Project One

Christopher D. Flowers
250897286

&

Philip Ceric
250866633

March 1, 2021

Dr. Basu
ASTRO 4101B
Stellar Astrophysics

All project code can be found at this clickable hyperlink on Github.

1 Introduction

The Lane-Emden equation is a dimensionless second-order ordinary differential equation that describes a self-gravitating, spherically symmetric, poly-tropic fluid, or more aptly, a star. It was first derived by the astrophysicists Jon Lane and Robert Emden, and is an important aspect of the modelling of stellar structures.

This project aims to solve this equation for different prescribed poly-tropic indexes and plot them. These plots will derive their data from both numerical solutions (using a 2nd order Runge-Kutta algorithm), as well as using analytical solutions (where possible). Further, the quantitative data that comes from these solutions will then be applied to find various physical parameters of a described star (central density, gravitational potential, etc).

2 Theory

2.1 The Lane-Emden Equation

As described previously, the Lane-Emden equation is an ODE that describes a self-gravitating, spherically symmetric, polytropic fluid. Thus, it is mostly concerned with modelling stellar structures in physical terms.

There are many ways to derive this equation, however it can most easily be done by starting with the concept of hydrostatic equilibrium and polytropic pressure.

$$\begin{aligned}
\frac{dP}{dr} &= \rho g \\
g &= -\frac{GM(r)}{r^2} \\
\frac{dM}{dr} &= 4\pi r^2 \rho(r) \\
P &= K\rho^{\gamma} = K\rho^{(n+1)/n}
\end{aligned}$$

Then, combining the first two, differentiating both sides, and then substituting the third and fourth equations, we obtain:

$$\begin{aligned}
\frac{r^2}{\rho}\frac{dP}{dr} &= -GM(r) \\
\frac{1}{r^2}\frac{d}{dr}\left[\frac{r^2}{\rho}\frac{dP}{dr}\right] &= -4\pi G\rho \\
\frac{n+1}{n}\frac{K}{r^2}\frac{d}{dr}\left[r^2\rho^{\frac{1-n}{n}}\frac{d\rho}{dr}\right] &= -4\pi G\rho
\end{aligned}$$

Then, transform both variables r and ρ into their respective dimensionless forms, θ^n & ξ:

$$\rho = \rho_c \theta^n \qquad \{\theta \mid 1 \geq \theta \geq 0\}$$

$$r = \alpha\xi \qquad \textbf{where} \;\; \alpha = \left[(n+1)\left(\frac{K\rho_c^{\frac{1-n}{n}}}{4\pi G}\right)\right]^{1/2}$$

From this, finally obtaining, the Lane-Emden equation:

$$\frac{1}{\xi^2}\frac{d}{d\xi}\left[\xi^2\frac{d\theta}{d\xi}\right] = -\theta^n$$

With the following boundary conditions:

$$\theta(\xi = 0) = 1, \quad \frac{d\theta}{d\xi}(\xi = 0) = 0$$
$$\textbf{from} \quad \rho(r = 0) = \rho_c, \quad \frac{d\rho}{dr}(r = 0) = 0$$

2.2 The 2nd Order Runge-Kutta Method

For a particular polytropic index value, there may or may not be an analytical solution (such as n = 0, 1, or 5. However, for most others, the only way to solve this equation is by way of numerical methods. This project utilizes a second-order Runge-Kutta method, which is more widely known as the midpoint method.

This iterative method can be used to find discrete solutions of ordinary differential equations (as well as systems, where x is treated as a vector). For an initial value problem:

$$\frac{dx}{dt} = f(t, x)$$
$$x(t_0) = x_0$$

The 2^{nd} order Runge-Kutta method that will solve this is as follows:

$$\mathbf{x}(t+\tau) = \mathbf{x}(t) + \tau\mathbf{f}\left(\mathbf{x}^*\left(\boldsymbol{t} + \frac{1}{2}\tau\right), t + \frac{1}{2}\tau\right)$$
$$\mathbf{x}^*\left(t + \frac{1}{2}\tau\right) \equiv \mathbf{x}(t) + \frac{1}{2}\tau\mathbf{f}(\mathbf{x}(t), t)$$

The global error of this method is $\mathcal{O}(h^2)$. This means that this method is an order of magnitude more accurate than Euler's method (also of the Runge-Kutta family), which has a global error of $\mathcal{O}(h)$.

3 Analysis

3.1 Part 1

The numerical solution of the Lane-Emden equation can be found by first reducing the 2nd order differential equation to a system of 1st order equations. These can be done by simply assigning a variable, like so:

$$\begin{aligned}
\frac{1}{\xi^2}\frac{d}{d\xi}\left[\xi^2\frac{d\theta}{d\xi}\right] &= -\theta^n \\
\frac{1}{\xi^2}\left[2\xi\frac{d\theta}{d\xi}+\xi^2\frac{d^2\theta}{d\xi^2}\right]+\theta^n &= 0 \\
\frac{d^2\theta}{d\xi^2}+\frac{2}{\xi}\frac{d\theta}{d\xi}+\theta^n &= 0 \\
v &= \frac{d\theta}{d\xi} \\
\frac{dv}{d\xi}+\frac{2}{\xi}v+\theta^n &= 0
\end{aligned}$$

This then leads us to the system of equations required to solve this second-order differential equation numerically:

$$\begin{bmatrix} v \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \frac{d\theta}{d\xi} \\ -\frac{2}{\xi}v-\theta^n \end{bmatrix}$$

It is, however, important to note that there is a divergent term in the second equation of the system. This issue is dealt with simply by utilizing a power series solution for $0 \leq \xi < 0.01$ (one solution step), and taking the endpoints of this solution (and its derivative) as initial conditions to the numerical solution. The power series solution (and its derivative) that were used are as follows:

$$\begin{aligned}
\theta &= 1-\frac{1}{6}\xi^2+\frac{n}{120}\xi^4 \\
\frac{d\theta}{d\xi} &= -\frac{1}{3}\xi+\frac{1}{30}\xi^3
\end{aligned}$$

Then, the prescribed Runge-Kutta numerical method was applied to the above system with a step size of $\tau = h = 0.01$. This step size was chosen to ensure that a smooth enough solution was obtained for the polytropic indexes prescribed. The choice to make this step size any smaller was in order to ensure a balance between the accuracy of the solution and run-time. The chosen h was sufficient enough where more accuracy was not desirable given this tradeoff.

The Runge-Kutta method was implemented by using a function of the state vector of the system, a function for a step of the RK2 method, and finally a set of nested for-loops to complete the solution for each prescribed polytropic index. The solution terminated once the x-intercept was reached or when the solution was no longer numeric (i.e. nan).

The following analytic solutions were also utilized in the plot:

$$\theta_0 = 1 - \frac{\xi^2}{6}$$
$$\theta_1 = \frac{\sin \xi}{\xi}$$
$$\theta_5 = \left(1 + \frac{\xi^2}{3}\right)^{-1/2}$$

The following graph was created with both the analytic and numerical solutions plotted together:

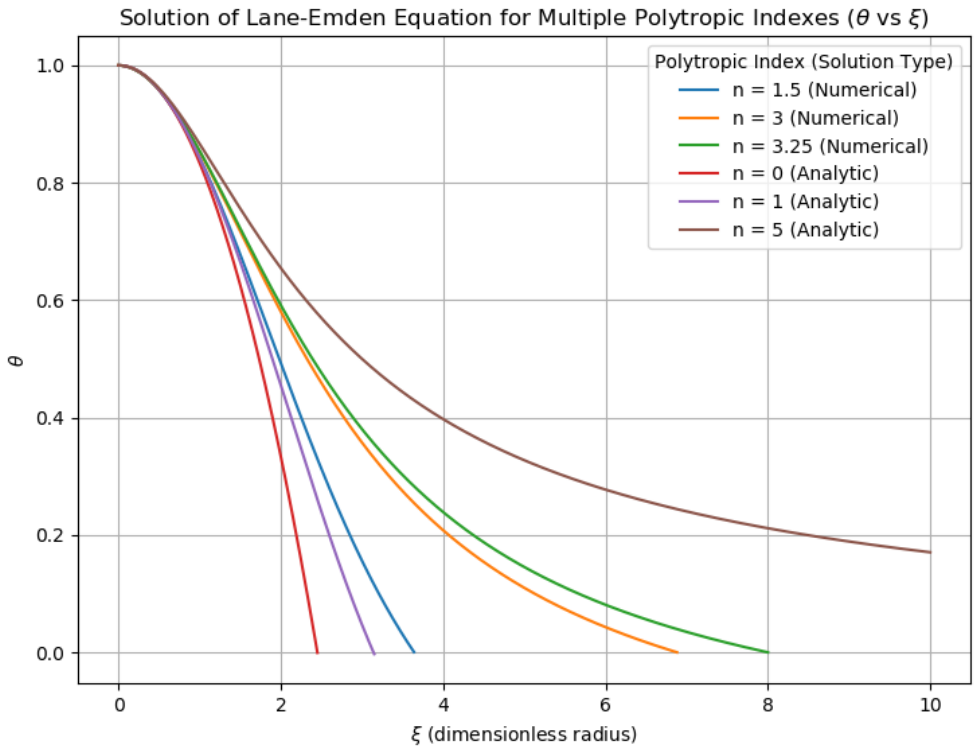

Figure 1: The plot of both analytic and numerical solutions for the specified polytropic indexes

3.2 Part 2

3.2.1 The Central Density, ρ_c

$$\begin{aligned}
M &= \int_0^R \rho(r) 4\pi r^2 dr \\
&= -4\pi\alpha^3 \rho_c \xi_1^2 \left.\frac{d\theta}{d\xi}\right|_{\xi=\xi_1} \\
\rho_c &= -\frac{M}{4\pi\alpha^3 \xi_1^2 \left.\frac{d\theta}{d\xi}\right|_{\xi=\xi_1}} \\
&= -\frac{M}{4\pi(\frac{R}{\xi_1})^3 \xi_1^2 \left.\frac{d\theta}{d\xi}\right|_{\xi=\xi_1}} \\
\rho_c &= -\frac{M\xi_1}{4\pi R^3 \left.\frac{d\theta}{d\xi}\right|_{\xi=\xi_1}} \\
\rho_c &= -\frac{(1.99\times10^{33}\text{ g})(7.99)}{4\pi(6.96\times10^{10}\text{ cm})^3(-0.03038)}
\end{aligned}$$

$$\boxed{\rho_c = 123.53\ \ \text{g cm}^{-3}}$$

3.2.2 The Length Scale, α

$$R = \alpha\xi_1$$

$$\begin{aligned}
\alpha &= \frac{R_\odot}{\xi_1} \\
&= \frac{6.96\times10^{10}\text{ cm}}{7.99}
\end{aligned}$$

$$\boxed{\alpha = 8.71\times10^{9}\ \ \text{cm}}$$

3.2.3 The Polytropic Constant, K

$$R = \left[\frac{(n+1)K}{4\pi G}\right]^{1/2} \rho_c^{\frac{1-n}{2n}} \xi_1$$

$$\begin{aligned} K &= \frac{4\pi G R^2}{\rho_c^{\frac{1-n}{n}} \xi_1^2 (n+1)} \\ &= \frac{4\pi(6.67259 \times 10^{-8}\ \mathrm{cm^3\ g^{-1}\ s^{-2}})(6.96 \times 10^{10}\ \mathrm{cm})^2}{(123.53\ \mathrm{g\ cm^{-3}})^{\frac{1-3.25}{3.25}}(7.99)^2(3.25+1)} \end{aligned}$$

$$\boxed{K = 4.2 \times 10^{14}\ \mathrm{cm^8\ g^{-2}\ s^{-2}}}$$

3.2.4 The Central Pressure, P_c

$$\begin{aligned} P_c &= \frac{GM^2}{R^4}\left[4\pi(n+1)\left(\frac{d\theta}{d\xi}\right)^2_{\xi=\xi_1}\right]^{-1} \\ &= \frac{(6.67259 \times 10^{-8}\ \mathrm{cm^3\ g^{-1}\ s^{-2}})(1.99 \times 10^{33}\ \mathrm{g})^2}{(6.96 \times 10^{10}\ \mathrm{cm})^4[4\pi(3.25+1)(-0.03038)^2]} \\ &= 2.28 \times 10^{17}\ \mathrm{g\ cm^{-1}\ s^{-2}} \end{aligned}$$

$$\boxed{P_c = 2.28 \times 10^{17}\ \mathrm{Ba}}$$

3.2.5 An Estimate of the Central Temperature of the Sun, $T_{\odot,c}$

$$
\begin{aligned}
PV &= NkT \\
PV &= (\frac{m}{\mu m_u})kT \\
P &= (\frac{m}{V})(\frac{1}{\mu m_u})kT \\
P &= \frac{\rho KT}{\mu m_u}
\end{aligned}
$$

$$
\begin{aligned}
T_{\odot,c} &= \frac{\mu m_u P_c}{\rho_c k} \\
&= \frac{mP_c}{\rho_c k} \\
&= \frac{0.6(1.6733 \times 10^{-24}\ \text{g})(2.28 \times 10^{17}\ \text{Ba})}{(123.53\ \ \text{g cm}^{-3})(1.3807 \times 10^{-16}\ \text{cm}^2\ \text{g s}^{-2}\ \text{K}^{-1})}
\end{aligned}
$$

$$\boxed{T_{\odot,c} = 13.4 \times 10^6\ \text{K}}$$

3.3 Part 3

The numerical data from the Runge-Kutta solution was transformed using the equations prescribed in section 2.1 above. It was then plotted along with the standard solar model prescribed by the project. The graph is as follows:

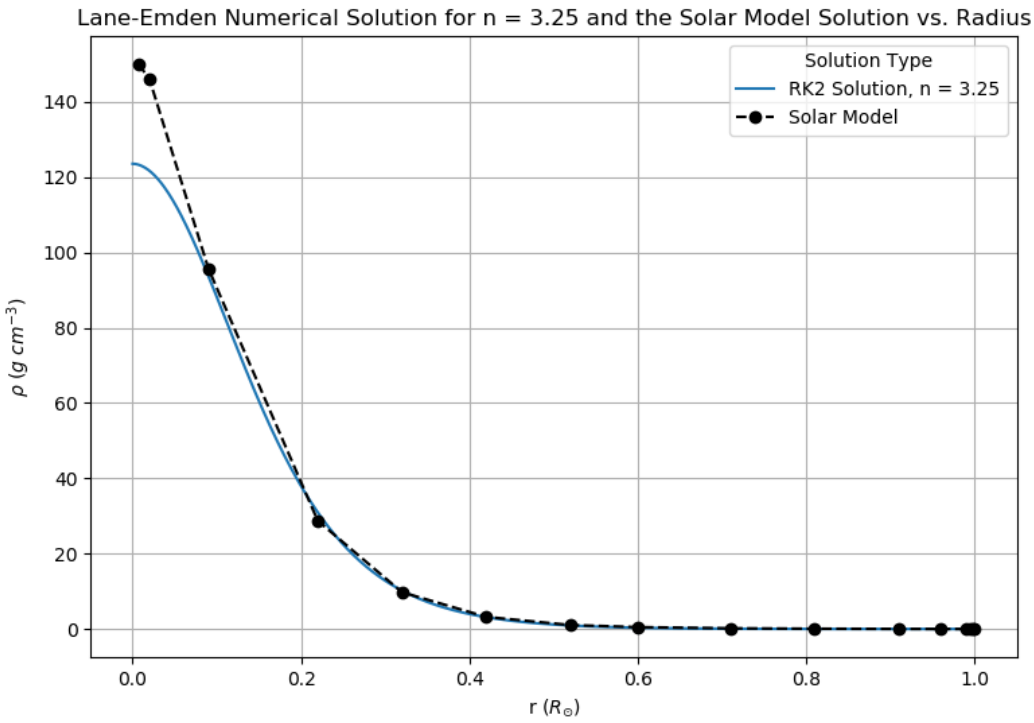

Figure 2: The plot comparing the standard solar model and our numerical model

There are few obvious differences present between the numerical model found and the standard solar model given for the project. Firstly, and most obviously, the central density differs by around 25 g cm^{-3} or by a percent difference of 19.3 %. This can most likely be attributed to the nature of the numerical solution and the limitations of the Lane-Emden model itself. The Lane-Emden equations appear to be formed from differential equations derived classically, however at such high temperatures and pressures, these models tend to breakdown in favour of quantum models. Numerically speaking, a 2nd order Runge-Kutta method is not the best in terms of error, there are stronger and less erroneous (e.g. RK4) methods that the data provided may have been sourced from. Finally, the solutions found will then be compared to and plotted with the standard solar model, in terms of density. Overall, when the solutions are compared, they are remarkably similar with the curves being essentially identical except as one approaches the star's centre.

4 Conclusion

This project was focused on solving the Lane-Emden equation using both a power series solution and a numerical (RK2) method. Firstly, the numerical solutions of the various indexes were plotted with some analytic solutions. It is clear that both types of solutions agreed (that is they were not widely different in their general shape) and that our numerical method allowed for an approximate solution to be found. Secondly, the numerical solution for $n = 3.25$ was used to find various physical parameters of a polytrope (i.e. our Sun). These parameters align somewhat accurately with accepted values in the field, most likely by virtue of our solution. Lastly, our $n = 3.25$ solution was plotted along with the standard model and was shown to agree closely, only differing near the centre of the star, where classic models presumably fall apart. Overall, the RK2 method for solving the Lane-Emden equation allows for a relatively close model of our Sun to be presented.

5 Code

```python
# ------------------------------------------------------------------------ #
#  File: project1.py                                                       #
#  Author: Christopher Flowers                                             #
#  Date: 2021-02-19                                                        #
#  Purpose: To solve the Lane-Emden equation numerically (with some        #
#           analytical solutions for comparison) via RK2 and plot.         #
#                                                                          #
# ------------------------------------------------------------------------ #

# ------------------- #
#    Dependencies     #
# ------------------- #

import numpy as np
import matplotlib.pyplot as plt
import math
import csv

# ------------------- #
# Solution Parameters  #
# ------------------- #

h = 0.01                # Step size
indexes = [1.5,3,3.25]  # Polytropic indexes to solve numerically
ps_end = 0.01           # Where to end power series, it is set to 1 since that is where the term stops going to infinity

ps = lambda xi,n : 1 - (1/6)*xi**2 + (n/120)*xi**4     # Lambda function for theta power series
dps = lambda xi,n : -(1/3)*xi + (n/30)*xi**3           # Lambda function for derivative of power series (to obtain dθ/dt for initial value of numerical solution)

# ---------------------------------------- #
# Initial Conditions and Solution Ranges  #
# ---------------------------------------- #

xi = np.arange(ps_end+h,10+h,h)  # Numerical portion of xi
theta = [[]]*3                   # Multi-dimensional array to store each solution of theta.
v = [[]]*3                       # Multi-dimensional array to store each solution of v.

# --------- #
# Functions #
# --------- #

def state(xi,x,n):       # State function
    theta,v = [x[0],x[1]]  # Unpack state variables
    return (np.array([v,-(2/xi)*v - theta**n]))  # Return derivatives

def rk2(f,x,t,n,tau):                  # 2nd order Runge-Kutta method, or Midpoint method
        x_star = x + tau*f(t,x,n)/2    # Midpoint Calculation
        x = x + tau*f(t+(tau)/2,x_star,n) # Full Calculation
        return x

# ------------ #
# Solution Loop #
# ------------ #

for j in range(len(indexes)):                                   # Loop through polytropic indexes
    theta[j] = [ps(z,indexes[j]) for z in np.arange(0,ps_end+h,h)]  # Calculate the power series solution for one stpe for a given n-value
    x = np.array([theta[j][-1],dps(ps_end,indexes[j])])         # Numpy array of the initial conditions of theta and v, is required to be a NP array for vector purposes
    for i in range(len(xi)):                                    # Loop to numerically solve for a given polytropic index
        prev_x = x                                              # Save last solution for use in the next iteration
        x = rk2(state,prev_x,xi[i],indexes[j],h)                # Solve and append
        if x[0] < 0 :                                           # Check if solution has passed through x-axis.
            break
        elif math.isnan(x[0]) == True:                          # If infinite value or non-numerical value is returned due to errors etc, remove and end solution.
            break
        else:
            theta[j].append(x[0])                               # Otherwise append new theta to appropriate solution array.
            v[j].append(x[1])


# ------------ #
#   Plotting   #
# ------------ #

xi = np.append(np.arange(0,ps_end+h,h),xi)  # Append the powerseries range and the numerical range together for plotting
analytic = [lambda z: 1 - z**2/6, lambda z: np.sin(z)/z, lambda z: (1 + z**2/3)**(-0.5)]  # Lambda functions for analytic solutions: n = 0, 1, and 5.


# Plot the numerical solutions found above

plt.plot(xi[0:len(theta[0])],theta[0],label='n = 1.5 (Numerical)')
plt.plot(xi[0:len(theta[1])],theta[1],label='n = 3 (Numerical)')
plt.plot(xi[0:len(theta[2])],theta[2],label='n = 3.25 (Numerical)')


# Plot the analytic solutions using the above lambda functions and the appropriate ranges up to the x-intercepts.

plt.plot(np.arange(0,np.sqrt(6)+h,h),analytic[0](np.arange(0,np.sqrt(6)+h,h)),label='n = 0 (Analytic)')
plt.plot(np.arange(0,np.pi+h,h),analytic[1](np.arange(0,np.pi+h,h)),label='n = 1 (Analytic)')
plt.plot(np.arange(0,10+h,h),analytic[2](np.arange(0,10+h,h)),label='n = 5 (Analytic)')

# Other misc. plotting parameters

plt.legend(title = "Polytropic Index (Solution Type)")   # Add legend
plt.grid()  # Add grid lines.
plt.title(r'Solution of Lane-Emden Equation for Multiple Polytropic Indexes ($\theta$ vs $\xi$)') # Add title
plt.xlabel(r'$\xi$ (dimensionless radius)') # Add x-axis label
plt.ylabel(r'$\theta$')  # Add y-axis label
plt.show()   # Show the plot

print("The value of dθ/dξ at the x-intercept of the polytrope for n = 3.25 is ",v[-1][-1])    # This is present to obtain a numerical value for Question 2a.
print("The value of ξ at the x-intercept of the polytrope for n = 3.25 is ",xi[len(theta[-1])-1])    # This is present to obtain a numerical value for Question 2a.

# ---------------------- #
#   Question 3 Plotting  #
# ---------------------- #

# Transform xi solution for n = 3.25 to units of solar radii
alpha = 8.71E9
Rsol = 6.96E10
r = alpha*np.array(xi[0:len(theta[2])])/Rsol

# Transform theta solution for n = 3.25 to units of g/cm
rho_c = 123.53
rho = rho_c*np.array(theta[2])**3.25

# Import CSV file to obtain solar model values
with open("solartable.csv", 'r') as f:
    reader = csv.reader("solartable.csv")
    data = list(csv.reader(f, delimiter = ","))

# Reverse data, pop headers out, re-reverse data to normal
data.reverse()
for i in range(1): data.pop()
data.reverse()

# Unpack required columns to use for plotting
r_solmodel = [float(data[i][0]) for i in range(len(data))]
rho_solmodel = [float(data[i][4]) for i in range(len(data))]


plt.plot(r,rho,label='RK2 Solution, n = 3.25')  # Plot RK2 Solution
plt.plot(r_solmodel,rho_solmodel,'o--',color = 'black',label = 'Solar Model')  # Plot solar model solution
plt.legend(title = "Solution Type")   # Add legend
plt.grid()  # Add grid lines.
plt.title(r'Lane-Emden Numerical Solution for n = 3.25 and the Solar Model Solution vs. Radius') # Add title
plt.xlabel(r'r ($R_{\odot}$)') # Add x-axis label
plt.ylabel(r'$\rho$ ($g \ cm^{-3}$)')  # Add y-axis label
plt.show()   # Show the plot
```