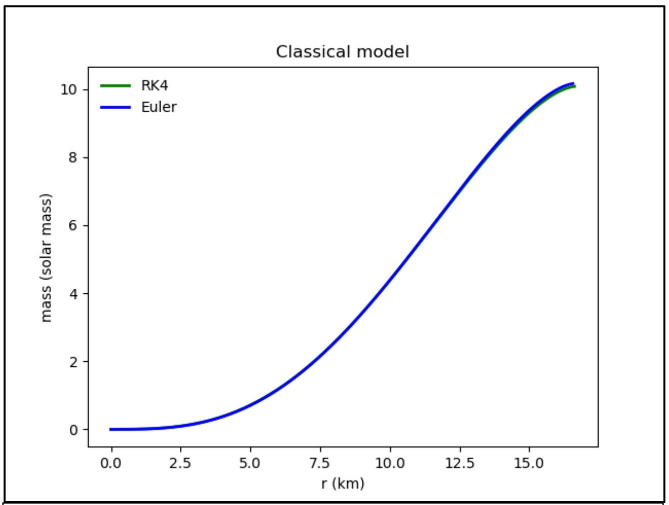


Figure 3: Graph of mass over radius.

This graph contains both RK4 and Euler approximations for the classical model. Both approximations were created using a step size of 0.01. The graph shows that both Euler and RK4 approximations produced nearly identical results. This graph shows that as the distance (radius) increases, so does the mass of the star. This increase inversely proportional for the pressure:

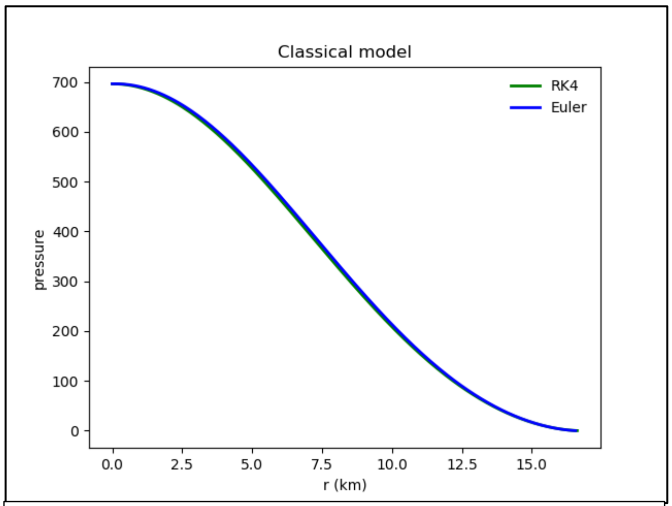


Figure 4: Graph of Pressure over radius.

Here, as the distance (radius) increases, the pressure decreases. This means that the further one got from the core of the star, the less pressure one felt. Just like the previous graph, the RK4 and Euler approximations are both very similar, and they both had a step size of 0.01. Figure three and figure four are inversely proportional along the x axis to one another.

These same graphs were then produced using a relativistic model of the star and the results were far different:

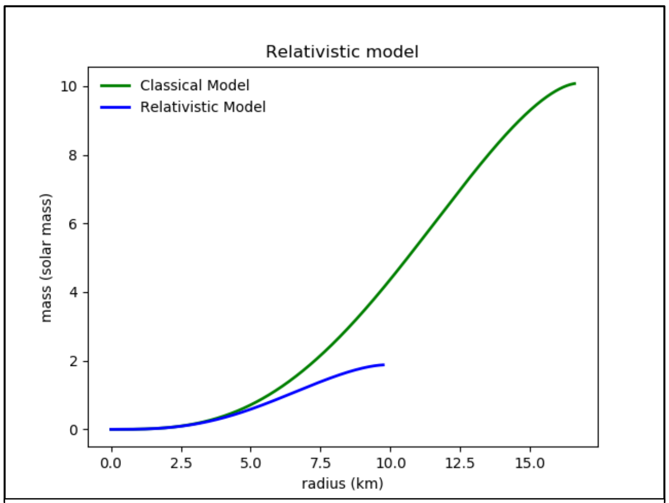


Figure 5: Graph of mass over radius.

This figure shows the classical versus relativistic models for a mass versus radius comparison. Here, you can see that the red relativistic line ends at nearly 10 km. This is because if the star were to get any larger, the star would collapse into a black hole. This was not accounted for by the classical model which is why the star is able to expand so much more. This same effect is also seen in the relativistic model of the pressure versus distance graph:

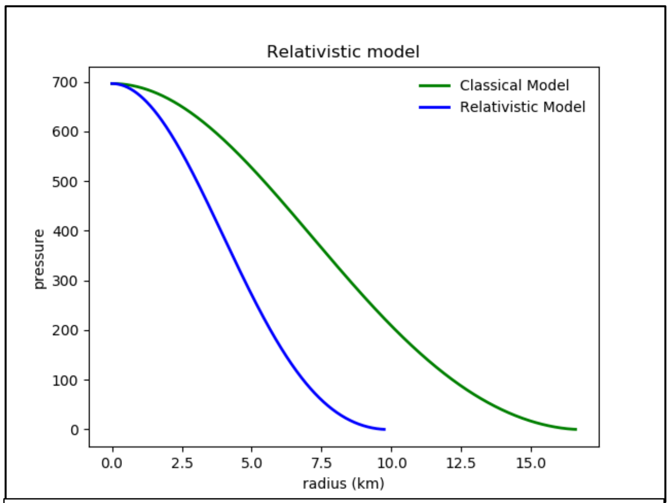


Figure 6: Graph of Pressure over radius.

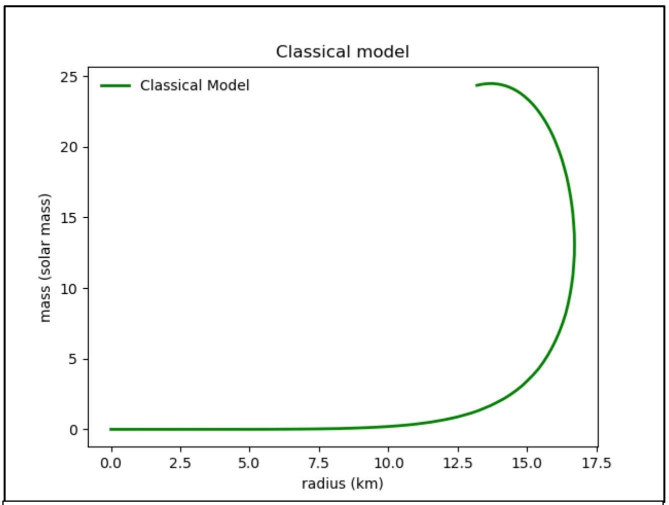


Figure 7: Visual of the runtime for classical model

Again, the red relativistic line falls below the classical line. This suggests that the classical method was not entirely accurate. The pressure has a much sharper fall off in the relativistic calculations and as such, the pressure becomes zero at just 10 km from the core, rather than the ~20 km for the classical model.

4. Conclusion

This project showed the range of values that a neutron star can be created within. Anything outside of this range will result in either the creation of a black hole. We also saw how the classical (Newtonian) model was no an effective solution for measuring these parameters for creating the star. This seems to be a recurring theme whenever large amounts of gravity are present the gravity begins to tamper with the classical model of looking at the universe. The relativistic model was able to compress on a solution much more quickly than the classical model was able to:

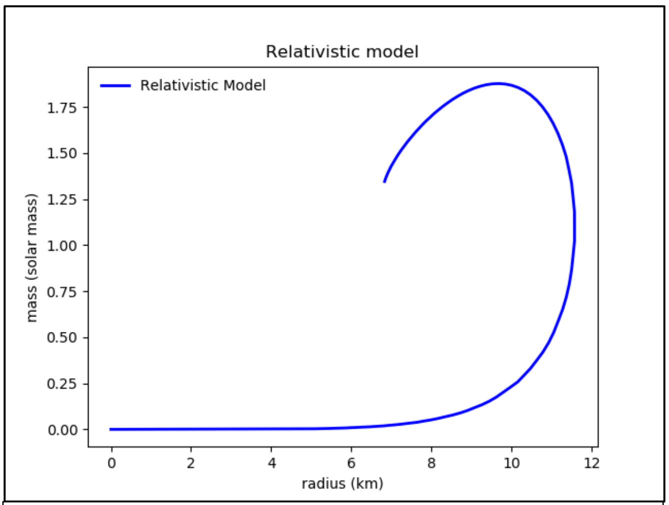


Figure 8: Visual of the runtime for relativistic model

As the figures seven and eight show, the runtime for the relativistic model was able to find a solution in about half the time it took the classical model. The solution is found when the the result parameters become absurd (e.g. the pressure becomes less than zero which is impossible). This was also true for the runtime that I personally experienced as my laptop was able to display a graph for the relativistic model almost a full second before the classical model was displayed. Both functions are exponential; each curve starts off gradually increasing in size before rapidly increasing and eventually turning back on itself.

References

[1] Faizi, Sarah, and Craig Fischer. “Pauli Exclusion Principle.” *Libretexts.org*, 5 June 2019, https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Electronic_Structure_of_Atoms_and_Molecules/Electronic_Configurations/Pauli_Exclusion_Principle.
[2] Myers, Steven T. “Lecture 18 - White Dwarfs & Neutron Stars (3/23/99).” *Nrao.edu*, 23 Mar. 1999, http://www.aoc.nrao.edu/~smyers/courses/astro12/L18.html.
[3] Nozawa, T, et al. “Construction of Highly Accurate Models of Rotating Neutron Stars - Comparison of Three Different Numerical Schemes.” *Aanda.org*, 6 May 1998, https://aas.aanda.org/articles/aas/full/1998/18/ds7680/node4.html.
[4] Burns, D, and R Dawson. “MODELLING OF NEUTRON STARS.” *University of Manchester School of Physics and Astronomy*, Apr. 2010, pp. 1–10., http://www.hep.man.ac.uk/u/russell/Neutron_Star_Report.pdf.

5. Appendix

The code for the project is listed in the following pages below:

```python
import numpy as np
import pylab as plt

hc = 197.327 # Conversion factor in MeV fm (hut * c)
G = hc * 6.67259e-45 # Gravitational constant
Ms = 1.1157467e60
density = 1665.3 # Central density (density at r = 0)
M0 = (4*3.14159265*(G**3)*density)**(-0.5)
R0 = G*M0
mn = 938.926 # Mass of neutron in MeV c^-2

def neutron():
    n = 1
    err = 1
    tol = 1e-15
    count = 0
    # Newton's method
    while err > tol :
        count += 1
        fn = n*mn + 236*n**(2.54) - density
        dfn = mn + 236*2.54*n**(1.54)
        temp = n - fn/dfn
        err = np.abs(n-temp)
        n = temp
    print ("Newton Converged after ", count, "iterations")
    return n

def density_function(p):
    n = (p*density/363.44)**(1./2.54)
    return (236. * n**2.54 + n *mn)/density

# Pressure vs radius graph, should decrease as r increases
def pressure_radius(r,m,p,side):
    if side == 0: # Classical model
        y = -m*density_function(p)/(r**2 + 1e-20)
    else: # Relativistic model
        rh = density_function(p)
        y = -(p+rh)*(p*r**3 + m)/(r**2 - 2*m*r + 1e-20)
    return y

# Mass vs radius graph, should increase as r increases
def mass_radius(r,m,p):
    return density_function(p)*r**2

def Euler(r,m,p,h,side):
    y = np.zeros(2)
    y[0] = m + mass_radius(r,m,p)*h
    y[1] = p + pressure_radius(r,m,p,side)*h
    return y

def RK4(r,m,p,h,side):
    y = np.zeros(2)
    a1 = mass_radius(r,m,p)
    b1 = pressure_radius(r,m,p,side)

    a2 = mass_radius(r+0.5*h,m+0.5*a1*h,p+0.5*b1*h)
    b2 = pressure_radius(r+0.5*h,m+0.5*a1*h,p+0.5*b1*h,side)

    a3 = mass_radius(r+0.5*h,m+0.5*a2*h,p+0.5*b2*h)
    b3 = pressure_radius(r+0.5*h,m+0.5*a2*h,p+0.5*b2*h,side)

    a4 = mass_radius(r+h,m+h*a3,p+h*b3)
    b4 = pressure_radius(r+h,m+h*a3,p+h*b3,side)

    y[0] = m + h*(a1 + 2.*a2 + 2.*a3 + a4)/6.
    y[1] = p + h*(b1 + 2.*b2 + 2.*b3 + b4)/6.
    return y

# This function plots the graph according to the inputs.
def mplot(fign,x,y,xl,yl,clr,lbl):
    plt.figure(fign)
    plt.xlabel(xl)
    plt.ylabel(yl)
    plt.title("Classical model")
    return plt.plot(x,y,clr, linewidth =2.0, label = lbl)

plt.figure(1)
plt.clf
plt.figure(2)
plt.clf

N = 1501
r = np.linspace(0,15,N)
h = r[1]-r[0]
m = np.zeros(N)
p = np.zeros(N)
rh = np.zeros(N)
ni = neutron()

# Initial values
r[0] = 0
m[0] = 0
```

```
p[0] = 363.44 * (ni**2.54)/density
rh[0] = 1
mf = 0
rf = 0

side_set = [0,1]
print ("Initial number density, ni =", ni)
print ("Initial Pressure, P[0] =", p[0])

print ("Simulation range, R = 0 to", r[-1]*R0*1e-18, "km")
tol = 9e-5

# Looping over 2 methods (k = 0 RK4 method, k = 1 Euler method)
# Only classical model is used in this case
for k in range(0,2):
    side = side_set[k]
    for i in range(0,N-1):
        if side == 0:
            [m[i+1], p[i+1]] = RK4(r[i],m[i],p[i],h,0)
        else:
            [m[i+1], p[i+1]] = Euler(r[i],m[i],p[i],h,0)

        rh[i+1] = density_function(r[i])
        if p[i+1] < tol:
            rf = r[i]
            mf = m[i]
            break

    print ("Running RK4")
    if i == N-2:
        print ("Program didn't converge to P = 0, extend the maximum value
    else:
        print ("P <", tol, "found after", i, "runs\n")

    m = m[0:i+2]
    p = p[0:i+2]
    rh = rh[0:i+2]
    r = r[0:i+2]

    if side == 0:
        lbl = "RK4"
        clr = "green"
    else:
        lbl = "Euler"
        clr = "blue"
    print ("----------------------------------------------")
    print (lbl, "Results")
    print ("----------------------------------------------")
    print ("Initial density, density =", density, "MeV/fm3")
    print ("Total mass = ", mf*M0/Ms, "times Solar mass")
    print ("Radius of the Neutron star =", rf*R0*1e-18, "km")

    mplot(1,r*R0*1e-18,p*density,'r (km)','pressure',clr,lbl)
    mplot(2,r*R0*1e-18,m*M0/Ms,'r (km)','mass (solar mass)',clr,lbl)

plt.figure(1)
q = plt.legend(loc = 0)
q.draw_frame(False)
plt.figure(2)
q=plt.legend(loc = 0)
q.draw_frame(False)

print ("\n----------------------------------------------")
print ("The constants R0 is =", R0*1e-18, "km")
print ("The constants M0 is =", M0/Ms, "times Solar mass")
```