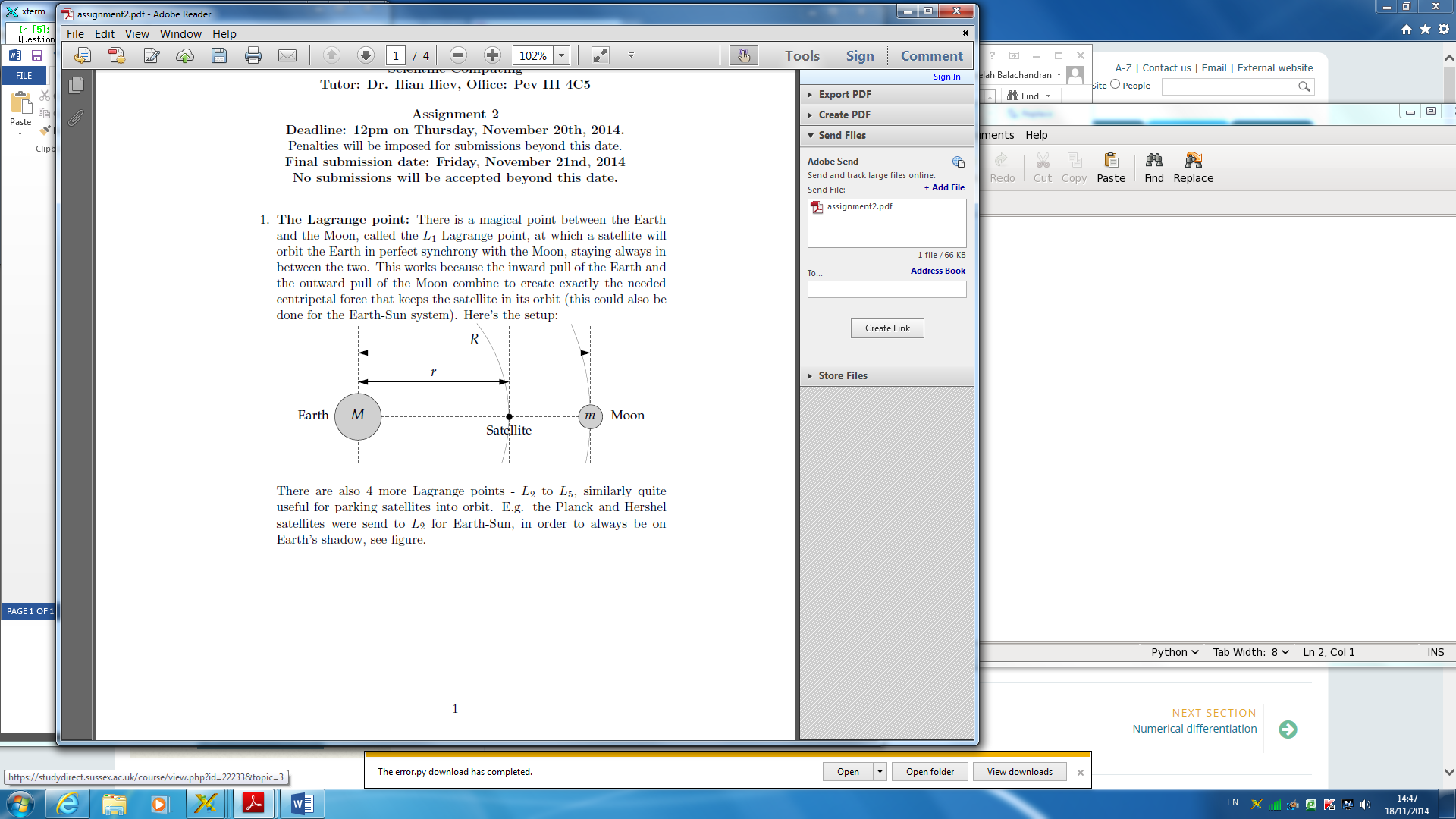
Scientific Computing

Python Assessment 2



1a)

**Using Newton’s Law of Gravitation to resolve forces**

Newton’s law f Gravitation is as follows:

Where F is the force between the masses, G is the Gravitational constant, M and m are respective masses of two separate bodies and r is the distance from the centre of each between them.

We form the following by resolving forces:

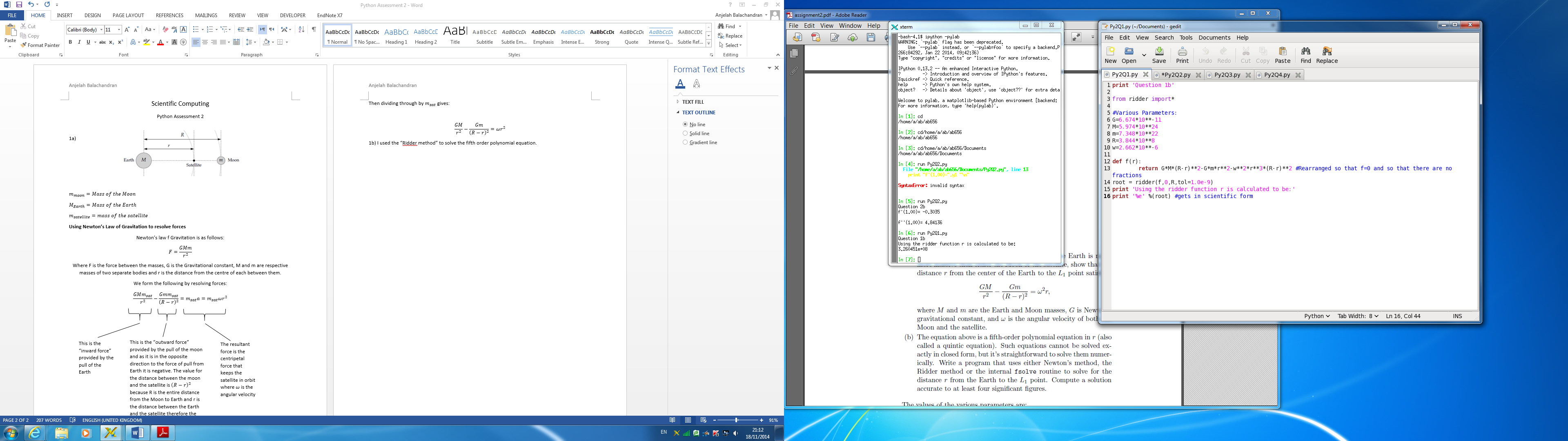
This is the “inward force” provided by the pull of the Earth

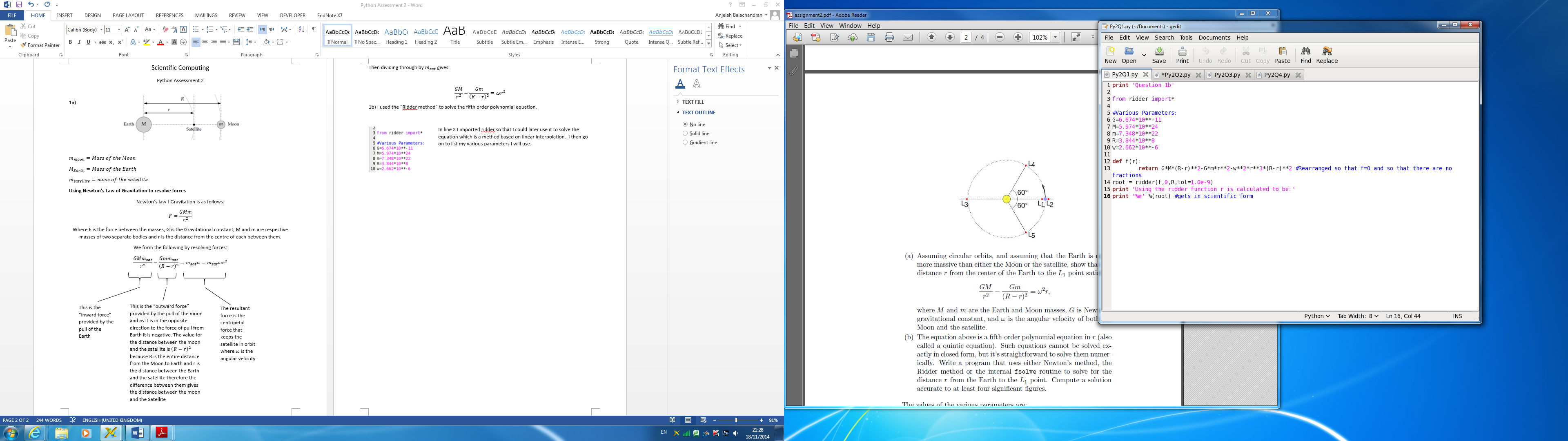
This is the “outward force” provided by the pull of the moon and as it is in the opposite direction to the force of pull from Earth it is negative. The value for the distance between the moon and the satellite is because R is the entire distance from the Moon to Earth and r is the distance between the Earth and the satellite therefore the difference between them gives the distance between the moon and the Satellite

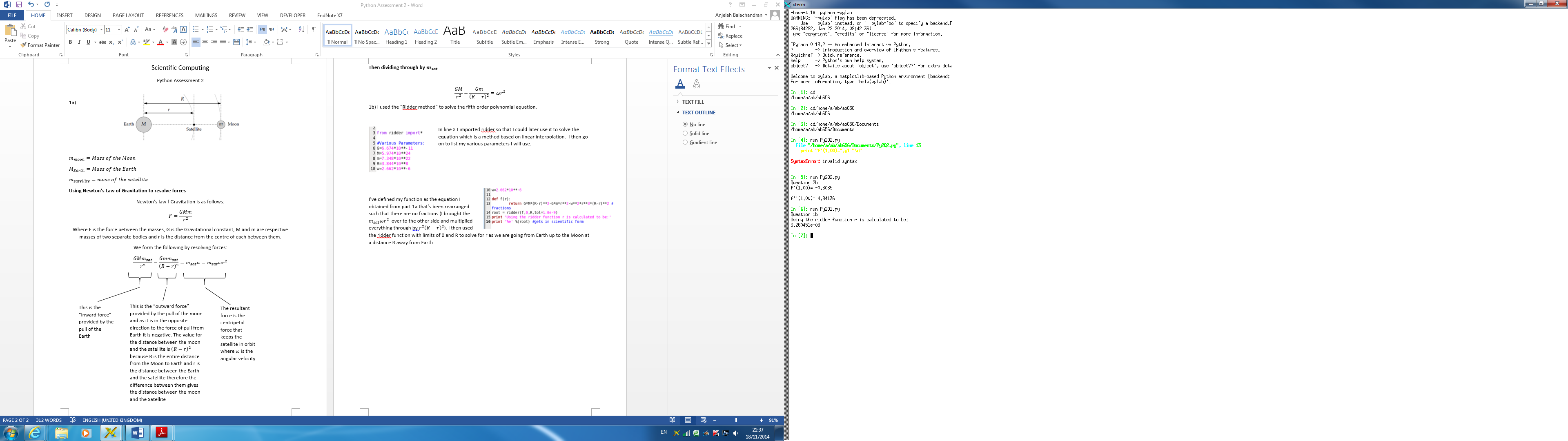
The resultant force is the centripetal force that keeps the satellite in orbit where is the angular velocity

**Then dividing through by**

1b) **Code attached** -I used the “Ridder method” to solve the fifth order polynomial equation.

In line 3 I imported ridder so that I could later use it to solve the equation which is a method based on linear interpolation. I then go on to list my various parameters I will use.

I’ve defined my function as the equation I obtained from part 1a that’s been rearranged such that there are no fractions (I brought the over to the other side and multiplied everything through by ). I then used the ridder function with limits of 0 and R to solve for r as we are going from Earth up to the Moon at a distance R away from Earth. Below is the solution after it has been run:



Therefore the solution to five significant figures would be m and this is a rough approximation of the location of L1.

2a)

The taylor series approximations are as follows

(1)

(2)

Adding the two taylor approximations [ (1) + (2) ] gives

(3)

Subtracting the two taylor approximations [ (1) – (2) ] gives

(4)

I can rearrange equation (4) to give

Divide through by

+ truncation error (5)

The truncation error for is as follows

Error on (6)

**The error on is therefore 1st order as**

I can rearrange equation (3) to give

Divide through by

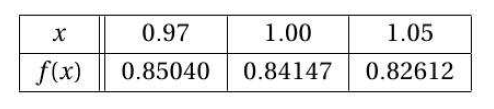
+ truncation error (7)

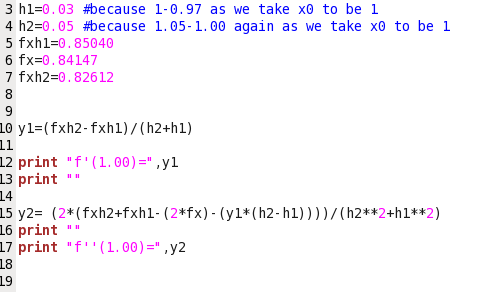
The truncation error for is as follows

Error on (8)

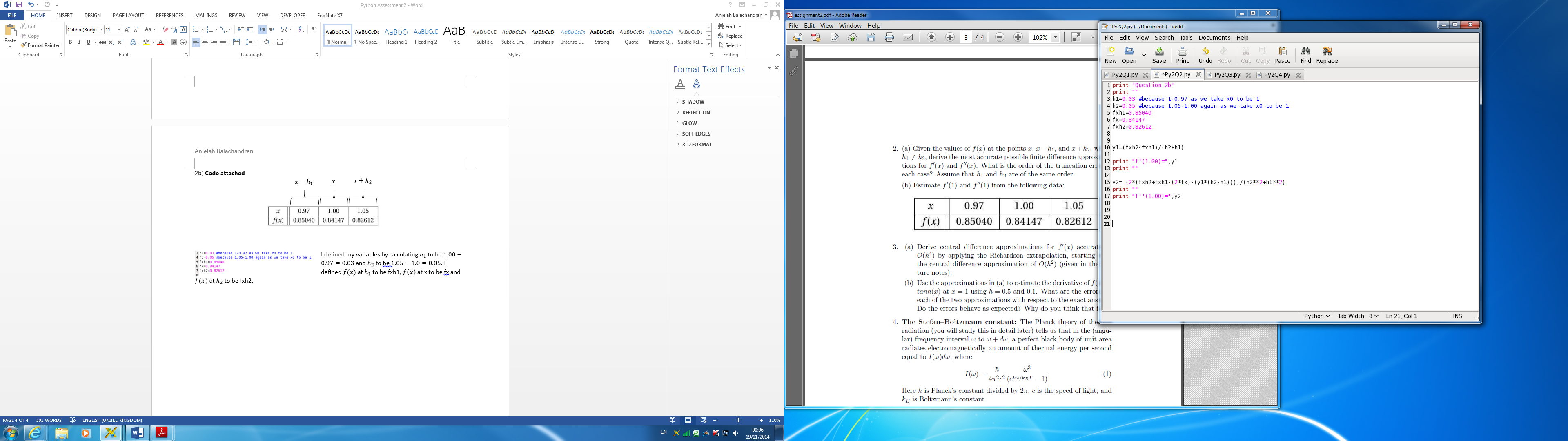
**The error on is therefore 1st order as**

2b) **Code attached**

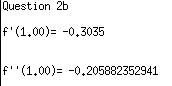




I defined my variables by calculating to be and to be . I defined at to be fxh1, at x to be fx and at to be fxh2.



I defined as y1 and as y2. I then went on to produce the solution for and i.e the variables defined earlier were substituted into the equations and an answer was generated. I used the solution for y1 and substituted it into y2 in order to ensure complete accuracy, as is in the y2 equation. The solutions obtained are as follows:



3a) Deriving the central difference approximations for accurate to O(h4). The centred approximation is as follows:

(1)

We want to compute some quantity G dependent on some parameter h

where is the approximation and is the error that takes the form , i.e

The Richardson extrapolation formula at h2= is as follows:

(2)

We are approximating O(h2) and so the power of h is 2, thus p is 2. Applying p=2 into equation (2) gives:

(3)

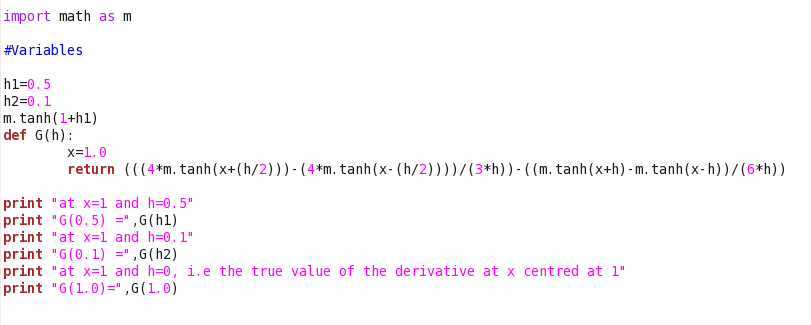
I will substitute (3) into (1) to give:

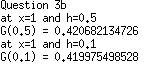
I will now put it in the form :

(4)

3b) **Code attached-**

Since we are approximating the derivative for tanh(x), I have substituted tanh(x) into (4) from part a



I imported math as m such that I can apply my tanh function. I defined a function G(h) that returned the equation I formed from part a. The function has 1 parameter h, which we choose suitably when we call the function (our values for h1 and h2). The values at h=0.5 and h=0.1 are as follows:



The true value of the first derivative of tanh(x) at 1 is

The error for when h=0.5

The error for when h=0.1

The errors do behave as expected. As 0.1<0.5, the width of the partition is smaller (and so there are more of them). As a result of having more (and smaller) partitions, the error (or over/underestimate) is less extreme, thus closer to the true value.

4a) A perfect black body of unit area radiates electromagnetic an amount of thermal energy per second where:

(1)

(2)

Therefore we have

(3)

I will perform an integration by substitution where I will let

Differentiating x with respect to gives

= rearranging this for gives

Substituting this into (3) gives

Since we can rearrange for as this is a substitution I will need to make

🡪

Now substituting this into our integral gives

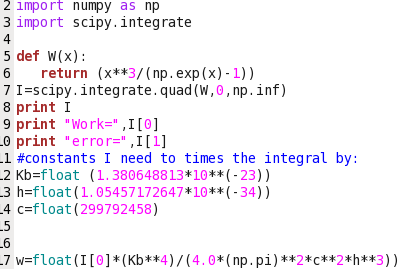
Since we can substitute into to give . Finally substituting this into our integral removes all terms and our substitution is complete:

(4)

Simplifying (4) gives

as required.

4b) Evaluating the integral from 0 to infinity



I imported the scipy.integrate package to compute my integral and defined my function as that returned the equation and used the scipy.integrate package to compute the integral between 0 and infinity shown by “0,np.inf”. This generated a solution that gave me the value for work (without constants) and the error on this value.



*The error is many magnitudes smaller than our value for work, therefore we can say that the approximation is accurate.*

As this only gave me an exact value for the integral without the constants, I went on to define the constants (lines 2,3,4) and then multiply this by my exact integrated solution. As scipy.integrate.quad outputs an array with two items, the first being the solution and the second being the error, my calculation was done using only the solution (the first item in the array) and thus I used I[0] multiplied by the constants:



The value for work was therefore calculated to be:

(When calculating the work done in Python, I didn’t reference the as it is a constant variable that will cancel when finding the final solution in part c)

4c) Stefan’s law:

Since we have W from part 4b to equal we can equate this to . This gives the following:

Will cancel giving Stefan Boltzmann’s constant W m−2 K−4. The known value of the Stefan-Boltzman constant is 5.67037321 × 10−8 W m−2 K−4, this is therefore a good agreement as the values are very close to one another.

Error percentage =

= is the relative error and % is the relative percentage error.