UNIVERSITY OF CALGARY

SCHULICH School of Engineering

ENME 631, Fall 2021 - Numerical Methods for Engineers

Chandrasekhar's White Dwarf Equation

Date Submitted: December 15, 2021

Instructor: Dr. Abdulmajeed Mohamad

Jackson Uhryn, 30026391
Tom Mathias, 30164379
Malcolm Macdonald, 30160983

Contents

List of Figures

Abstract

Numerical methods enable mathematical problems to be solved with arithmetic calculations and are a powerful tool, especially in complex non-linear systems. We have utilized the Explicit Euler and 4th Order Runge-Kutta methods to solve the Chandrasekhar white dwarf ordinary differential equation numerically using the Python programming language. Chandrasekhar's equation is a coupled ordinary differential equation that relates the mass and radius of a white dwarf star using a Fermi gas pressure equation of state. The results of radius versus mass for white dwarf stars are plotted including non-relativistic and relativistic cases. Experimental observations of white dwarfs are also shown in order to validate the accuracy of predictions made by the Chandrasekhar equation and numerical implementations presented in this paper. We found that our numerical results match well with the current experimental observations of white dwarf radius-mass relations. The python code including hand implementation of the two numerical algorithms used is provided at the end of the document if the reader is interested in further reading or modifications of this work.

1 Introduction

Numerical methods are techniques by which mathematical problems are formulated so that they can be solved with arithmetic calculations [2]. These methods often involve numerous repetitive calculations and with the invention of modern computing, have become much easier to apply to real world problems. Although analytical solutions still provide valuable insights, numerical methods greatly expand one's capabilities to solve and confront difficult problems that involve large systems of equations, nonlinearities and complicated geometries [2].

Equations which are composed of an unknown function and its derivatives are called *differential equations* [2]. These equations play a fundamental role in engineering because many processes are best described by their derivative(s). The quantity being differentiated is called the *dependent variable* and the quantity it is being differentiated with respect to is called the *independent variable*. When there is only a single independent variable, the function is called an *ordinary differential equation* (ODE). The order of the equation is coined based on the highest derivative used. For example, an equation which has the second derivative as its highest differential term is called *second order*. However, higher order equations can always be re-written as first order systems by introducing new variables. Therefore, methods to solve first order equations can be used to solve any order of ODE [2].

A solution to an ODE is a specific function of the independent variable and parameters that satisfies the original differential equation. However, through this process a constant of integration will appear which suggests that an infinite number of solutions exist. In order for a unique solution to be found, there must be additional conditions stated. Typically, these are either initial values or boundary values. In general, an nth order differential equation will require n initial conditions to obtain a unique solution. If all conditions are specified at the same value of the independent variable (i.e. t = 0), then it is an *initial value problem*. However, if the conditions are specified at different values of the independent variable it is called a *boundary value problem*. Initial conditions usually have intuitive values for physical based problems. For example, if an ODE describes the falling motion of an object, typically the object will start from rest [2].

1.1 Objective

The objective of this report is to analyze *Chandrasekhar's white dwarf equation* using the Explicit Euler and 4th Order Runge Kutta methods. The results will be compared against each other and validated with real data observations from astronomers.

2 Literature Review

Chandrasekhar's white dwarf equation describes the relationship and behavior between density and radius of a white dwarf star. It governs the structure of degenerate matter in gravitational equilibrium and is represented by a initial value problem ordinary differential equation [3]. Before introducing the equation and numerical derivation in full, we will provide background information regarding white dwarfs, electron degeneracy, and Fermi gases: all of which are critical for proper understanding of the problem.

2.1 White Dwarfs

White dwarfs, also known as degenerate dwarfs, are stellar core remnants composed of mainly electron-degenerate matter. They are extremely dense objects, for example, a white dwarf the size of earth would weigh the same as our Sun [3]. No fusion takes place in a white dwarfs core and they are the final life stage of moderate size stars. Following the hydrogen fusion stage of a main-sequence star of low to medium mass, the star will expand to a red giant during which it fuses helium to carbon then oxygen and so forth, eventually ending with nuclear stable iron in its core. The outer layers of the red giant will eject ionized gas in an expanding glowing shell called a *planetary nebula*, after a sufficient amount of time, all that is left behind is a solid core, this is the remnant white dwarf. It should be noted that the majority of white dwarfs consist mostly of carbon and oxygen, however iron white dwarfs are possible too, it just depends on which stage of evolution the star stops producing nuclear fusion products [3, 4].

While nuclear fusion no longer occurs in white dwarfs, that does not mean they are cold. In fact, white dwarfs are extremely hot, the surface temperature of a white dwarf is equivalent to the core temperature of a red giant due to the immense mass condensed into a relatively minuscule volume. The black body radiation curve in the visible range for a white dwarf is essentially a flat line, this is why they appear white. The temperature of a white dwarf will slowly decrease over time as electromagnetic energy is radiated away from the star [3, 4].

Since no fusion occurs in the white dwarf core it can no longer support itself by the heat generated from fusion against gravitational collapse. Instead, collapse is prevented due to electron gas degeneracy pressure. In other words, the interior thermal energy is much smaller than the Fermi energy for white dwarfs, this is the opposite of main stage stars such as our Sun [3].

2.2 Electron Degeneracy

Electron degeneracy is a subset of degenerate matter, which is a highly dense state of fermionic (electrons, protons, neutrons) matter in which the Pauli exclusion principle governs the fermionic interactions [4].

In an ordinary gas, thermal effects dominant the interactions and system behaviour, meaning most electron energy levels are unfilled and electrons are free to move between them. However, as the particle density increases, electrons are stripped from their parent atoms creating a sea of electrons and eventually the lowest energy levels become fully occupied, meaning electrons are forced to occupy higher energy levels even at low temperatures. These electrons cannot shed energy by dropping to lower energy levels, therefore their momentum creates a *degeneracy pressure* that strongly resists further compression of the gas [3, 4].

Degenerate gas is a near perfect conductor of heat and does not obey the regular gas laws, instead it is governed by Fermi gas interactions, which is a quantum mechanical equation of state. It states that the pressure of a degenerate gas does not depend on temperature. The kinetic energy of the electron gas is high, but the rate of collision is low, therefore degenerate electrons can travel at speeds close to the speed of light, introducing relativistic effects. Additionally, the degeneracy pressure mainly depends on the speed of the electrons due to being stuck in their fully occupied quantum energy states [3, 4].

The physics of electron degeneracy yields a maximum mass of 1.44 $M_\odot$ (solar masses) for a non-rotating white dwarf as described by Chandrasekhar, after which point electron degeneracy pressure cannot support it, and it collapses to a neutron star or a black hole [3, 4]. A white dwarf that approaches this Chandrasekhar limit may explode as a type 1a supernova if carbon re-ignition causes a runaway chain reaction before collapse occurs [3, 4].

Due to the immense density of matter contained within a white dwarf, electrons are no longer bound to single nuclei and instead form a degenerate electron gas. Thus, they are accurately modelled by an ideal Fermi gas equation of state [3, 4].

2.3 Fermi Gas

An ideal Fermi gas denotes a state of matter in which non-interacting half-integer spin quantum particles, which are out the scope of this paper, interact. The behaviour of these gas systems are statistical in nature and in general, depend on the number density, temperature, and set of available energy states [3, 5]. A Fermi gas is simply a quantum model of a gas, it is similar to the ideal gas equation. In fact, at sufficiently low particle number densities and high temperatures, a Fermi gas can be accurately described as a classical ideal gas [5].

From the Pauli exclusion principle, fermions with the same quantum state cannot occupy the same quantum energy level. Therefore, once ground states are occupied, the fermions interact and create a sort of pressure within the gas. This interaction pressure is often called *degeneracy pressure* and as discussed before, is responsible for preventing gravitational collapse in white dwarf stars below the Chandrasekhar mass limit. The degeneracy pressure is derived from a thermodynamic analysis of the gas' internal energy [5, 4]. Due to its quantum mechanical nature, the full derivation is lengthy, complex, and not necessary to understand the Chandrasekhar equation derivation. Therefore, we will not provide a full derivation, but rather we will simply use the degeneracy pressure equation derived from Fermi gas analysis in its form typically stated in literature as seen in the following section.

3 Chandrasekhar's White Dwarf Equation

The traditional form of Chandrasekhar's equation is shown in eq: 1.

$$\frac{1}{\eta^2}\frac{d}{d\eta}(\eta^2\frac{d\phi}{d\eta}) + (\phi^2 - C)^{3/2} = 0, \quad \phi(0) = 1, \quad \phi'(0) = 0 \tag{1}$$

where η is the non-dimensionalized radial distance from the center of the star, ϕ is the density of the white dwarf, and C is a constant related to the density at the center point of the star. Note, the density will decrease radially and approaches zero as radius tends to infinity. Mathematically this is formulated as, $\sqrt{C} \leq \phi \leq 1$ [6].

While the form stated in Eq. 1 is succinct and elegant, it does not easily lend itself to numerical approximation. Therefore, a more straightforward approach is taken below in order to derive all terms in a dimensionless form (adapted from *Pols 2011*) [4]. First, we start with the equation $\rho = \frac{m}{V}$, which for a radial shape can be written in the differential form seen in Eq. 2.

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

where, *m* is the mass, *r* is the radius, and ρ is the density.

Secondly, variations in density with respect to radius is a function of mass, radius, and pressure as seen in Eq. 3.

$$\frac{d\rho}{dr} = -\left(\frac{dP}{d\rho}\right)^{-1} \frac{Gm}{r^2}\rho \tag{3}$$

where, *G* is the gravitational constant, and *P* is the pressure. However, Eq. 3 also depends on another differential equation derived from Fermi gas momentum (Fermi pressure is the outward force preventing gravitational collapse). The case of a relativistic Fermi gas is shown in Eq. 4.

$$\frac{dP}{d\rho} = Y_e \frac{m_e c^2}{m_p} \gamma \left(\frac{\rho}{\rho_0}\right) \tag{4}$$

where, γ is a factor given by Eq. 5, Y_e is the number of electrons per nucleon, m_p is the mass of a proton, m_e is the mass of an electron, ρ_0 and R_0 are the density and radius of the average white dwarf, respectively.

$$\gamma(y) = \frac{y^{2/3}}{3(1+y^{2/3})^{1/2}} \tag{5}$$

where, $y = \frac{\rho}{\rho_0}$.

In order to non-dimensionalize the derived equations, several relations need to be formulated for mass, density, and radius such that they only vary from 0 to 1. By forming dimensionless equations, the numerical approximations become more general and also have increased stability owing to the fact that extremely small and large values associated with dimensional quantities often cause divergence of numerical techniques.

$$M = \frac{m}{\frac{4}{3}\pi R_0^3 \rho_0} \tag{6}$$

$$q = \frac{\rho}{\rho_0} \tag{7}$$

$$x = \frac{r}{R_0} \tag{8}$$

After substitution, the final differential equations for mass and density can be written as Eqs. 9, 10.

$$\frac{dM}{dx} = 3x^2 q \tag{9}$$

$$\frac{dq}{dx} = -C\frac{qM}{x^2\gamma(q)} \tag{10}$$

where the constant, $C \approx 0.86$, is given by Eq. 11

$$C = \frac{4\pi G R_0^2 \rho_0 m_p}{3 Y_e m_e c^2} \tag{11}$$

While Eqs. 9, 10 describe the Chandrasekhar equation for a relativistic Fermi gas, the non-relativistic case also provides a useful comparison. Following a similar procedure as above, we will also derive the non-relativistic equations and numerically solve both sets of equations for the sake of comparison later.

In the non-relativistic case, the gas velocity simply becomes the ratio of electron momentum to mass, $v(p) = \frac{p}{m_e}$. Therefore, the pressure of the Fermi gas is,

$$P = \frac{1}{3}\int_0^{p_F} n(p)\frac{p^2}{m_e}dp \tag{12}$$

Using the same procedure as the relativistic case, the differential pressure can be written as,

$$\frac{dP}{d\rho} = Y_e\frac{m_e c^2}{3m_p}\left(\frac{\rho}{\rho_0}\right)^{2/3} \tag{13}$$

Performing the same non-dimensional techniques as before, the differential equations of mass and density can be derived as shown in Eqs. 14, 15.

$$\frac{dM}{dx} = 3x^2 q \tag{14}$$

$$\frac{dq}{dx} = -D\frac{q^{1/3}M}{x^2} \tag{15}$$

where the constant, *D*, is given by Eq. 16.

$$D = \frac{4\pi G R_0^2 \rho_0 m_p}{Y_e m_e c^2} \tag{16}$$

Eqs. 9, 10 and 14, 15 will be solved numerically in order to compute the radius versus mass of white dwarfs for relativistic and non-relativistic Fermi gases, respectively. The specific methods used are the Explicit Euler and 4th Order Runge-Kutta approximations, both of which will be discussed further in the following section.

4 Numerical Methods

Based on the derivation of the white dwarf equation completed above in Eqs. 9, 10, and 14, 15 it can be seen that these are 1st order ordinary differential equations (ODE's). From chapter 6 in the lecture notes, there are many methods to solve ODE's and for our case we chose two of the most common and well known: *Explicit Euler* and *4th Order Runge-Kutta* methods.

4.1 Explicit Euler

The Explicit Euler method was primarily chosen for its great simplicity and ease of use. The downside of this simplicity is that the Explicit Euler is 2nd order locally and 1st order globally accurate and conditionally stable. However, the ODE's to solve for this problem are 1st order so the Euler method works quite well given the step size is small enough to ensure convergence.

The need for the these methods arise from the kind of problem: "New value = old value + slope * step size".

$$y_{i+1} = y_i + \phi h \tag{17}$$

The equation constantly predicts the next value from the previous position used as an initial condition and traces the trajectory of the solution. Euler's Method is thus: the slope at the previous position is taken to be the slope over the whole of the subsequent interval, and so on. The first derivative of a function is an estimate of the slope at the point of interest, x_i. Where ϕ is $f(x_i, y_i)$ and $f(x_i, y_i)$ is the differential equation evaluated at x_i, and y_i. Substituting this back into the initial equation results in,

$$y_{i+1} = y_i + f(x_i, y_i)h \tag{18}$$

There are two major issues one runs into when solving ODE's numerically: truncation errors and round-off errors. While rounding errors are more self-evident and a result of the limitation of the tools being used, the truncation error is an error with the method itself. Truncation errors may be separated into two further categories: local and propagated truncation errors. A local truncation error results from using a method over a single step; a sudden change in the function over the step size of the method would cause this. A propagated truncation error occurs from the stacking of the approximation errors from all the previous steps.

The function in question may be expressed as a Taylor Series if the derivatives are continuous,

$$y_{i+1} = y_i + y_i'h + \frac{y_i''}{2!}h^2 + ... + \frac{y_i^{(n)}}{n!}h^n + R_n \tag{19}$$

Where h is $x_{i+1} - x_i$ and,

$$R_n = \frac{y^{(n+1)}(\xi)}{(n+1)!}h^{n+1} \tag{20}$$

and ξ is between x_i and x_{i+1}.

A more general form of the Taylor Series is given as:

$$y_{i+1} = y_i + f(x_i, y_i)h + \frac{f'(x_i, y_i)}{2!}h^2 + ... + \frac{f^{(n-1)}(x_i, y_i)}{n!} + O(h^{n+1}) \tag{21}$$

Where $O(h^{n+1})$ is the local truncation error in the Taylor Series, proportional to the step size. The error in the Euler method comes from only using a finite number of terms and excluding the rest from the Taylor Series. The truncation error may be expressed by subtracting Euler's method from the Taylor Series, yielding,

$$E_t = \frac{f'(x_i, y_i)}{2!}h^2 + ... + O(h^{n+1}) \tag{22}$$

For a small enough step size, the error may be further simplified,

$$E_a = \frac{f'(x_i, y_i)}{2!}h^2 \tag{23}$$

The Euler method uses straight line segments based on the slope of the point in question. It presumes that the function remains close enough to the slope over the size of the step. From the previous equation, the size of the error is proportional to the square of the step. Thus, shortening the step size will decrease the total error. Error can be completely removed as Euler's method is

a first order solution and if the function in question is a linear function, then the second derivative would be zero. This is slightly unfeasible for two reasons: finite amount of computational power and time, and the functions in question are often non-linear. The Euler method may be used to obtain a "quick and easy" but rough explicit solution or a precise but intensive explicit solution.

4.2 4th Order Runge-Kutta

The benefit of the Runge-Kutta methods is that they are more accurate owing to the usage of a Taylor series expansion. Runge-Kutta functions concern themselves with solving ODE's of the form,

$$\frac{dy}{dx} = f(x, y) \tag{24}$$

The Runge-Kutta method takes the general form of,

$$y_{i+1} = y_i + \phi(x_i, y_i, h)h \tag{25}$$

Where ϕ is the increment function; a weighted average of the slopes.

$$\phi = (a_1k_1 + a_2k_2 + ... + a_nk_n) \tag{26}$$

The weights a_i, $i = 1, 2, \ldots, n$ are constants which generally satisfy $w_1+w_2+\ldots+w_n = 1$ and the values of each k_i is the function in question being evaluated at some point between $x_n \leq x \leq x_{n+1}$. Expanded, this results in k values of,

$$k_1 = f(x_i, y_i) \tag{27}$$

$$k_2 = f(x_i + p_1h, y_i + q_{11}k_1h) \tag{28}$$

$$\begin{aligned} k_3 &= f(x_i + p_2h, y_i + q_{21}k_1h + q_{22}k_2h) \\ &\vdots \end{aligned} \tag{29}$$

$$k_n = f(x_i + p_{n-1}h, y_i + q_{n-1,1}k_1h + q_{n-1,2}k_2h + ... + q_{n-1,n-1}k_{n-1}h) \tag{30}$$

p's and q's are constant, and the previous k's recur in each subsequent iteration. This means that there is less data to store in a computer simulation involving the Runge-Kutta method at higher orders. n is the order of the method. By setting $n = 1$, $w_1 = 1$ and $k_1 = f(x_n, y_n)$. Thus, the first-order Runge-Kutta method results in the original Euler method.

As previously stated, *n* indicates the order of the Runge-Kutta method. For a chosen *n*, the values for the *w*'s, *p*'s and *q*'s are evaluated by setting the Runge-Kutta equation equal to the terms of the Taylor series expansion. The order of the method is suitable for the order of the differential equation. A second-order Runge-Kutta solution will be exact for a quadratic differential equation, a third-order for a third degree polynomial and so on. Additionally, as the order of the methods increase, the approximations become more precise, and the error is decreased as well.

The following is the derivation of the second-order Runge-Kutta method:

Start with the second-order version of the method,

$$y_{i+1} = y_i + (a_1k_1 + a_2k_2)h \tag{31}$$

Where k_1 is Eq. 27 and k_2 is Eq. 28. Eq. 31 needs values for the four constants a_1, a_2, p_1 and q_{11}. Recall the second order Taylor series, Eq. 21,

$$y_{i+1} = y_i + f(x_i, y_i)h + \frac{f'(x_i, y_i)}{2!}h^2 \tag{32}$$

Determine $f'(x_i, y_i)$ by the chain-rule,

$$f'(x_i, y_i) = \frac{\partial f(x, y)}{\partial x} + \frac{\partial f(x, y)}{\partial y}\frac{dy}{dx} \tag{33}$$

Now substitute Eq. 32 into Eq. 33,

$$y_{i+1} = y_i + f(x_i, y_i)h + \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\right) \tag{34}$$

The goal now is to find values for a_1, a_2, p_1 and q_{11} that will make Eq. 31 and Eq. 34 equivalent. Use a Taylor series to expand Eq. 28,

$$f(x_i + p_1h, y_i + q_{11}k_1h) = f(x_i, y_i) + p_1h\frac{\partial f}{\partial x} + q_{11}k_1h\frac{\partial f}{\partial y} + O(h^2) \tag{35}$$

The result along with Eq. 27 may be substituted back into Eq. 31,

$$y_{i+1} = y_i + a_1hf(x_i, y_i) + a_2hf(x_i, y_i) + a_2p_1h^2\frac{\partial f}{\partial x} + a_2q_{11}h^2f(x_i, y_i)\frac{\partial f}{\partial y} + O(h^3) \tag{36}$$

Collect the terms,

$$y_{i+1} = y_i + [a_1f(x_i, y_i) + a_2f(x_i, y_i)]h + \left[a_2p_1\frac{\partial f}{\partial x} + a_2q_{11}f(x_i, y_i)\frac{\partial f}{\partial y}\right]h^2 + O(h^3) \tag{37}$$

For Eq. 37 to be equivalent to Eq. 36, The following must be true: $a_1 + a_2 = 1$, $a_2p_1 = \frac{1}{2}$ and $a_2q_{11} = \frac{1}{2}$. Expanding the Runge-kutta method for the second order case results in three equations with four unknowns. Therefore, a value must be assumed for one of the unknowns which means there is an infinite number of second-order Runge-Kutta methods. The solution will be in the same family of equations and will yield the same results if the solution to the ODE is quadratic or linear. There will be a different result if the ODE is higher order. Therefore, higher order methods are required.

The fourth-order Runge-Kutta method solves the main issue of the explicit Euler method in that it takes multiple estimates of the slope to come up with a greatly improved average slope over the interval in question. The fourth-order Runge-Kutta method is also the more data efficient method

compared to the Explicit Euler method and is therefore more practical and efficient. The following is the final form of the fourth-order Runge-Kutta and the constants k.

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \tag{38}$$

$$k_1 = f(x_i, y_i) \tag{39}$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right) \tag{40}$$

$$k_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h\right) \tag{41}$$

$$k_4 = f(x_i + h, y_i + k_3 h) \tag{42}$$

Both Explicit Euler and 4th Order Runge-Kutta methods stated above were implemented using Python in order to solve the Chandrasekhar white dwarf equation. The results are discussed in the following section.

5 Results and Discussion

After implementing the white dwarf equations for the non-relativistic and relativistic cases as well as the numerical methods for each case, the plots of the numerical results were found. This can be seen in Fig. 1 where the 4th Order RK and Euler methods are given. In Figs. 1, 2 the step value (called *tau* in the code) is at 0.0001. With this step value the RK4 and Euler methods give the same results with the curves overlapping each other. From testing, it was found that step values of greater than 0.01 for the relativistic case and 0.15 for the non-relativistic case led to the curves diverging, highlighting the unstable nature of numerical methods in general. This is due to the greater complexity in the relativistic case with the Lorentz factor term, $\gamma(q)$, being included. It should be noted that the due to it's nuclear stability, the heaviest plausible white dwarf would consist of an iron core. On the other hand, carbon (or equivalent systems such as helium-oxygen) would represent the least dense Fermi gas possible for a white dwarf [3, 4]. Therefore, by plotting the solutions for both a pure iron core and a pure carbon core we can create bounds in which stellar observations are most likely to occur.

The results of the non-relativistic and relativistic equations vary quite a lot as the mass increases. At low solar masses up to $\sim 0.6 M_\odot$ the curves give the same results but at larger solar masses the relativistic case follows an asymptotic trajectory toward the Chandrasekhar limit. The limit shown on the curves is that of an average white dwarf composition (i.e. carbon or oxygen) while more dense white dwarfs (up to iron) will have a smaller Chandrasekhar limit [3].

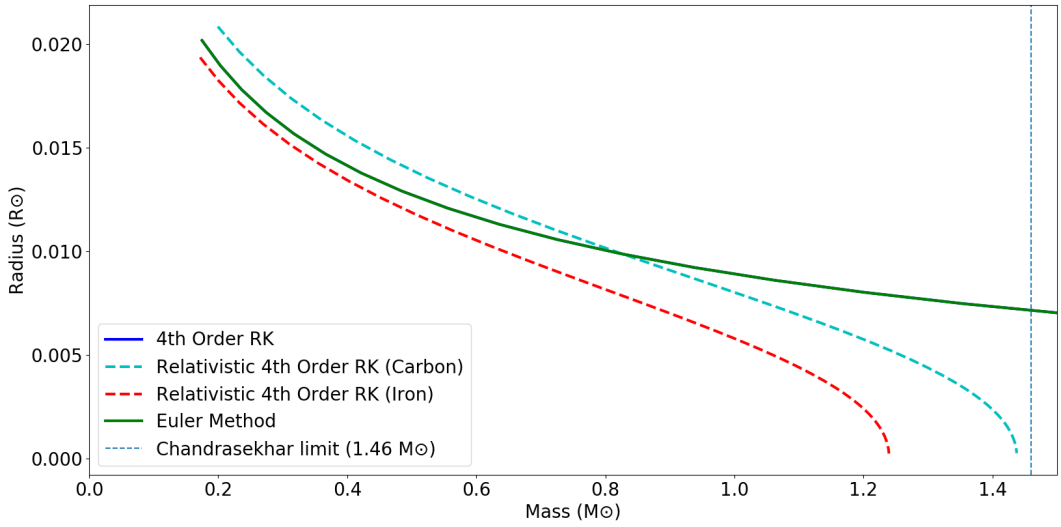

Figure 1: Numerical solutions of the non-relativistic and relativistic white dwarf equations using both the *4th Order Runge-Kutta* (RK4) and *Explicit Euler* methods. Note how the chemical composition of the core (carbon or iron) changes the curves and also the Chandrasekhar limit.

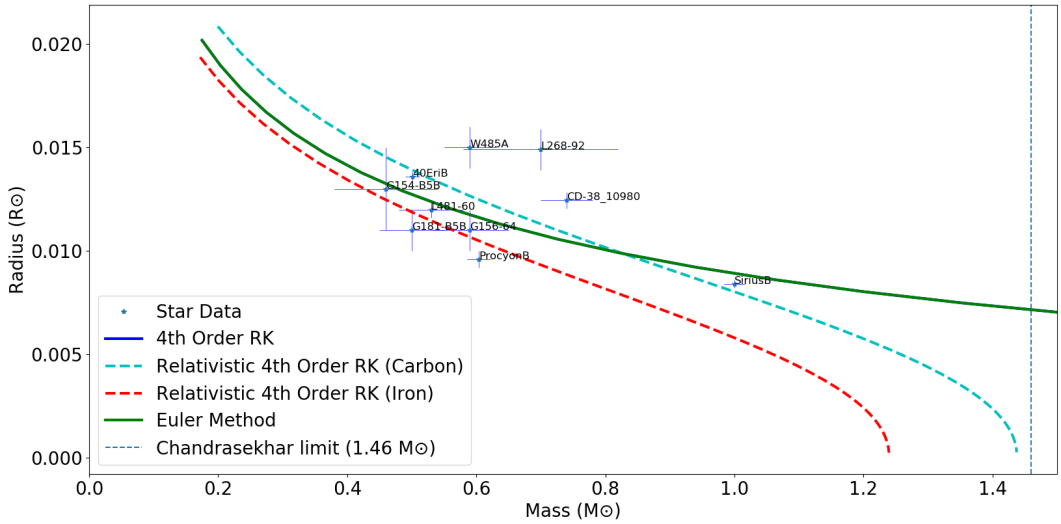

Figure 2: Numerical simulations of the white dwarf equations with included real white dwarf data. Star data is extracted from *Provencal et al. (1997)* [1].

From the results shown in Fig. 2 the numerical simulations closely match those of the real astronomy data from *Provencal et al. (1997)* [1]. It can be seen that most of the star data fits between the carbon and iron curve bounds nicely with only the stars *W485A*, *L268-92*, and *CD-38 10980*

straying outside the curves. This error can result from multiple sources such as the errors in the astronomical data (see error bars in Fig. 2) and differences in the star composition from what was selected in the numerical simulations. Additionally, the values of ρ_0 and R_0 in Eqs. 7 and 8 are that of average white dwarfs. The white dwarfs outside the curves are outliers and so these average values are likely not applicable. Overall the numerical simulations very accurately model the given ODE's for white dwarfs and that closely matches the real world observations.

6 Conclusion

In this project, the governing models of white dwarf stars, the non-relativistic and relativistic cases, developed by *Subrahmanyan Chandrasekhar* were numerically modelled using two different techniques being the *Explicit Euler* and *4th Order Runge-Kutta*. The models in question are 1st order *Ordinary Differential Equations* (ODE's) and they were successfully implemented using the Python programming language by iterating through successive central density (ρ) values until an equilibrium mass and radius were found. This was done both for white dwarfs composed of carbon and iron, thus forming a lower and upper bound for the white dwarf composition. The results of these simulations correctly predicted the Chandrasekhar limit for white dwarf masses being approximately 1.46 solar masses ($1.46 M_\odot$) [4]. These numerical results were then compared to real astronomy data (Fig. 2) and it was found qualitatively that the white dwarf equations fall within the astronomical data's margin of error. These results validate the accuracy of the explicit Euler and 4th Order Runge-Kutta methods as well as Chandrasekhar's white dwarf equations.

7 References

[1] J. L. Provencal, H. L. Shipman, E. Høg, and P. Thejll, "Testing the white dwarf mass-radius relation with hipparcos," *European Space Agency, (Special Publication) ESA SP*, vol. 494, pp. 375–376, feb 1997.

[2] S. C. Chapra and R. P. Canale, *Numerical methods for engineers*, vol. 1221. Mcgraw-hill New York, 2011.

[3] N. K. Glendenning, *Special and general relativity: with applications to white dwarfs, neutron stars and black holes*. Springer Science & Business Media, 2010.

[4] O. R. Pols, *STELLAR STRUCTURE AND EVOLUTION*. Astronomical Institute Utrecht, sep 2011.

[5] F. Schwabl, *Statistical Mechanics*. Springer, 2nd ed., 2013.

[6] H. Davis, *Introduction to nonlinear differential and integral equations*. New York: Dover Publications, new dover ed., 1962.

Appendix

Python Code

Note: Most comments are not shown here to clear up space on pdf document. Also, the solar symbol unicode cannot be rendered by the python to latex package we are using, so those show up as blank spaces in the code below.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

def func_eval(funcName, *args):
    return eval(funcName)(*args)


def Euler(function, y, t, tau):

    slope = function(y, t)

    y_new = y +tau*slope

    return(y_new)

def RK4(function, y, t, tau):

    k1 = function(y, t)

    k2 = function(y + (tau/2)*k1, t+tau/2)

    k3 = function(y+(tau/2)*k2, t+tau/2)

    k4 = function(y+(tau/2)*k3, t+tau/2)

    y_new = y + (tau/6)*(k1+(2*k2)+(2*k3)+k4)

    return(y_new)


#Constants
mu_e_c = 12/6
mu_e_fe = 56/26

p_slash_c = np.geomspace(0.1, 2.5e6, num=100)
p_init = p_slash_c
p_0_c = 9.74e5*mu_e_c
p_0_fe = 9.74e5*mu_e_fe
p_Eu = np.zeros((len(p_slash_c)))
p_RK4 = np.zeros((len(p_slash_c)))
p_Rel_RK4 = np.zeros((len(p_slash_c)))
p_AB = np.zeros((len(p_slash_c)))

tau = 0.0001

```

```
R_0_c = 7.72e8/mu_e_c
R_0_fe = 7.72e8/mu_e_fe
R_sun = 69600000000.0
r = np.arange(tau, 4+tau,tau)

m_init_Eu = 0.0
m_init_RK4 = 0.0
m_init_Rel_RK4 = 0.0
m_init_AB = 0.0

M_0_c = 5.67e33/(mu_e_c**2)
M_0_fe = 5.67e33/(mu_e_fe**2)
M_sun = 1.989e+33
M_limit = 1.46


def system(y,r):

    p = y[0]
    m = y[1]

    dpdr = -m*p/(p**(2/3)*r**2/(3*(1+p**(1/3))**(0.5)))
    dmdr = r**2*p
    return np.array([dpdr, dmdr])


def gamma(y):

    gamma = y**(2/3)/(3*(1+y**(2/3))**0.5)

    return gamma

def RelCase(y,r):

    p = y[0]
    m = y[1]

    dpdr = -m*p/(r**2*(gamma(p)))
    dmdr = r**2*p

    return np.array([dpdr, dmdr])


radius_RK4 = []
mass_RK4 = []


#RK4 METHOD-----------------------------------------------------

#NON-RELATIVISTIC-----------------------------------------------
for p_RK4 in np.geomspace(0.1, 2.5e6, num=50):
    y = np.array([p_RK4, m_init_RK4])

    for i in range(len(r)):
```

```
        y_prev = y
        y = RK4(system,y,r[i],tau)

        if np.isnan(y[0]) == True:
            radius_RK4.append(r[i-1])
            mass_RK4.append(y_prev[1])
            break

radius_solar_RK4 = np.array(radius_RK4)*R_0_c/R_sun
mass_solar_RK4 = np.array(mass_RK4)*M_0_c/M_sun


#RELATIVISTIC-----------------------------------------------------
radius_Rel_RK4 = []
mass_Rel_RK4 = []

for p_Rel_RK4 in np.geomspace(0.1, 2.5e6, num=50):
    y = np.array([p_Rel_RK4, m_init_Rel_RK4])

    for i in range(len(r)):
        y_prev = y
        y = RK4(RelCase,y,r[i],tau)

        if np.isnan(y[0]) == True:
            radius_Rel_RK4.append(r[i-1])
            mass_Rel_RK4.append(y_prev[1])
            break

#Carbon Core
radius_solar_Rel_RK4 = np.array(radius_Rel_RK4)*R_0_c/R_sun
mass_solar_Rel_RK4 = np.array(mass_Rel_RK4)*M_0_c/M_sun

#Iron Core
radius_solar_Rel_fe_RK4 = np.array(radius_Rel_RK4)*R_0_fe/R_sun
mass_solar_Rel_fe_RK4 = np.array(mass_Rel_RK4)*M_0_fe/M_sun



#EULER METHOD-----------------------------------------------------

radius_Eu = []
mass_Eu = []

for p_Eu in np.geomspace(0.1, 2.5e6, num=50):
    y = np.array([p_Eu, m_init_Eu])

    for i in range(len(r)):
        y_prev = y
        y = Euler(system,y,r[i],tau)

        if np.isnan(y[0]) == True:
            radius_Eu.append(r[i-1])
            mass_Eu.append(y_prev[1])
            break
```

```
155
156 radius_solar_Eu = np.array(radius_Eu)*R_0_c/R_sun
157 mass_solar_Eu = np.array(mass_Eu)*M_0_c/M_sun
158
159
160 plt.rcParams.update({'font.size': 20})
161 #Plots mass vs radius----------------------------------------
162 plt.plot(mass_solar_RK4,radius_solar_RK4, 'b',linewidth=3, label="4th Order RK
    ")
163 plt.plot(mass_solar_Rel_RK4,radius_solar_Rel_RK4,'c',linestyle='--',linewidth
    =3 ,label="Relativistic 4th Order RK (Carbon)")
164 plt.plot(mass_solar_Rel_fe_RK4,radius_solar_Rel_fe_RK4,'r',linestyle='--',
    linewidth=3, label="Relativistic 4th Order RK (Iron)")
165 plt.plot(mass_solar_Eu,radius_solar_Eu, 'g',linewidth=3, label="Euler Method")
166 plt.axvline(x=M_limit ,linestyle='--',label="Chandrasekhar limit (1.46  M  )")
167 plt.legend()
168 plt.ylabel('Radius ( R  )')
169 plt.xlabel('Mass ( M  )')
170 plt.xlim([0, 1.5])
171 #plt.title("White Dwarf Radii vs Mass")
172 plt.show()
173
174 #Plotting vs example stars-----------------------------------
175 stars = np.loadtxt('wd_data.txt',usecols=(1,2,3,4))
176 star_names = np.genfromtxt('wd_data.txt', dtype=str, usecols=(0)).tolist()
177
178 plt.plot(stars[:,0],stars[:,2], '*', label="Star Data")
179 plt.errorbar(x=stars[:,0],y=stars[:,2],yerr=stars[:,3],xerr=stars[:,1], fmt='
    ',color='b', elinewidth=0.5)
180 for i in range(len(star_names)): plt.annotate(star_names[i],xy=(stars[i,0],
    stars[i,2]),fontsize=12)
181 plt.plot(mass_solar_RK4,radius_solar_RK4, 'b',linewidth=3, label="4th Order RK
    ")
182 plt.plot(mass_solar_Rel_RK4,radius_solar_Rel_RK4,'c',linestyle='--',linewidth
    =3, label="Relativistic 4th Order RK (Carbon)")
183 plt.plot(mass_solar_Rel_fe_RK4,radius_solar_Rel_fe_RK4,'r',linestyle='--',
    linewidth=3, label="Relativistic 4th Order RK (Iron)")
184
185 plt.plot(mass_solar_Eu,radius_solar_Eu, 'g',linewidth=3, label="Euler Method")
186 plt.axvline(x=M_limit ,linestyle='--',label="Chandrasekhar limit (1.46  M  )")
187 plt.legend()
188 plt.ylabel('Radius ( R  )')
189 plt.xlabel('Mass ( M  )')
190 plt.xlim([0, 1.5])
191 #plt.title("White Dwarf Radii vs Mass")
192 plt.show()
```