SEAN HUGGINS PHYSICS 3926 PROJECT 2

Table of Contents

1. Introduction

The purpose of this project is to estimate the Chandrasekhar limiting mass of White Dwarf stars. White dwarfs are packed extremely tightly, so free electrons in the interior of a White Dwarf exert a strong degeneracy pressure which arises from the Pauli Exclusion Principle. Stars with cores that have a mass greater than this limit will have more self gravity than outward degeneracy pressure, and their lives will end in a supernova explosion.

This mass limit will be estimated by examining the mass-density relation of several White Dwarf stars with different central densities, and by integrating over the whole radius of each star until its density drops to zero. When we plot radii versus masses obtained from this integration we will be able to estimate the Chandrasekhar limit from our plot.

Numerical integration will be performed using our **RK2Integrator()** function as well as with Matlab's built-in **ODE45()** ODE solver.

Our Chandrasekhar limit will be compared with the limit estimated by Kippenhahn & Weigert (1990). We will also overlay several White Dwarf masses and radii obtained experimentally by Provencal et al. (1998) with their error bars to make sure that our mass-radius relation is consistent.

2. The RK2Step() Function

2.1 Description

The first necessary function, dubbed **RK2Step()** is used to numerically integrate one constant tau step forward using the second order Runge Kutta scheme.

RK2Step() takes the name of a function pertaining to a problem we wish to solve, an independent variable, a constant independent variable step, and state variables.

It then calls the function pertaining to our problem in a manner outlined by the RK2 scheme, and returns the updated state variables at a forward step.

It was desirable to separate a single RK step from a full fledged integration. The next function will deal with the whole Runge Kutta integration.

3. The RK2Integrator() Function

3.1 Description

RK2Integrator() utilizes **RKStep()** to carry out a full integration.

It takes name of the function pertaining to a problem we wish to solve, an independent variable span (e.g $0 < t < 100$s), a constant independent variable step, and boundary/initial conditions for the state variables.

RK2Integrator() then calls **RKStep()** over the span of the independent variable divided by the step size, and returns the state variables vs. the independent variable for the entire integration.

RK2Integrator() in combination with **RKStep()** can now be used to solve any system of coupled first order differential equations we wish. We will investigate this further.

3.2 Testing RK2Integrator()

We want to make sure that **RK2Integrator()** works how we would like it to before we go on to estimating our Chandrasekhar Limit. Let's test it using some simple but non-trivial examples.

3.2.1 ballrk()

ballrk() returns dv/dt and dx/dt for the motion of a ball through the air with some air resistance. Let's use **RK2Integrator()** to plot the trajectory of a ball using these ODEs.

```
%Initialize ball position vector
r = [0; 1];

%Initialize the ball
ballSpeed = 50;

%Initialize ball velocity vector
v = [ballSpeed*cos(deg2rad(45)); ballSpeed*sin(deg2rad(45))];

%State is of the form [vx; vy; rx; ry]
[time, state] = RK2Integrator('ballrk', [0 5], 0.1, [v; r]);

%Plot the ball trajectory
plot(state(:,3), state(:,4), 'b.');
xlabel('x (m)');
ylabel('y (m)');
title('Trajectory of a Baseball with Air Resistance');
```

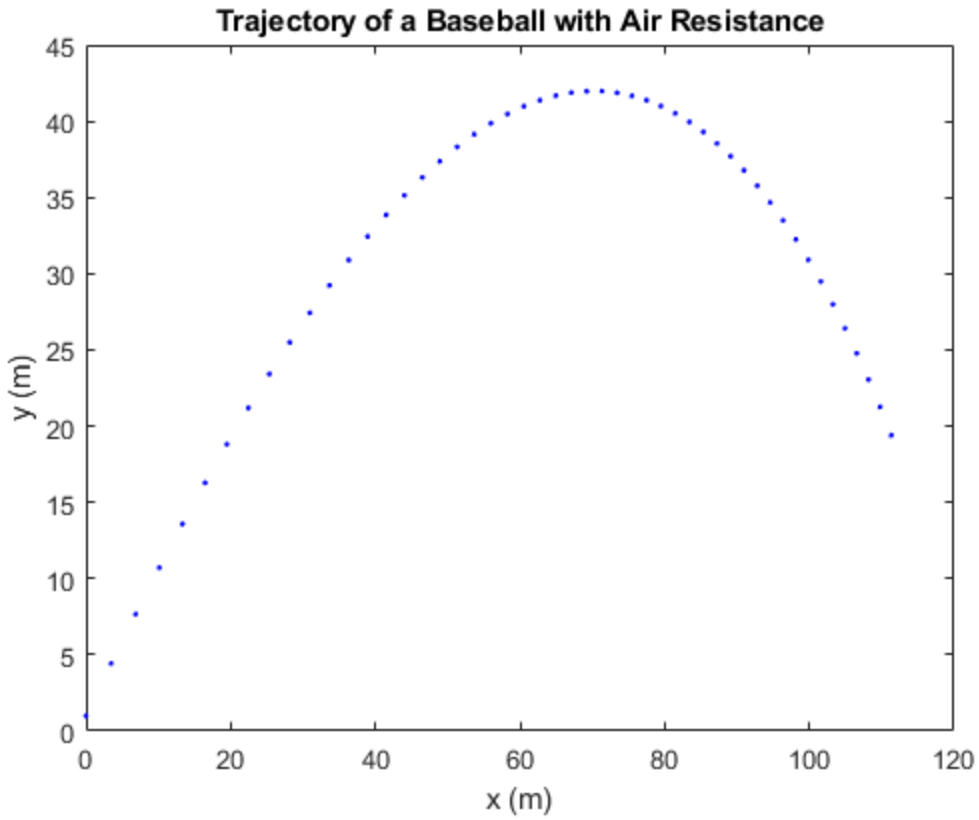

Figure 1: ballrk() integrated using **RK2Integrator()** for an initial height of 1 m, initial speed of 50 m/s, angle of 45 degrees, and a time step of 0.1s. Constants pertaining to the properties of the ball, and air resistance are specified within **ballrk()**. Time spans from 0 to 5 seconds.

This looks great! We have recreated Figure 2.3 from p45 in the textbook, except this time using our RK2 integrator instead of the Euler Method. Let's try one more example just to be safe.

3.2.2 springrk()

springrk() returns dv/dt and dx/dt for a damped harmonic oscillator. Let's use **RK2Integrator()** to retrieve and plot the position of the oscillator vs. time for these ODEs.

```
%Initialize the position of the oscliator
x = 2;

%Initialize the speed
v = 3;

%State is of the form [v; x]
[time, state] = RK2Integrator('springrk', [0 100], 0.01, [v; x]);

%Plot the spring oscillations with time
plot(time, state(:,2), 'b');
xlabel('t (s)');
ylabel('x (m)');
title('Damped Simple Harmonic Oscillator');
```

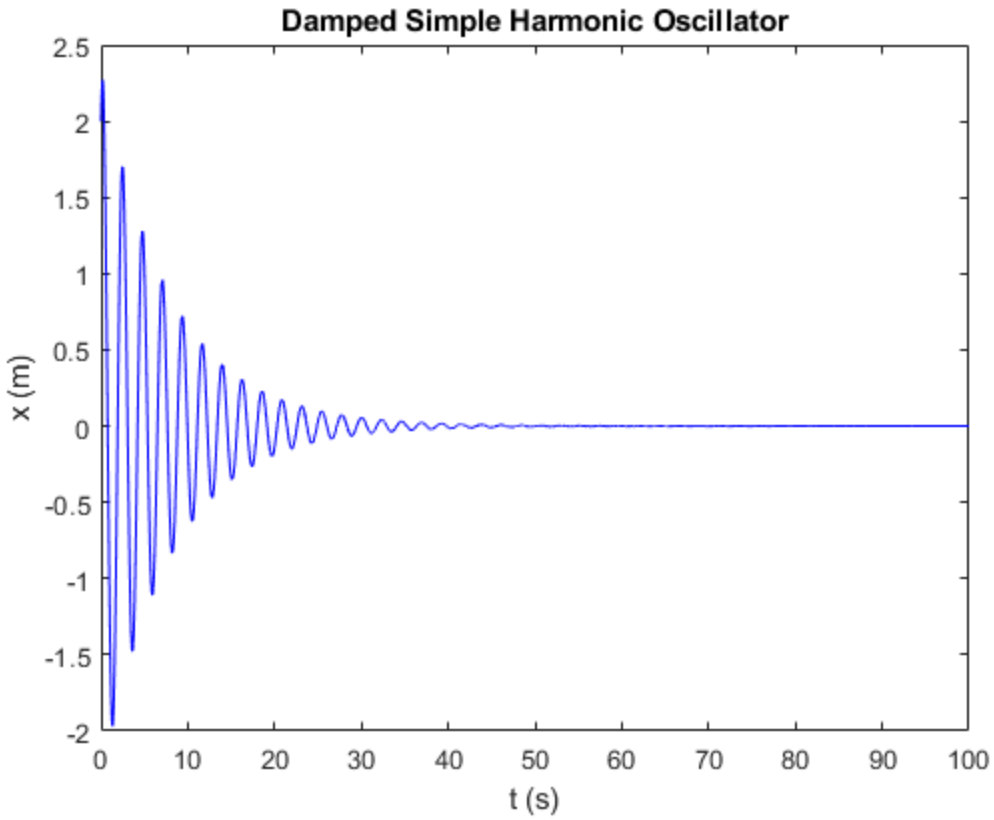

Figure 2: springrk() integrated using **RK2Integrator()** for an initial position of 1 m, initial speed of 3 m/s, and a time step of 0.01s. Constants such as a spring constant, a mass, and a damping coefficient are specified within **springrk()**. Time spans from 0 to 100 seconds.

Another successful test! This is exactly what we would expect for the motion of a damped harmonic oscillator with time. Our **RK2Integrator()** is definitely ready to be used in our analysis of the Chandrasekhar limiting mass of a White Dwarf. Let's get going!

4. White Dwarf Masses and Radii

4.1 dwarfRK()

dwarfRK() returns the rates of change of White Dwarf mass and density with respect to radius, calculated using scaled variables. **RKStep()** numerically integrates these ODEs one step forward, and **RKIntegrator()** calls **RKStep()** n times, where n is specified by the given independent variable span and the independent variable step.

4.2 Numerically Integrating dwarfRK() to find masses and radii of White Dwarfs

Let's get to the root of our problem. We first need to setup some constants and boundary conditions.

```
%Set constants for the problem we're trying to solve
ue = 2;    %number of nucleons per electron
```

```
Msun = 1.989e33; %g
Rsun = 6.955e10; %cm

%Set boundary conditions for state variables
centralMass = 0;
%Create a logspace array of White Dwarf central densities to consider
centralDensity = logspace(-1,7, 100);
%Remove densities greater than 2.5x10^6
centralDensity(centralDensity>2.5e6) = [];
%Add back the max central density
centralDensity = [centralDensity, 2.5e6];

%choose a constant radius step
radStep = 0.0005;

%Initialize arrays for white dwarf masses and radii
RKM = zeros([length(centralDensity) 1]);
RKR = zeros([length(centralDensity) 1]);
ODEM = zeros([length(centralDensity) 1]);
ODER = zeros([length(centralDensity) 1]);
```

Now we consider White Dwarf stars with central densities ranging from 10^-1 to 2.5x10^6 and divided up logarithmically. We are solving for their masses and radii by numerically integrating from the centre of the star to where its density falls to zero. This will be the outer radius of the star, and the mass at this radius will be the star's total mass.

For all the central densities, we retrieve all the masses and radii using both **RK2Integrator()** and **ODE45()**.

```
for j = 1:length(centralDensity)
    %Use our RK2 integrator to retrieve R vs. M in the scaled
 variables for a White Dwarf with a particular central density
    [scaledRadius, scaledState] = RK2Integrator('dwarfrk', [eps,4],
 radStep, [centralMass; centralDensity(j)]);

    %Remove imaginary parts from the scaled state variables
    scaledState = real(scaledState);

    %Remove negatives from the state variables
    negatives = find((scaledState(:,1)<0)|(scaledState(:,2)<0));

    %Anything value beyond the first negative mass encountered is non-
physical behaior, so we dispose of it
    %Leave 1 negative value for the interpolation
    scaledRadius(negatives(2):end,:) = [];
    scaledState(negatives(2):end,:) = [];

    %Interpolate to find the mass and radius of the White Dwarf
    [x, index] = unique(scaledState(:,2));
    RKscaledM = interp1(x, scaledState(index,1), 0, 'pchip');
    RKscaledR = interp1(x, scaledRadius(index,1), 0, 'pchip');

    %Unscale the radius and store the value
    RKM(j) = ((5.67*(10^33))/(ue^2))*RKscaledM;
    %Unscale the mass and store the value
    RKR(j) = ((7.72*(10^8))/(ue))*RKscaledR;
```

```
   %We now have the mass and radius of our star at this central
density

   %Lets do it again, just now with ode45

   %We chose a relative error tolerance 'RelTol' of 10^-4, which is
better than the default 10^-3
   [scaledRadius, scaledState] = ode45(@dwarfrk,[eps,4],[centralMass;
centralDensity(j)],odeset('RelTol', 1e-4));

   scaledState = real(scaledState);

   negatives = find((scaledState(:,1)<0)|(scaledState(:,2)<0));
   scaledRadius(negatives(2):end,:) = [];
   scaledState(negatives(2):end,:) = [];

   [x, index] = unique(scaledState(:,2));
   ODEscaledM = interp1(x, scaledState(index,1), 0, 'pchip');
   ODEscaledR = interp1(x, scaledRadius(index,1), 0, 'pchip');


   ODEM(j) = ((5.67*(10^33))/(ue^2))*ODEscaledM;

   ODER(j) = ((7.72*(10^8))/(ue))*ODEscaledR;

   %We have now found the mass and radius of our White Dwarf star
using
   %our RK2 integrator and ODE45
end
hold on
```

We have acquired masses and radii for White Dwarf stars with central densities ranging from 1e-1 to 2.5e6. We can now get some results.

5. Results and Analysis of the Chandrasekhar Limit

5.1 Results from our plot

We can plot radius on the Y-axis and mass on the X-axis to show the relationship between radius and mass for our White Dwarf stars. Let us plot both the results from RK2, as well as those from ODE45.

```
%Plot our results from the RK2 Integrator
p1 = plot(RKM,RKR, 'b.');

%Plot the results found using ODE45
p2 = plot(ODEM,ODER, 'r.');

%Plot the estimated Chandrasekhar limiting masses found through RK2
 and
```

```
%through ODE45
climitRK = plot([RKM(end) RKM(end)], [RKR(end) RKR(end)], 'ko');
climitODE = plot([ODEM(end) ODEM(end)], [ODER(end) ODER(end)], 'go');


%Label the axes
xlabel('White Dwarf Mass (g)');
ylabel('White Dwarf Radius (cm)');

%Title the plot
title('Theoretical Radius vs. Mass for White Dwarf Stars');

%Set the y limit
ylim([0 RKR(1)]);

%Masses and radii from the table in Q5
masses = [0.604 1.000 0.501 0.74 0.59 0.70 0.53 0.46 0.50 0.59].*Msun;
radii = [0.0096 0.0084 0.0136 0.01245 0.0150 0.0149 0.0120 0.0130
 0.0110 0.0110].*Rsun;
merror = [0.018 0.016 0.011 0.04 0.04 0.12 0.05 0.08 0.05 0.06].*Msun;
rerror =[0.0004 0.0002 0.0002 0.0004 0.001 0.001 0.0004 0.002 0.001
 0.001].*Rsun;

%Plot error bars
ebars = errorbar(masses,radii,-rerror, rerror, -merror,
 merror, 'b','LineStyle','none');

%Make a legend
legend([p1 p2 ebars climitRK climitODE],'RK2','ODE45',...
    'Provencal et al. (1998) experimental masses & radii', ...
    sprintf('RK2 Chandrasekhar limiting mass: %1.3e g', RKM(end)),...
    sprintf('ODE45 Chandrasekhar limiting mass: %1.3e g', ODEM(end)),
  'Location', 'southwest');
```

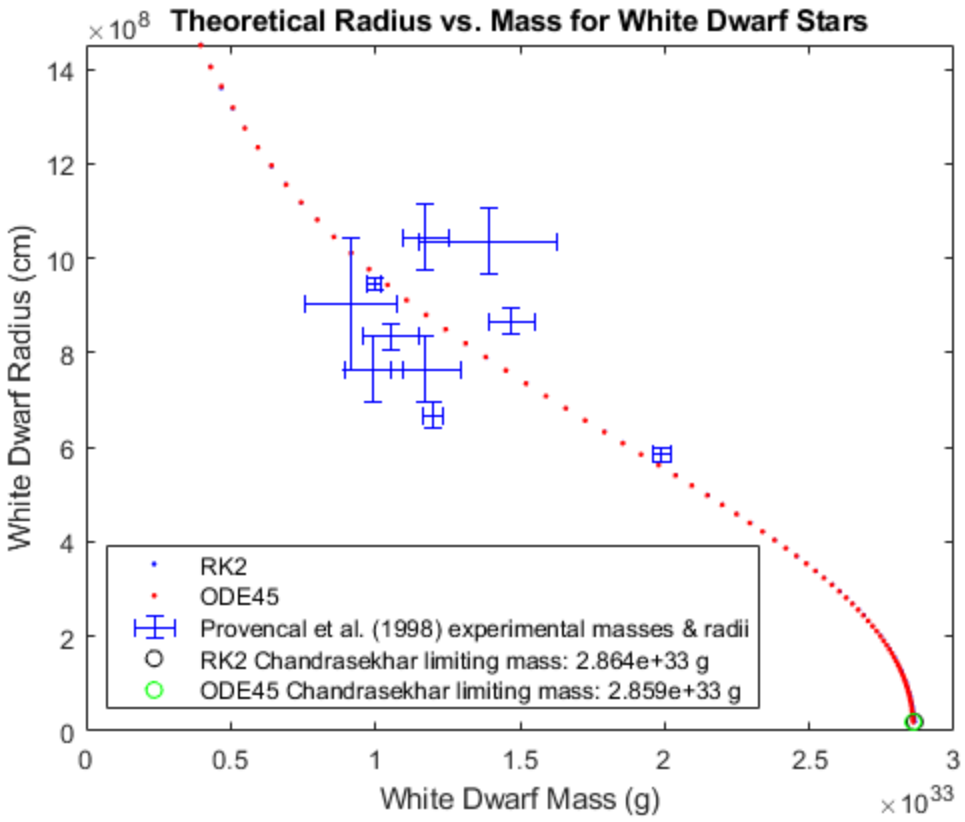

Figure 3: Theoretical radius vs. mass for White Dwarf Stars for central densities ranging from 1e-1 to 2.5e6. Also shown are experimental measurements with errors for 10 known White Dwarf stars. The estimated Chandrasekhar limiting mass found by RK2 is represented by the black circle and the limit found by ODE45 is represented by the green circle. The values are displayed in the legend

The results from RK2 line up very nicely with the results from ODE45. The discrepancy between the two results becomes more apparent closer to the Chandrasekhar limit.

The difference between the ODE45 result and the RK2 result is resolved when we choose a smaller integration step. A percent difference calculation comparing the RK2 & ODE45 limits is done below:

```
%Display the percent difference
fprintf('Percent difference between our RK2 and ODE45 result = %%
%.3f', ...
    (abs(RKM(end)-ODEM(end))/((RKM(end)+ODEM(end))/2))*100    );
```

```
Percent difference between our RK2 and ODE45 result = %0.168
```

5.2 Analysis

5.2.1 Comparison with Kippenhahn & Weigert

Kippenhahn & Weigert (1990) cite a value of 2.9020e33 g for the Chandrasekhar limiting mass. Looking at our estimates, they agree nicely with this figure. A percent diffence calculation is done below:

```
%Kippenham & Weigert Result
```

```
KWM = 2.9020e33;

%Display the percent difference
fprintf('Percent difference between our RK2 limit and the Kippenham &
 Weigert result = %%%.3f', ...
    (abs(RKM(end)-KWM)/((RKM(end)+KWM)/2))*100    );
```

```
Percent difference between our RK2 limit and the Kippenham & Weigert
 result = %1.322
```

5.2.2 Comparisons with Provencal et al.

White Dwarf masses and radii obtained from Provencal et al. (1998) are shown on Figure 3 with error bars. For the most part, the values and their experimental errors fall within our computed mass-radius relation.

6. Conclusion

In conclusion, using our extensively tested second order Runge Kutta integrator **RK2Integrator()**, as well as Matlab's **ODE45()** with a relative error tolerance of 1e-4, we were able to numerically integrate over the radius of various White Dwarf stars with different central densities. Doing this, we obtained a mass-radius relation which allowed us to estimate the Chandrasekhar limiting mass for White Dwarf stars.

It was found that the profiles obtained from **ODE45()** and **RK2Integrator()** were in good agreement with each other, and were in better agreement when a smaller integration time step was used for RK2.

Our result for the limit agreed nicely with the result obtained from Kippenhahn & Weigert (1990). Several White Dwarf experimental masses and radii obtained from Provencal et al. (1998) were overlayed on our plot. These values with their errors were in good agreement with our mass-radius relation.

Published with MATLAB® R2018b