Red Giant Simulator

Nicholas Clancy

Abstract

A computer program was written in order to make a simulation of the suns evolution from its present day state to the start of the red giant stage. The program solves the four fundamental equations of stellar structure numerically and works out the change in hydrogen mass fraction per mass step per time interval (assuming constant reaction rates per time interval). It also computes the pressure, temperature, luminosity and radius of the sun. The model of the present day sun gives a core temperature, core pressure, luminosity, mass and radius of $2.07\text{x}10^7$K, $4.8\text{x}10^{16}$Pa, $3.37\text{x}10^{27}$W $2.02\text{x}10^{30}$Kg and $3.46\text{x}10^8$m respectively. These values are close in astrophysical terms to the accepted values of $1.56\text{x}10^7$K, $2.29\text{x}10^{16}$Pa, $3.86\text{x}10^{26}$W, $1.99*10^{30}$Kg and $6.96\text{x}10^8$m from the standard solar model (Ref.1.).

The model was then run over n time intervals in order to find out if the sun model could be made to evolve into a red giant as the helium mass fraction increased. Moving the model forward in time steps of 100 million years, my model made six time steps, reducing the core hydrogen mass fraction to 0.19, before the model became seriously unstable and couldn't be made to form a star with anywhere near the correct mass. I concluded from this that a program of this limited complexity can only handle performing stellar evolution up to a point, and is not suitable for simulation of the red giant stage of the sun.

Contents page

1. Introduction and aims

For the first part of this project, I had to make a computer program that analytically solved the four fundamental equations of stellar structure in terms of mass steps. The four differential equations that give the changes in important parameters per mass step are shown below (Ref.2). These were used instead of their equivalents in terms of radius because it will be necessary later to alter the hydrogen mass fraction per mass step, so the radius may change but the mass must stay approximately constant.

$$\frac{dP}{dM} = -\frac{GM}{4\pi r^4}$$ (Ref.2) Eqn. (1.1)

$$\frac{dr}{dM} = \frac{-1}{4\pi r^2 \rho}$$ (Ref.2) Eqn. (1.2)

$$\frac{dL}{dM} = \varepsilon$$ (Ref.2) Eqn. (1.3)

$$\frac{dT}{dM} = \frac{-2\kappa L}{64\pi^2 acr^4 T^3}$$ (Ref.2) Eqn. (1.4)

P = pressure (pa)
r = radius (m)
L = luminosity (W)
T = temperature (T)
M = total mass enclosed(kg) in radius r
c = $3*10^8 ms^{-1}$ (speed of light in a vacuum)
G = $6.67*10^{-11} Nm^2kg^{-1}$(Gravitational constant)
a = $7.565*10^{-16} J K^{-4} m^{-3}$ (radiation density constant)

The other parameters ε, κ and ρ are the energy density, opacity and density. These are given by the equations shown below. Note: these are approximations, refer to the literature for details.

$$\varepsilon = 9.5 \times 10^{-37} X_1^2 \rho^2 T^4 \text{ Wm}^{-3}$$ (Ref.2) Eqn. (1.5)

$$\rho = \frac{M_a P}{kT}$$ (Ref.2) Eqn. (1.6)

$$\kappa = 0.02(1 + X_1)$$ (Ref.1) Eqn. (1.7)

X_1 = hydrogen fraction
M_a = average particle mass

Altogether this gives four highly non-linear, coupled, differential equations for r(M), T(M), P(M) and L(M) with eight boundary conditions. Of these four are known –

P(M) and T(M) must be zero at the surface, and r(M) and L(M) must be zero at the centre. The determination of the other four boundary conditions – the core pressure, core temperature, the luminosity of the star and the stars radius formed the bulk of this project. In addition these equations had to be solved in such a way that the pressure and temperature converged to crossed zero at approximately the same mass, giving an overall mass for the sun.

After the sun model had been made, it was then extended to calculate the mass fraction of hydrogen per mass step that was burnt in one billion years (assuming the reaction rate stayed unchanged this long). The model was also altered to accommodate the effects of electron degeneracy and of helium burning, which becomes important as the stars hydrogen fraction becomes smaller and the central pressure increases. Then the model was run several times to simulate the progression of the hydrogen burning and the start of helium burning.

actually 10^8 (100 million) looking at program

2. Theory

Reaction rates

The fusion of hydrogen and helium inside the star model can be simply modelled by the simulation of the slowest stage of the process, the other stages are assumed to take no time at all (in the time scales involved this assumption is justified). The fusion rate of two nuclei A and B at temperature T and density ρ can be shown to be:

$$R_{AB} = 6.48\times10^{-24}\frac{n_a n_b}{A_r Z_A Z_B}S(E_0)\left(\frac{E_G}{4kT}\right)^{2/3} e^{\left(-3\left(\frac{E_G}{kT}\right)^{1/3}\right)}\ \mathrm{s^{-1}m^{-3}}\ \text{(Ref.2) Eqn. (2.1)}$$

A_r = reduced mass of the nuclei (u)
n_A, n_B = number of nuclei of type A and B per m^3
Z_A, Z_B = proton no of nuclei A and B
$S(E_0)$ = nuclear fusion factor (assumed to be constant for purposes of model)

E_G is the Gamow energy, and is given by:

$E_G = (\pi\alpha Z_A Z_B)^2\, 2m_r c^2$ (Ref.2) Eqn.(2.2)

Where m_r = reduced mass

Note that in the model, an approximation to this full equation is employed. For hydrogen fusion, the slowest step is the initial fusing of two protons to form a deuteron as shown below

$$p + p \rightarrow d + e^+ + \nu_e$$

Equation (2.1) approximates to the equation below:

$R_{pp} = 1.88x10^{-24} X_1^2 \rho^2 T^4\ \mathrm{s^{-1}m^{-3}}$ (Ref.1) Eqn.(2.3)

This reaction is extremely slow, so slow that it has not been possible to measure its cross section in an experiment. This makes it difficult to know what the fusion factor $S_{pp}(E_0)$ is. However theoretical calculations suggest it is 3.8×10^{-22}keV barns, and it will assumed to be so for this model.

For helium burning, the situation is different. First two Helium nuclei join to create ^{8}Be, as is shown below, but this is an unbound state by 92keV and rapidly decays back into two alpha particles.

$$^4He + {}^4He \longleftrightarrow {}^8Be$$

However, if the temperature and density are high enough then an equilibrium concentration of ^{8}Be is present as they are created and then decay quickly. The last step in this process is joining of another Helium nuclei to the ^{8}Be to form an excited state of ^{12}C. This normally decays straight back into 3 Helium nuclei, but occasionally it decays by two gamma rays or by pair production back to the ground state. This unorthodox production route can be shown to occur with a reaction rate given by

$$\frac{dn_{12}}{dt} = \frac{n_4^3}{\tau(^{12}C^* \rightarrow {}^{12}C)} 3^{3/2} \left[\frac{h^2}{2\pi m_4 kT}\right]^3 \exp\left[-(m_{12}^* - 3m_4)c^2 / kT\right] \text{s}^{-1}\text{m}^{-3} \text{(Ref.1) Eqn2.4}$$

Where:

$\tau(^{12}C^* \rightarrow {}^{12}C) = 1.8x10^{-16} s$ (Mean time for carbon-12 decay)

Subscripts 4 and 12 denote Helium or Carbon

n = number of nuclei per m^3

m = mass of a single nuclei

This releases energy at the rate below

$$\varepsilon_{3\alpha} = (3m_4 - m_{12})c^2 \frac{dn_{12}}{dt} \qquad \text{(Ref.1)} \qquad \text{Eqn.2.5}$$

$(3m_4 - m_{12})\, c^2 = 7.275$ MeV

3. Execution of project

3.1 The creation of a working sun model

My project is similar in some respects to an earlier project done by Paul Batham, which was a sun simulation done in steps of radius from the centre outwards but based around the same equations as my project but rearranged. Using his earlier program as a frame work I modified the central equations to the new arrangements and made other alterations in order to make it work in mass steps. This was necessary because later I will be altering the model to reduce the hydrogen mass fraction per mass step, and so while the radius will be changing, I will need to be able to carry over the hydrogen mass fraction per mass step to the next time step. The mass will be kept approximately constant throughout the evolutionary steps I will be making the star perform so my array of the hydrogen mass fraction will be transferable.

There were several problems with the program at these early stages, including it giving a floating exception at the centre of the star because several of the new equations divide by R or M, both of which are zero at the centre. This was solved with the aid of a simple IF statement. This set dP and dT to zero at the centre if the mass enclosed was less than one Kg. It also used an approximation to r that assumed dr could be approximated to a sphere at the centre. of constant ~~mass~~ density

One large change I made was removing the iteration algorithm that would find the correct core temperature. This would have to be reintroduced later but would need to be quite different due to the different nature of the program.

Once the program appeared to be running correctly, I ran it using the pressure value that had been left in there by Batham (2.29×10^{16}Pa)(Ref.4), but stepping down the central temperature every time. I made a graph of the central temperature against surface temperature, which gave nonsensical results when extrapolated (The central temperature had to be negative in order to make the surface temperature cross 0). Clearly this was not an accurate indicator of what the solution was for this pressure guess. I then tried reducing the pressure by a factor of ten while keeping the core temperature the same. This simply made the temperature reduce too slowly and I abandoned that line of enquiry. I decided to then set the pressure at 5×10^{16} Pa and reduce the temperature slowly. This quickly produced results that were unrealistic- after about two steps the temperature would become random, either going negative or becoming higher than the core temperature. Deciding this core pressure was too high, I then tried 2.5×10^{16} Pa and 2.29×10^{16} Pa again. After some more iterating, I realised that there was a problem with my methodology – or that solutions to these equations are ridiculously difficult to find. Basically whenever I reduced the core temperature it went random very quickly.

I decided to plot the graphs of individual stars in Sigmaplot. I discovered that when the core temperature is too low, the temperature crosses zero and then starts again at a random point, as shown in the graph below.

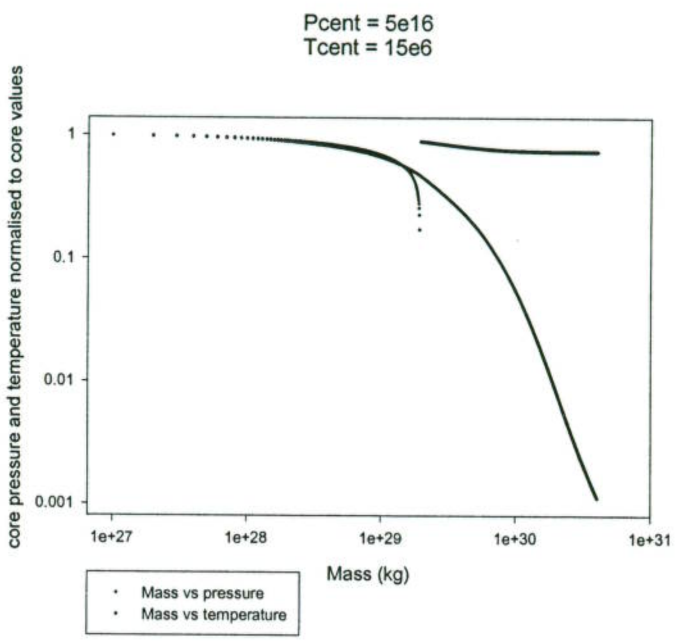


Fig (3.1) this graph is of mass enclosed against temperature and pressure as a fraction of the core values. Note that the temperature reduces too quickly and crosses zero before reappearing at a random value. This is an example of the core temperature being set too low. This was for a central pressure of $5x10^{16}$Pa and a central temperature of $1.5x10^{7}$.

This meant that many of the results I had got so far were actually valid, and I could reanalyse the data I already had in a much more informative way. My next discovery was that I had set the model to stop at one solar mass, which eliminated the possibility of finding a solution that was slightly over that value. I changed it to two solar masses to give myself plenty of leeway and to make sure the program didn't just keep going on forever if the pressure or temperature was asymptotic.

As an experiment I also set the mass step to half its current value in the hope that more accurate results would make it easier to find a result. In addition at this point I added IF statements that stopped the program when temperature and pressure were either zero or negative, and when the temperature went higher than the core temperature. This helped reduce the amount of data that I was storing and also made iterating that much quicker. With these new tools, I started finding zero-crossing values of temperature and pressure quite readily. The challenge then was to find a solution for a working star i.e. the pressure and temperature both crossed zero at the same (or a close approximation to the same) mass.

While trying to find a valid star solution for 2.29×10^{16}Pa, my results seemed to indicate that there were two solutions to the equations. To investigate this I made my program write out the mass crossing values of pressure and temperature as it found them. This was done by interpolation since the temperature and pressure drop by very large steps even if you examine every step it makes. For this a constant gradient was assumed, which is close enough for our purposes. Using this more sensitive technique I discovered that neither of the previously discovered "solutions" actually was real and the actual solution was just before them . The two false solutions were just coincidences thrown up by the chaotic behaviour that occurs when the core temperature is too low. These gave me a working solution for a star with core pressure and temperature of 2.29x1016Pa and 4.832x106K respectively, and mass 1.653×10^{29}Kg. This is equivalent to only 0.0825 solar masses, but now that I had a working technique for finding solutions I could step up the pressure in an attempt to find a solution that approximated to the sun. The graph of pressure and temperature against time for this solution is shown below.

superscript

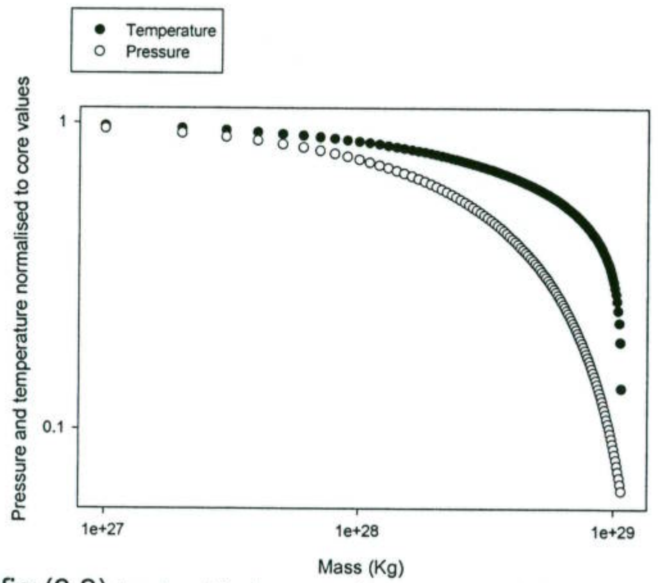


fig (3.2) Log-log plot of mass against pressure and temperature
T_{cent} = 4.7e6 , P_{cent}=2.29×10^{16}, star mass is 1.653×10^{29}Kg

Using my new- found iteration technique I tried to find masses for core pressure values both higher and lower than 2.29×10^{16}, in order to see how the mass varied with pressure. While I was fairly certain that higher pressure would mean a larger mass for the star, I wanted to validate my solution by collecting more data points and then use that knowledge to guess what central pressure would be needed to achieve an approximation to the Sun's mass. Higher central pressures gave higher mass stars, as I had inferred. Using this method I made a star with mass 1.25×10^{30}Kg, central pressure 4.5×10^{16}Pa and central temp 1.8×10^{7}K. I was getting close but the time involved in iterating manually was prohibitive due to the large amounts of data generated by the model. To get around this I created an iterative algorithm that finds the correct core temperature for a give core pressure. This works iterating down from a suitably high core temperature, backing up and reducing the core temperature step by a factor ten if the system overshoots and the temperature becomes too low. Due to the chaotic nature of the system when the temperature is too low, the iterative program is not totally reliable but produces solutions in most cases. Using this algorithm I quickly found a central pressure which gives a solution that is very close to the accepted values for the sun. The initial result of the sun model is shown below.

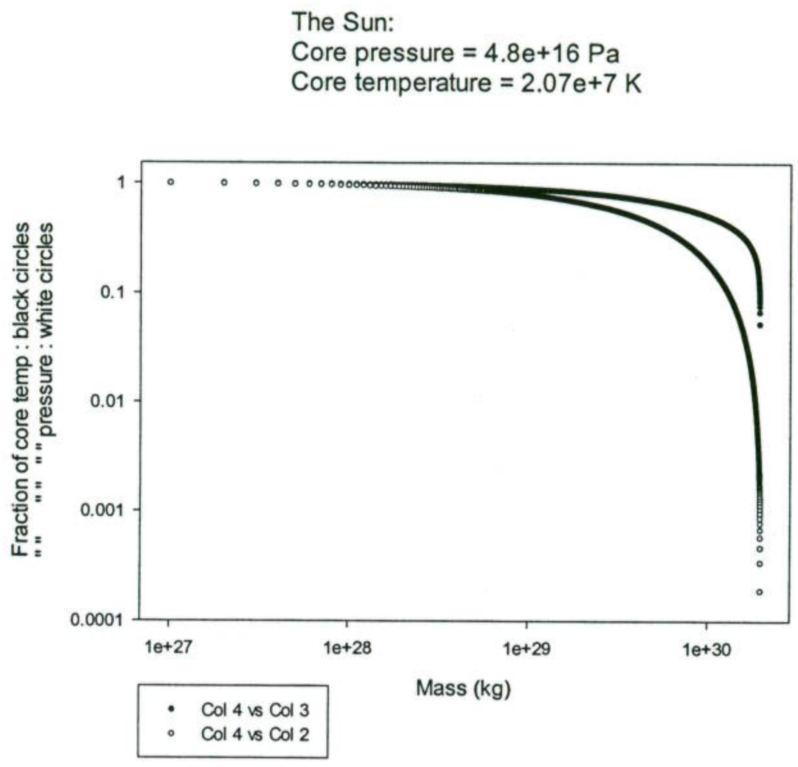


Figure (3.3) The mass enclosed against pressure and temperature as a fraction of the core values for my sun model (The temperature is the topmost line).

3.2 Simulating its evolution into a red giant

Now that I had a working sun model I now had to simulate its evolution. From the start I recognised that this model does not take account of convection so I could simulate the change in hydrogen mass fraction per mass step and assume no movement of materials between the different mass shells. I made my sun model as the starting point of the evolution i.e. I would try and make my model evolve from a starting hydrogen mass fraction $X_1 = 0.71$. At first I tried to make it alter the mass fraction as the program made the star, but after a while I realised that would make the mass fraction get altered every time the temperature DO loop ran. Once I realised this I put the evolution algorithm as a separate DO loop at the end of the program that called up the required parameters and used them to alter the mass fraction at each step. This was all based around arrays, the manipulation of which proved difficult due to my inexperience in using them. The equation below was used to work out the new array points:

$$X_{1(new)} = X_{1(old)} - \left(\frac{4R_{pp} \times timestep \times m_H}{\rho} \right) \qquad \text{Eqn.(3.2.1)}$$

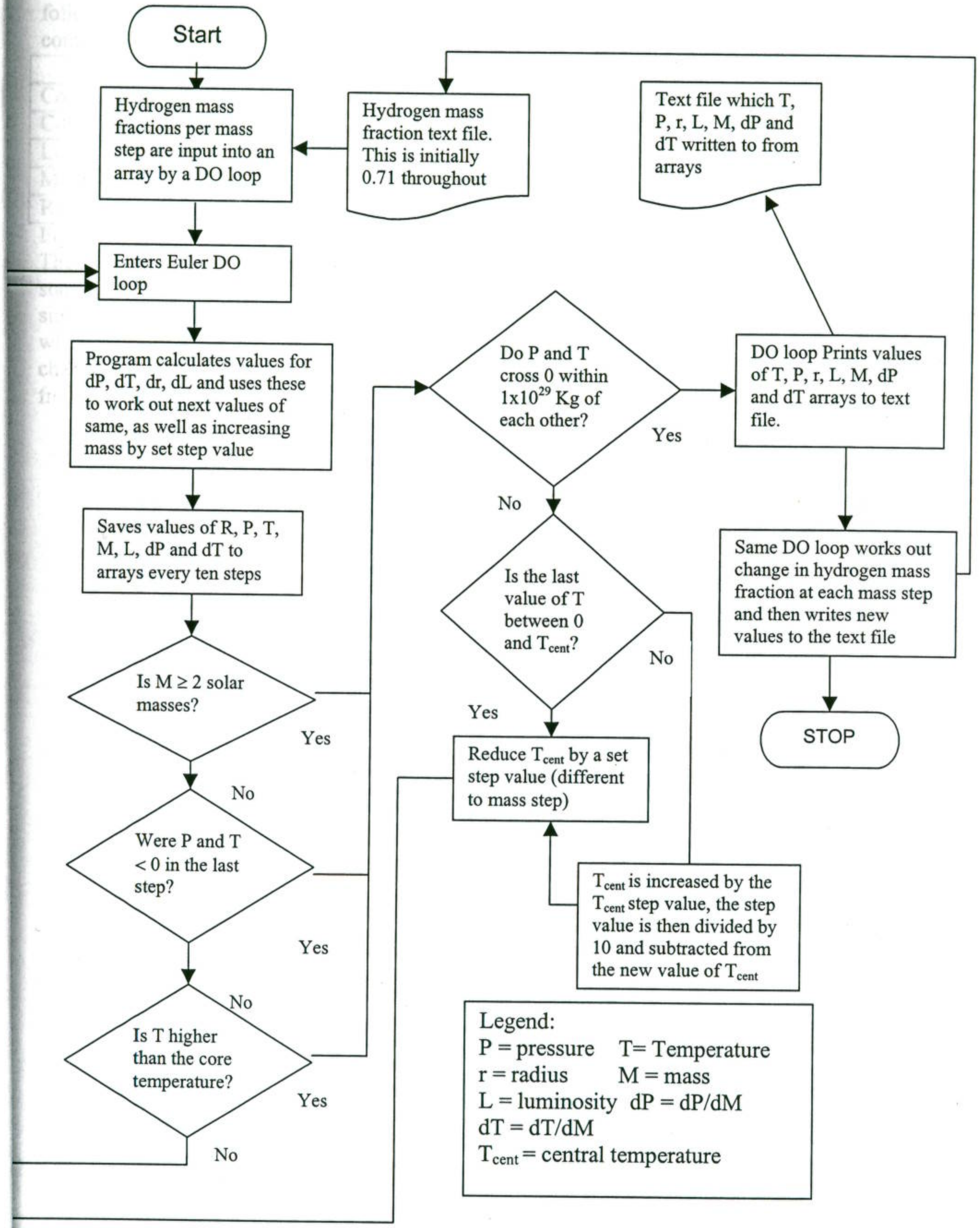
Once this was working, I now had a model that would advance one time step from the initial values but could not be made to evolve any further. I set the time interval to 100 million years as a reasonable value for a time step where a constant reaction rate could be assumed but that would also make for noticeable differences in the hydrogen mass fraction.

To implement further time steps, it would be necessary to read in to an array the text file that contained the hydrogen mass fractions for the last time step, then use this as a basis for the calculation of the new properties of the star. This was done by way of a DO loop at the start of the program that used a read statement to read in the mass fractions from a file. Then inside the main star creation loop it calls up the required array variable in order to calculate the luminosity at that point. At this point I discovered that just running the program again did alter the mass fraction but wouldn't necessarily return the correct mass value. To account for this I had to iterate again towards the correct central pressure, while also making sure that when an incorrect result was returned I deleted the altered mass fractions and replaced them with the originals. I had to settle in some cases for an approximation to the correct mass value, since the program would often behave unpredictably, returning floating exceptions at some values for no clear reason. This problem became worse the more evolution steps I took. After the sixth time interval however, the hydrogen mass fraction at the core was reduced to 0.19 and I couldn't find a core pressure value that returned a mass anywhere near one solar mass. Most values simply returned a floating exception, and only a couple of times did I manage to make a working star, but these had mass approx. 2 solar masses, nowhere near accurate enough to justify continuing the star evolution. So, due to time constraints I was forced to stop here, without a working red giant model.

The iterating towards a correct solar mass value at later stages of evolution highlighted the instability of the program. A large random factor seemed to be involved in what mass star it returned, making iterating very unreliable.

4.Program algorithm

The finished program algorithm is illustrated by the flow diagram below. (Fig(4.1))


Start
Hydrogen mass fractions per mass step are input into an array by a DO loop
Hydrogen mass fraction text file. This is initially 0.71 throughout
Text file which T, P, r, L, M, dP and dT written to from arrays
Enters Euler DO loop
Program calculates values for dP, dT, dr, dL and uses these to work out next values of same, as well as increasing mass by set step value
Do P and T cross 0 within 1x10^29 Kg of each other?
Yes
No
DO loop Prints values of T, P, r, L, M, dP and dT arrays to text file.
Saves values of R, P, T, M, L, dP and dT to arrays every ten steps
Is the last value of T between 0 and Tcent?
No
Yes
Same DO loop works out change in hydrogen mass fraction at each mass step and then writes new values to the text file
Is M ≥ 2 solar masses?
Yes
No
STOP
Reduce Tcent by a set step value (different to mass step)
Were P and T < 0 in the last step?
Yes
No
Tcent is increased by the Tcent step value, the step value is then divided by 10 and subtracted from the new value of Tcent
Is T higher than the core temperature?
Yes
No
Legend:
P = pressure T= Temperature
r = radius M = mass
L = luminosity dP = dP/dM
dT = dT/dM
Tcent = central temperature

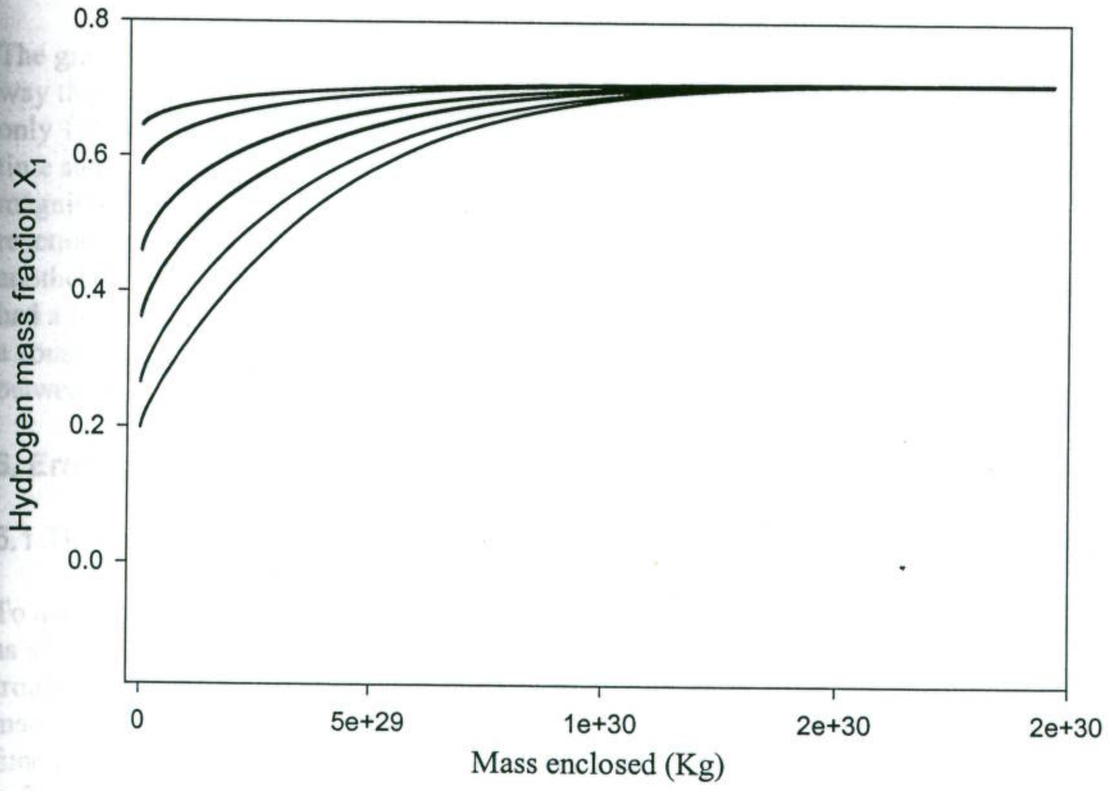
The results above indicate that my program's results are of the same form and mostly of the same order of magnitude as is found in the literature (fig.5.1.6). If the literature is considered a measure of accuracy then my program performs well and seems to produce results that are consistent with what's been inferred about stellar structure.

Most of the figures above are rendered in terms of mass and all of them are linear-linear plots. However I was unable to find a graph which had the form of mass against pressure for a 1M⊙ star in the literature so was forced to substitute a radius against pressure graph (fig.5.2), with a comparison result of the same form.

Three of the graphs deviate from the form of the textbook results at the highest masses. The radius against mass graph has a tail that should not be present, as does the pressure against radius graph, while the temperature against mass graph has a few negative points, which should not be present. This is a consequence of the temperature-crossing zero before the pressure, once this happens the temperature behaves in a random way that can sometimes skew results slightly.

5.2 Results from the star evolution

The results from the star evolution have been combined into the graph shown below:



Fig(5.2.1)Graph of hydrogen mass fraction against mass for six 1×10^8yr time intervals for my Sun mode

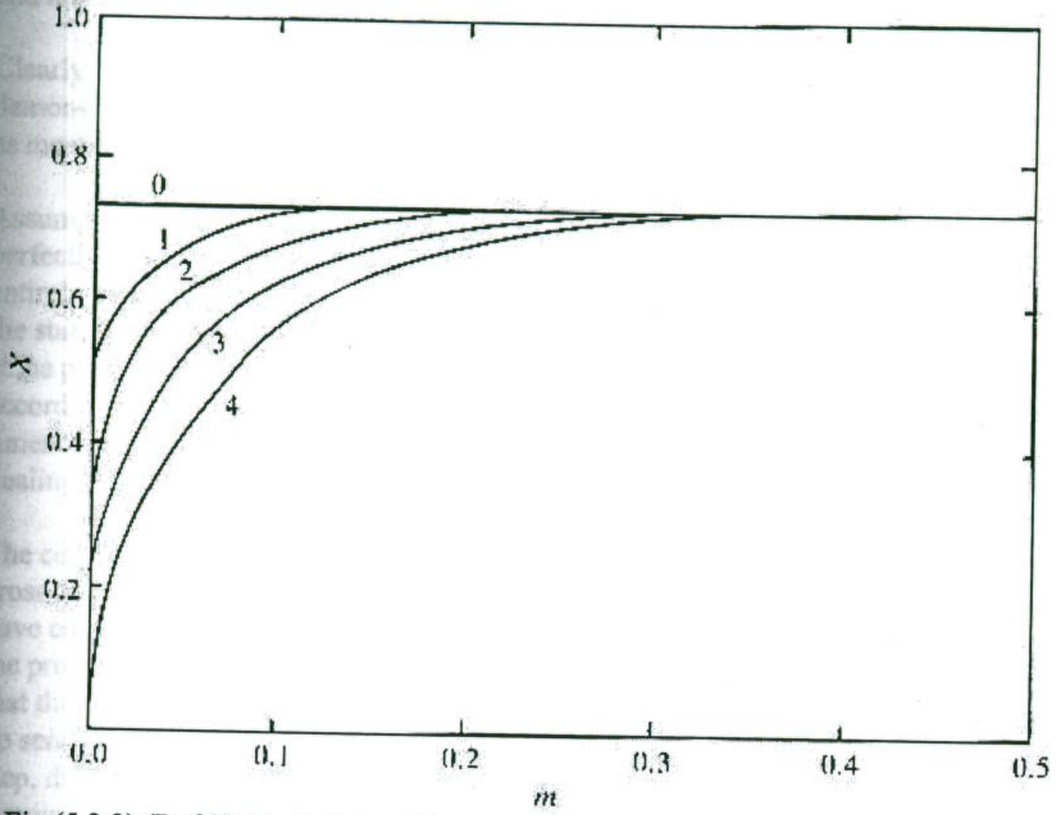


Fig (5.2.2) (Ref.2) The depletion of hydrogen in a low mass star. The numbered curves refer to successive stages in the star's evolution. Unfortunately the reference did not specify what ages corresponded to which numbers, but the graph form is what's important.

The graphs above demonstrate that the hydrogen burning in my star progressed in the way that the literature claims, but that it progressed very quickly. My time steps were only 10^8yrs each, yet my core had a hydrogen mass fraction of only 0.19 after only six time steps. I think this may be a consequence of my luminosity being an order of magnitude higher than the observed value, which implies that it has a much higher reaction rate than the real sun. The real sun is expected to burn hydrogen for approx. another 5×10^9yrs (Ref.2). Two other large assumptions that were made that may have had a large effect on the speed of the evolution was the fact that my program assumed a constant reaction rate for the duration of the time interval and that movement between the different mass shells was ignored also.

6. Errors and accuracy

6.1 The sun model

To accurately model the sun would require a large amount of computing time as well as a team of researchers, so with my rather more limited resources and time, only a 'rough sketch' of the sun is possible. This project has concentrated on generating the macroscopic properties of the sun to within a reasonable accuracy, using the four fundamental equations of stellar structure as its basis. The only error that is sufficiently large enough to be considered "wrong" is the luminosity value, which is an order of magnitude higher than the Sun's. The sun my model created has very nearly the same mass as the real sun but has half the radius, twice the central pressure and has a third higher core temperature.

Clearly it is not an accurate representation of our Sun, but its macroscopic properties demonstrate the same behaviour as is observed in much more complex models as well as mostly being of the same order magnitude.

Assumptions that were made in order to simplify the program considerably include; perfectly spherical Sun, radiation pressure is negligible, convection has been ignored entirely and pressure and temperature have been assumed to be zero at the surface of the star. In addition simplified versions of some equations have been employed at some points. Of these, the lack of convection is the biggest assumption, though according to some sources (Ref.2) large scale convection in stars occurs with timescales of the order of 10^{12} years – a lot longer than the timescales I will be dealing with – which are of the order of 10^{8-9}yrs.

The central problem of this program has been making the pressure and temperature cross zero at the same point, which has required iteration procedures. The method I have employed for this project has been to keep the core pressure the same and run the program with a progression of lower core temperatures, until the program detects that they both cross zero within $1\text{x}10^{29}$Kg. If the surface temperature it returns makes no sense, then the program increases the core temperature by the core temperature step, divides the core temperature step by 10 and then subtracts that from the core temperature. In this way it iterates in finer and finer steps until a working star solution is achieved. This process however is unreliable for at least two important reasons

1) When the core temp is too low, it does not always give a result that makes no sense. The program checks for the temperature being either higher than the central temperature or less than zero. In this way the iteration program can think the temperature is too high and miss out a solution.

2) If the mass step size is too large, then the approximations $dP \approx \Delta P$, $dT \approx \Delta T$ etc is not valid and inaccurate results are returned.

I tested to see if reason two was causing any inaccuracies. This was a simple matter of halving the step value and re-running the program. This didn't cause any significant changes, so it appears that the step value I'm using is appropriate for the task in hand. Problem one however is quite possibly a source of some of the program instabilities, though trying small differences in the central pressure can be employed to get around this, since temperature behaviour is chaotic once its crossed zero if the program is still running.

6.2 The sun evolution

The evolution of the star progressed in the way that the textbooks predicted, with smooth curves of hydrogen mass fraction slowly moving down as the core burnt its hydrogen, but it proceeded at too fast a rate. I believe this is due to the order of magnitude higher luminosity error, the only significant error in the original model. This implies that my Sun model fuses its hydrogen at a significantly higher rate, which is borne out by the fact that it was down to 0.20 hydrogen mass fraction at the centre after only six 10^8yr time steps. This is wrong compared to the predicted hydrogen burning time of the real sun (about $5x10^9$yrs) but is consistent with the higher temperature, pressure and luminosity of the model.(Ref.1)(Ref.2)

Again the model performed analogously to the real thing, but does not return very accurate values, which were exacerbated by the evolution process

7. Conclusions

The sun model performed well, possibly better than expectations. The only significant deviation from the standard solar model values (which agree well with the observed properties of the Sun) was the luminosity being an order of magnitude higher than the observed value. This seems to have had a knock-on effect of making the hydrogen fusion of the star proceed at a quicker rate, reducing the expected hydrogen burning time from the literature value of $5x10^9$yrs to an extrapolated 8-9 $x10^8$yrs. This had to be extrapolated since the program became highly unstable after the program had run six 10^8yr steps, refusing to return any usable values. I took this as an indicator that the program in its current form couldn't be made to form a red giant.

8. Acknowledgements

This project would not have been possible without my supervisor Dr. Ben Murdin, who I collaborated with closely on all aspects of this project. I also would like to thank Paul Batham for his program framework and advice, and Dr Paddy Regan for his help with the physics of the triple-alpha process. Finally I would like to thank my father Anthony Clancy for proof-reading my program.

9. References

(1) A.C. Phillips, The Physics of Stars, John Wiley and Sons (1994)
(2) R.J. Tayler, The Stars: their structure and evolution (1994)
(3) D.A. Ostlie and B.W. Carroll, An Introduction to Modern Astrophysics (1996)
(4) P. Batham, Stellar Structure Models (2000)

Appendix A: The computer program

Shown below is the computer program used to create the evolving sun model

```
!Nicholas Clancy, final year project, Red giant simulator
!This program can be used to analytically solve the four fundamental equations of stellar
!structure for a given core pressure, and also simulates the fusion of hydrogen for a
!given time interval from either a given hydrogen mass fractionstarting point read in
!from a file or a homogeneous value specified from inside the program. In addition, the
!power generation of Helium burning is implemented in the program as well as electron
!degeneracy pressure
Program Stellar_evolution
Implicit none
Double Precision :: R, P, L, M, T, R_Next, P_Next, M_Next, L_Next,evk,mpu
Double Precision :: K, G, Ma, X1, pi, h, dP, dM, dL, dT, Delta_R, T_Cent,dr
Double Precision :: Density, Step, Opacity, Power_density, a,mh,Rpp,Rpp1
Double Precision :: T_Cent_Step, T_Next, P_Cent, c, T_Effective,Rpp2,Rpp3
Double Precision :: Sigma,M_step,Msun,McrossP,McrossPflag,McrossT,McrossTflag
Double Precision :: delta_X1, interval, Hydrogen_fract(1:1000000)
Double Precision :: Powerpp, Power3alpha,dn12,NHe,mHe,mHeu
Real :: Dist(1:1000000), Pres(1:1000000), Temp(1:1000000)
Real :: Mass(1:1000000), Power(1:1000000), Pres_Grad(1:1000000)
Real :: Temp_Grad(1:1000000)
Integer :: j, No_Points, i, count,u,f,ios

! Sets the values of constants
h = 6.626d-34
K = 1.38D-23
evK = 8.6174D-5
G = 6.67D-11
pi = 3.141592654D0
c = 3D+8
a = 7.565D-16
Sigma = 5.67D-8
Msun = 2d30
mh =  1.673D-27
mpu = 1.00727647
mHe = 6.64449075d-27
mHeu = 4.0015
no_points = 1000000

! Sets the input parameters
step = 1D+26
T_Cent_Step = 1D+6
interval = 60**2*24*365*1e+8

!Loads the Hydrogen mass fraction at each mass step from a text file into an array
open (2, file="hydrogenfract.txt")
i=1
do i=1,no_points
        Read (2,*,iostat=ios) hydrogen_fract(i)
        IF (ios == -1) then
                exit
        end if
End do
close(2)

! Sets the boundary conditions
R = 0.D0
P_Cent = 24d+16
L = 0.D0
T_Cent = 112D+6
M = 0.D0

Temperature: do u=0,500

        !The Resets the values of T, P, R, M, and L
        T = T_Cent
        P = P_Cent
        R = 0.D0
        L = 0.D0
        M = 0.D0
        j = 1
        count = 0
        McrossP = -0.5E+30
        McrossT = 0
        McrossPflag = 0
        McrossTflag = 0

        !Calculates values at points in the star
        i=0
        Euler: do
                !This if assigns the current hydrogen fraction from the array to the value
X1
                if (count == 0) then
                        i = i +1
                        X1 = hydrogen_fract(i)
                end if
                !Determines values at points in the star
                Ma = 2.D0*1.66D-27/(1.D0+(3.D0*X1)+(0.5D0*(1-X1)))
                !If statement determines if electron degeneracy pressure is present
                IF (2.3E-38*((1d0+X1)/(2d0*mh)*(Ma*P)/(k*T))**(5d0/3d0) > P) THEN
                        Density = (P/2.3E-38)**(3d0/5d0)*((2d0*mh)/(1d0+X1))
                ELSE
                        Density = ((P*Ma)/(k*T))
                END IF
                !These lines calculate power generation inside the current mass step
                opacity = (1.D0+X1)*0.02D0
                NHe = (((1-X1)*density)/mHe)
                dn12 = ((NHe**3)/1.8d-16)*3**(3/2)*((h**2/(2*pi*mHe*k*T))**3)*exp(-
379.38d3/(evk*T))
                Power3alpha = 1.164d-12*dn12
                Powerpp = 9.5D-37*X1**2*density**2*T**4
                Power_density = Powerpp + Power3alpha
                !Determines dP, dT,dR and dL for the current mass step, If statement is to
                !prevent floating exception at M=0
                IF ( M <= 1.D0 ) THEN
                        dp = 0.D0
                        dT = 0.D0
                        dr = (3.d0*step/(4.d0*pi*density))**(1.d0/3.d0)/step
                        dL = Power_density/density

                ELSE
                        dp = -1.d0*(G*M)/(4.d0*pi*R**4.d0)
                        dT = -1.d0*(3.d0*opacity*L)/(64.d0*pi**2.d0*a*c*r**4*T**3.d0)
                        dr = 1/(4.d0*pi*R**2.d0*density)
                        dL = Power_density/Density
                END IF

                ! Determines the next value of the pressure, enclosed mass,
                ! power generated and the change in H fraction
                R_Next = R + dr*Step
                P_Next = P + dp*Step
                M_Next = M + Step
                T_Next = T + dT*Step
                L_Next = L + dL*Step

                ! This saves the values of R, P, T, M, L, dP and DT to the Distance,
                ! Pressure, Temperature, Mass and Power arrays every 10 steps to
                ! prevent the data file getting too big.
                if (count == 0) then
                        Dist(j) = R
                        Pres(j) = P
                        Temp(j) = T
                        Mass(j) = M
                        Power(j) = L
                        Pres_Grad(j) = dP
                        Temp_Grad(j) = dT
                        j = j + 1
                        count = 10
        end if

        ! Sets values of pressure and radius to next values
        P = P_Next
        R = R_Next
        M = M_Next
        T = T_Next
        L = L_Next
        count = count - 1

        !This exits the euler loop when the mass reaches 2 solar masses
        if (M > 4D+30) exit

        !These if statements determine if either pressure or temperature have
        !crossed zero and switch on a flag if so
        if (P*(P-dp*step) <= 0d0) then
                Print *, 'McrossingP = ', M-step+P/dP
                McrossP = M-step+P/dP
                McrossPflag = 1

        End if

        if (T*(T-dT*step) <= 0d0) then
                Print *, 'McrossingT = ', M-step+T/dT
                McrossT = M-step+T/dT
                McrossTflag = 1
        End if

        !This if checks if both pressure and temperature have crossed zero
        if (McrossTflag == 1 .and. McrossPflag == 1) then
                Print *,'Temperature & Pressure have crossed zero'
        Exit
        End if

        !This if checks to make sure that the temperature is not higher than the
        !core temp
        if (T > T_cent ) then
                Print *,'Temperature has gone higher than core temp'
        Exit
        End if

end do Euler

no_points = j - 1

!This if checks to see if P and T have crossed zero within 1x10**29 kg of each
!other and hence are a solution
if (McrossT <= (McrossP +0.1E+30).and. McrossT >= (McrossP -0.1E+30)) then
        if (McrossPflag==1 .and. McrossTflag==1) then
                PRINT *,'Working star created!'
                PRINT *,'McrossP =',McrossP
                PRINT *,'McrossT =',McrossT
        exit
        end if
        PRINT *,'Tcent =',T_cent
!The else if statements here calculate how much the core temperature is reduced
!for the next iteration
else  if ( T >= 0.d0 .and. T <= T_cent ) then
        T_cent = T_cent - T_cent_step
        PRINT *,'Tcent =',T_cent
else if (T_cent_step > 10D+3) then
        T_Cent = T_Cent + T_Cent_Step
        T_Cent_Step = T_Cent_Step / 10.D0
        T_Cent = T_cent - T_Cent_Step
        PRINT *,'Tcent =',T_cent
else
        T_Cent = T_cent - T_Cent_Step
        PRINT *,'Tcent =',T_cent
end if
```

```
        if (T_cent <= 0) then
              exit
        end if

        PRINT *,'Temperature iteration IF completed'

end do Temperature

!This determines the effective temperature.
T_Effective = ( L / (4.D0*pi*sigma*R*R) )**0.25D0

!prints the values of important parameters for the star
Print *, 'Central Pressure (Pa)' ,P_Cent
Print *, 'Optimum Central Temperature (K)',T_cent
Print *, 'Radius of star (m)',R
Print *, 'Mass of the star (Kg)' ,M
Print *, 'Total Power generated (W)',L
Print *, 'Surface density (Kg/m3)',Density
Print *, 'Surface Temperature (K)',T
Print *, 'Effective Temperature (K)',T_Effective
Print *, 'Surface Pressure (Pa)',P
Print *, 'Central temperature Step (K)',T_Cent_Step

!Opens a file to write the values to
open (1, file="valuesdoesthiswork.txt")
open (2, file="hydrogenfract.txt")

!Prints the Labels
75 format(T8, A8, T24, A8, T39, A11, T56, A9, T68, A15, T89, A5, T103, A5)
write (1,75) 'Distance','Pressure','Temperature','mass encl','power
generated','dP/dr','dT/dr', P_cent, T_cent

!This do loop prints out the values of r,P,T,M and L contained in the appropriate arrays
!to a text file, and then calculates the change in hydrogen mass fraction at the given
!mass shell for a given time interval assuming a constant reaction rate, before writing
!the new mass fraction values to another text file.
i = 1
firsttimestep: do i = 1,no_points

        write (1,*) Dist(i), Pres(i)/P_cent, Temp(i)/T_cent, Mass(i), Power(i),
Pres_Grad(i), Temp_Grad(i)
        !hydrogen_fract(i) = 0.71
        Density = ((Pres(i)*Ma)/(k*Temp(i)))
        Rpp=1.88D-25 *hydrogen_fract(i)**2*density**2*(Temp(i))**4
        hydrogen_fract(i) = hydrogen_fract(i)-((4*Rpp*interval*mh)/density)
        if (hydrogen_fract(i) <= 0) then
               hydrogen_fract(i)=0
        end if
        Write (2,*) hydrogen_fract(i)

end do firsttimestep

close (1)
close (2)


end program Stellar_evolution
```