Lab 4: Fourier Analysis using Python

Thomas Brosnan

February 1, 2023

1 Introduction

The aim of this lab was to perform an investigation into Fourier analysis using both Fourier series and Fourier transforms to generate different functions. Both these methods require one to compute several complicated integrals so the first task was to develop a numerical integration method that approximately found a solution to these integrals. The method used is called Simpsons rule. This approximation has implications on the accuracy of the resulting Fourier series or transform but makes up for this in its small computational cost.

A Fourier series is a series of sin or cosine functions of different frequencies that when scaled and added together approach the form of a non sinusoidal function. The more terms added the closer the Fourier series resembles the function and the function itself can be said to equal an infinite sum of these sin/cosine terms. To calculate the coefficients of n terms in the series, n integrals need to performed, this is where Simpsons rule was used. The first exercise was to build a program that calculated the Fourier coefficients and put them together to display the function, The second exercise applied this to the more difficult to deal with step functions and the final exercise was apply the discrete Fourier transform. This is a method of obtaining a non-periodic function in terms of sines and cosines, however, the frequencies then become a continuous spectrum and the terms become part of an integral instead of a sum.

This Lab was carried out using python 3 and was written in the Spyder application.

2 Methodology

2.1 Exercise 1: Simpson's rule

The first task as stated earlier was to use Simpson's rule which for any function $f(t)$, approximates the area under a small section (-h,+h) of this curve as a parabola. This gives a simple formula for the area of this section in terms of the function evaluated at the boundaries of the section. This can then be extended by adding up the areas for each interval until the whole desired range has been covered, giving the final result for a function $f(x)$ evaluated between a and b as,

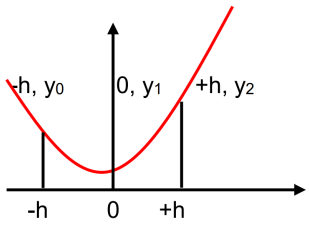

Figure 1: *Example interval*

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(x_0) + 2\sum_{i=1}^{n/2-1} f(x_{2i}) + 4\sum_{i=1}^{n/2} f(x_{2i-1}) + f(x_n)\right]. \quad (1)$$

Where here the number of terms approximated n is arbitrary but increases the accuracy and the the size of the intervals (-h,+h) depends on n by,

$$h = \frac{b-a}{n}. \tag{2}$$

To test the application of the above numerical integration method a know definite integral was performed and its result compared to the analytical value. For this n was taken to be 8. The know integral was,

$$\int_0^1 e^x dx. \tag{3}$$

2.2 Exercise 1: Evaluating Fourier series

The Simpson's rule method was then used to calculate the Fourier coefficients of the Fourier series for any given function $f(t)$. As outlined earlier any periodic function with a period T can be represented by an infinite sum of sin and cosine terms with different frequencies. Each frequency is scaled by some factor a_n or b_n which can be thought of as a measure of how much of that frequency is in the function. In practise we can only calculate a finite amount of these terms but as will be shown these can still be quite accurate at representing the function. The full form of a Fourier series is,

$$f(t) \;=\; a_0 + \sum_{n=1}^{\infty}(a_n cos(n\omega t) + b_n sin(n\omega t)). \tag{4}$$

Then to calculate the coefficients the following integrals were performed. These isolate the coefficients due to the orthogonality properties of sines and cosines.

$$a_0 \;=\; \frac{1}{T}\int_0^T f(t)dt, \tag{5}$$

$$a_k \;=\; \frac{2}{T}\int_0^T f(t)cos(k\omega t)dt, \tag{6}$$

$$b_k \;=\; \frac{2}{T}\int_0^T f(t)sin(k\omega t)dt. \tag{7}$$

To show the results, plots of the coefficients a_k and b_k as discrete points were made as well as the actual Fourier series themselves. To show just how accurate the series were, graphs of the analytical functions were overlay-ed with the Fourier series.

2.3 Exercise 2: Fourier series of square and rectangular waves

Square and rectangular wave functions are periodic functions so they can be represent by a Fourier series. However, they take a lot of terms (large n) to produce an accurate result due to the discontinuous nature of their first derivatives, i.e. they have sharp bends that are hard to replicate with smooth continuous functions like sin and cos. The square wave function for a period $T = 2\pi$ is defined as below, as well as a rectangular wave function with a period a and some parameter $0 < b <$ a that changes the asymmetry.

$$f(t) \;=\; \begin{cases} 1 & 0 \le t \le \pi \\ -1 & \pi \le t \le 2\pi \end{cases}, \tag{8}$$

$$f(t) = \begin{cases} 1 & 0 \leq t \leq b \\ -1 & b \leq t \leq a \end{cases}. \tag{9}$$

The Fourier series coefficients were calculated and compared to the known analytical result and the resulting Fourier series was plotted and compared to the actual square wave function.

2.4 Exercise 3: Fourier Transform

As stated earlier the Fourier transform is a method of obtaining regular non-periodic functions as a function of sin and cosine terms. However, now instead of being made of terms with integer frequencies, they are made up of a continuous spectrum of frequencies as the period T $\to \infty$, making the fundamental $\omega = 2\pi/T \to 0$, changing the infinite sum to an integral over all the frequencies. This turn what was discrete list of Fourier coefficients into a continuous function. Using complex notation to write both the sin and cos coefficients in terms of one complex exponential results in the following formula for the Fourier transform i.e. the continuous coefficient spectrum.

$$F(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-i\omega t} dt. \tag{10}$$

This then raises the question of how to get back to the original function given the Fourier transform. This turns out to once again be an integral as shown below and is called the back/inverse transform.

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega t} d\omega. \tag{11}$$

2.5 Discrete Fourier transform

In practice we cannot perform the above calculations for all functions and exact integrals are replaced the discrete Fourier transform which samples the function at small steps. If the function is a function of t, then $f = f(t_m)$ where $t_m = mh$, for some arbitrarily small h. This results in a fundamental frequency $\omega_0 = 2\pi/(Nh)$ that all other frequencies are multiples of, $\omega_n = n\omega_0$. This then turns equations 10 and 11 into the below form. They are now a list of discrete complex values.

$$Fn = \sum_{m=0}^{N-1} f_m e^{-\frac{2\pi i m n}{N}}, \tag{12}$$

$$fm = \frac{1}{N} \sum_{n=0}^{N-1} F_n e^{\frac{2\pi i m n}{N}}. \tag{13}$$

Using this definition of F_n 12 the python script for the Fourier series was modified to calculate the the discrete Fourier transform. Then plots were made of the real and imaginary components of F_n for the function $f(t) = sin(0.45\pi t)$. As well as this, to show that the DFT was being properly calculated, the back transform as seen in the above equation 13 was performed on F_n. The resulting function was plotted against the original function to compare any changes. This method was then carried out on another function $f(t) = cos(6\pi t)$ while varying the values of N and h to see how they changed the outputted DFT and back transformed function.

3 Results

3.1 Exercise 1: Simpson's rule

The results from Simpsons rule were extremely accurate. On the test integral 3 the programmed Simpsons rule produced a value of 1.718284155, for n=8. Here n is basically the number of intervals the function was split up into between the bounds. This is very close to the known analytical value of $e - 1 = 1.718281284$ making the error just 3.266×10^{-6}. This shows that this method is highly accurate, even for small values of n, which is good for computational cost.

3.2 Exercise 1: Evaluating Fourier series

Simpsons rule was effectively used once more as a part of calculating the Fourier coefficients and series. Shown below are the plots of 4 Fourier series compared to plots of the actual functions. In each case the parameter n was n=100 and the number of coefficients calculated $k_max = 100$to be accurate.

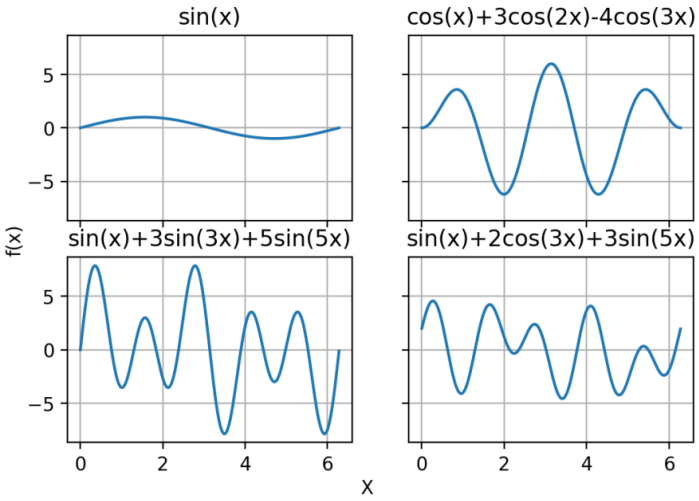

Figure 2: *Plots of the calculated Fourier series for the shown functions*

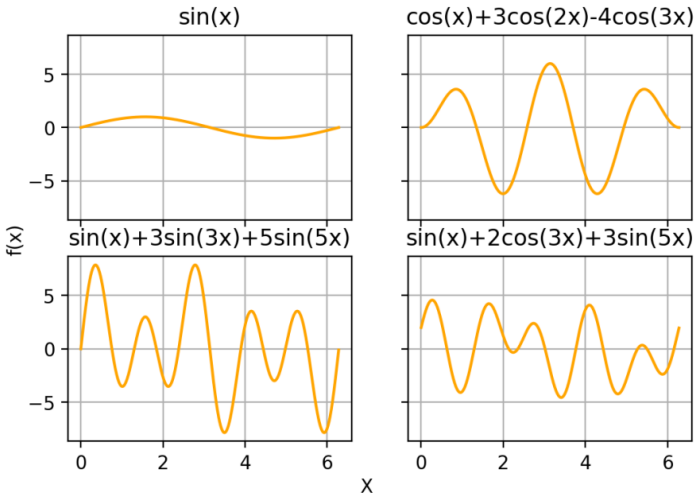

Figure 3: *Plots of the analytic functions*

These two figures showing the Fourier series and the analytic functions are practically indistinguishable, showing just how effective the program made was. This was however, quite easy to achieve with these 4 functions as they are all all-ready sinusoidal functions with discrete frequencies, meaning the Fourier series calculated are the exact functions.

As well as the Fourier series plots of the Fourier series coefficients were also made. Shown below are the plots of the coefficients for the above 4 functions 2,3.

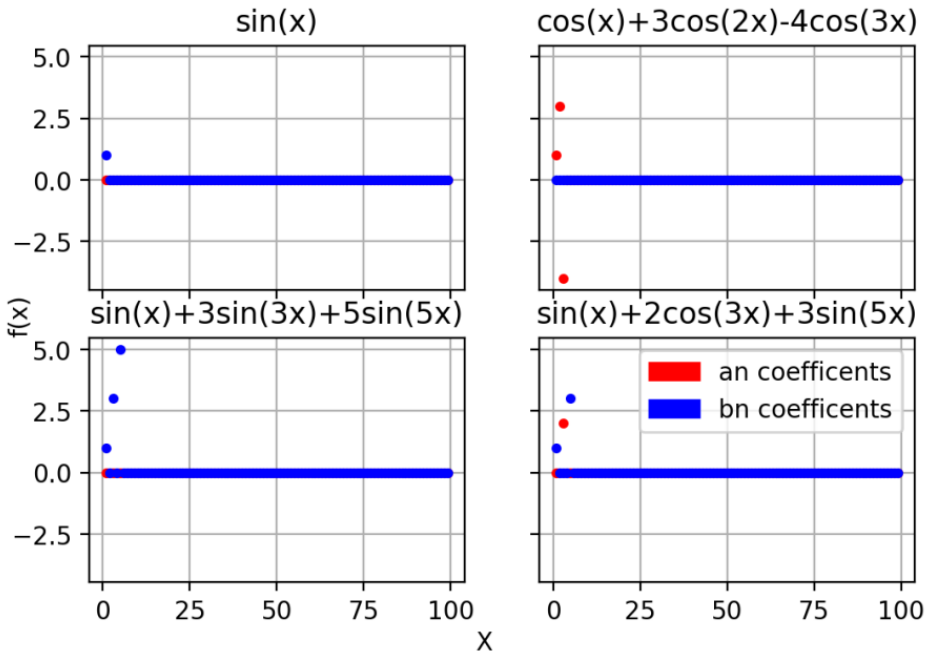

Figure 4: *Plot of Fourier coefficients for the shown functions*

These coefficients confirm that the Fourier series being calculated are exactly the same as the original functions. This can be seen from the fact that each each frequency found in f(x) is matched in the coefficient plot by the corresponding dot at the value of the magnitude of its coefficient. In the above plot, blue points are the bn terms meaning the correspond to the coefficient of sin terms, where as red points are for cos terms.

3.3 Exercise 2: Fourier series of square and rectangular waves

The Fourier transform that was used in the above section did not need to be modified to calculate the Fourier series of the square wave function. However, it was clear from a few runs of the program that in order to achieve an accurate series the number of samples of the function taken (n) and the number of coefficients calculated (k_{max}) needed to be high. Thus k=10,20,100,200 and n=500 were used. For the square wave this resulted in the following plotted Fourier series.

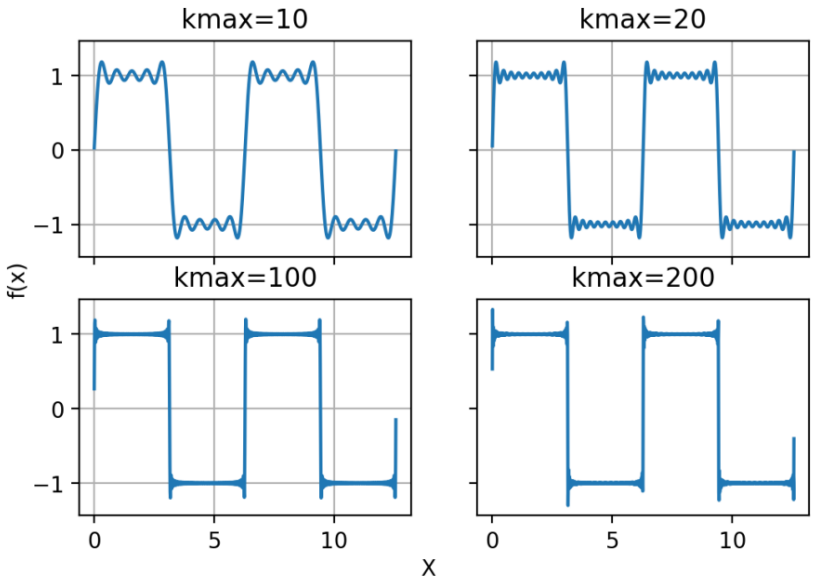

Figure 5: *Square wave Fourier series for different* k_{max}

When overlay-ed with the actual square wave function it looks like this:

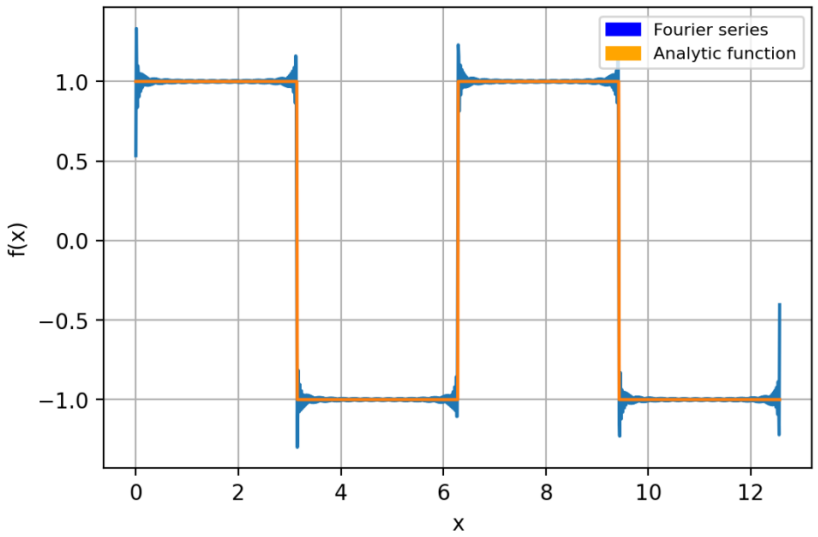

Figure 6: *Analytic and Fourier series comparison for* $k_{max} = 200$

This shows that for most the function the Fourier series is an accurate representation but, around the points of first derivative discontinuity (i.e. the corners), the Fourier series is not so good at emulating the sharp turn. This is expected due to the nature of sin and cos being smooth and continuous.

Graphs were also made of the coefficients a_n and b_n.It is expected from the analytical solution that all the coefficients a_n should be 0 as well as all the even b_n coefficients. However, the odd b_n should have values of $4/n\pi$. Thus to show this the coefficients were plotted over the curve $4/n\pi$

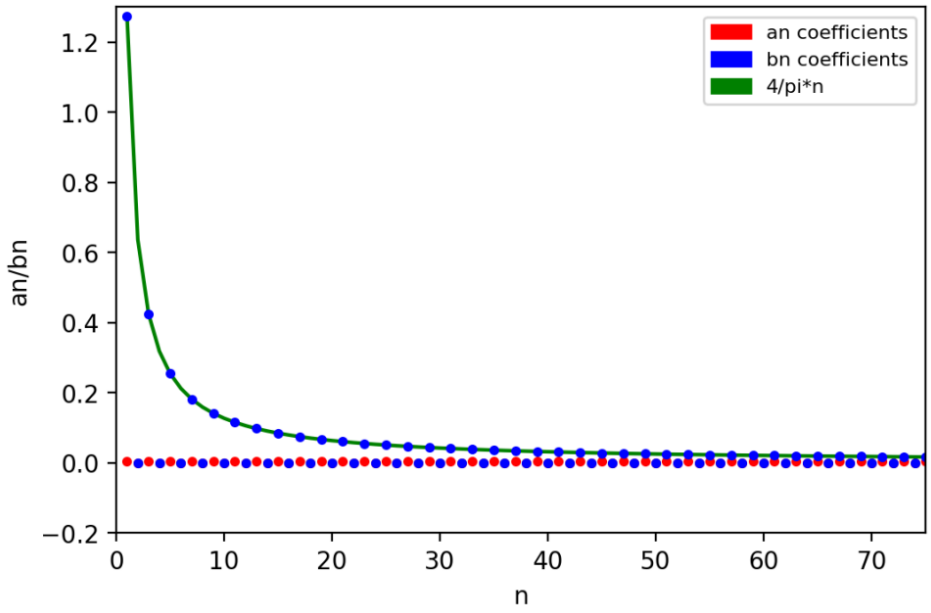

Figure 7: *Fourier series coefficients for square wave*

These points clearly match up well with what was expected from the analytic solution.

The Fourier series for the rectangular wave function was calculated in a similar manner. These also followed the analytical function quite well at all points except the discontinuous corners.

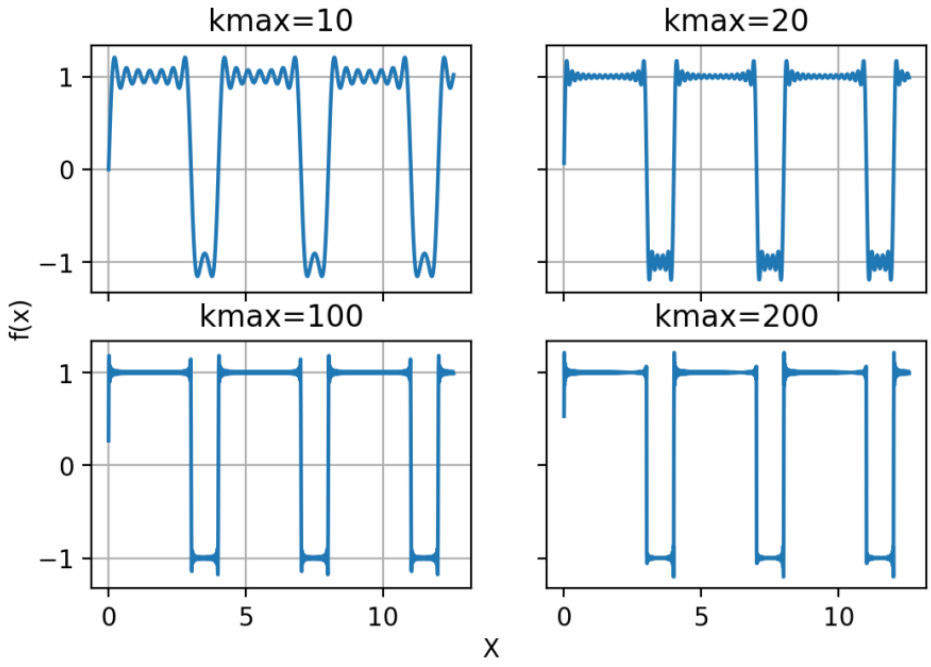

Figure 8: *Rect wave Fourier series for different* k_{max}

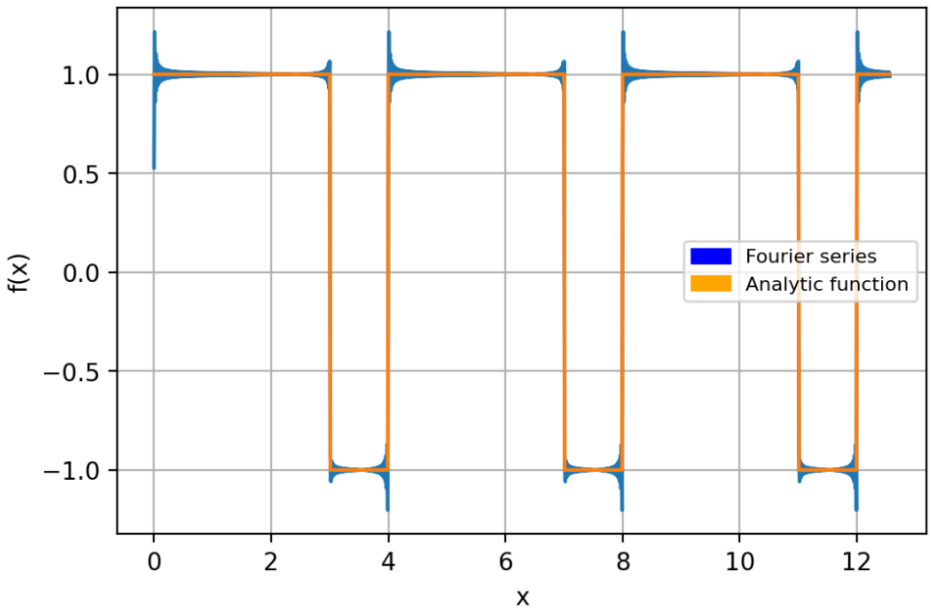

Figure 9: *Analytic and Fourier series comparison for* $k_{max} = 200$

3.4 Discrete Fourier transform

The first graph plotted shows how the function $f(x) = \sin(0.45\pi x)$ was sampled. Each triangle on the graph indicates a point where the value of f(x) was used as part of the DFT calculation.

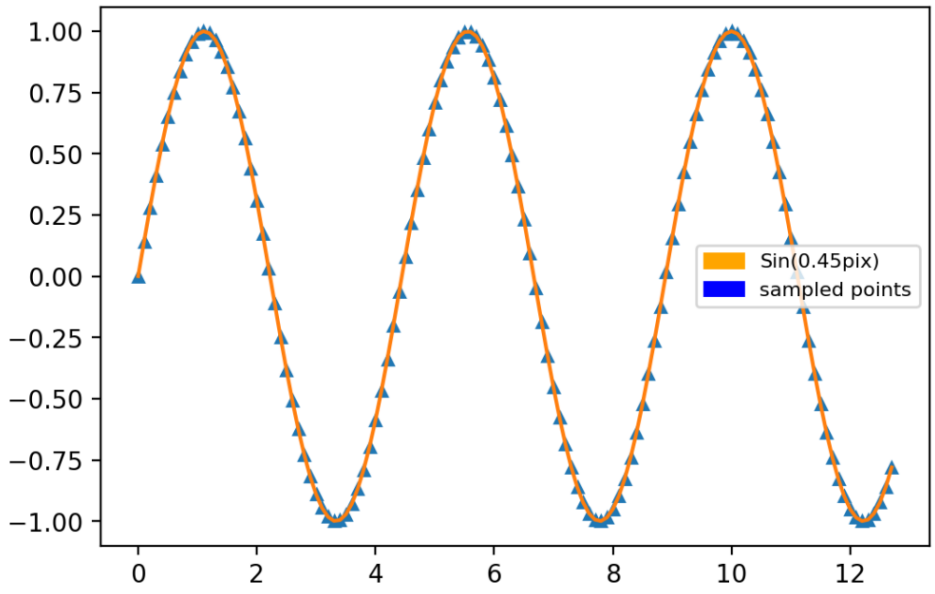

Figure 10: *Sampled points for DFT on the function* $f(x) = sin(0.45\pi x)$

Then the real and imaginary Fourier coefficients series F_i and F_r were plotted against n, using N=128 and h=0.1.

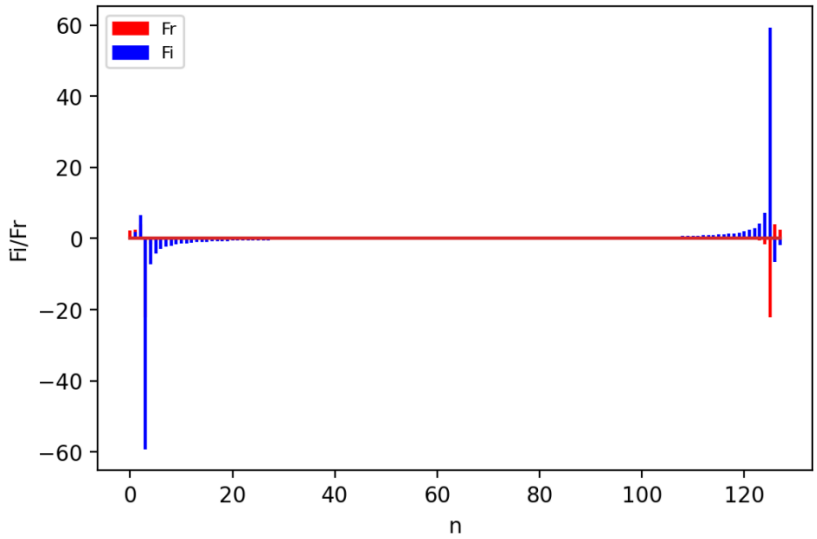

Figure 11: *Real and imaginary series of the transform of* $f(x) = sin(0.45\pi x)$*, for h=0.1 and N=128*

It was found that it is ideal to pick a value of h such that the fundamental frequency (ω_1) of the DFT was the same as the frequency of the original function. $\omega_1 = 2\pi/Nh$ and $\omega = 0.45\pi$, thus h was chosen to be $h = 2/0.45N$. This changed the real and imaginary Fourier coefficients series, which are shown in the plot below.

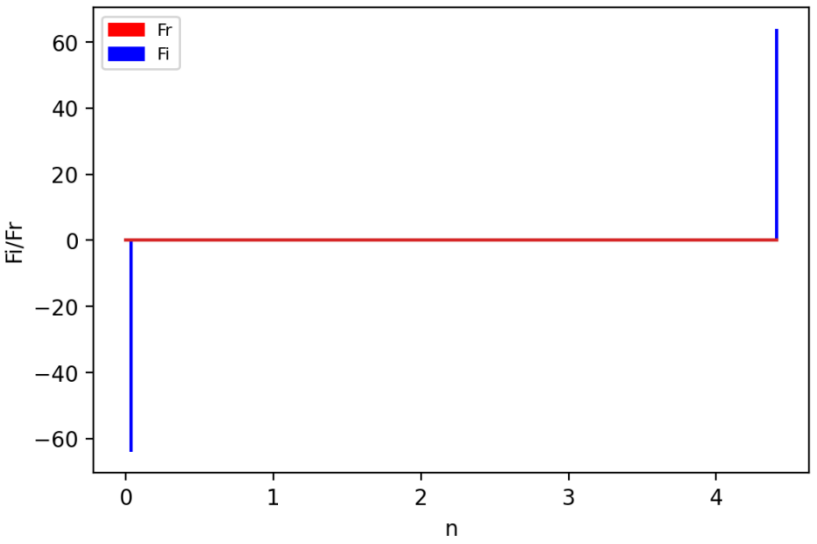

Figure 12: *Real and imaginary series of the transform of* $f(x) = sin(0.45\pi x)$ *for* $h = 2/0.45N$ *and N=128*

After this the components of the back Fourier transform were calculated, which outputted the reconstructed function. This is shown below compared to the actual original function.

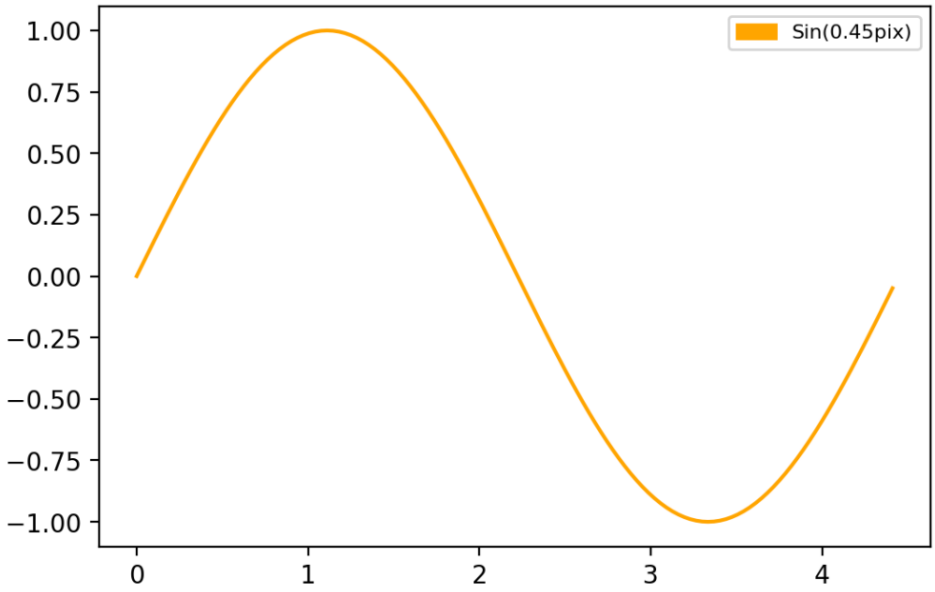

Figure 13: *original function*

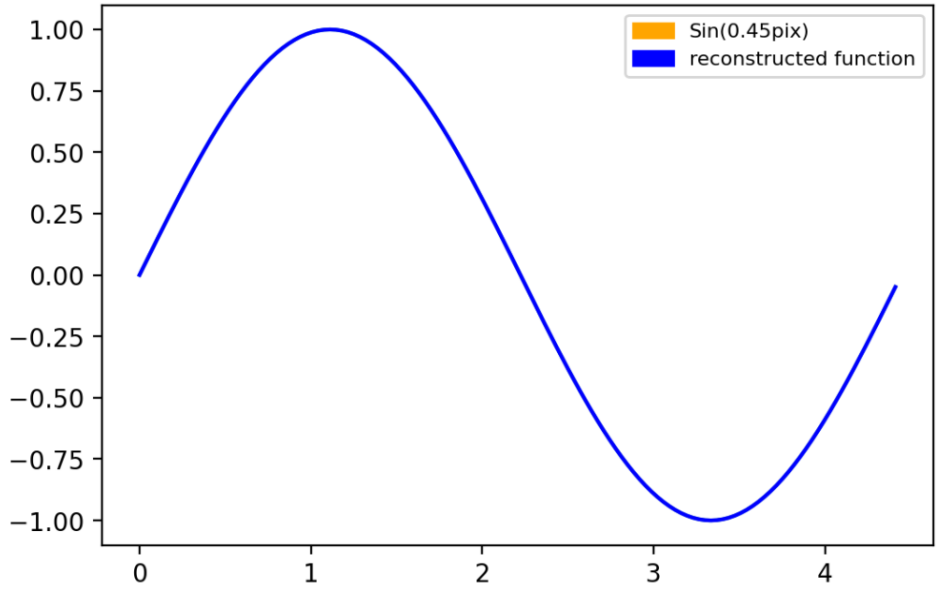

Figure 14: *Reconstructed function and original function*

Next the new function $f(x) = \cos(6\pi x)$ was looked at. Shown below are the Fourier coefficients series, power spectrum's and reconstructed functions, for different values of h. These plots show how some values of h are better than others at approximating the function.

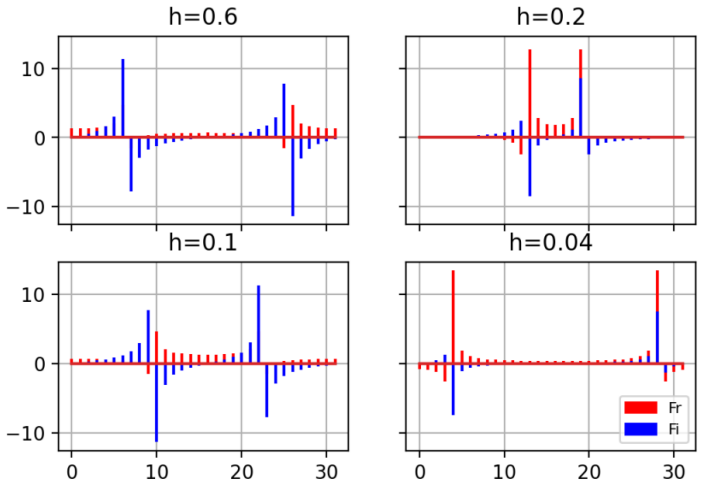

Figure 15: *Fourier coefficients series of* $f(x) = cos(6\pi x)$ *for different values of* h

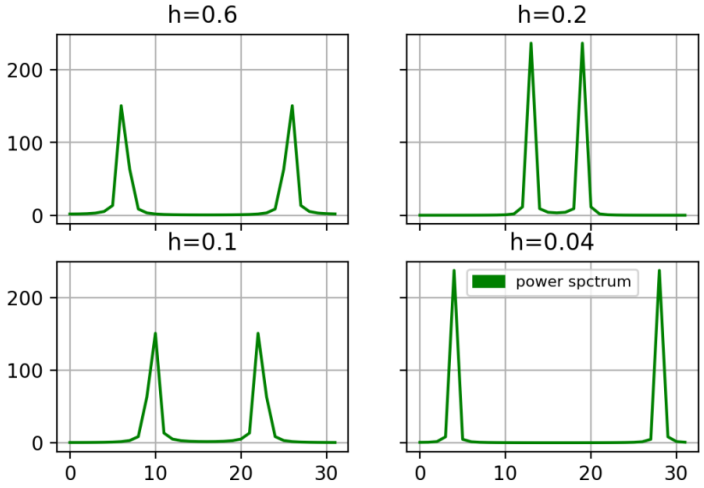

Figure 16: *Power spectrum for different values of* h

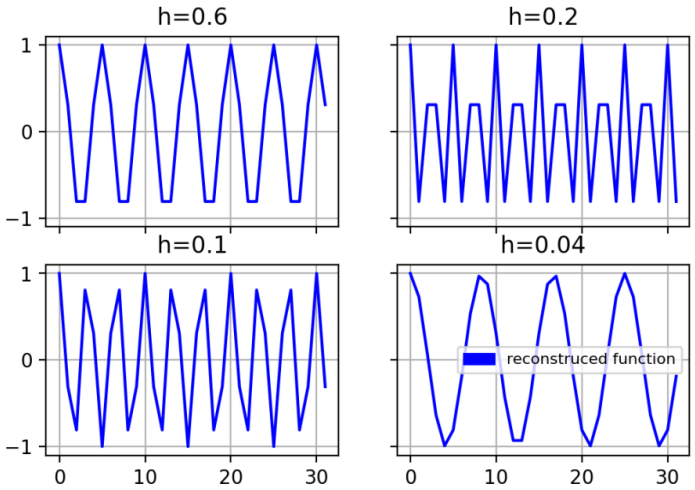

Figure 17: *Reconstructed function for different values of* h

4 Conclusion

In conclusion a thorough investigation was performed into Fourier analysis using python. An effective method of integral approximation was made using Simpson's rule, which was found to be accurate and computationally efficient. Then using this program a function was made to calculate the Fourier coefficients for any given periodic function. This was then shown to be fully functional the Fourier series of sinusoidal functions were exactly the same as the original functions. This was then repeated for square and rectangular functions, to not quite the same level of success due to the discontinuous nature of the "corners" of the functions. Lastly the discrete Fourier transform was used to sample a given function, periodic or not, and represent now as a series of sin and cosine terms , that now had non-integer frequencies. This was proved effective as the back transform perfectly plotted the original function. It was also investigated how changing the sampling size changed the outputted function.