SHPC4001 - Assignment 5

Ayden S. McCann

March 2021

Code Listing at the end

Introduction

The purpose of this assignment was to implement the Fourier transform in various forms and analyse them, this included personally writing functions to perform these transforms as well as utilising the well known module SciPy and the scipy.fftpack module. Question 1 focuses on the numerical implementation of the standard Fourier series approximation of a periodic triangle wave and sawtooth wave. The number of terms as well as the x dependence on error scaling is explored before the Lanczos sigma factors were implemented to counteract Gibbs phenomenon. Question 2 focuses on the discrete Fourier transformation as well as its inverse, which is applied to a function for a number of discretisations to assess their accuracy. In question 3 the 1D wave-function of an electron localised in both position and momentum space is analysed through the application of the scipy.fftpack module in order to gain its distribution in both position and momentum space.

1 Question 1

1.1

A partial Fourier series (1) was implemented in order to approximate both the triangle and sawtooth waves given. These approximations are plotted below in Figures 1 and 2 for 5, 10 and 20 terms. The difference between the analytical function and these partial sums are shown in Figures 3 and 4.

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \, cos\left(\frac{2\pi kx}{L}\right) + b_k \, sin\left(\frac{2\pi kx}{L}\right) \right) \tag{1}$$

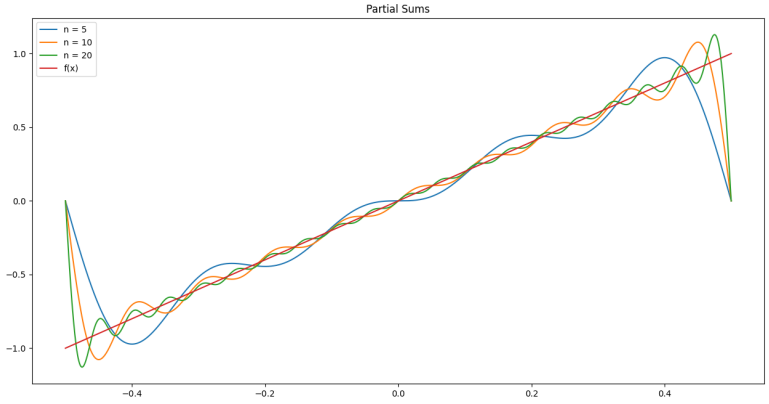

Figure 1: The partial Fourier series approximation to the sawtooth wave function for various values of n.

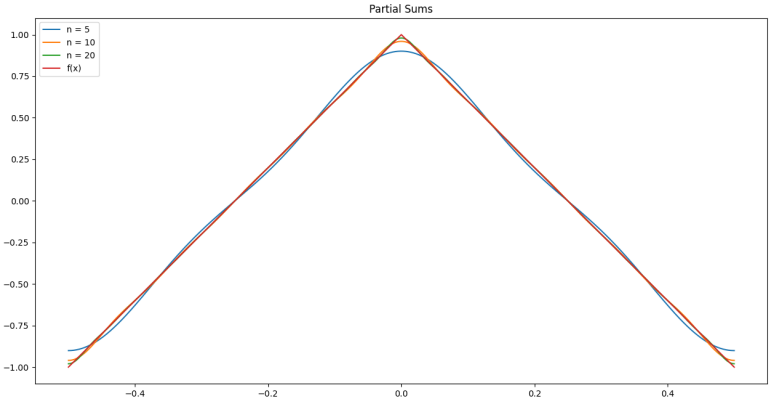

Figure 2: The partial Fourier series approximation to the triangle wave function for various values of n.

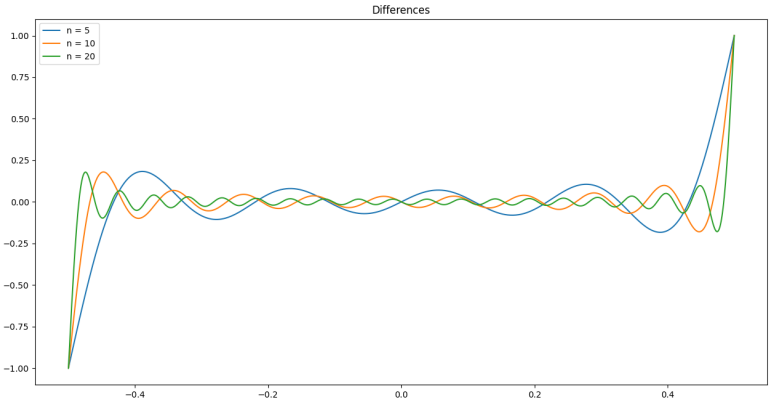

Figure 3: The differences between various partial Fourier series approximations and the sawtooth wave function

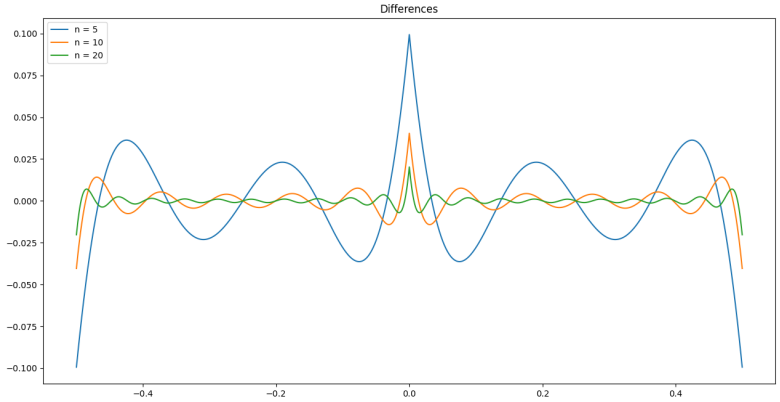

Figure 4: The differences between various partial Fourier series approximations and the triangle wave function.

1.2

Another through which the errors of this approximation can be analysed is by maintaining x and varying the number of terms. As can be seen in Figure 5, the rate of convergence is in fact dependent on x. The errors in Figure 5 were analysed with the np.polyfit function and shown to have errors scaling $\sim O(1/n^1)$ for $x = 0$ and $\sim O(1/n^2)$ for $x = 0.125$

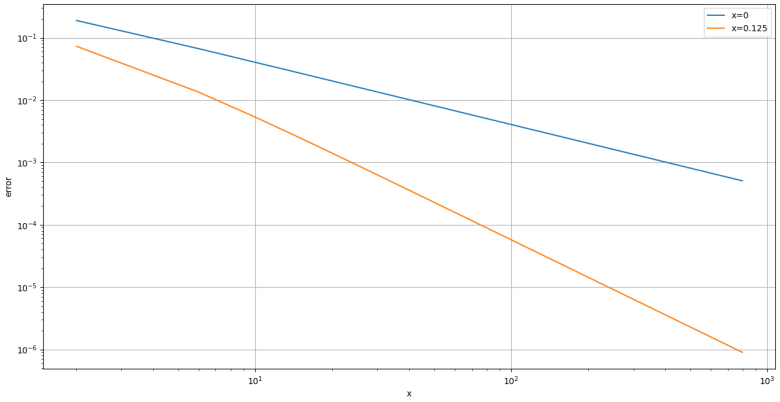

Figure 5: The absolute error of the partial sums at $x = 0$ and $x = 0.125$

1.3

Seen in Figure 1, Gibbs phenomenon acts to force Fourier approximations to consistently overshoot and undershoot the desired function in the presence of discontinuities (such as those present in the sawtooth wave of Figure 1) In order to counteract this error the Lanczos sigma factors (2) were used as a low pass filter to attenuate high frequency oscillations.

$$f(x) \sim \frac{a_0}{2} + \sum_{k=1}^{\infty} sinc\left(\frac{k}{n+1}\right)\left(a_k\, cos\left(\frac{2\pi kx}{L}\right) + b_k\, sin\left(\frac{2\pi kx}{L}\right)\right) \quad (2)$$

As seen in Figure 6 this clearly improved the quality of the Fourier approximation around the aforementioned discontinuity.

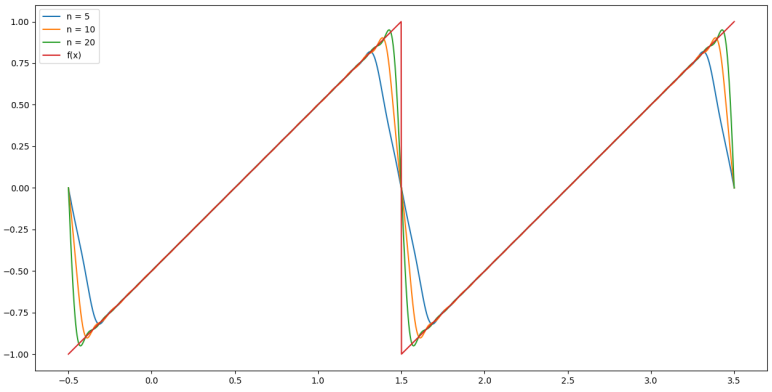

Figure 6: The partial sums of the sawtooth wave function for varying n values in the Lanczos sigma factors.

2 Question 2

2.1

In q2.1.py naive discrete Fourier transform (DFT) and inverse discrete Fourier transform (IDFT) was implemented and applied to (3) in a non-physical context, The use of the DFT function is shown in Figure 7. In order to assess the accuracy of the results of Figure 7, the IDFT was then applied. Analytically, it is possible to perform these two transformations without the loss of information. Clearly seen in Figure 8 however, the choice of discretisation clearly impacts the level of information permanently lost and subsequently the accuracy of the transformation.

$$f(x) = e^{cos(x)} \tag{3}$$

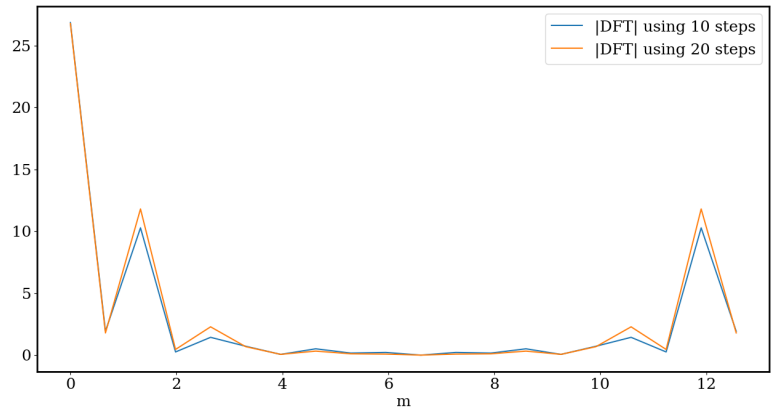

Figure 7: The absolute of the DFT function applied to (3) over the range $0 \leq x \leq 4\pi$

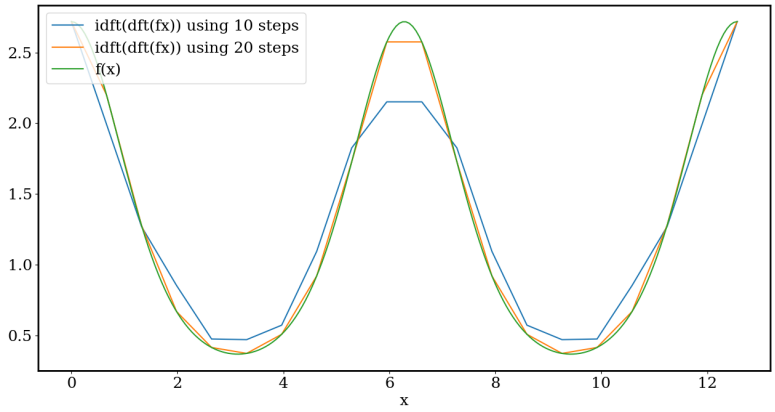

Figure 8: The result of applying the IDFT function to the DFT function of (3) over the range $0 \leq x \leq 4\pi$.

3 Question 3

As discussed in the assignment pdf the above implementation scales as $O(N^2)$. As a result, for large numbers of N it is highly impractical. There is another algorithm however, the Fast Fourier Transform (FFT) which scales as $O(Nlog(N))$. This algorithm is implemented in scipy.fftpack, which was utilised in this question.

The function to which the FFT was performed was the Gaussian wave packet (4), which describes an electron localised in both position and momentum.

$$\psi(x,t) = \frac{e^{-\frac{x^2}{2(1+it)}}}{\sqrt{\sqrt{\pi}(1+it)}} \tag{4}$$

In order to obtain the probability density for the described electron the following operation must be performed:

$$|\psi(x,t)|^2 = \psi(x,t)^* * \psi(x,t) \tag{5}$$

Where $\psi(x,t)^*$ represents the complex conjugate of the wave function $\psi(x,t)$. Plotting this distribution gives Figure 9.

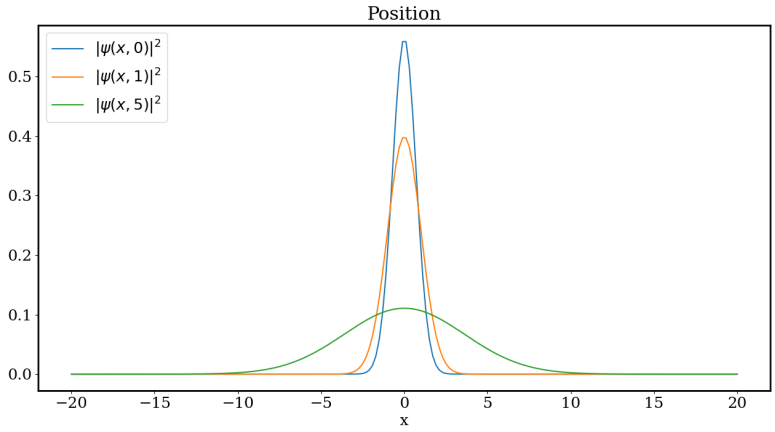

Figure 9: The probability density $|\psi(x,t)|^2$ at t = 0, 1 and 5.

scipy.fftpack.fft was then utilised to apply the FFT to the wave function $\psi(x,t)$ and then scipy.fftpack.fftshift was used to shift the zero frequency of the result to the center of the spectrum so that it can be visually interpreted more intuitively. The squared magnitude of this result is plotted below in Figure 10.

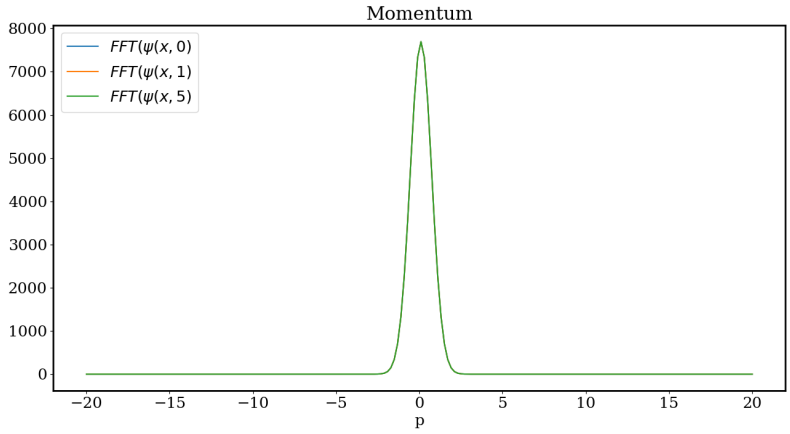

Figure 10: The squared magnitude of the FFT of $\psi(x,t)$ at t = 0, 1 and 5

The physical interpretation of these two plots (Figures 9 and 10) is as follows: According to the density functions in both position and momentum space, we know that the most likely configuration of the electron at t=0 is $x = 0$ and $p = 0$. However, given that both of these distributions are Gaussian in nature, there is a range of positions and momenta that the electron could possess at this initial timestep. As time progresses, the possible x positions of the particle increase. This due to the initial momenta it had at t = 0 acting to continually move it away from the origin, a momenta which it still possesses at later timesteps. This is why we see a widening of the Gaussian distribution for position but all 3 investigated timesteps share the same distribution of momenta.

Conclusion

As demonstrated through the implementation of the partial Fourier series, the accuracy of these approximations depends on a large number of factors such as the number of terms, the presence of discontinuities and even the x position within a function as demonstrated in Figure 5. The discrete Fourier transformation and its inverse was analysed for various values of N however given this algorithms scaling ($O(N^2)$) it is impractical for large N and as a result the fast Fourier transform through the SciPy module scipy.fftpack was implemented due to its far more impressive ($O(Nlog(N))$ scaling. This algorithm was then applied to the scenario of an electron wavepacket, whereby information regarding its position and momentum throughout time can be derived.

4 Code Listing

Question 1.1: q1.1.py

Question 1.2: q1.2.py

Question 1.3: q1.3.py

Question 2: q2.py

Question 3: 3.py