Report

Ahmad Izzuddin

December 8, 2022

Chapter 1

Introduction

1.1 Background

While the sun is in our view for most of human history, no direct observations of the interior of the sun, or any other star for that matter, have been taken. The closest spacecraft has been to the interior of the sun is the Parker Solar probe NASA launched in 2018 (Whittlesey et al., 2020). This probe was closer to the sun than any other spacecraft humans have built. However, this is still far from being able to conduct observations of the sun's interior. This is why the use of modeling is the only way we can "see" inside. The interior of a star is described by PDEs which are coupled and nonlinear. These equations are discussed in 2.1. There is no analytical general solution to the equations. Because of this, the numerical method is needed for stellar modeling. This project aims to model zero-age main sequence stars that are in hydrostatic and thermal equilibria. The goal is to model the density, pressure, temperature, and mass of the star's interior depending on the radius from the center of the star.

Chapter 2

Method

2.1 Equations

The equations for stellar structure are as follows (Hansen et al., 2004):

$$\frac{\partial P}{\partial M_r} = -\frac{GM_r}{4\pi r^4} - \frac{1}{4\pi r^2}\frac{\partial^2 r}{\partial t^2} \tag{2.1}$$

$$\frac{\partial r}{\partial M_r} = \frac{1}{4\pi r^2 \rho} \tag{2.2}$$

$$\frac{\partial T}{\partial M_r} = -\frac{GM_r}{4\pi r^4}\frac{T}{P}\nabla \tag{2.3}$$

$$\frac{\partial L_r}{\partial M_r} = \epsilon - T\frac{\partial S}{\partial t} \tag{2.4}$$

When the model is simplified and hydrostatic and thermal equilibria are assumed, the equations become

$$\frac{\partial P}{\partial M_r} = -\frac{GM_r}{4\pi r^4} \tag{2.5}$$

$$\frac{\partial r}{\partial M_r} = \frac{1}{4\pi r^2 \rho} \tag{2.6}$$

$$\frac{\partial T}{\partial M_r} = -\frac{GM_r}{4\pi r^4}\frac{T}{P}\nabla \tag{2.7}$$

$$\frac{\partial L_r}{\partial M_r} = \epsilon \tag{2.8}$$

Where,

$$M_r = \int_0^r 4\pi r'^2 \rho \, \mathrm{d}r \tag{2.9}$$

which is the mass at radius r from the center of the star, and

$$L_r = 4\pi r^2 F \tag{2.10}$$

is the luminosity or total flux at radius r. Where ∇ is approximated with

$$\nabla = \begin{cases} \nabla_{rad} & \nabla_{rad} \leq \nabla_{ad} \\ \nabla_{ad} & \nabla_{rad} > \nabla_{ad} \end{cases} \tag{2.11}$$

These gradients are

$$\nabla_{rad} = \frac{\kappa L_r}{16\pi c G M_r} \frac{3P}{aT^4} \tag{2.12}$$

which is the radiative gradient, and

$$\nabla_{ad} = \frac{\partial \ln T}{\partial \ln P} = -\frac{4\pi r^4 P}{G M_r T} \frac{\partial T}{\partial M_r} \tag{2.13}$$

which is the adiabatic gradient. Also,

$$\kappa = \frac{4acT^3}{3\rho} \frac{1}{\lambda} \tag{2.14}$$

which is the coefficient of radiative opacity (per unit mass), c is the speed of light, and a is the radiation constant which is valid if the heat transport is due to radiation. However, these equations still depend on the mass at a certain radius which in turn is dependent on the density at the specified radius. This is solved by using polytropes which define density and in turn also pressure. The Lane-Emden equation which defines this is

$$\frac{1}{\xi^2} \frac{\partial}{\partial \xi} \left(\xi^2 \frac{\partial \theta}{\partial \xi} \right) = -\theta^n \tag{2.15}$$

where n is the polytropic index and θ_n is the "polytropic temperature". In this context, density, pressure, and radius are

$$\rho = \rho_c \theta_r^n \tag{2.16a}$$

$$P = P_c \theta_r^{n+1} \tag{2.16b}$$

$$r = r_n \xi \tag{2.16c}$$

where $\rho_c = \rho(r=0), P_c = P(r=0), r_n = \frac{(n+1)P_c}{4\pi G \rho_c^2}$. Pressure is also defined as

$$P = k\rho^{1+\frac{1}{n}} \tag{2.17}$$

Then, combining eqs. (2.16c) and (2.20d) at the total radius $r = R$

$$\frac{k}{GN_n} = R^{\frac{3-n}{n}} M^{\frac{n-1}{n}} \tag{2.18a}$$

$$N_n = \frac{(4\pi)^{\frac{1}{n}}}{n+1} \left(\left[-\xi^2 \frac{\mathrm{d}\theta}{\mathrm{d}\xi} \right]_{\xi=\xi_R} \right)^{\frac{1-n}{n}} \xi_R^{\frac{n-3}{n}} \tag{2.18b}$$

the polytropic constant k becomes

$$k = R^{\frac{3-n}{n}} M^{\frac{1-n}{n}} G N_n \tag{2.19}$$

which depends on mass M, total radius R, and the polytropic index n. Mass itself becomes Substituting with eqs. (2.16a) and (2.16c) the equation becomes

$$M_r = \int_0^r 4\pi r'^2 \rho \, \mathrm{d}r \tag{2.20a}$$

$$M_\xi = 4\pi r_n^3 \rho_c \int_0^\xi \xi^2 \theta^n \, \mathrm{d}\xi \tag{2.20b}$$

$$M_\xi = 4\pi r_n^3 \rho_c \int_0^\xi -\mathrm{d}\left(\xi^2 \frac{\partial \theta}{\partial \xi}\right) \tag{2.20c}$$

$$M_\xi = 4\pi r_n^3 \rho_c \left[-\xi^2 \frac{\partial \theta}{\partial \xi}\right]_\xi \tag{2.20d}$$

Another useful quantity is the average density which is defined as

$$\rho_{av} = \frac{3M}{4\pi R^3} \tag{2.21a}$$

$$= \frac{3}{\xi_R^3} \rho_c \left[-\xi^2 \frac{\partial \theta}{\partial \xi}\right]_{\xi_R} \tag{2.21b}$$

$$\frac{\rho_c}{\rho_{av}} = \frac{\xi_R}{3\left[-\frac{\partial \theta}{\partial \xi}\right]_{\xi_R}} \tag{2.21c}$$

For temperature T, we can define it using the ideal gas law and substituting with the polytropic equations for pressure (2.17)

$$P = \frac{\rho}{\mu m_p} k_B T \tag{2.22a}$$

$$k\rho^{1+\frac{1}{n}} = \frac{\rho}{\mu m_p} k_B T \tag{2.22b}$$

$$T = k\rho^{\frac{1}{n}} \frac{\mu m_p}{k_B} \tag{2.22c}$$

$$T_c = k\rho_c^{\frac{1}{n}} \frac{\mu m_p}{k_B} \tag{2.22d}$$

$$T = T_c \theta \tag{2.22e}$$

where m_p is the proton mass, k_B is the Boltzmann constant, and μ is the mean molecular weight with X is the mass fraction of hydrogen, Y is the mass fraction of helium, and Z is mass fraction of "metallic" elements

$$\mu^{-1} = 2X + \frac{3}{4}Y + \frac{1}{2}Z \tag{2.23}$$

2.2 Boundary Conditions

The boundary conditions for this system of equations are:
at the surface, where $M_r^* = M^*$:

$$\rho = 10^{-12}, \quad T = 2^{-\frac{1}{4}} T_{eff} \tag{2.24}$$

and, at the center where $M_r^* = 0$:

$$r^* = 0, \quad L_r^* = 0 \tag{2.25}$$

And the boundary conditions for the Lane-Emden equation are:
at the center, where $\xi = 0$:

$$\theta = 1, \quad \frac{\partial \theta}{\partial \xi} = 0 \tag{2.26}$$

and the surface of the star where θ is very small such that the density ρ is equal to the boundary condition for the surface density as defined in equation (2.24), ξ is the dimensionless radius of the star.

2.3 Numerical Solution

Here we use the Runga-Kutta 4^{th} order method on the Lane-Emden equation. Which is

$$k1 = y^{'}(x_i, y_i) \tag{2.27a}$$

$$k2 = y^{'}(x_i + \frac{h}{2}, y_i + \frac{h}{2}k1) \tag{2.27b}$$

$$k3 = y^{'}(x_i + \frac{h}{2}, y_i + \frac{h}{2}k2) \tag{2.27c}$$

$$k4 = y^{'}(x_i + h, y_i + h \times k3) \tag{2.27d}$$

$$y_{i+1} = y_i + (h/6)(k1 + 2 \times k2 + 2 \times k3 + k4) \tag{2.27e}$$

where h is the step size along x. y is the integrated value of $y^{'}$. Using this scheme, the mass, density, pressure, and temperature can be calculated. The order of calculations is:

1. Calculate constants and polytrope values

   (a) "Polytropic temperatures" θ for the corresponding values of the dimensionless radial coordinate ξ

   $$\frac{\partial}{\partial \xi}\left(\xi^2 \frac{\partial \theta}{\partial \xi}\right) = -\xi^2 \theta^n \tag{2.28a}$$

   $$2\frac{\partial \theta}{\partial \xi} + \xi \frac{\partial^2 \theta}{\partial \xi^2} = -\xi \theta^n \tag{2.28b}$$

   $$\frac{\partial^2 \theta}{\partial \xi^2} = -\theta^n - \frac{2}{\xi}\frac{\partial \theta}{\partial \xi} \tag{2.28c}$$

   to avoid division by 0 at $\xi = 0$, $\frac{\partial^2 \theta}{\partial \xi^2} = -\frac{1}{3}$ Hansen et al., 2004.

   (b) Dimensionless total radius ξ_R where θ_n <<<
   The total dimensionless radius ξ_R is when $\theta_{i+1} < 0 < \theta_i$.

   (c) Constant N_n (2.18b)

   (d) Polytropic constant k (2.19)

   (e) Radius constant r_n

   $$r_n = \frac{R^*}{\xi_R} \tag{2.29}$$

2. Calculate $M^*/\Delta M_r^*$ discrete points for

(a) Density ρ
Determine core center density ρ_c using eq. (2.21c)

(b) Mass M_r (2.20d)

(c) Pressure P (2.17)

(d) Temperature T (2.7)

3.

2.4 Program

The program will be shown below. The GitHub repository for the project containing the code is available here: https://github.com/nidduzzi/PHYS461-Final-Project. To run the program, follow the following instructions:

1. Download the whole repository to your local machine
2. Make sure you have at least python 3.8 installed
3. Then install the required packages (numpy scipy matplotlib seaborn pandas). Do this with your environment management. For example by executing the following commands (pipenv or conda respectively):

```
pip install numpy scipy matplotlib seaborn pandas
#or
conda install numpy scipy matplotlib seaborn pandas
```

4. Then run the program:

```
#linux
python starNum.py
#Windows
python .\starNum.py
```

To alter any parameters use the following flags:

```
python starNum.py [-h] [-M M] [-R R] [-X X] [-Y Y]
    ↪  [-Z Z] [-n N]
                  [-tol TOL] [-h0 H0] [-MAXITER
                      ↪ MAX_ITER] [-help]

Models a star with the Lane-Emden Equation. By
    ↪ default this uses the mass,
radius, and composition of the sun with a
    ↪ polytropic index of 3

optional arguments:
-h, --help show this help message and exit
-M M Total mass of the star in units of mass of
    ↪ the sun
                  (default 1.0 MSUN)
```

```
            -R R Total radius of the star in units of radius
                ↪ of the sun
                                (default 1.0 RSUN)
            -X X Hydrogen mass fraction of the star (default
                ↪ 0.73)
            -Y Y Helium mass fraction of the star (default
                ↪ 0.26)
            -Z Z Metalic mass fraction of the star (default
                ↪ 0.01)
            -n N Polytropic index (default 3.0)
            -tol TOL Tolerance between model radius
                ↪ approximation and actual
                                radius for a given polytropic
                                    ↪ index (default 1e-13)
            -h0 H0 Initial integration step size for RK4 (
                ↪ default 1e-05)
            -MAXITER MAX_ITER Maximum number of steps for RK4
                ↪ integration (default
                                10000000000)
            -help Shows the help

            by Ahmad Izzuddin
```

```
import numpy as np
from scipy.constants import G, Boltzmann, atomic_mass,
    ↪ proton_mass
from scipy.special import logsumexp
import matplotlib

matplotlib.use("TkAgg")
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import argparse
from rk4 import RK4


def ThetaXi(n=3.0, dxi=0.01, MAX_XI=6.9, MAX_ITER=10000):
    def h_func(x: np.float64, q: "np.ndarray[int,␣np.dtype[np.
        ↪ float64]]", h: np.float64):
        R_est = x - q[0] / q[1]
        if x + h > R_est:
            h = -q[0] / q[1]
        return h

    def rhs_func(x: np.float64, q: "np.ndarray[int,␣np.dtype[np.
        ↪ float64]]"):
        """the righthand side of the LE system, q' = f"""
        f = np.zeros_like(q)
```

```
        # y' = z
        f[0] = q[1]
        # for z', we need to use the expansion if we are at x = 0,
        # to avoid dividing by 0
        if x == 0.0:
            f[1] = (2.0 / 3.0) - q[0] ** n
        else:
            f[1] = -2.0 * q[1] / x - q[0] ** n

        return f

    solver = RK4(
        rhs_func,
        np.float64(1.0e-5),
        h_func,
        np.array([1.0, 0.0]),
        x0=np.float64(0.0),
        tol=np.float64(1.0e-12),
        MAX_ITER=MAX_ITER,
    )
    xi = np.array(solver.xi)
    theta, dthetadxi = np.array(solver.result)

    return xi, theta, dthetadxi


def Xi2dThetadXi(n, xi, theta, dxi=0.001):
    xi2dthdxi = [0.0, -(xi[1] ** 2 * theta[1] ** n)]
    for i in range(2, len(xi)):
        xi2dthdxi.append(-2.0 * dxi * xi[i] ** 2.0 * theta[i] ** n
            ↪  + xi2dthdxi[-2])
    return xi2dthdxi


def polytropicNn(n, xiR, xi2dthdxiR):
    return (
        (((4 * np.pi) ** (1 / n)) / (n + 1))
        * ((-xi2dthdxiR) ** ((1 - n) / n))
        * (xiR ** ((n - 3) / n))
    )


MSUN = 1.989 * 10.0**30.0 # Mass of the sun
RSUN = 696.342 * 10.0**6.0 # Radius of the sun


def run(
    PIN=3.0, # Polytropic index number
    M=1.0 * MSUN, # Mass of star
    R=1.0 * RSUN, # Radius of star
```

```
    X=0.73, # ratio of hydrogen
    Y=0.26, # ratio of helium
    Z=0.01, # ratio of "metalic" elements
    MAX_ITER = 10000000000
):
    MU = 1 / (2 * X + 0.75 * Y + 0.5 * Z) # Mean molecular wheight
    # Calculate Polytropic temperatures and dimensionless radius
        ↪ coordinate (Lane Emden)
    xi, theta, dthetadxi = ThetaXi(n=PIN, MAX_ITER=MAX_ITER)
    xi2dthdxi = xi**2 * dthetadxi
    # Calculate constant N_n
    Nn = polytropicNn(PIN, xi[-1], xi2dthdxi[-1])
    # Calculate Polytropic constant k
    k = R ** ((3 - PIN) / PIN) * M ** ((PIN - 1) / PIN) * G * Nn
    # Calculate radius constant r_n
    r_n = R / xi[-1]
    # Calculate core density and pressure
    rho_c = (3 * M) / (4 * np.pi * R**3) * (xi[-1]) / (3 * -
        ↪ dthetadxi[-1])
    # Calculate density
    rho_xi = rho_c * theta**PIN
    # Calculate Mass_r
    M_xi = 4 * np.pi * r_n**3 * rho_c * -xi2dthdxi
    # Calculate Pressure
    P_c = k * rho_c ** ((PIN + 1.0) / PIN)
    P_xi = k * np.sign(rho_xi) * np.abs(rho_xi) ** ((PIN + 1.0) /
        ↪ PIN)
    # Calculate Temperature
    T_c = k * rho_c ** (1 / PIN) * (MU * atomic_mass) / Boltzmann
    T_xi = T_c * theta
    # Calculate radius
    R_xi = r_n * xi

    # Plot
    df_scaled = pd.DataFrame(
        {
            "xi": xi / xi[-1],
            "dthetadxi": dthetadxi,
            "theta": theta,
            "rho": rho_xi / rho_c,
            "M": M_xi / M,
            "P": P_xi / P_c,
            "T": T_xi / T_c,
            "R": R_xi,
        }
    )
    df_scaled.set_index("R")
    sns.lineplot(pd.melt(df_scaled, ["R"]), x="R", y="value", hue=
        ↪ "variable")
    plt.savefig("out.png")
```

```
    plt.cla()
    df = pd.DataFrame(
        {
            "xi": xi,
            "dthetadxi": dthetadxi,
            "theta": theta,
            "rho": rho_xi,
            "M": M_xi,
            "P": P_xi,
            "T": T_xi,
            "R": R_xi,
        }
    )
    df.to_csv("./sim_LE.csv")
    df.set_index("R")
    sns.lineplot(df, x="xi", y="theta")
    plt.savefig("xitheta.png")
    plt.cla()
    sns.lineplot(df, x="R", y="rho")
    plt.savefig("RRho.png")
    plt.cla()
    sns.lineplot(df, x="R", y="M")
    plt.savefig("RM.png")
    plt.cla()
    sns.lineplot(df, x="R", y="P")
    plt.savefig("RP.png")
    plt.cla()
    sns.lineplot(df, x="R", y="T")
    plt.savefig("RT.png")
    plt.cla()


if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        prog="Star Polytrope",
        description="Models a star with the Lane-Emden Equation. 
            ↪ By default this uses the mass, radius, and 
            ↪ composition of the sun with a polytropic index of 3
            ↪ ",
        epilog="by Ahmad Izzuddin",
    )
    parser.add_argument(
        "-M",
        help="Total mass of the star in units of mass of the sun (
            ↪ default %(default)s MSUN)",
        dest="M",
        default=1.0,
        type=float,
    )
    parser.add_argument(
```

```
    "-R",
    help="Total␣radius␣of␣the␣star␣in␣units␣of␣radius␣of␣the␣
        ↪ sun␣(default␣%(default)s␣RSUN)",
    dest="R",
    default=1.0,
    type=float,
)
parser.add_argument(
    "-X",
    help="Hydrogen␣mass␣fraction␣of␣the␣star␣(default␣%(
        ↪ default)s)",
    dest="X",
    default=0.73,
    type=float,
)
parser.add_argument(
    "-Y",
    help="Helium␣mass␣fraction␣of␣the␣star␣(default␣%(default)
        ↪ s)",
    dest="Y",
    default=0.26,
    type=float,
)
parser.add_argument(
    "-Z",
    help="Metalic␣mass␣fraction␣of␣the␣star␣(default␣%(default
        ↪ )s)",
    dest="Z",
    default=0.01,
    type=float,
)
parser.add_argument(
    "-n", help="Polytropic␣index␣(default␣%(default)s)", dest=
        ↪ "n", default=3.0, type=float
)
parser.add_argument(
    "-tol",
    help="Tolerance␣between␣model␣radius␣approximation␣and␣
        ↪ actual␣radius␣for␣a␣given␣polytropic␣index␣(default
        ↪ ␣%(default)s)",
    dest="tol",
    default=1.0e-13,
    type=float,
)
parser.add_argument(
    "-h0",
    help="Initial␣integration␣step␣size␣for␣RK4␣(default␣%(
        ↪ default)s)",
    dest="h0",
    default=1.0e-5,
```

```
        type=float,
    )
    parser.add_argument(
        "-MAXITER",
        help="Maximum␣number␣of␣steps␣for␣RK4␣integration␣(default
            ↪ ␣%(default)s)",
        dest="MAX_ITER",
        default=10000000000,
        type=int,
    )
    args = parser.parse_args()
    run(args.n, args.M*MSUN, args.R*RSUN, args.X, args.Y, args.Z,
        ↪ args.MAX_ITER)
```

```
import numpy as np
from typing import Callable


class RK4:
    def __init__(
        self,
        rhs_func: Callable[
            [np.float64, 'np.ndarray[int,␣np.dtype[np.float64]]'],
            'np.ndarray[int,␣np.dtype[np.float64]]',
        ],
        h0: np.float64,
        h_func: Callable[
            [np.float64, 'np.ndarray[int,␣np.dtype[np.float64]]',
                ↪ np.float64], np.float64
        ],
        q0: 'np.ndarray[int,␣np.dtype[np.float64]]',
        x0=np.float64(0.0),
        tol=np.float64(1.0e-12),
        MAX_ITER=100000,
    ) -> None:
        self.xi: list[np.float64] = []
        self.result: list['np.ndarray[int,␣np.dtype[np.float64]]']
            ↪  = []
        self.q0 = q0
        self.x0 = x0
        self.h0 = h0
        self.rhs_func = rhs_func
        self.h_func = h_func
        self.tol = tol
        self.MAX_ITER = MAX_ITER
        self.integrate(h0, tol, q0, x0, MAX_ITER)

    def integrate(
        self,
```

```
    h0: np.float64,
    tol: np.float64,
    q0: 'np.ndarray[int, np.dtype[np.float64]]',
    x0: np.float64,
    MAX_ITER: int,
) -> None:
    # q = np.zeros_like(q0, dtype=np.float64)
    x = x0
    h = h0
    q = q0.copy()
    iter = 0
    while h > tol and iter < MAX_ITER:
        self.result.append(q.copy())
        self.xi.append(x)
        # fourth order runge-kutta
        k1 = self.rhs_func(x, q)
        k2 = self.rhs_func(x + 0.5 * h, q + 0.5 * h * k1)
        k3 = self.rhs_func(x + 0.5 * h, q + 0.5 * h * k2)
        k4 = self.rhs_func(x + h, q + h * k3)

        q += (h / 6.0) * (k1 + 2 * k2 + 2 * k3 + k4)
        x += h

        h = self.h_func(x, q, h)
        iter += 1
    self.result.append(q.copy())
    self.xi.append(x)
    self.result = list(np.array(self.result).transpose())
```

Chapter 3

Results

The overall results of the model generally follow what is expected in a star. As can be seen in figure 3, the pressure drops the fastest, then the density follows, and the temperature drops off the slowest. The mass accumulates slowly at first and picks up pace after 50 million meters. Since the temperature is just the polytropic "temperature" θ multiplied by the core temperature the curves are overlapping.

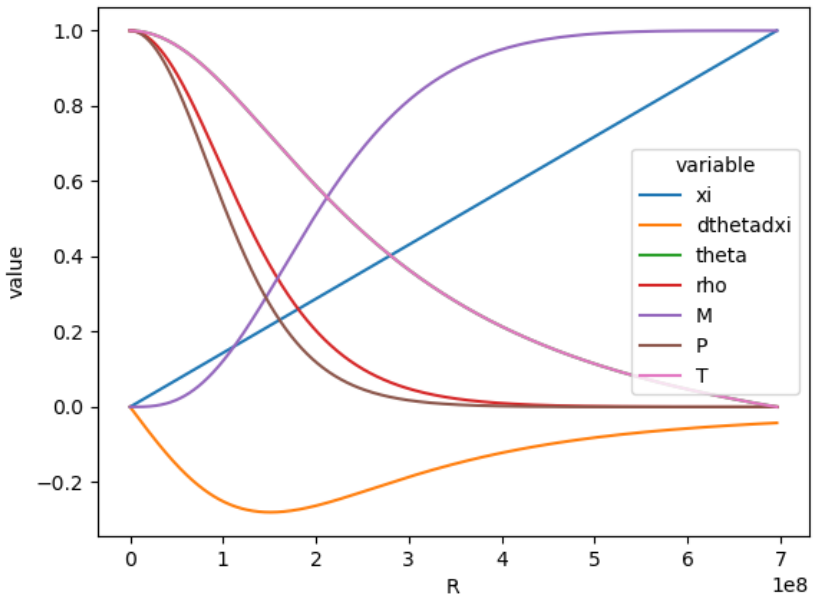

Figure 3.1: Overall scaled results of density, mass, pressure, temperature, polytropic "temperature", dimensionless radius

3.1 Lane-Emden

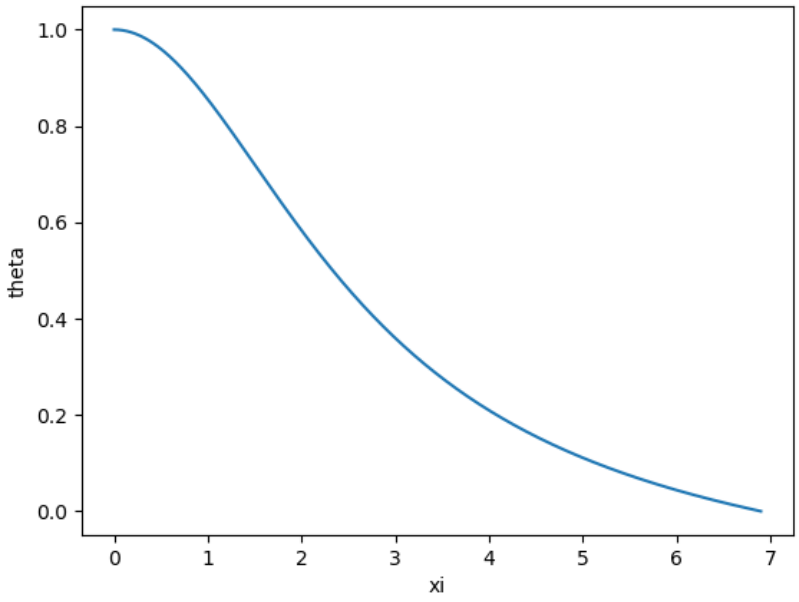

Figure 3.2: Polytropic "temperature" θ vs dimensionless radius ξ

3.2 Density

Density ρ drops off sharply at around 50 million meters and continues to do so until 300 million meters. This is shown in figure 3.2. One big problem is the ends of the curve are not accurate. This is because the surface of the sun doesn't have a density that drops to zero. And in comparison to a reference model which is more accurate, the Standard Solar Model (SSM) (Turck-Chièze, 2016), the central density in our model is half of that in the SSM, $73.196 \times 10^3 kgm^{-3}$ and $153 \times 10^3 kgm^{-}3$ respectively.

3.3 Mass

The mass itself does follow what is expected of a star. It starts with a value of 0 at $r = 0$ and ends at $r = 696.342 \times 10^6$ with a value of $1.989 \times 10^3 0$.

3.4 Pressure

Pressure is another one where it is problematic at both ends. For the center, the SSM has a value of $2.34 \times 10^1 6 Nm^{-2}$ and our model has a value of $1.24 \times$

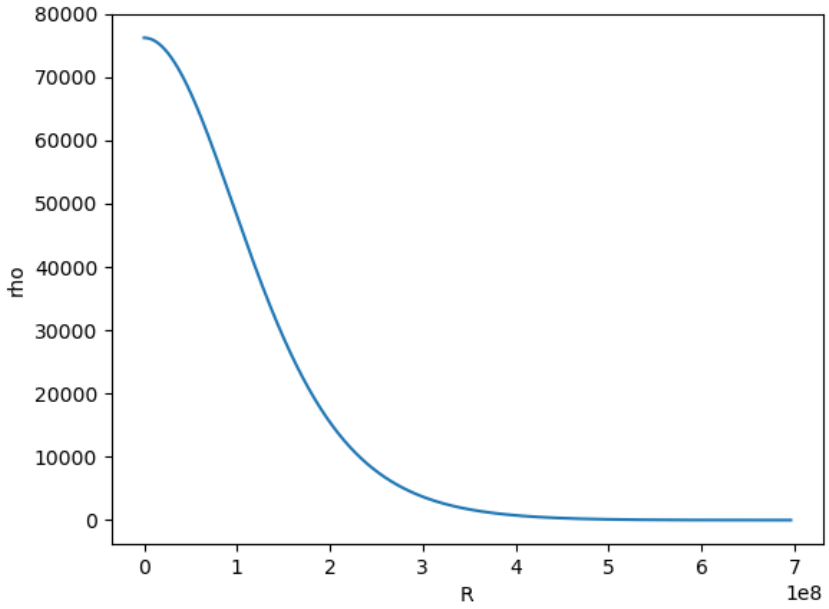

Figure 3.3: Density vs Radius

10^16Nm^{-2}. The discrepancy is once again about half the value from the SSM. At the surface, our model's pressure drops to zero. This is inaccurate because the sun has an atmosphere with a pressure that is not zero.

3.5 Temperature

The same with density and pressure, the result for temperature is not accurate for the ends of the curve. The SSM has a value of $15.74 \times 10^6 K$ at the center and our model has a value of $11.8 \times 10^6 K$. The discrepancy this time is not as severe and at most is about a third of the value from the SSM. The surface value is also inaccurate since it should be somewhere in the range of $5800K$.

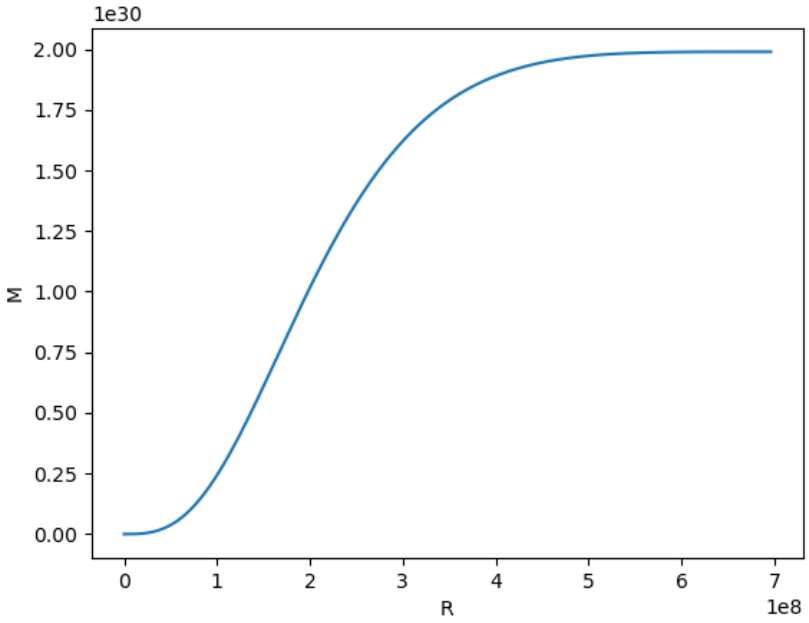

Figure 3.4: Mass vs Radius

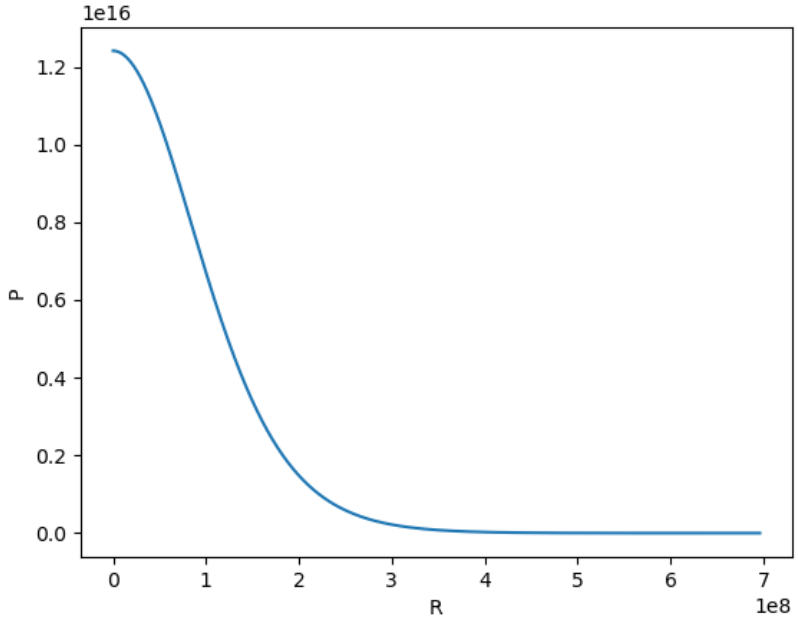

Figure 3.5: Pressure vs Radius

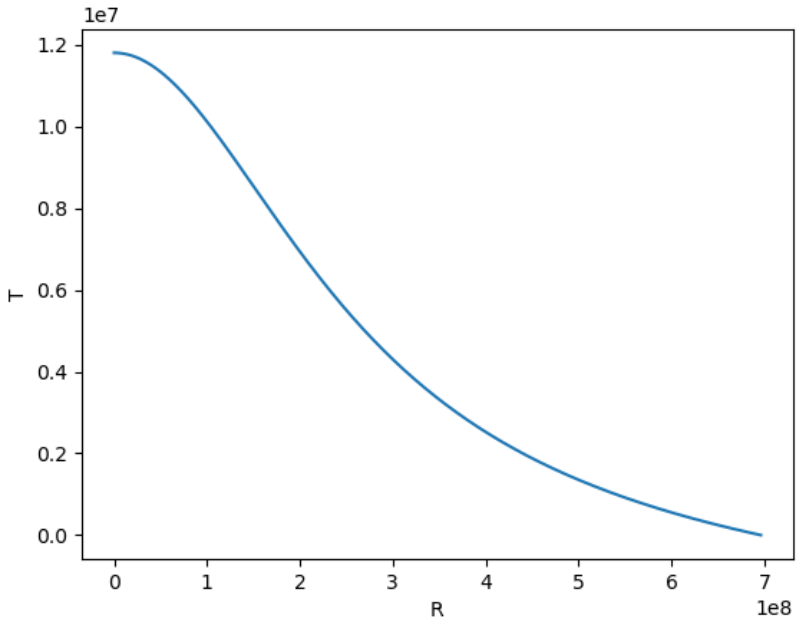

Figure 3.6: Temperature vs Radius

Bibliography

Hansen, C. J., Kawaler, S. D., & Trimble, V. (2004). *Stellar Interiors*. Springer New York. https://doi.org/10.1007/978-1-4419-9110-2

Turck-Chièze, S. (2016). The Standard Solar Model and beyond. *Journal of Physics: Conference Series*, *665*, 012078. https://doi.org/10.1088/1742-6596/665/1/012078

Whittlesey, P. L., Larson, D. E., Kasper, J. C., Halekas, J., Abatcha, M., Abiad, R., Berthomier, M., Case, A. W., Chen, J., Curtis, D. W., Dalton, G., Klein, K. G., Korreck, K. E., Livi, R., Ludlam, M., Marckwordt, M., Rahmati, A., Robinson, M., Slagle, A., ... Verniero, J. L. (2020). The Solar Probe ANalyzers—Electrons on the Parker Solar Probe. *The Astrophysical Journal Supplement Series*, *246*(2), 74. https://doi.org/10.3847/1538-4365/ab7370