Laboratory 4: Fourier Analysis using Python

Jordan Ahern 21363697

December 2, 2022

1 Introduction

This report investigated a wide variety of useful techniques in Fourier analysis. These tools included Fourier series, Fourier transforms, and discrete Fourier transforms (DFT). These tools have a large range of uses in the modern world, particularly in the analysis of wave signals. Frequently these techniques are employed as fast Fourier transforms [1] which extend the tools explored in this report. Analytical Fourier analysis relies on the use of integration of functions, which was emulated using the Simpson method in Python. Simpson's rule is a numerical method for computing the integral of a given function. This method approximates the area under a curve by summing the areas under a sequence of parabolic curves in even steps across an interval. It is similar to, but generally more accurate than, the trapezoid rule, theoretically being able to give exact results for polynomials up to cubic order [2]. The Fourier series is practical for the analysis of waves with finite, definite, periods. It can be used to exactly express a function in terms of an infinite series of sine and cosine terms with known coefficients, although an excellent approximation is achievable with a finite sum. This was shown to be particularly useful in finding an expression for functions which must otherwise be expressed piece-wise, such as a square wave. Where a function is non-periodic, or the period is unknown, it must be instead analysed using Fourier transformations. Wheres the Fourier series summed discrete terms, the Fourier transform integrates over a continuous frequency. Such a complex integration would be unsuitable for rapid computing so numerical methods are used to create a DFT. In DFT, the Fourier transform is numerically approximated by summation of the product of the function and a complex exponential evaluated at constant intervals over a wide range. The original function can then be recovered by a similar, but reversed, process. The accuracy of this method was examined. The investigation was completed using Python 3, with the Numpy and Matplotlib libraries within the Anaconda Spyder environment.

2 Methodology

2.1 Simpson's Rule

Where a function was to be integrated, the numerical method of Simpson's Rule was used. This method approximates the area under a curve by splitting the total interval into an even number of equal segments. For every two segments the area under a parabola, which meets the curve at the three boundaries of the two segments, is calculated analytically and all the areas are summed. This is expressed according to

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{\frac{n}{2}-1} f\left(x_{2j}\right) + 4\sum_{j=1}^{\frac{n}{2}} f\left(x_{2j-1}\right) + f(x_n)\right] \tag{2.1}$$

where $h = (b - a)/3n$, $x_0 = a$, $x_n = b$, and $x_j = a + j \cdot h$ for $j = 0, 1, \ldots, n$.
This method, Equation 2.1, was implemented and tested against the analytically calculated solution for the integration of e^x from 0 to 1, and found to be an excellent approximation at just $n = 8$.

2.2 Fourier Series

The Fourier series expresses functions according to

$$f(t) = a_0 + \sum_{k=1}^{\infty}\left[a_k \cos\left(k\omega t\right) + b_k \sin\left(k\omega t\right)\right] \tag{2.2}$$

where $\omega = \frac{2\pi}{T}$ is the fundamental frequency and T is the period. As an infinite series cannot be numerically computed, Equation 2.2 was instead an approximation summed to a defined maximum "k". The coefficients are defined as

$$a_0 = \frac{1}{T}\int_0^T f(t)\,dt \tag{2.3}$$

$$a_k = \frac{2}{T}\int_0^T f(t)\cos(k\omega t)\,dt \tag{2.4}$$

$$b_k = \frac{2}{T}\int_0^T f(t)\sin(k\omega t)\,dt \tag{2.5}$$

To implement each of Equations 2.3, 2.4, and 2.5, the Simpsons method, Equation 2.1, was used. These implementations were then tested by plotting the Fourier series, Equation 2.2, and comparing to the true plots of a number of known functions. A square function was then implemented using the SciPy Signal library. This function is most easily described in piece-wise terms with a period of 2π making it a good candidate to be analysed using a Fourier series. The numerical methods above were used to plot a square function, with the relationship between accuracy and number of terms observed.

2.3 Discrete Fourier Transform

The discrete Fourier transform is a numerical approach to computing the Fourier transform, which is theoretically a continuous integration, by summing finite terms. This involves breaking the Fourier transformation into real and imaginary parts. These parts are computed by

$$F_{n,real} = \sum_{m=0}^{N-1} f_m \cos\left(\frac{2\pi mn}{N}\right) \tag{2.6}$$

$$F_{n,imag} = -\sum_{m=0}^{N-1} f_m \sin\left(\frac{2\pi mn}{N}\right) \tag{2.7}$$

with N the number of samples, n the sample frequency.
The reverse transform is defined as

$$f_m = \frac{1}{N}\sum_{n=0}^{N-1} F_n \left(\cos\left(\frac{2\pi mn}{N}\right) + i\sin\left(\frac{2\pi mn}{N}\right)\right) \tag{2.8}$$

with $F_n = (F_{n,real} + iF_{n,imag})$ a complex number.

3 Results

3.1 Exercise 1: Simpson's Rule and Fourier Series

The implementation of the Simpson's rule was tested by calculating $\int_0^1 e^x\,dx$. This is trivially calculated to be $[e^1 - 1] \approx 1.7182818$. The Simpson's method numerically calculated, at n = 8 steps, $\int_0^1 e^x\,dx \approx 1.718284$, which is accurate up to 10^{-6} order, an excellent approximation.
The Fourier series for a number of 2π periodic functions[1] was then calculated numerically and plotted against the exact curve, as discrete points at each time.
An excellent Fourier series fit was seen for f(t) = sin(ωt), Figure 3.1, all plotted points are overlapping the plotted function, the blue line. This was helped by the simplicity of the function. Inspection of the printed coefficients gave $b_1 = 1$, with all other coefficients effectively zero. This was exactly as expected as to recreate the function only one term was needed, sin(ωt) itself which corresponds to b_1 as in Equation 2.2.
Similar inspection of the coefficients for Figure 3.2, further demonstrated the inner workings of the Fourier series method. Only three coefficients were significant $a_1 = 1$, $a_2 = 3$, and $a_3 = -4$. Since this function was clearly made up of only even terms, only a_k coefficients would be expected, specifically three such coefficients corresponding to the true coefficients. The accuracy of this method was further supported by the graph produced with all Fourier points on the curve.

[1]Giving $\omega = 1$ for all graphs in this exercise.

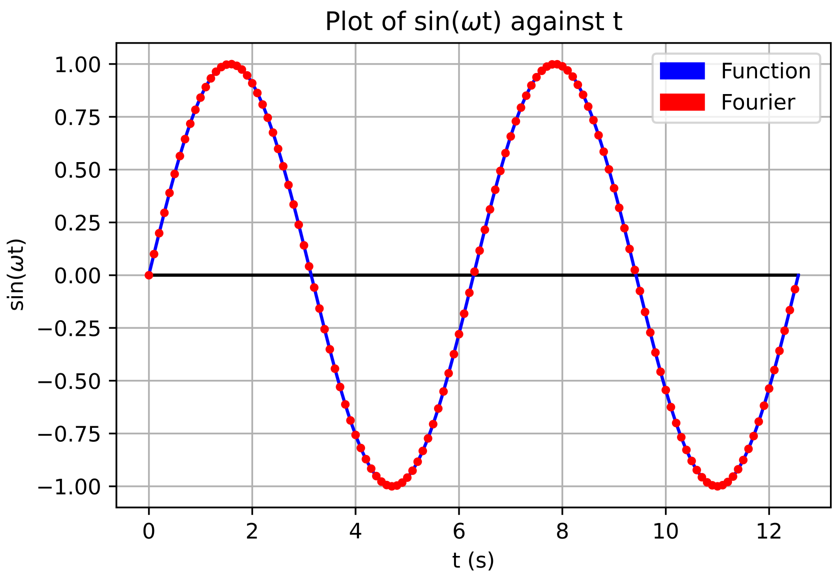

Figure 3.1: Graph of sin(ωt), n=200, k=20, Period=2π, Range=4π s, Step size = 0.1 s

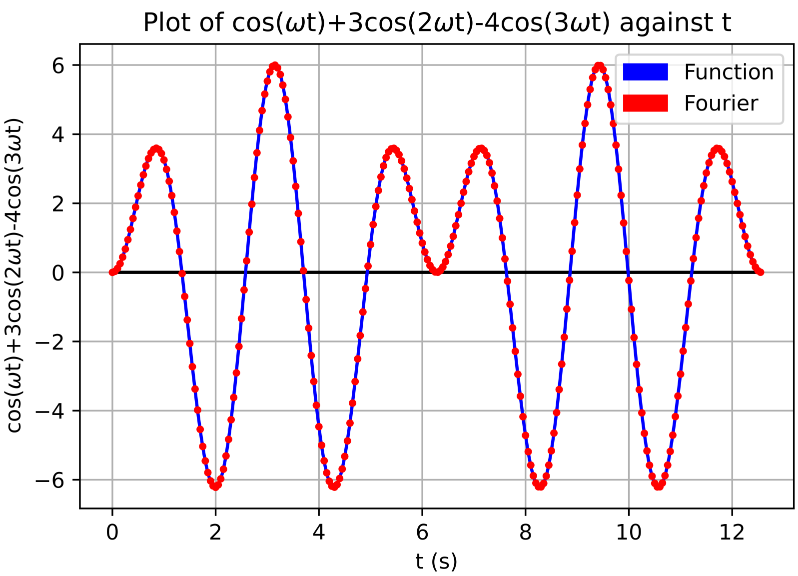

Figure 3.2: Graph of cos(ωt)+3cos(2ωt)-4cos(3ωt), n=200, k=20, Period=2π, Range=4π s, Step size = 0.05 s

For the function f(t) = sin(ωt) + 3sin(3ωt) + 5sin(5ωt), the three significant coefficients were $b_1 = 1$, $b_3 = 3$, and $b_5 = 5$. Which was predictable as the function was purely odd. Also, the function f(t) = sin(ωt) + 2cos(3ωt) + 3sin(5ωt), gave three significant coefficients $b_1 = 1$, $b_5 = 3$, and $a_3 = 2$. The mix of coefficients occurred, of course, because this function was a mixture of two odd and one even terms. Additionally, it demonstrated that the number of the coefficient does not necessarily correspond to the value of the coefficient. Figures 3.3 and 3.4 each further demonstrated the accuracy of this numerical implementation of the Fourier series, even as functions became more complicated, with all points plotted on the correct line.

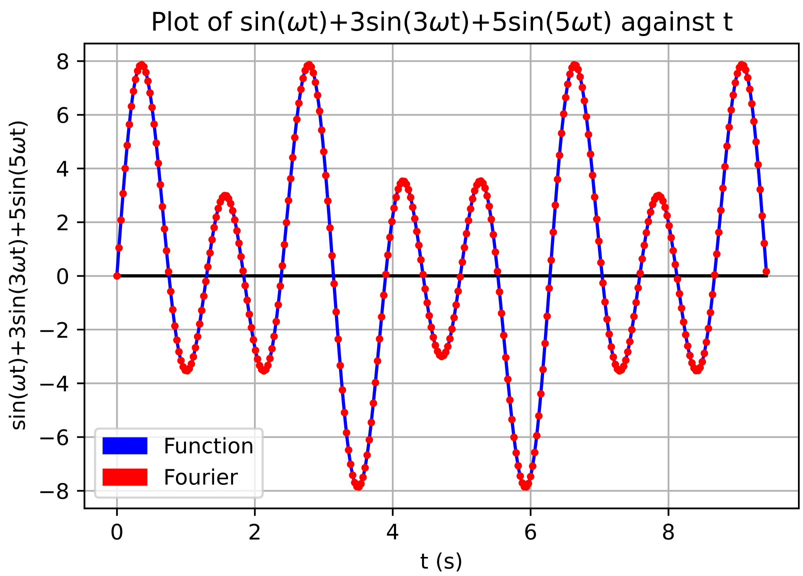

Figure 3.3: Graph of sin(ωt)+3sin(3ωt)+5sin(5ωt), n=200, k=20, Period=2π, Range=3π s, Step size = 0.03 s

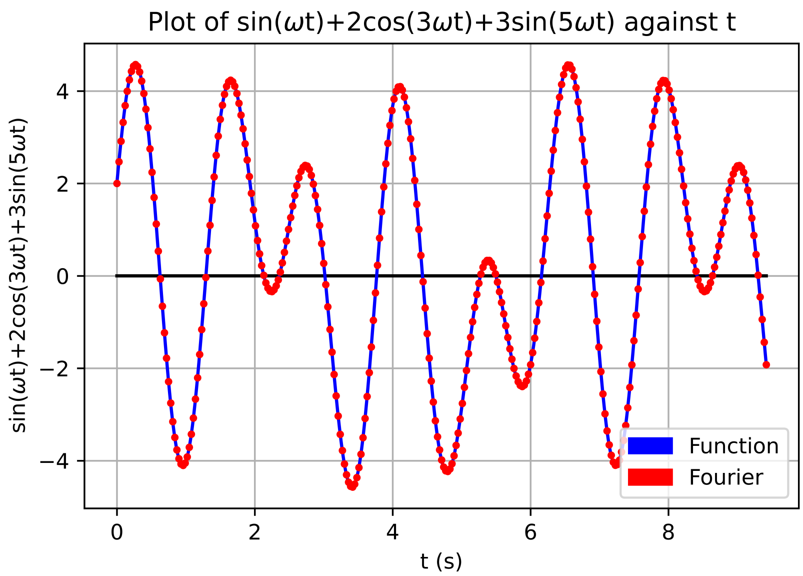

Figure 3.4: Graph of sin(ωt)+2cos(3ωt)+3sin(5ωt), n=200, k=20, Period=2π, Range=3π s, Step size = 0.03 s

3.2 Exercise 2: Fourier Series of Step Waves

A square wave function was defined using the SciPy Signal library. This function was then analysed using the Fourier series and Simpson's rules from Exercise 1.

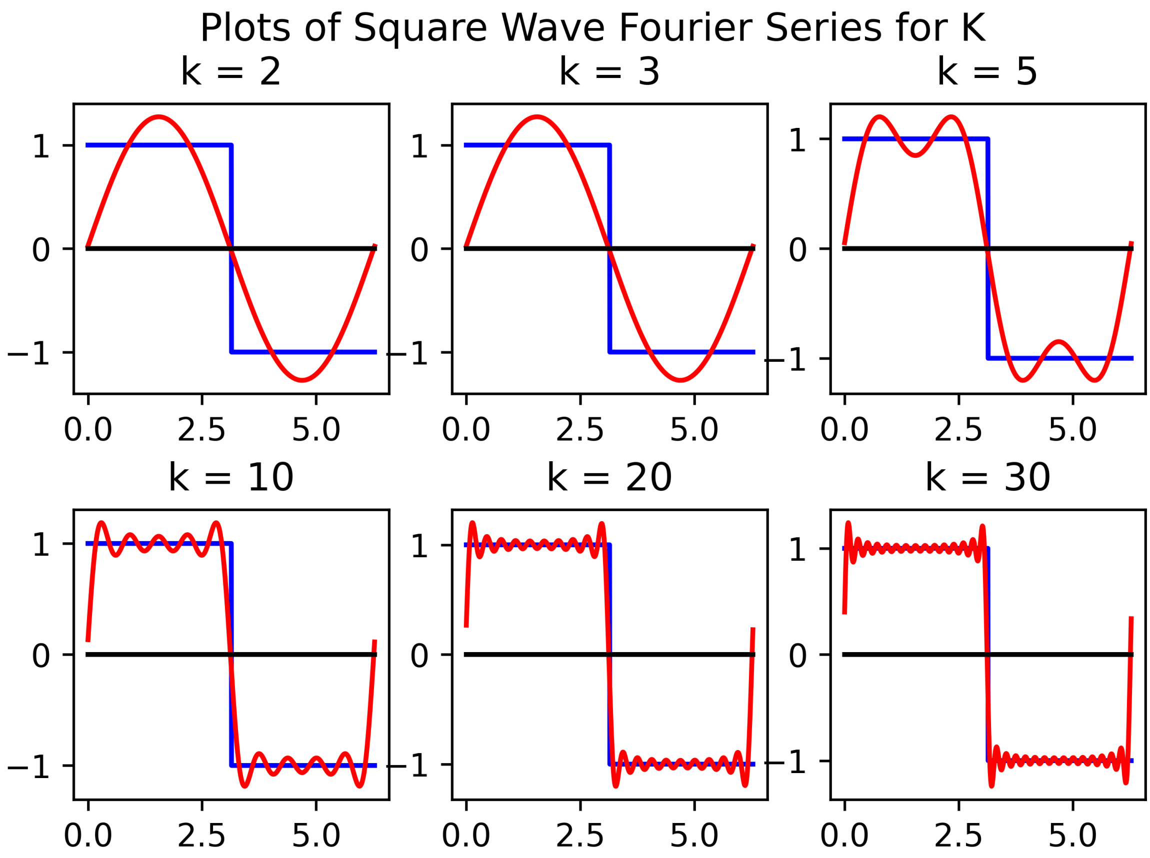

Figure 3.5: Graph of square wave reconstructions for a range of k, n=100, Period=2π, Range=2π, Step size = 0.01

In Figure 3.5, the blue line was the directly plotted function and the red was the plotted Fourier series. A clear progression was seen in matching a Fourier series to the correct wave, starting from a single sine wave. By k = 30, a square wave shape could be said to be very noted, although the matching was in no way perfect. There was clearly a great difficulty in smoothing the wave form down to a straight line and such a task would likely require a vast number of terms.

Figure 3.6, explored greatly increasing the number of terms used. Vertical lines were seen to be relatively well matched but the overall graph is still imperfect. The horizontal lines were still fluctuating from the amplitude of ± 1. The overshoots were also uneven across the entire function. These results suggest some need to improve the numerical methods in use to analyse a step function accurately.

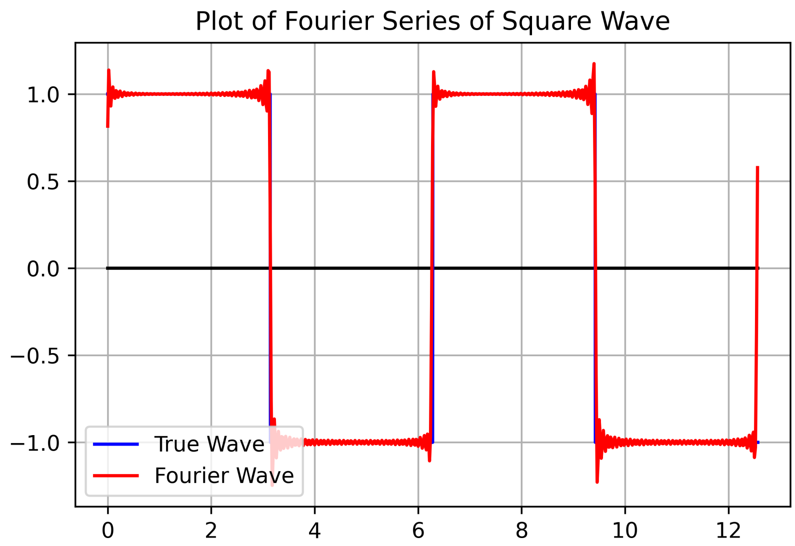

Figure 3.6: Graph of a square wave reconstruction, n=200, k=100, Period=2π, Range=4π, Step size = 0.005

Further difficulties for the numerical methods used were seen by directly inspecting the calculated coefficients. Analytically, only odd, b_n, terms would be expected but the method printed small but not insignificant even, a_n, terms.

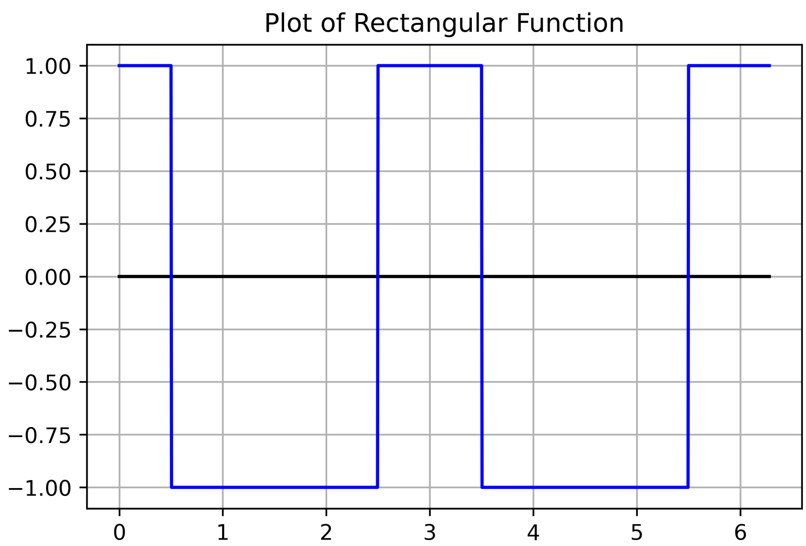

Figure 3.7: Graph of rectangular wave, Range=2π, Step size = 0.005

A rectangular wave was made using a number of if statements in a function and was plotted in Figure 3.7. This new function was then analysed using the same techniques.

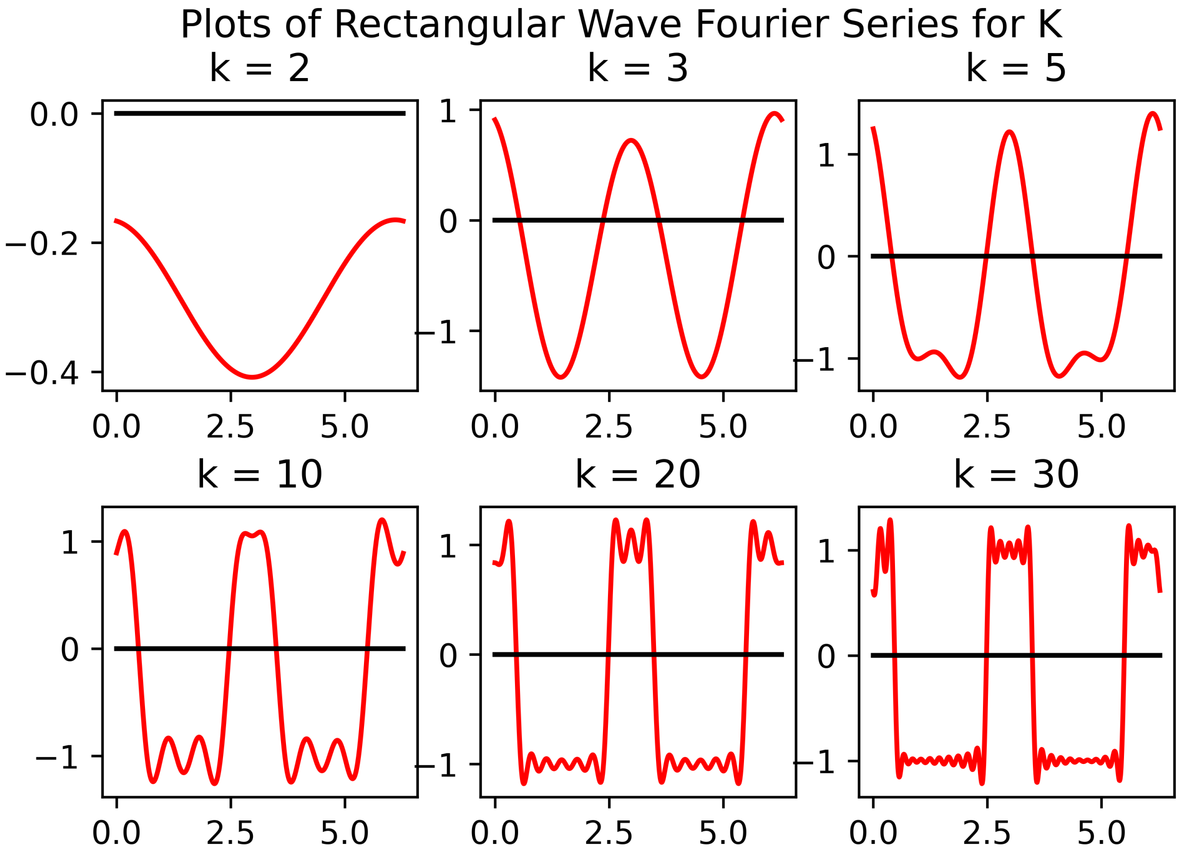

Figure 3.8: Graph of square wave reconstructions for a range of k, Period=2π, Range=2π, Step size = 0.01

The rectangular wave was seen to be more complicated to analyse in Figure 3.8. The early plots were biased towards the $x < 0$ axis, this was seen to level out from k = 5 onward. As with the square wave plots, Figure 3.5, the Fourier series continued to have difficulty in creating straight horizontal lines. The Fourier series was able to accurately plot the proportions in the distances between peaks and troughs.

As with the square wave, the printed Fourier series coefficients were not perfectly in line with the expected analytical coefficients, but were broadly in line in the aggregate. These results could be expected by observing the reconstructed waves, which while close to the expected graphs, were not smooth or perfect.

Figure 3.9, demonstrated the reconstruction of a rectangular wave for k = 100. This reconstruction was similar to the equivalent for the square wave, Figure 3.6. The fit continues to improve with increased k but large jumps still remain and a perfect fit does not appear to be in sight with this implementation.

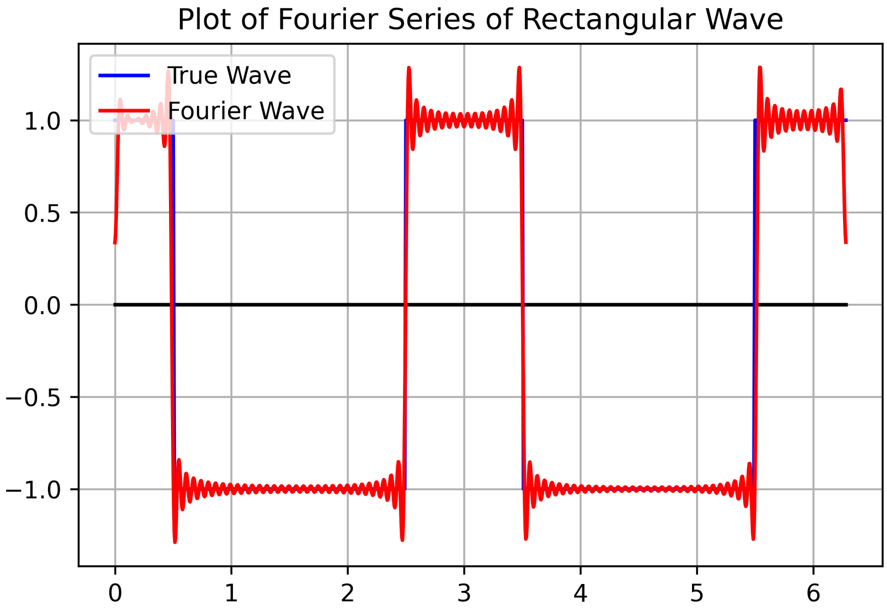

Figure 3.9: Graph of rectangular wave reconstruction, n=200, k=100, Period=2π, Range=2π, Step size = 0.005

3.3 Exercise 3: Discrete Fourier Transforms

The Fourier transform analysed given waves by sampling the wave at a discrete number of points, shown in Figure 3.10.

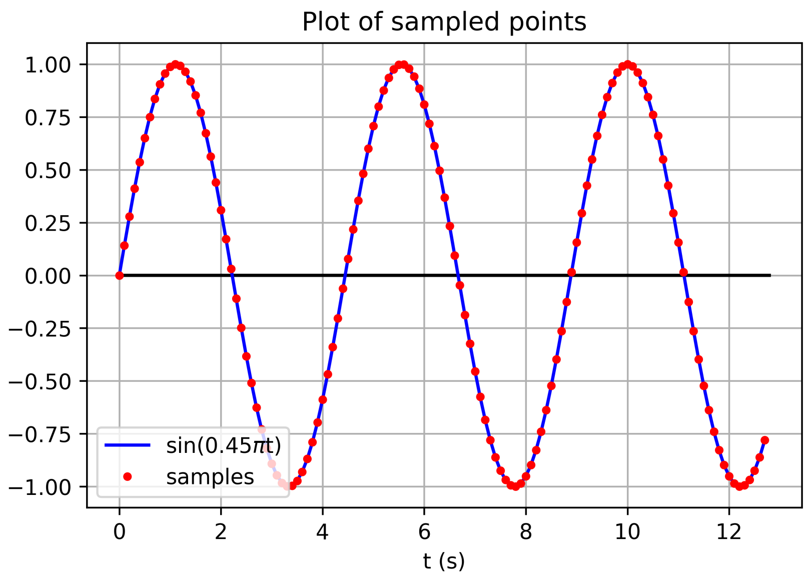

Figure 3.10: Graph of sampled points on f(t) = $\sin(0.45\pi t)$, N=128, h=0.1, t=12.8 s

These samples occur at intervals of 'h', which gives a sampling rate of $v_s = \frac{1}{h}$. In Figure 3.10, $h = 0.1$ therefore $v_s = 10$ samples per unit time. The fundamental frequency for a wave was taken to be $\omega_1 = \frac{2\pi}{N \cdot h}$, which gave the effective frequency at $h = 0.1$ to be $\frac{2\pi}{12.8} \approx 0.49$ which was shorter than the known frequency of 0.45π.

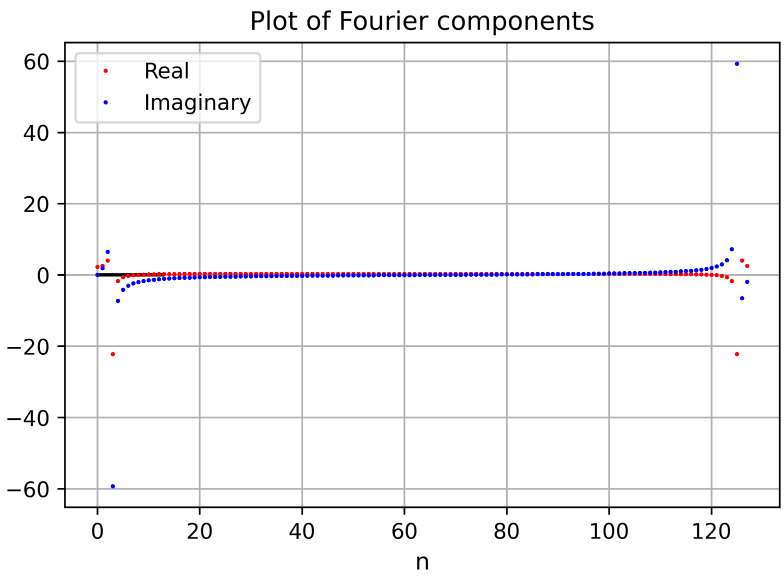

Figure 3.11: Graph of Fourier coefficients of f(t) = sin(0.45πt) against n, N=128, h=0.1

The graph of real and imaginary Fourier coefficients of the function plotted a relatively large number of significant points, Figure 3.11. These results included two very large imaginary coefficients which were mirrored over approximately 128 steps, which showed that the DFT takes $N \cdot h$ to be the period of the function. The other significant, and smaller, points were not expected as the function is made up of one sine function, where multiple points suggested a composite function. The largest significant points were imaginary which corresponds to the sine function as expected.

While these components were not ideal the DFT implementation was verifiable through a back Fourier transform according to Equation 2.8. The back transform was plotted in Figure 3.12 and accurately recreated the graph according to Figure 3.10. It was encouraging to see that the full graph could be recreated by only taking a finite number of samples, and this supported to use of a DFT for analysing waves.

Since the true fundamental frequency of the wave is known to be 0.45π the optimal sampling interval is $h = \frac{2\pi}{128 \cdot 0.45\pi} \approx 0.0347$ for N = 128. This would give a period of of $T \approx 4.44$ as expected. The wave was reanalysed for these optimal conditions.

At the optimal sampling conditions for sampling the Fourier coefficients came into line with theoretical expectations. Only two non-zero components occurred and these were at n=1 and n=128 as expected. No other interference occurred, Figure 3.13.

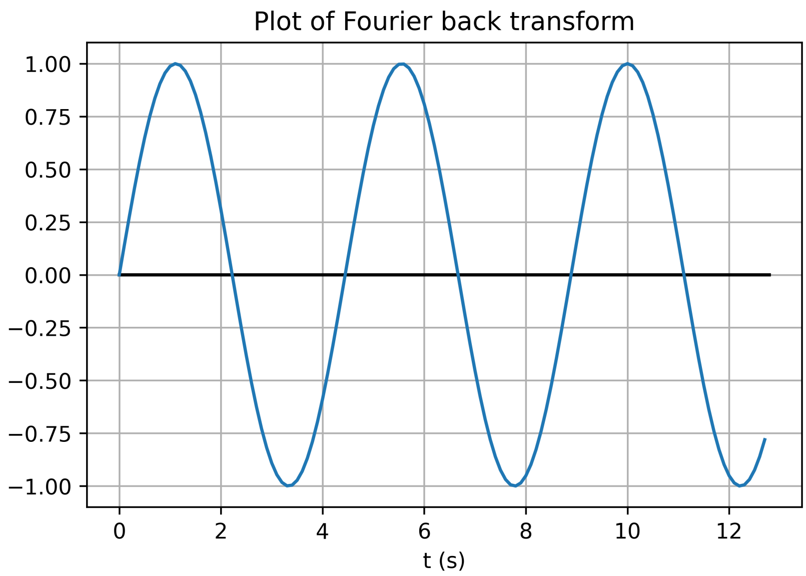

Figure 3.12: Graph of Fourier back transform of f(t) = $\sin(0.45\pi t)$, N=128, h=0.1

The back transform at optimal sampling, Figure 3.14, the DFT method is further supported as the expected wave over a single period is plotted.

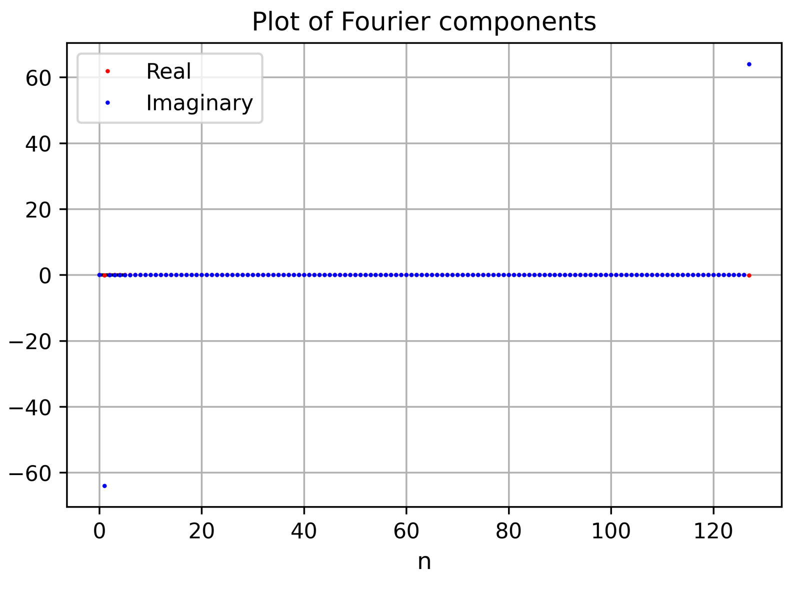

Figure 3.13: Graph of Fourier coefficients of f(t) = $\sin(0.45\pi t)$ against n, N=128, h=0.0347

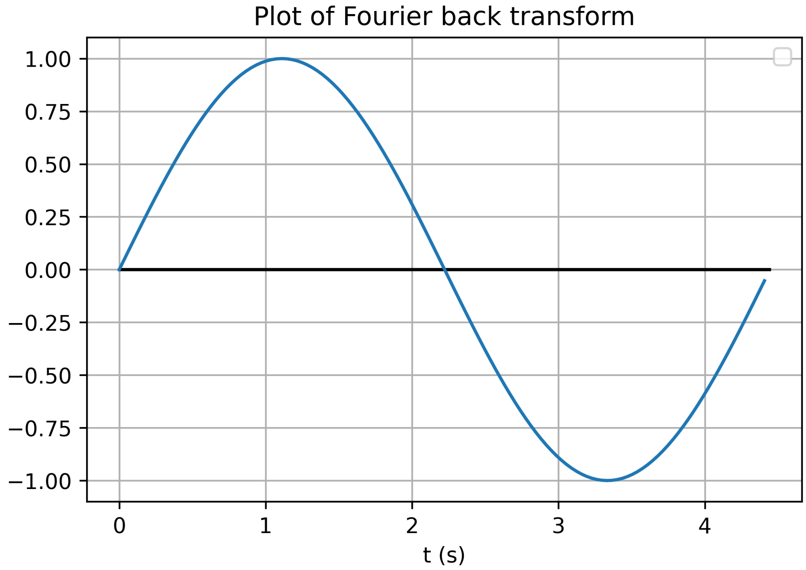

Figure 3.14: Graph of Fourier back transform of f(t) = sin(0.45πt), N=128, h=0.0347

The Nyquist frequency of a signal can be calculated to be

$$v_n = \frac{1}{2h} - \frac{1}{N \cdot h} \tag{3.1}$$

with N and h defined as previously in this report. If this value is less than the true fundamental frequency of one of the components of a signal then the sampling rate will not be adequate to analyse the frequency using DFT.

Figure 3.15 plotted the Fourier transform coefficients of f(t) = cos(6πt) at N = 32, for a range of specified 'h' values. Using Equation 3.1, the Nyquist frequency for all $h = 0.6, 0.2, 0.1, 0.04$ was less than the fundamental frequency of 6π. As a result the graphed coefficients are very messy and the signal did not appear to have been well analysed. The sample size $h = 0.01$ had a Nyquist frequency greater than 6π and the coefficients were much more ordered. The optimal sample length was found to be $h = \frac{1}{96}$ at this sample size a very good analysis was seen. Only two significant coefficients were found with both being real, as expected for a cosine function.

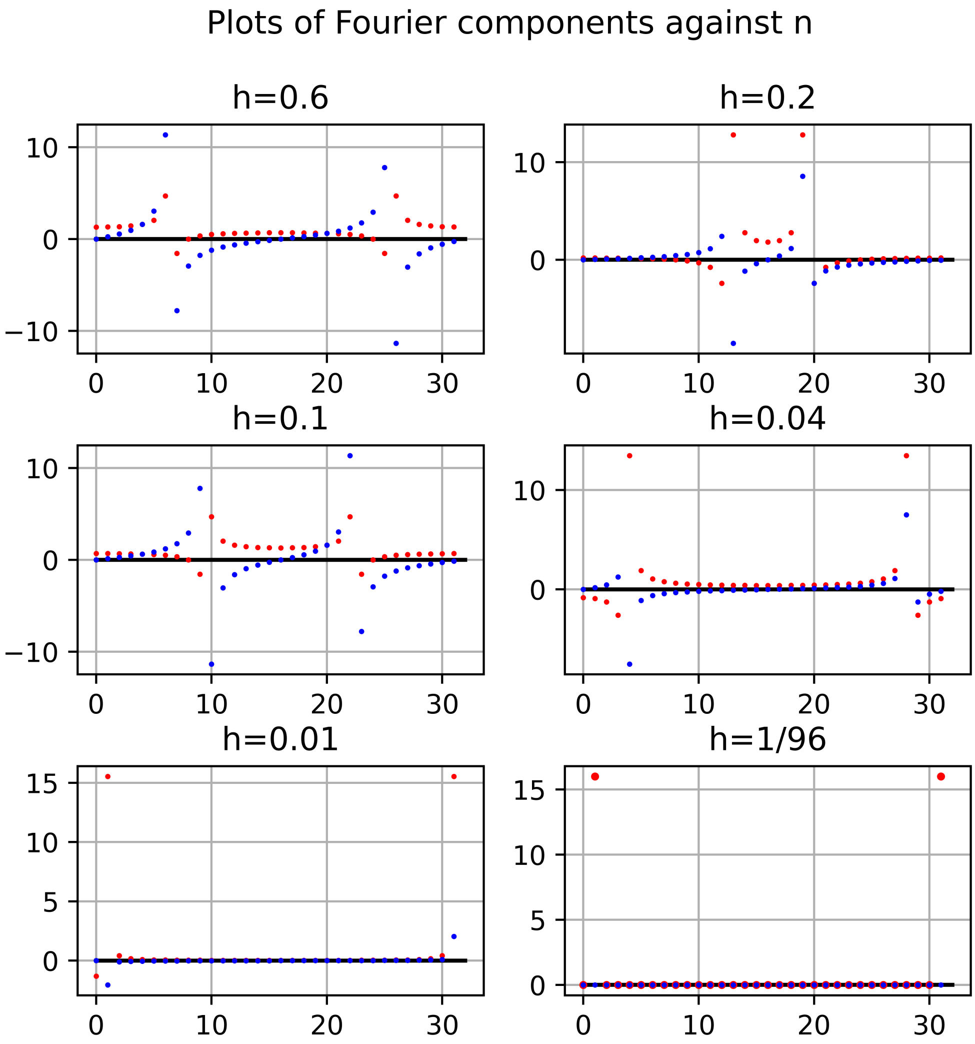

Figure 3.15: Graphs of Fourier components over n for various h values, N=32, f(t) = $\cos(6\pi t)$, red is real, blue is imaginary.

4 Conclusion

This report explored the analysis of wave signals using numerical Fourier analysis. The main methods investigated were Simpson's rule, Fourier series, and discrete Fourier transformations. Simpson's rule was seen to be an accurate method of numerically integrating functions. With the help of Simpson's rule the Fourier series of a number of trigonometric functions with known periods were analysed. This method was then applied towards analysing piece-wise step functions, both square and rectangular. The Fourier series was not able to perfectly recreate the step functions for the number of terms computed but the general shape of both waves was formed. The computation of a Fourier transform through the sampling of a finite number of discrete points, discrete Fourier transformation, was also investigated. This method was seen to be able to accurately reform a wave from only a relatively small number of sampled points. The computation of the Fourier transform was also seen to be achievable only when the sampling rate was greater than the Nyquist frequency, Equation 3.1. Overall the numerical methods were seen to be very adequate for the analysis of wave signals.

5 Bibliography

References

[1] J. C. Schatzman, "Accuracy of the discrete fourier transform and the fast fourier transform", SIAM Journal on Scientific Computing **17**, 1150–1166 (1996).

[2] E. W. Weisstein, "Simpson's rule", `https://mathworld.wolfram.com/SimpsonsRule.html`, From MathWorld–A Wolfram Web Resource. (2004).

List of Figures