Computational Lab Report

Euler & Runge Kutta Modelling Differential Equations

Report by: **Jamie Somers**

November 8, 2021

Abstract

The purpose of this experiment was to compare two different methods of approximately modelling differential equations for certain physical systems in Python. We first attempted using Euler's method for modelling differential equations, in these experiments we compared the graphs given by the Euler method to the expected graphs. This allowed us to see the various shortcomings of the Euler method which will be discussed in-depth. We will attempt to show that the Runge Kutta Modelling is a much more robust modelling technique which continues to accurately describe physical systems where Euler's method breaks down. It should be noted that the comparative analysis being done in the scope of this report mostly covers the differential modelling of a harmonic oscillator and damped harmonic oscillator, this report can not make any definitive claims of the usefulness of either method for other physical systems and more experiments should be performed to determine either methods suitability when attempting to model other physical systems.

1 Introduction

When trying to model physical systems described by differential equations, physicists ran into the issue that it is extremely difficult to find an analytic solution to a differential equation that is more complex than a standard linear differential equation. It is for this reason that physicists and scientists in general have moved towards using computers to solve differential equations numerically, computers are capable of carrying out extremely complex equations over a large data set in a short amount of time. Despite the convenience of computers this method still isn't perfect as there are many different methods for getting approximate solutions to differential equations, this leads to the question of which method is best as not every method is equally as good at modelling physical systems. In this report the question of which method is best will be applied to two different Runge-Kutta (RK) methods, specifically the Euler method and a particular form of the RK method (RK4) a fourth order scheme which we hope will produce better representations of physical systems over the Euler method.

Everyday physicists are modelling the way our universe interacts and how matter behaves on a fundamental level, when discussing simple systems it is very easy to do these calculations by hand and achieve a final answer for a particular set of initial conditions, however once we want to start modelling how things behave over time it is no longer feasible to carry out these calculations by hand and the job rests on a computer to do these calculations as quickly and efficiently as possible. When physicists first started using the power of computers to carry out these models the Euler method was sufficient as it could very easily produce simple models of systems which behaved according to differential equations, nowadays as physicists continue to understand more about the universe and produce more complicated systems the simplicity of Euler's method is no longer sufficient and the errors produced from this method are too prominent to be ignored.

This report will give an adequate depiction of the increasing errors given by Euler's method and also explore why the Runge-Kutta fourth order scheme (RK4) is so much more accurate.

2 Background and Theory

We can't understand how the two methods of modelling differ until we understand how each methods work, first we will start by explaining Euler's method. Euler's method involves assuming the next value in a curve by drawing a tangent to the slope of the previous point and choosing the new point that is a particular distance away from the previous point, this chosen distance is known as the step size. If we take a differential equation such as

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

And we start off with one known point $y(0)$, we can draw a tangent to the slope of that point and pick our next point to be on that tangent a distance (say h) away from the previous point. This process of essentially "guessing" the next point as being along a tangent at a certain distance away means there is a sizeable room for error, in fact the error is described as the local error (error per step) is proportional to the square of the step size, and the global error (error at a given time) is proportional to the step size. This means that as you increase the step size you should see a more accurate model and a larger step size means a greater margin of error.

We can visualise the Euler method using the figure included below:

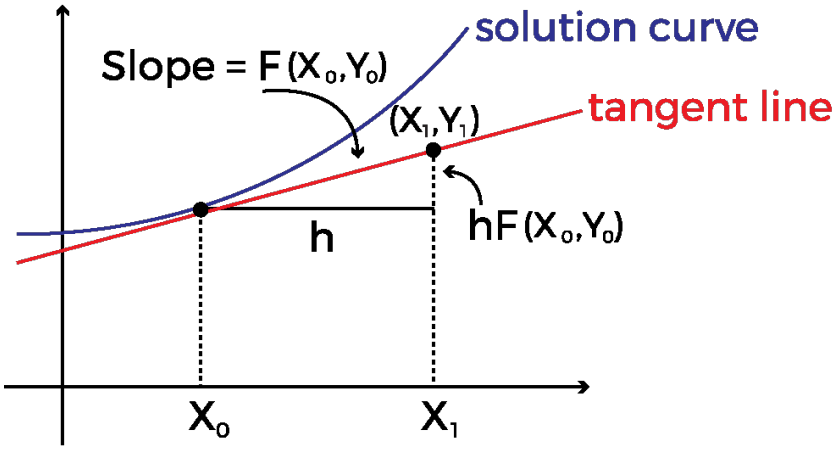

Figure 1: Representation of Euler Method

From Fig. 1 we can see that to go from X_0 to X_1 we simply add h or $X_1 = X_0 + h$, to get the slope of the tangent line $(F(X_0, Y_0))$, we do the difference in y over the difference in x or $F(X_0, Y_0) = \frac{\Delta y}{h}$ This gives us the following equation for the change in y:

$$\Delta y = hF(X_0, Y_0)$$

If we want to find the next value for y Y_1, we simply add Y_0 to this equation like so:

$$Y_1 = Y_0 + hF(X_0, Y_0)$$

We can generalise this for any number of iterations of x and y and we get the following equations for Euler's method:

$$X_{n+1} = Y_n + hF(X_n, Y_n)$$

$$Y_{n+1} = Y_n + hF(X_n, Y_n)$$

This is the entire mathematical concept behind Euler's method, and the first method we will be using in Python to model our physical systems. By the end of this report we should have a reasonably accurate picture of Euler's methods advantages and disadvantages for modelling these particular physical systems.

In order to determine the shortcomings of Euler's method we have to compare it to some other method of modelling differential equations in Python, for this purpose we will use The Runge-Kutta method or more specifically the Runge-Kutta fourth order scheme (RK4).

Euler's method of solving differential equations can often be described as introductory and simple as I have said previously above, and the RK4 method is a great way of showing how a method of graphing differential equations can scale in complexity. It should be clear by now that the technique described above involving using a tangent protruding from a slope to estimate the next point has a lot of uncertainty and doesn't account for changes in the curvature of the graph from one point to the next. This is why RK4 attempts to use the slope of a line at one point and expand it to more than a single point and find the next point as an amalgamation of these multiple points, in the case of a fourth order scheme specifically four points are used to extrapolate the next point in time.

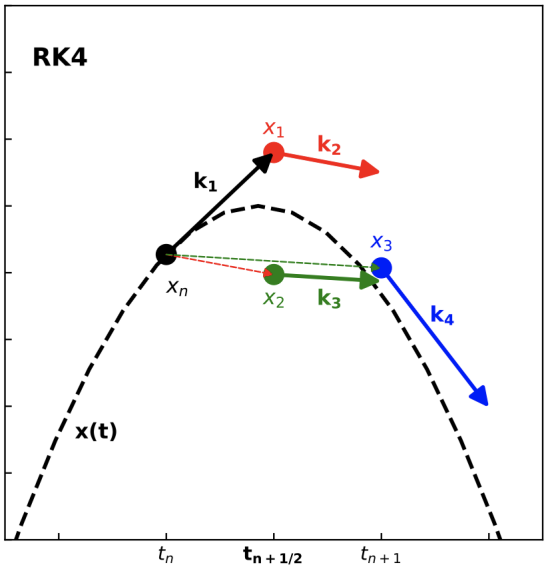

Figure 2: Representation of Runge-Kutta fourth order scheme

In Fig 2 we see a similar process as with the Euler method however there are clearly four different approximations of the slope within the time interval, the points labelled k_1 to k_4 represents these four approximations. As before we write the differential notation as follows:

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

Therefore our first point x_n gives us our initial slope the same as the one from Euler's method k_1.

$$k_1 = f(x_n, t_n)$$

however instead of going a full step h, we limit the distance of this point new point to $h/2$. This is how we arrive at x_1.

$$x_1 = x_n + \frac{h}{2}k_1$$

We can then move on to evaluating the next slope k_2 by using the initial equation again at our newly defined point x_1.

$$k_2 = f(x_1, t_{n+\frac{1}{2}}) = f(x_n + \frac{h}{2}k_1, t_n + \frac{h}{2})$$

We now move back to point x_n and repeat this process to find a new point x_2 instead of carrying on in the direction of x_1.

$$k_3 = f(x_2, t_{n+\frac{1}{2}}) = f(x_n + \frac{h}{2}k_2, t_n + \frac{h}{2})$$

Now we have moved one full step from our starting point x_n so we no longer half our step h. This gives us a final slope k_4 using the equation on our newly discovered point x_3.

$$k_4 = f(x_3, t_{n+1}) = f(x_n + hk_3, t_n + 1)$$

We know take all of these points and try to come up with a final point and slope that makes the most sense following this progression using a weighted average as we are most confident in the points a half step away from our starting point x_n and least confident in the point furthest away.

This weighted average equation gives us the following:

$$x_{n+1} = x_n + \frac{1}{6}(k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4) \cdot h$$

This is a considerably much more precise method of finding the next point and its error is stated to be in the order of h^5, this means that even when attempting to perform this over a large number of values the error isn't nearly as large as it is using the Euler method or that larger step sizes can be used and still be as accurate. This method is so widely used and trusted by scientists that it is incorporated into the SciPy package which many physicists use on a daily basis.

Now that we have a foundational understanding of how both methods go about extrapolating the next point in a curve we can begin analysing the results of the experiments carried out using both methods and see if the experimental data lines up with our theoretical understanding of how the two models work.

3 Results

In this experiment we carried out the two method analysis on multiple different python programmes. In order to make this repeat as comprehensible as possible, the results section has been split up into subsections of Euler's method, which consists of programmes E1 to E4, Runge-Kutta method which consists of programmes R1 to R8, and a conclusion on both methods.

3.1 Euler's method

E1: Radioactivity

Radioactive decay is an excellent physical system for representing exponential decay. In this program we are trying to accurately graph the decay of a Uranium-235 source, we can't accurately depict when decay will take place but we can determine the probability for decay to take place. The behaviour of ^{235}U decay is defined by the following differential equation:

$$\frac{dN}{dt} = -\frac{N}{\tau} \tag{1}$$

Where N is the number of nuclei and τ is the time constant (mean lifetime) for the decay.

Integrating this equation gives:

$$N(t) = N_0 \exp(-t/\tau) \tag{2}$$

Where N_0 is the number of nuclei present at time t = 0.

For our purposes we will be using Euler's method to describe this differential equation and determine N as a function of t. Given an initial value for N and t, we want to be able to estimate the value of N for any value of t. In order to achieve this goal we must first expand N as a taylor series about $t = 0$ (this will be our initial value for t).

Using the method of Taylor Series described previously we arrive at the following equation:

$$N(t + \Delta t) \approx N(t) - \frac{N(t)}{\tau}\Delta t \tag{3}$$

This equation is the one used within the python code to generate a figure which demonstrates radioactive decay. Since we know the exact solution should follow the equation

$$N(t) = N_0 e^{-\frac{t}{\tau}} \tag{4}$$

We can include the exact slope in our figure and compare how the Euler method stacks up with increasing time steps.

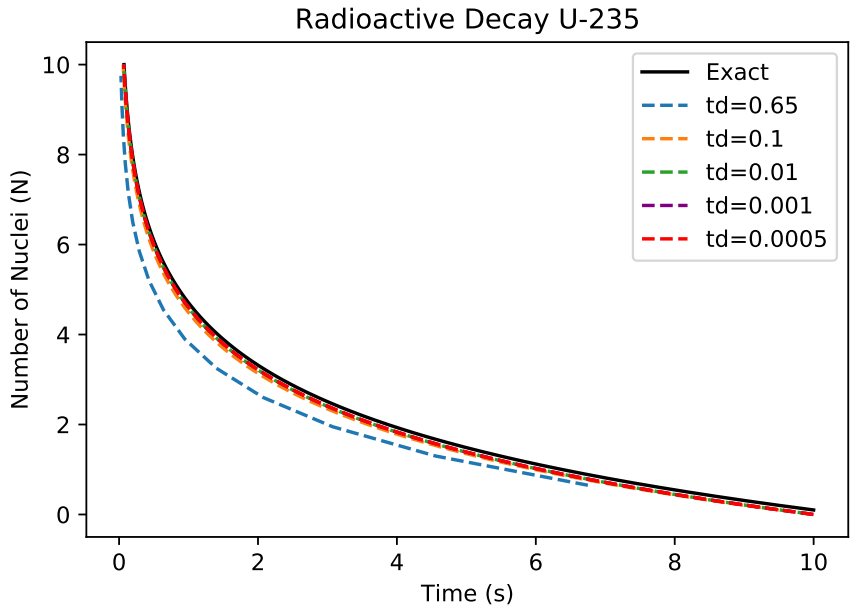

Figure 3: Representation of Euler's Method for Radioactive decay

Euler's method works surprisingly well at mapping the radioactive decay of a Uranium 235 isotope, however we can clearly see that it does a better job at it for smaller steps such as dt = 0.0005 than it does for larger values like 0.1. If the steps were to be increased further we would see this affect worsen as discussed in the theory of Euler's method, I have included a step size of 0.65 to show this affect.

E2: A non-linear example

Moving on from a physical system like radioactive decay which has an easy to reproduce in Python exact solution we now turn our attention to a non-linear function. The function in question is:

$$\frac{dx}{dt} = t - x^2, \text{ where } x = x_0 \text{ at } t = 0 \tag{5}$$

This function has no exact solution in terms of elementary functions so it will be interesting to analyse how Euler's method handles the mapping the function.

Using a technique we learned during the second experiment we can plot two graphs of this function side by side, with different tmax values for both. We set the step size to $h = 0.05$ for 5 different initial conditions namely x_0 = 3,1,0,-0.7 and -0.75.

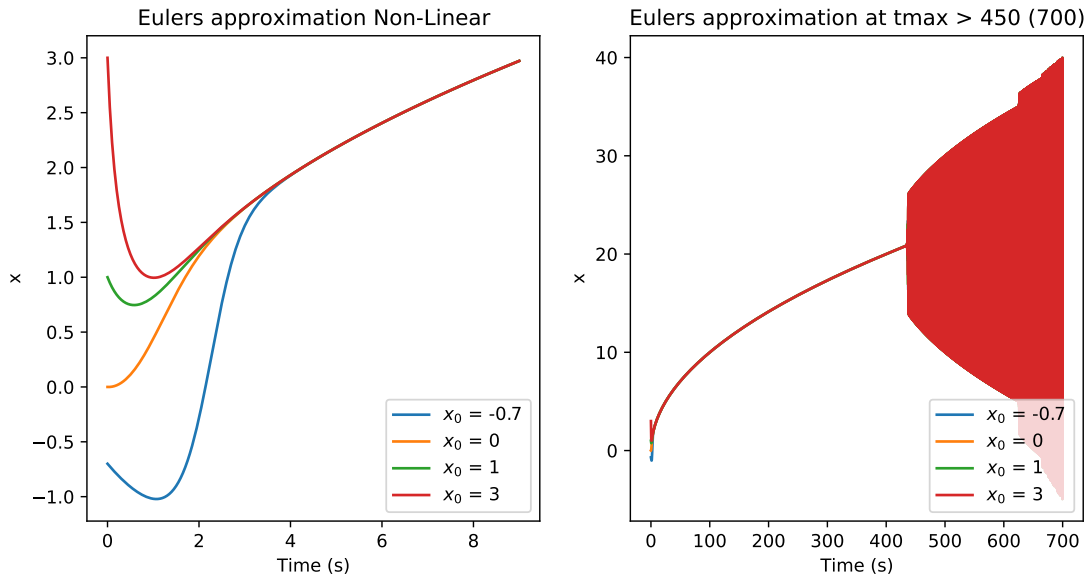

Figure 4: Representation of Euler's Method for Non Linear Eq.

Our graph shows a relationship where all of the different initial conditions will eventually converge at one point given enough time. We allowed the graph to run for an extremely long amount of time, in this case 700 seconds, what starts to form is that after approximately 450 seconds the x value starts to oscillate back and forth across a wide range of values, this range increases even more as time goes on. This is a perfect example as to why Euler's method does not work for large steps, as time goes on in the figure on the right the value becomes less and less accurate and the graph is almost unrecognisable to the one where tmax $<$ 450.

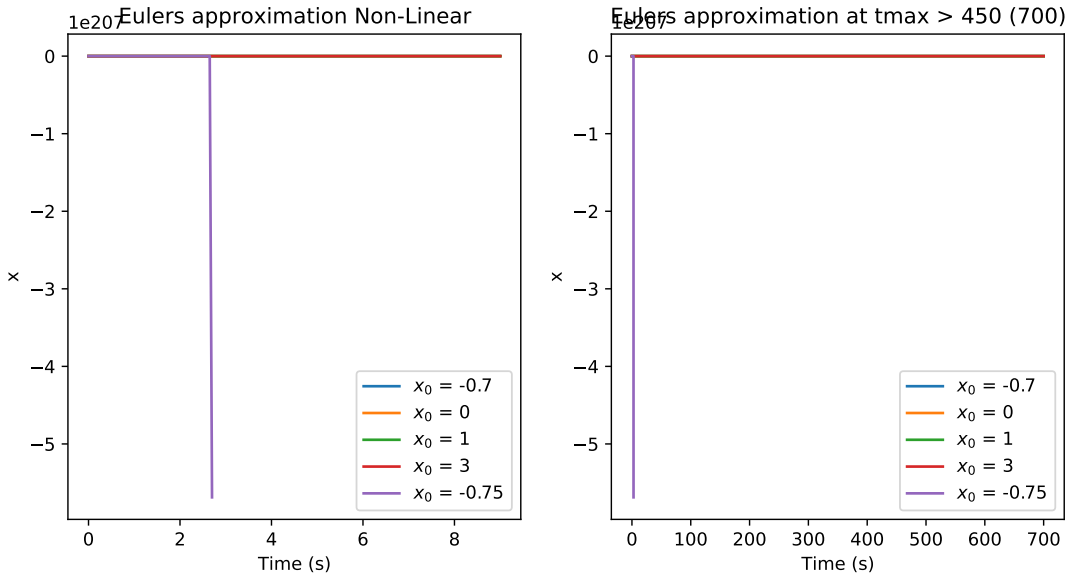

Figure 5: Representation of Euler's Method for Non Linear Eq.

Euler's method is incompatible with values that are deemed as "limits". When we set our initial condition of x_0 to some value which there is no solution for (in this case $x_0 = -0.75$). The graph completely breaks and is no longer capable of being analysed.

E3: Coupled equations

Although what we have shown Euler's equation to be capable of so far is of great use to physicists trying to model physical systems, another type of system which the physical universe often behaves as is a second order differential equation. For example waves and anything related to a harmonic oscillator will need to be mapped as a second order differential equation. In order for us to represent second order differential equations using Euler's method we must first split them out into a set of first order equations, these sets are known as coupled equations and Euler's method can be extended to solve such coupled equations within Python. The example we do below will be for mapping a simple harmonic oscillator.

The equation we will be using is for a simple harmonic oscillator and a simple harmonic oscillator will follow this second order differential equation:

$$\frac{d^2x}{dt^2} + \omega^2 x = 0 \tag{6}$$

Once again this is an equation with an exact solution:

$$x = \frac{v_0}{\omega}\sin(\omega t) \tag{7}$$

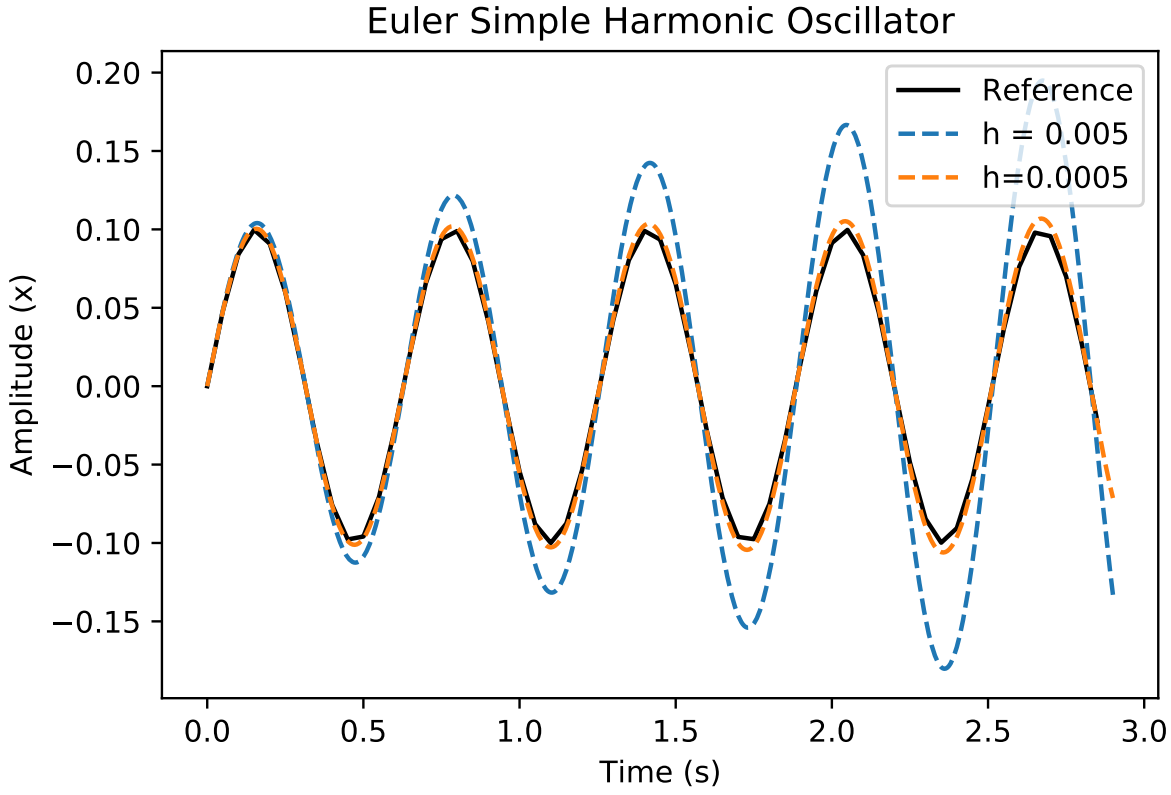

Figure 6: Euler's method simple harmonic oscillator

We can clearly see the accuracy decreasing from the reference wave with each oscillation, we do observe an increase in the amplitude which is noted in the manual as a limitation of the Euler's method. We can also see that even a small increase in the step size results in a huge change in the graph after only a few oscillations.

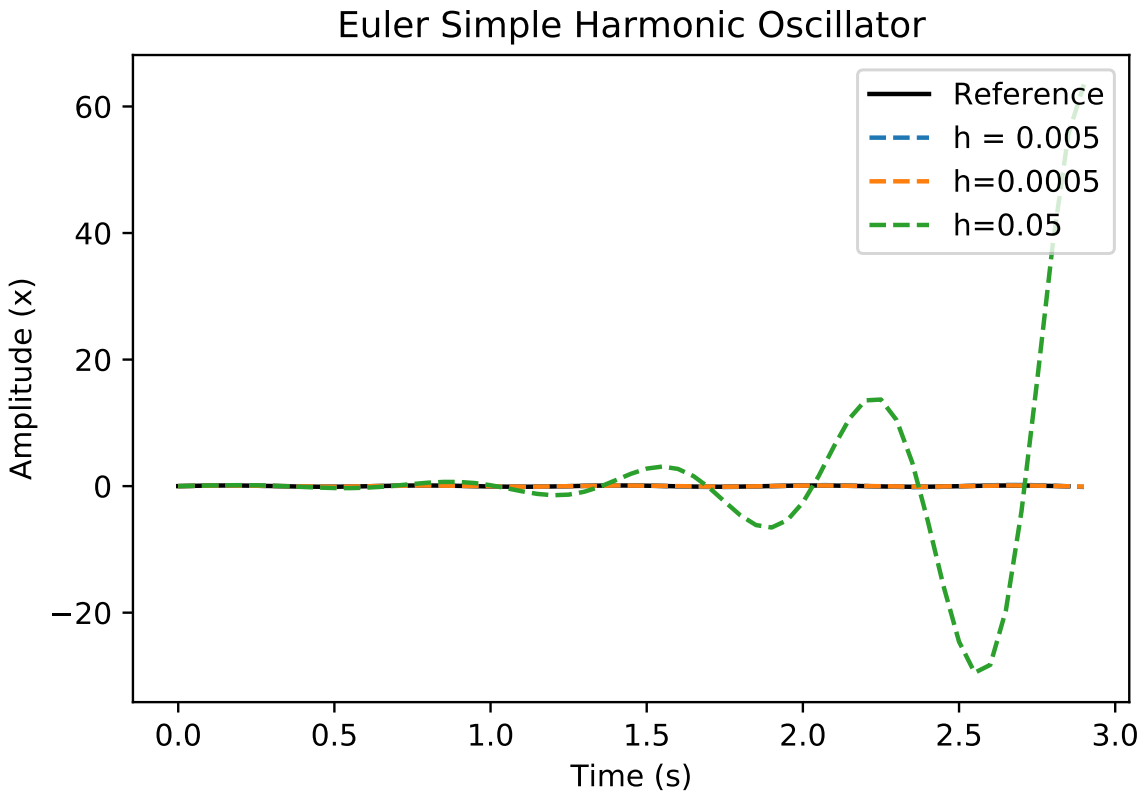

Figure 7: Euler's method simple harmonic oscillator

In this graph we have included a step size of $h = 0.05$ we can see that at this step size Euler's method produces a signal that isn't even a sine wave. Once the step size is larger than 0.04 it no longer represents a harmonic oscillator. This is most likely a limitation of the Euler method as when the step size is too large the points are set at a greater distance from one another and quick changes in slope such as the ones seen in sine waves are lost in the approximation.

E4: Modified Euler Method

By now we have clearly shown that the Euler method is extremely limited, to the point where its in the best interest of everyone analysing data to use a different method of mapping, the simplicity of the Euler method is matched only by the size of the error given for any physical system mapped on a reasonably long step time.

However, if one was really determined to use the Euler method over one of the other available methods within SciPy, there is one modification you can implement to reduce the compounding error that occurs on larger time scales. The modified Euler's method improves the accuracy by taking the weighted average between the current point and the extrapolated point. This is done using the following equation:

$$x_{n+1} = x_n + 0.5h(f(x_n, t_m) + f(x_{n+1}, t_{n+1}) \tag{8}$$

Using this formula should greatly improve the Euler method for equations where there is a huge change in slope, specifically the last program where the resultant wave was a sine wave with constantly changing slope. For this reason we will attempt to implement the modified Euler method into the previous program and see if it improves the mapping as much as we expect it to.

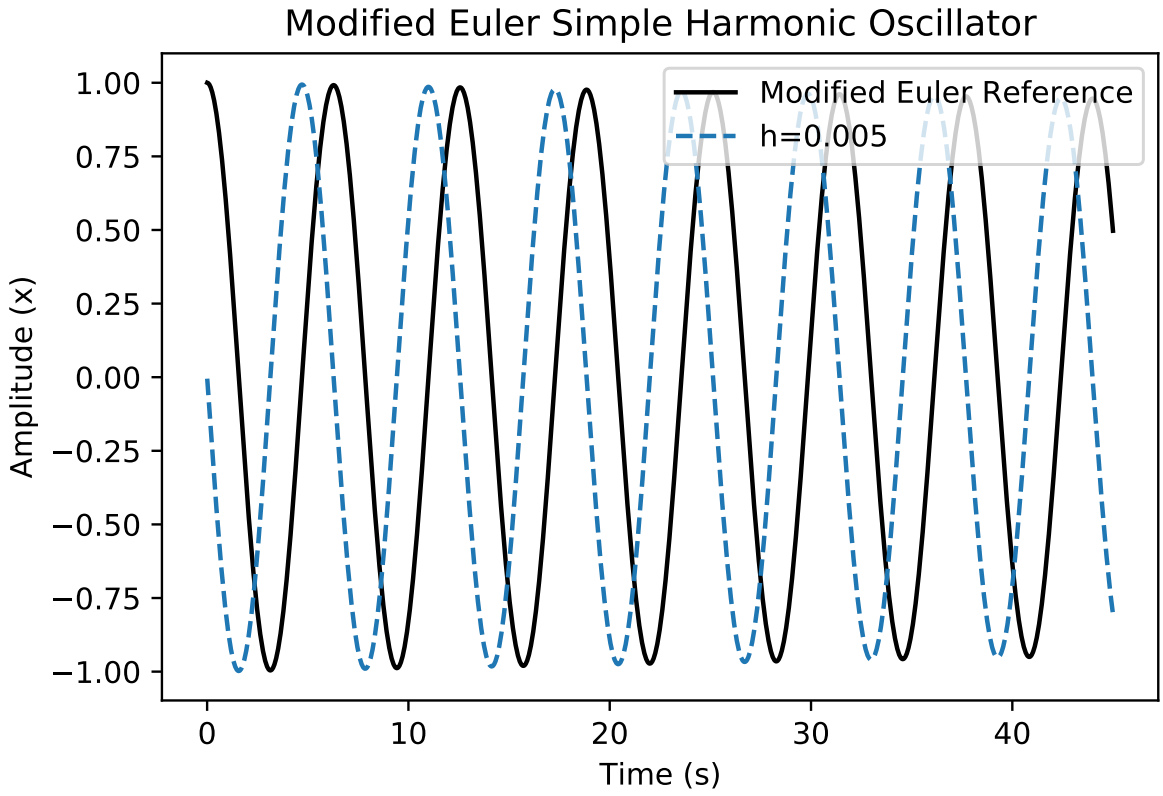

Figure 8: Modified Euler's method simple harmonic oscillator

We can see a huge improvement between the mapping of the reference signal to the Euler approximated signal. The two signals appear to be out of phase however this can be easily corrected by adding a phase shift to the modified Euler equation used in the python script.

3.2 Runge-Kutta method

R1: Using SciPy to solve ODEs

As mentioned previously in this report, the Runge-kutta method is already available inside the package SciPy as it is a widely used method of solving Ordinary Differential Equations. From this point on we will be using "from scipy import integrate" in our Python scripts in place of Euler's method.

We will be using the RK45 method to map the following equation:

$$\frac{dy}{dt} = -ay^3 + b\sin(t) \tag{9}$$

We mapped the differential equation for 4 different initial values of a and b:

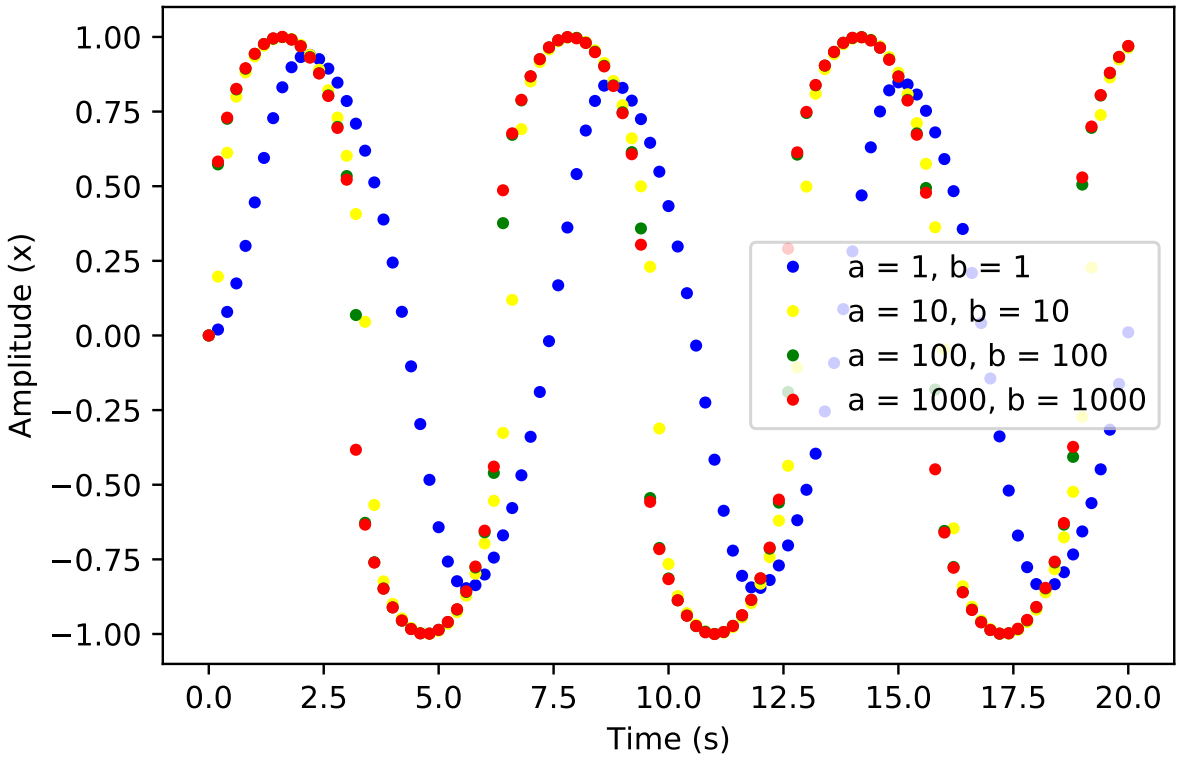

Figure 9: Representation of RK4 Method for Differential Eq.

Now that we have used the SciPy integrate function for RK4 and we are confident it works as intended we can move on to our next program.

R2: RL circuit

The next program involves the use of a hypothetical circuit, the circuit is made up of a resistor R in series with a conductor L, a voltage source V and a switch. We can determine the current I for when the switch is closed at a particular time t.

The differential equation we will be using to map this circuit is as follows:

$$L\frac{dI}{dt} = V - RI \tag{10}$$

This particular physical system also has an exact solution which we can use to determine the accuracy of the RK4 method of mapping. The equation for the exact solution is as follows:

$$I(t) = \frac{V}{R}\left[1 - \exp\left(-\frac{Rt}{L}\right)\right] \tag{11}$$

As we have done previously with the Euler method we will be mapping both the RK4 method and the exact solution to compare them and analyse how accurate the RK4 method really is.

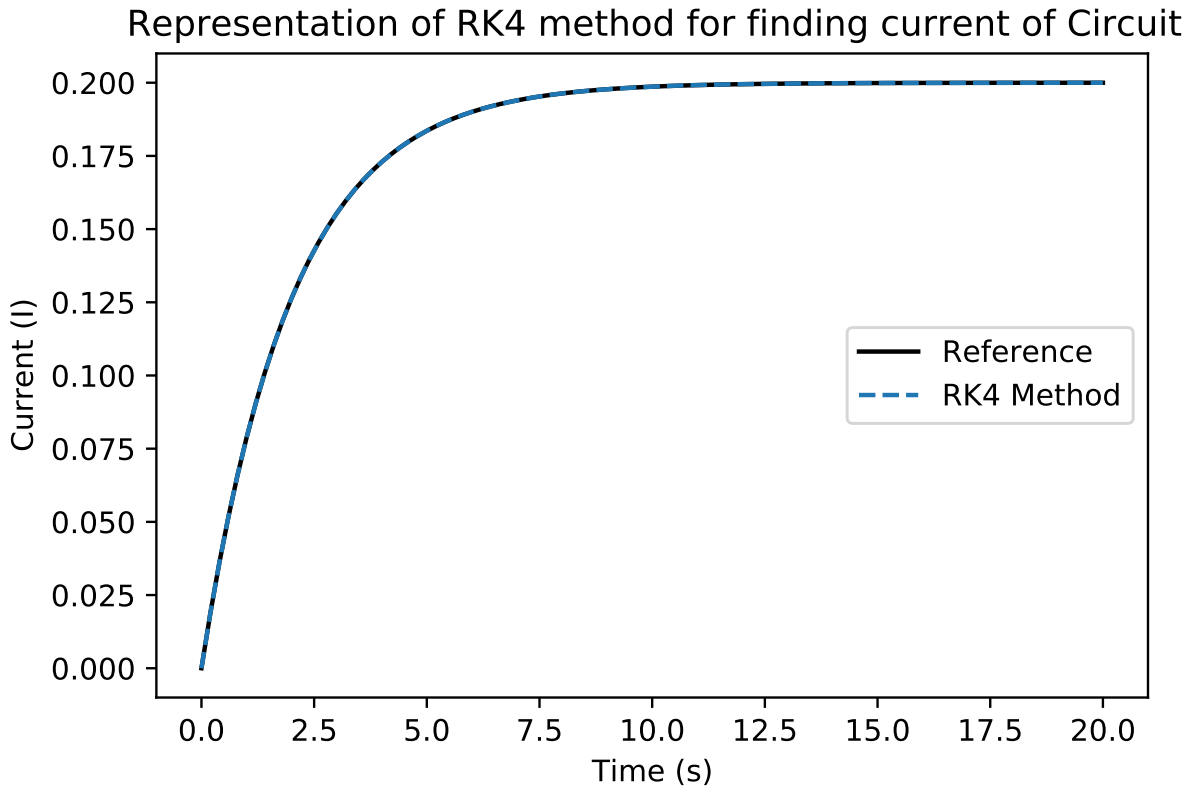

Figure 10: Representation of RK4 Method for Circuit Physical system.

We can see that our approximation maps pretty much perfectly on to the exact solution. This perfectly shows how much better the RK4 method is at mapping physical systems by using a moving average of four points instead of just the one. In this example there is a very light slope followed by a steep slope and then another light slope, the RK4 method has no problem adapting to these conditions and correcting extremely quick changes in slope.

R3: Harmonic Oscillator

Now that we know the RK4 method is the superior method for mapping quick changes in slope, we can once again go back to the harmonic oscillator example as see if this method has greater success accurately mapping the physical system. We will once again be using Eq. 6 and Eq. 7 so there is no need to repeat them here.

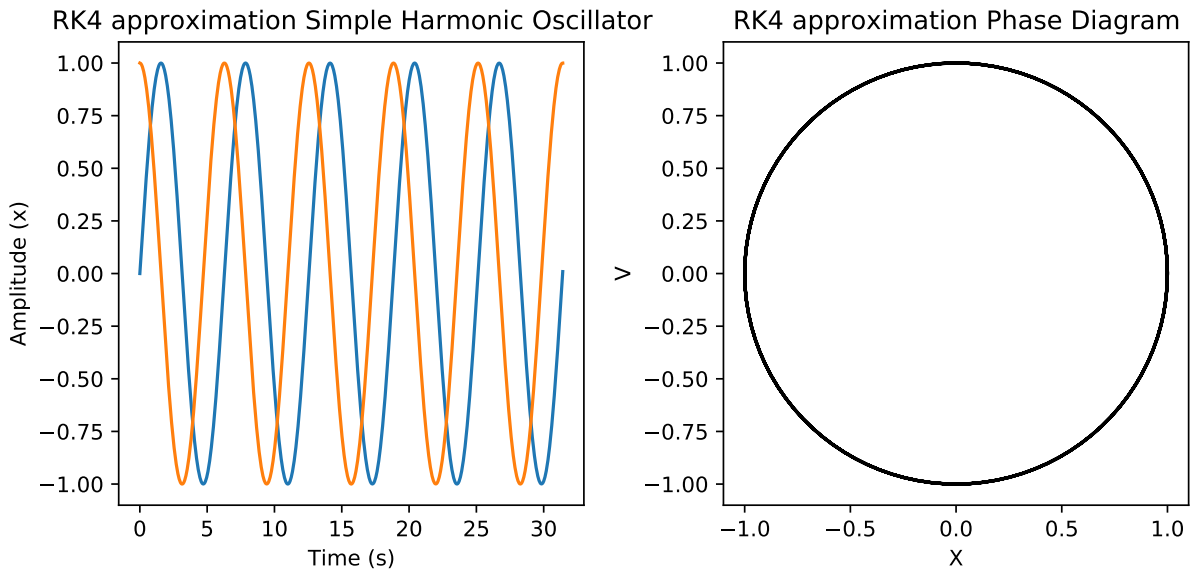

Figure 11: Representation of RK4 Method for Simple Harmonic Motion.

Here we have included a graph to represent how the oscillator moves with respect to time on the left hand side and how the velocity changes with respect to displacement. Once again the two waves are out of phase however they appear to be perfectly equal in amplitude and would map pretty much perfectly if a phase shift was included in the RK4 equation. This is yet another example of the RK4 method being extremely accurate and the ability to generate a phase diagram such as the one on the right hand side is a really important inclusion when analysing real world systems that move with simple harmonic motion.

R4: Damped Harmonic Oscillator

One of the experiments undertaken in the fourth year labs is to measure the Simple Harmonic Motion of a pendulum when a source of magnetic damping is introduced. In this program we will be introducing a damping coefficient to our simple harmonic oscillator which should result in a continuous reduction in the amplitude of the oscillator. The damped oscillator can be described with the following differential equation:

$$\frac{d^2x}{dt^2} + b\frac{dx}{dt} + \omega_0^2 = 0 \tag{12}$$

The added term 'b' is our damping coefficient which affects the oscillator with respect to velocity $\frac{dx}{dt}$. We were able to replicate the expected figures for damping:

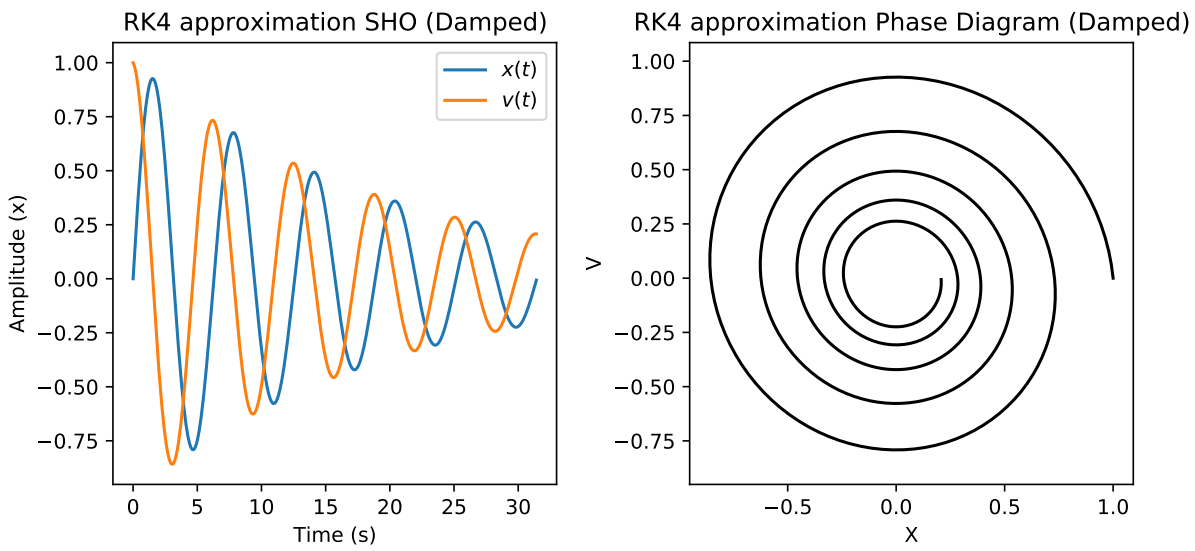

Figure 12: Representation of RK4 Method for Simple Harmonic Motion.

As theorised above, the amplitude of the oscillator decreases in an exponential fashion as the damping coefficient continuously affects the calculated value during each oscillation. Similarly the phase diagram shows a decrease in the velocity being reached by the oscillator as it exponentially decreases until the oscillator will eventually come to rest.

R5: Lambda function

Lambda functions are extremely powerful functions in python which allow us to define a new function which can be called without defining all of the parameters. For this program we will use the power of lambda function using the same simple harmonic oscillator system as before. Once we get the script to run using lambda functions we can change the parameters of the damping coefficient and the resonant frequency to one of our choosing.

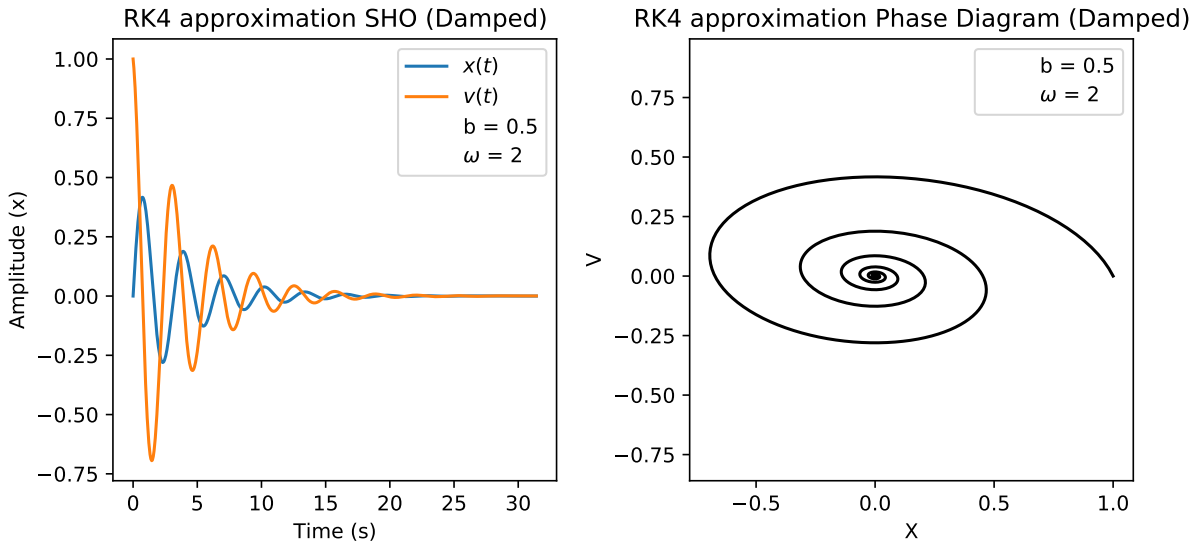

Figure 13: Representation of RK4 Method for Simple Harmonic Motion.

We successfully managed to implement lambda functions into our script and chose a damping coefficient of 0.5 and a resonant frequency of 2.

R6: Driven oscillator

This example is another take on modifying the simple harmonic oscillator, this time instead of inducing some damping force such as a magnet at the base of a pendulum we are instead inducing some driving force at the pivot point of the pendulum such as moving the pivot point up and down the frequency of this driving force ω_d, as long as we keep this frequency similar to the natural frequency of the system we should see some resonance. The equation for this physical system is as follows:

$$\frac{dx}{dt} + b\frac{dx}{dt} + \omega_0^2 x + A\sin(\omega_d t) = 0 \tag{13}$$

Where A is the amplitude of this newly introduced frequency and ω_d is the oscillation of this newly introduced frequency.

This equation was split into coupled equations

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = -bv - \omega_0^2 - A\sin\omega_d t$$

Below is the graph for a driven system with initial conditions A = 1, b = 0.1, ω_0 = 1, and ω_d = 0.9.

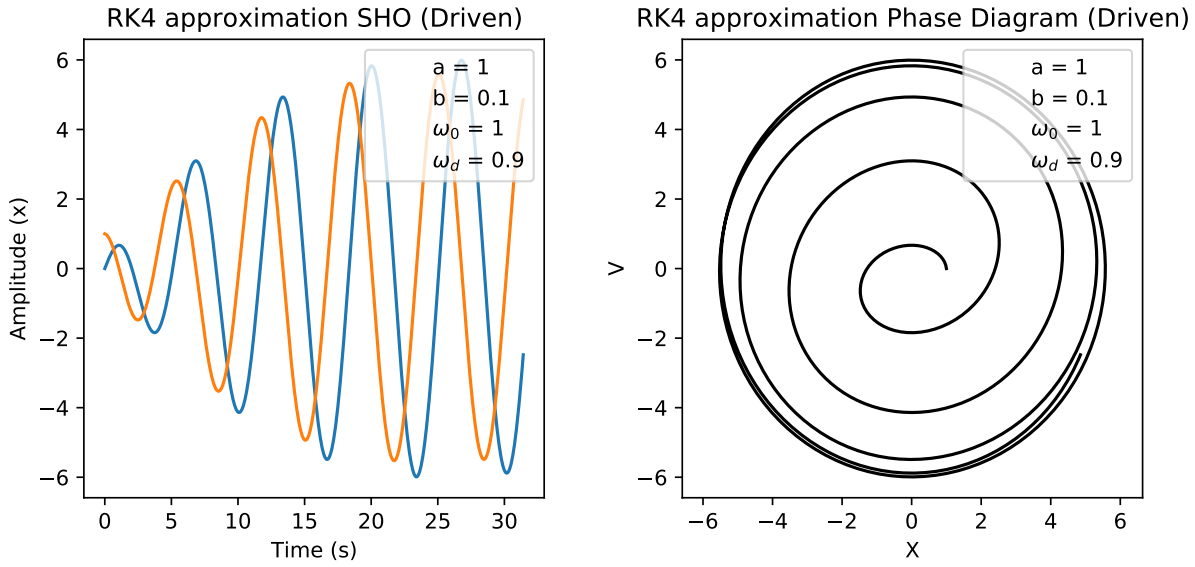

Figure 14: Representation of RK4 Method for Driven Oscillator.

We then kept the A and ω_0 value constant and varied the b and ω_d values. This is what the graph looked like with $b = 0.5$ and $\omega_d = 1.5$

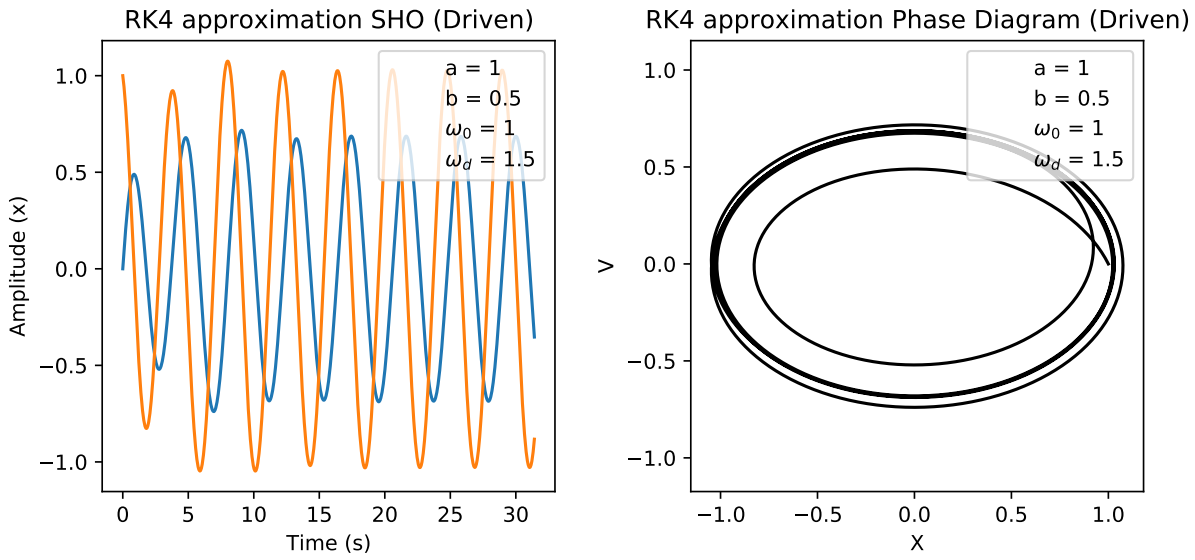

Figure 15: Representation of RK4 Method for Driven Oscillator.

It appears as though the Driven oscillator has a large amount of change in its state at the very beginning for the first 5 seconds before settling into a steady amplitude and phase. This corresponds to when the oscillator is moving out from the centre circle of the phase diagram into the continuous circle it eventually enters. This would be the initial transient state for the first 5 seconds and then the steady state thereafter.

We also tested what happens when the driving frequency is close to and far away from the natural frequency.

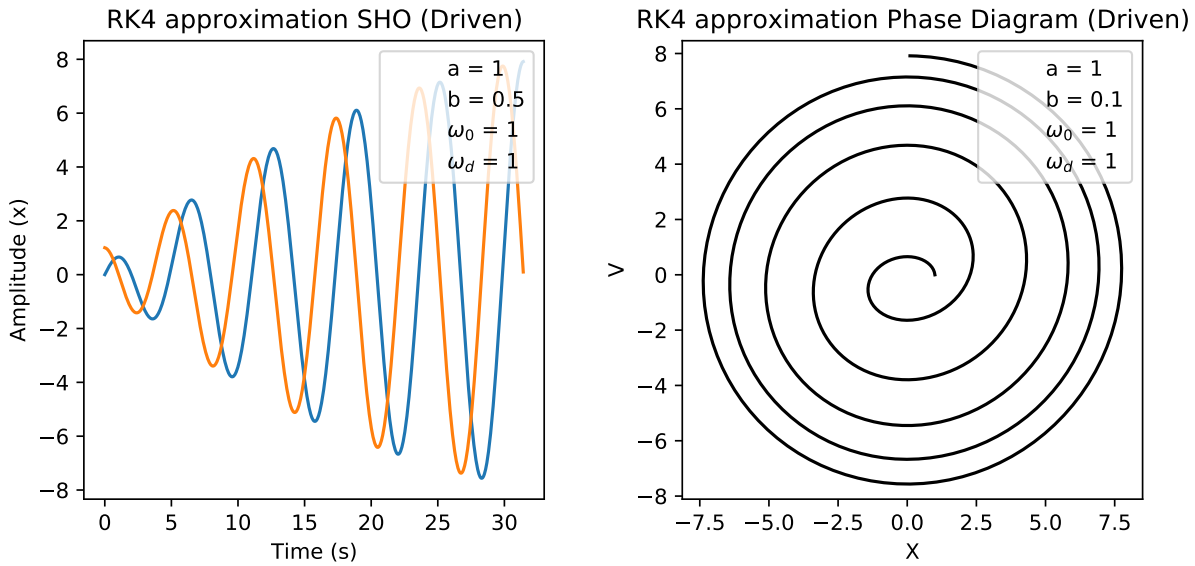

Figure 16: Representation of RK4 Method for Driven Oscillator.

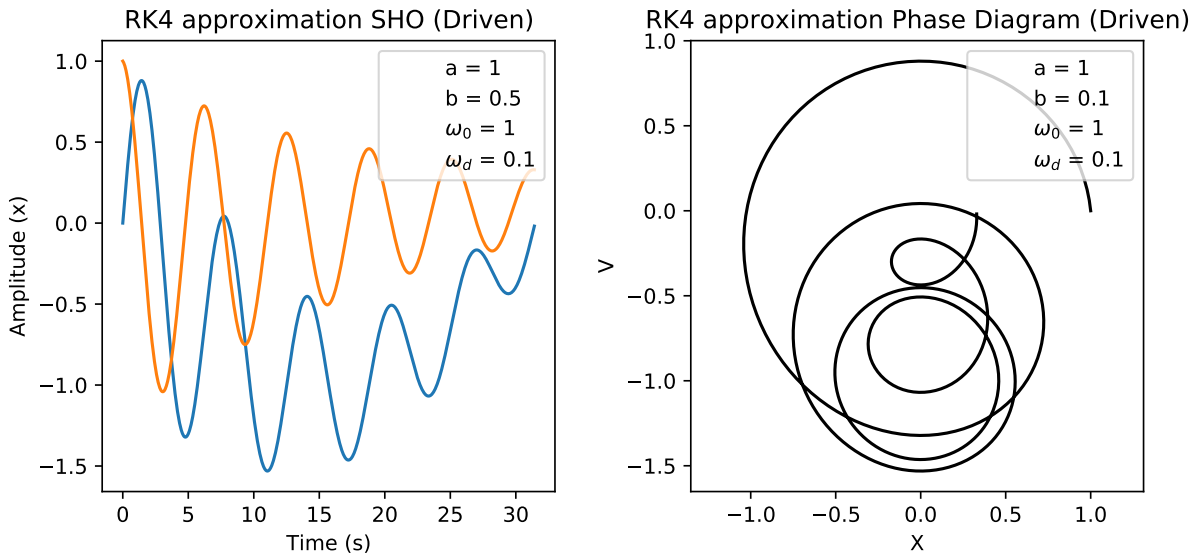

Figure 17: Representation of RK4 Method for Driven Oscillator.

We can clearly see from our two graphs that when the natural frequency and the driving frequency are in sync there is a resonance and they appear similar perfectly matching each others frequency and when the frequencies are extremely different there is disorder as the frequencies no longer interact in a way that causes them to be in balance.

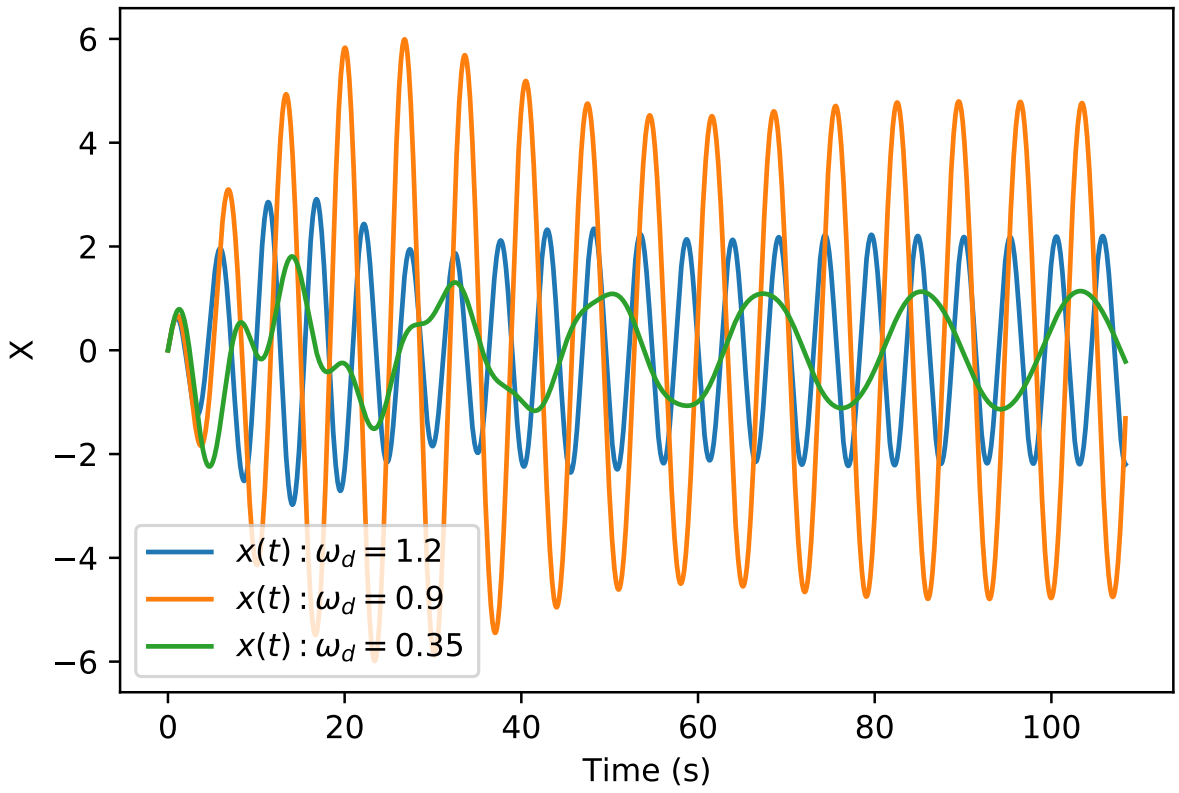

Figure 18: Driven Oscillator for A=1, b=0.1, ω_0=1, and varying ω_d

The graph above shows how the system varies as the driving oscillation ω_d is changed. One of the values 0.9 is the natural frequency and the other two values are chosen to represent values above the natural frequency and values below the natural frequency.

It appears as if the wave at the natural frequency finds its stable state sooner than the two other waves, and the value we chose for above the natural frequency finds its steady state sooner than the one below. This implies that the rate at which the oscillator finds its steady state is proportional to how close the oscillation value of the driving force is to the natural frequency which is what we expect to observe in a system like this.

R7/R8: Shape of the resonance / Amplitude of the resonance

Finally we mapped the shape of the resonance frequency and also the amplitudes for different values of b with ω_d varying in both cases.

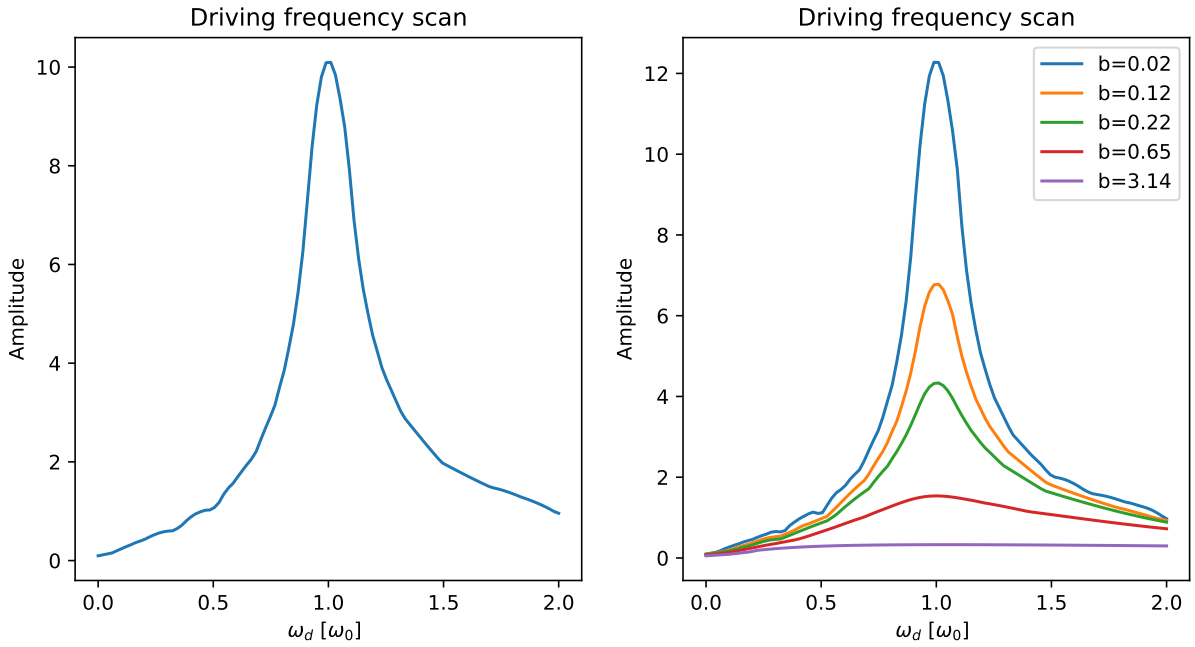

Figure 19: Amplitudes of solutions to (15) for A=1, b=0.1, ω_0=1, and varying ω_d .

Figure 20: Amplitudes of solutions to (15) for A=1, b=0.1, ω_0=1, and varying ω_d and b .

We can clearly see from Fig. 20 that as the value of the damping coefficient b increases, the sharpness decreases and the curve becomes more spread out. This describes an inverse relationship between the sharpness and the damping coefficient.

Conclusion on Euler's Method & RK4 Method

Throughout the course of this data analysis we have effectively shown the implementation of both the Euler method as well as the RK4 method. Carrying out a wide range of programs on multiple physical systems allowed us to see the strengths and weaknesses of using one method over the other and how to efficiently solve differential equations through Python using a variety of techniques including Lambda functions and coupled equations.

Where possible we tried to plot each method against the known solution which clearly showed that Euler's method did not map the differential equations nearly as accurately as the RK4

method could. The Euler method also suffers greatly from compounding errors, although we were able to negate these compounding errors by using a modified version of Eulers method for the Simple Harmonic Oscillator it still can't be ignored that the Euler Method was far inferior to the RK4 method which almost perfectly managed to map each of the physical systems included in this report which had an exact solution to compare it to.

If we couple this analysis along with the use of Lambda functions and the RK4 method's ability to include a phase diagram along side it, its clear that for modelling physical systems the RK4 method is the best way to go. Although the Euler method is great for introductory modelling modules and getting early career physicists started, its important to show students how inaccurate models created using the Euler method can become, especially over large steps.

4 Discussion and Conclusion

It was found during E1 that Eulers method differed somewhat from the expected values of radioactive decay. This is most likely due to Eulers method's inability to handle changes in curvature.

In E2 we observed that having extremely large time scales causes Eulers method to completely stop working $t_{max} > 450$ and also at "limits" where there is no defined solution to the differential equation e.g $x_0 = -0.75$.

For E3 it was discovered how to use Euler's method on second order differential equations using coupled equations, we also found that Euler's method becomes less accurate over time for systems such as a simple harmonic oscillator when compared to the expected solution.

For E4 showed us that it is possible to improve Euler's method and make it more accurate for systems that have large changes in slope in short amounts of time. This was once again easy to implement but not as accurate as we would hope for when modelling physical systems.

In R1 We successfully imported the RK4 method into Python using SciPy and mapped a first order differential equation for multiple initial values.

In R2 we modelled a circuit system attempting to find the current at a given point in time knowing the initial conditions of the circuit at time $t = 0$. This was implemented successfully and mapped on to the expected values exactly showing the accuracy of the RK4 model.

R3 and R4 showed how much more accurate the RK4 model is at graphing simple harmonic systems, this model mapped not only the harmonic oscillator but also a damped oscillator and was incredibly accurate compared to the Euler model. This shows that the RK4 technique of extrapolating a slope using a moving average of the calculated points is a far better method of graphing than using a point found along a tangent of a slope.

In R5 we incorporated Lambda functions into our equations which are extremely powerful functions which make graphing differential equations in Python easier.

In R6, R7 and R8 we worked with a driven oscillator which added both the damping coefficient and a driving frequency and amplitude to the simple harmonic motion, this allowed us to analyse the relationship that exists between natural frequency and driving frequency and how resonance that is produced by a driving oscillator changes over driving oscillation and with

varying damping coefficients. All of this was made possible by the power of the RK4 method which has been proven to be far superior to the inferior Euler method.

It can be stated that for the purposes of this experiment and for each of the programs that were run replicating physical systems the RK4 method was the better method for extrapolating data points and graphing differential equations.

5 References

Figure [1] Calcworkshop. Dec 31, 2019. How to do Euler's method? Simply Explained in 3 Powerful Examples https://calcworkshop.com/first-order-differential-equations/eulers-method-table/

Figure [2] DCU Physics Department Nov 8, 2021. PS351 – Computational lab – Week 2: Solving ODEs with SciPy – The Runge-Kutta Method and Phase Space