Simulation of the aging of the Sun into a Red Giant

Steven Li
Phys 140, Fall 2017, San Jose State University

The sun and all stars undergoes an evolutionary track similar to the lifecycle of a human being. I have created a python program that attempts to stimulate the process that the sun undergoes into becoming a red giant through utilizing a polytrope to model the sun. Modeling the sun after a polytrope allows me to find a portion of density and pressure from a given n value which behaves like specific stars. From that I will able to find physical properties of the sun and utilizing a luminosity function over time, I can find physical properties of the sun throughout it's lifetime on the main-sequence before becoming a red giant.

INTRODUCTION

All stars including the sun undergo an evolutionary track called stellar evolution where stars undergo a sequence of dramatic changes that warp the nature of the star resulting in changes of size, temperature, luminosity and behaviour throughout it's lifetime ending with a bang or a quiet whimper. Stellar evolution is a process which we can't directly observe because the amount of time for a star to perform one of these phase shifts is way beyond our time on earth. So based on the observation of the types of stars in the universe, we have a general idea of how the evolution causes a star change over the course of it's lifetime. Most stars in the universe are main-sequence stars which are in a state of equilibrium and it will be in this state for the majority of it's life. Any star that diverge from the main-sequence will rapidly change signifying the beginning of the end of it's life.

THEORY

In order to model the sun, I will make the assumption the star is a non-rotating spherically symmetric, an ideal gas and is isolated from other stars that might affect it's evolutionary track.
The stellar structure of a star following these assumption is dictated by 5 stellar equations.[1]

Hydrostatic equilibrium:

$$\frac{dP}{dr} = -G\frac{M_r\rho}{r^2} \tag{1}$$

where this equation describes the pressure gradient on the star in a static distribution of pressure and density to keep the star in equilibrium. Otherwise, the gravity towards the star created by the star will be greater than the pressure exerted by the star, causing the star to collapse on itself and that would move it away from the main-sequence signifying the beginning of it's end.

Mass Conservation:

$$\frac{dM_r}{dr} = 4\pi r^2\rho \tag{2}$$

Energy Conservation:

$$\frac{dL_r}{dr} = 4\pi r^2\rho\epsilon \tag{3}$$

These two conservation stellar equations are important to keep the star isolated because the mass conservation relates the relationship between mass and density and how it has to be equivalent to each other. The same goes for energy conservation which describes the relationship between the energy generation of the star and the energy distributed in the form of light measured in luminosity.

Radiative Transport:

$$\frac{dT}{dr} = -\frac{3}{16\pi a_r c r^2}\frac{\kappa\rho}{T^3}L \tag{4}$$

Convective Transport.

$$\frac{dT}{dr} = \frac{\gamma-1}{\gamma}\frac{T}{P}\frac{dP}{dr} \tag{5}$$

The remaining two equations describes the transport of energy either in the form of radiative or convective.

To simply the simulation, I'm making the assumption that energy generation and transport will be ignored resulting in only a relationship between pressure and density to remain. This results in a polytrope with a polytropic index that describes the behavior of the star such that:

$$P(r) = K\rho^{1+1/n}(r) \tag{6}$$

METHOD

Solving the Lane-Emden 2nd Differential Equation required a while-loop because I didn't know the number of time steps required to reach the maximum radius of the star. The time-step varied from different values of n. The code ran until right before the dimensionless radius became zero signifying the radius of the star, giving the constant ξ_1 which is the dimensionless length right before the radius was reached and $\frac{d\theta}{d\xi}_{\xi_1}$ which is the rate of radius change at the maximum point.

```
for n in nArray:

    #Boundary Conditions/Initial Values
    theta = 1. #Dimensionless Scaling Length
    dthetadxi = 0.

    #Setup
    xi = 0.0 #Dimensionless Radius
    dxi = 0.000001 #Time Step,
    #lower the more accurate it is

    #Arrays
    thetaArray = []
    thetaArray.append(theta)
    dThetaArray = []
    dThetaArray.append(dthetadxi)
    xiArray = []
    xiArray.append(xi)

    #Loop until Scaling Length
    #goes below time-step.
    #(Roughly right before it
    # becomes negative)
    while theta > dxi:
        if(xi == 0):
            dthetadxi -= (theta**n * dxi)
        else:
            dthetadxi -= ( 2 * dthetadxi/xi
            + theta**n) * dxi
        theta += dthetadxi * dxi
        xi += dxi

        thetaArray.append(theta)
        dThetaArray.append(dthetadxi)
        xiArray.append(xi)


    #Results
    print('\n-------------------',\
    \nxi_1:',xi)
    print('dTheta/dXi_1:',dthetadxi)

    xi_1 = xi
    dthetadxi_1 = dthetadxi
```

The remaining physical properties can be achieved after achieving the constants for the polytrope above. The K will remain constant for the main-sequence star, so I can achieve density from the changing mass while keeping K the same value.

```
def kConstant(mass, density):
    k = (((-2)**(2/3) * np.pi**(1/3)
    * G * mass**(2/3)
    * density_c**(1/3)*(1-(3/n)))\
        /    ((-xi_1**2 * dthetadxi_1)**(2/3)
        * (n**3 + 3*n**2 + 3*n + 1)**(1/3)))
    return abs(k)

def densityFromK(mass):
    return (-((((dthetadxi_1) * K**(3/2)
    * xi_1**2
    * np.sqrt((n/G) + (1/G)))/(2 \
             * np.sqrt(np.pi) * G))
             +(((dthetadxi_1)
             * K**(3/2) * n * xi_1**2
             * np.sqrt((n/G) + (1/G)))/(2 \
             * np.sqrt(np.pi) * G)))/mass)
             **((2*n)/(n-3)).real
```

Radius and Pressure can be achieved with the changing central density of the star through the manipulated equations:

```
def radius(density_c):
    return ((n+1)/(4*np.pi*G))**0.5 * K**(0.5)
    * density_c**((1-n)/(2*n)) * xi_1

def pressure(density):
    return K * density**(1+(1/n))
```

The remaining physical properties to achieve for the first star creation object so I can iterate it over a luminosity time-dependence function would be the initial temperature and luminosity of the star. These functions can be reused during the iteration to achieve the new values from the evolving star. *effRatio* is currently a placeholder way of obtaining the Effective temperature of the star through the ratio between the central temperature and the effect temperature of the sun.

```
effRatio = 2722 #Ratio between
#Central Temperature and Effective Temp

def temp(density):
    return (((pressure(density)
    * mu*m * (K**n)
    * (k / (mu*m))**(-n))/k)**(1/(n+1))
    /effRatio).real

def Lum(temp, radius):
    return 4 * np.pi * radius**2
    * sigma * temp**4
```

With Luminosity, I can derive a function that returns the mass from the new luminosity from the time-dependence function.

```
def L(t,mass):
    return (initialTemp) * (1
    - (5/4)*(psi + 1)*
    ((mu * initialTemp)/(mass * Q))*t)\
    **(psi/(psi+1))

def massFromLum(lum, time):
    return (5 * lum * (psi + 1)
    * time * mu
    *(lum/initialTemp)**(1/psi))
    / (4 * Q \
            * ((lum/initialTemp)**(1/psi)
            + lum*initialTemp))
```

Utilizing the derivated equations above, I obtained a set of equations to describe and find the physical property of the sun at every time interval for an N number of steps up to any amount of years.

```
N = 20 #Number of Time Slices
time_space = np.linspace(1,8e9,N)
#Time from 1 Year to 8 Billion Years

for t in time_space:

    #Mass Change from temperatrure difference
    lum = L(t,randMass)
    mass = massFromLum(lum,t)
    centralDensity = densityFromK(mass)
    radiusStar = radius(centralDensity)
    centralPressure
    = pressure(centralDensity)
    temperature = temp(centralDensity)

    newStar =
    Star(mass,lum,radiusStar,temperature)
    starArray.append(newStar)
```

RESULTS

Figure 1. shows the solution for N = 1.5 Lane-Emden ODE achieved from the ODE method used which is very similar to the table of constants that I am suppose to get.

Figure 2. shows the solutions for other N other than 1.5, I stopped at 4 because 5 doesn't converge and goes to infinity. The values are still consistent with the table of constants for Lane-Emden solutions for each N value.

Once the constants for the Lane-Emden is solved for each N, it doesn't have to be re-calculated because it will remain the same alongside K. Only the central density changes that results in the time-dependence variable to change the other physical properties are different time-steps.

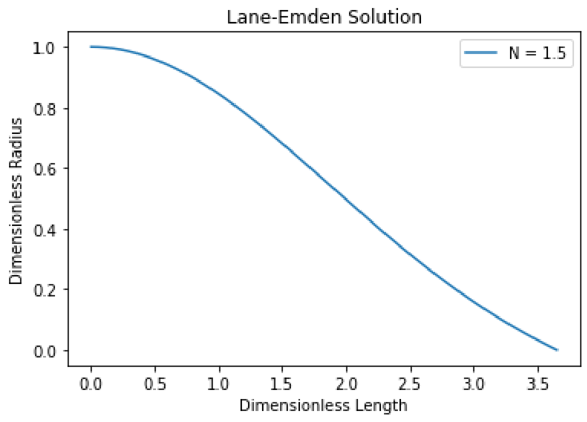

FIG. 1. Solution to Lane-Emden for N = 1.5

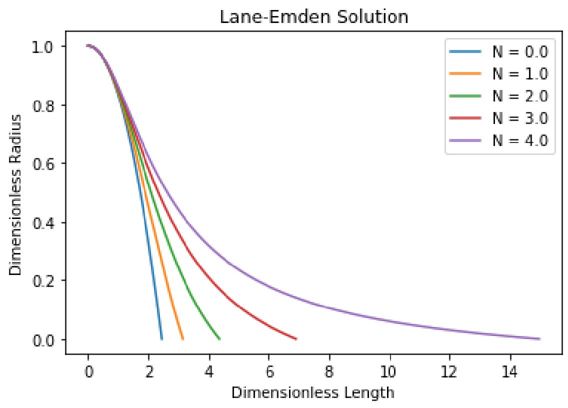

FIG. 2. Solution to Lane-Emden for N = 0 to 4

Still got some errors with the math that results in weird numbers mainly with the time-dependence function. Such as Figure 3 should have a steady increase of effective temperature in similar to Figure 4 because when the Sun moves away from the main-sequence, it's effective temperature and luminosity should be growing proportionally to each other.

The results for mass and radius shown by Figure 5 and Figure 6 seem to be very unexpected since mass should be decreasing since the conservation of mass is utilized meaning that mass shouldn't at any point increase in an isolated system.

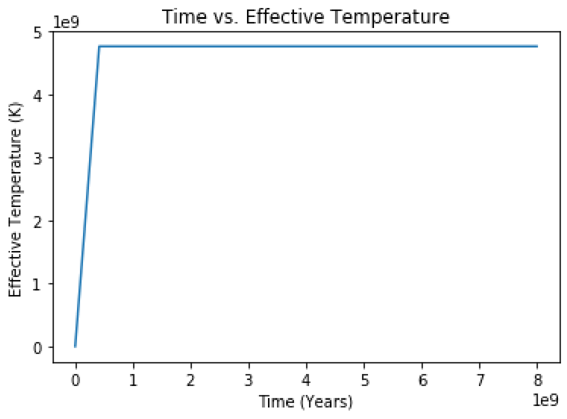

FIG. 3. Graph for the value of Effective Temperature in K as the Sun ages

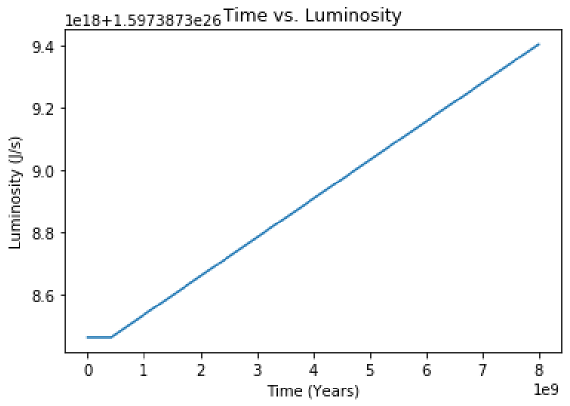

FIG. 4. Graph for the value of Luminosity in J/s as the Sun ages

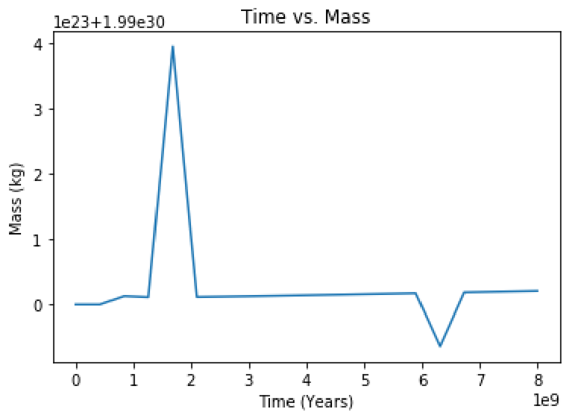

FIG. 5. Graph for the value of Sun's Mass in kilograms as the Sun ages

The mass should be decreasing since the composition of the sun is constantly being used up as the time-steps increase meaning that mass is being converted into energy. Radius similar to mass should be swelling up and increasing but instead I'm getting an opposite result.

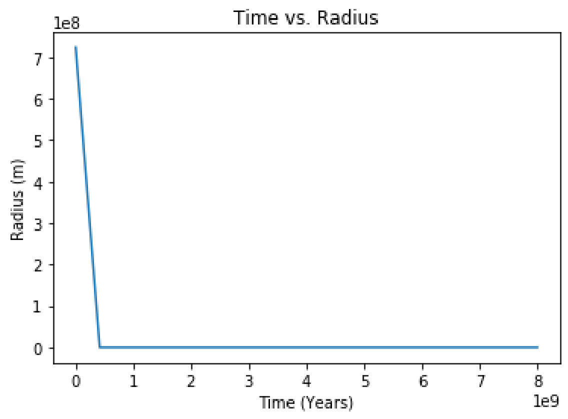

FIG. 6. Graph for the value of the Sun's Radius in meters as the Sun ages

The result are not in compliance to what I predicted or expected the stimulation to do. There seems to be some error in the derivation of the equations from their original state in order to manipulate the variables around to solve for a specific variable or there is some aspect of stellar structure that I didn't include or excluded that is creating these anomalies in the results that I don't have an explanation for.

DIFFICULTIES

IMPROVEMENT SUGGESTIONS

SUMMARY

[1] W. Carroll, Bradley & A. Ostlie, Dale (2006). An Introduction to Modern Astrophysics (2nd Edition)