Fourier Analysis Coding project in C

Candidate Number: 21056 Date: 20/3/18

Question 1: Fourier Transform Theory

Discrete Fourier Transform: $H_n(\omega_n) = \sum_{k=0}^{N-1} h_k(t_k)e^{-i2\pi nk/N}$

$h(t_k) = e^{i\omega_1 t_k}, \omega_1 = 1, T = \frac{2\pi}{\omega} = 2\pi, N = 4, \Delta = \frac{T}{N} = \frac{\pi}{2}, \ t_k = k\Delta \therefore h(t_k) = e^{ik\Delta} = e^{ik\frac{\pi}{2}}$

$$e^{ik\frac{\pi}{2}} \times e^{-i2\pi nk/N} \equiv e^{ik\frac{\pi}{2}(1-n)}$$

$$H_0 = \sum_{k=0}^{3} e^{ik\frac{\pi}{2}} = 1 + e^{i\frac{\pi}{2}} + e^{i\pi} + e^{i\frac{3\pi}{2}} = 1 + i - 1 - i = 0$$

$$H_1 = \sum_{k=0}^{3} e^{0} = 1 + 1 + 1 + 1 = 4$$

$$H_2 = \sum_{k=0}^{3} e^{-ik\frac{\pi}{2}} = 1 + e^{-i\frac{\pi}{2}} + e^{-i\pi} + e^{-i\frac{3\pi}{2}} = 1 - i - 1 + i = 0$$

$$H_3 = \sum_{k=0}^{3} e^{-ik\pi} = 1 + e^{-i\pi} + e^{-i2\pi} + e^{-i3\pi} = 1 - 1 + 1 - 1 = 0$$

The spectrum of $h(t_k)$ shows that it is only comprised of one frequency at $n = 1$ and therefore the real and imaginary parts of the function in the time domain are equivalent to $\sin t$ or $\cos t$. This is what you would expect as $e^{it_k} \equiv \cos t_k + i \sin t_k$. As the frequency component of function occurs at $n = 1$ this indicates that $\omega = 1$ which is correct as the question specifies that $\omega_1 = 1$.

Question 2: The outline and approach of the program

The first task that will need to be completed is the ability to deal with complex numbers, this will be achieved by defining a struct called 'complex'. As all complex numbers can be written in the form $a + bi$, where a and b are real numbers, the struct will contain two parts each of type double, one to store the real part and one to store the imaginary part. Complex numbers will need to be multiplied and added together and so the next two functions will take two variables of type complex as their arguments and one function will multiply the two numbers together and will return the resulting complex number and the other will return the addition of the two complex inputs. These functions will be tested using a series of complex numbers with known answers to their multiplication and addition.

The next task will be to define the functions h_1(t) and h_2(t) as given in question 3a, these will both have one argument of type double and will both return complex numbers. The real and imaginary parts will be calculated separately within the function as part of a single complex number. Using a for loop with increments of $\pi/50$ between 0 and 2π each of the functions will be sampled 100 times with the real and imaginary part being printed to two separate text files (h_1.txt and h_2.txt) for plotting.

When performing a Fourier transform on a function, each sampled value must be multiplied by $e^{-i2\pi nk/N}$, the 'Fourier Factor', as this will be used multiple times it will be most efficient to separate it as a function of its own. It will have three arguments all of type integer; n, k and N and will return a complex number. The real and imaginary parts will, again, be calculated separately, using the mathematical identity $e^{it} \equiv \cos t + i \sin t$, before being returned as a complex number. The same will apply for the 'Inverse Fourier Factor', $e^{i2\pi nk/N}$, which is used when calculating inverse Fourier transform values. This will be defined in the same way with three arguments; n, k and N and will return a complex number.

Each discrete Fourier transformed value, H_f,n, is calculated using the equation $\sum_{k=0}^{N-1} h_k(t_k)e^{-i2\pi nk/N}$, therefore each value requires a sum from $k = 0 \rightarrow (N-1)$ and there are $N-1$ discrete Fourier values (H_f,0 → H_f,N-1). This will be achieved by using two for loops, one inside the other each performing $N-1$ iterations, one will represent the k values and the other the n values. To calculate each H_f,n value the value for the chosen function at t_k will be multiplied by the 'Fourier Factor' for that n, k and N using the complex multiplication function. This will be added to the previously calculated value using the complex addition function. k will then be incremented by one and the process will repeat until $k = N-1$ at which point the Fourier transformed value, H_f,n, has been calculated, this will then be stored in an array. n will then be incremented by one and the whole process will repeat until the H_f,N-1 has been calculated and stored. To make the Fourier Transform function work on any arbitrary function it will use a pointer to the function as an argument. It will also have arguments Delta of type double which will indicate the spacing of the samples, for example for function h_1 Delta will be $2\pi/100$. It will also have arguments of N (type integer), the number of samples and store[] (type complex) which will store the Fourier transformed values. The final argument it will have will be an integer, check, that will be used in a case statement to correctly output the results of the Fourier transforms.

The Inverse Fourier Transform function will work in a very similar way to the Fourier Transform function, most notably it will still have the two for loops, one inside the other, that will both run from $0 \rightarrow (N-1)$. The differences will come from the use of the 'Inverse Fourier Factor' not the 'Fourier Factor' and instead of using the value of the function at t_k the stored Fourier transform values will be used (store[n]). Each value of h_f',k will also be divided by N as per the inverse discrete Fourier transform equation. It will also have arguments of N (type integer) and store[] (type complex) which will already contain the stored Fourier transformed values for a particular function. It will also have the 'check' integer as an argument which will again be used in a case statement so that the results of each inverse Fourier transform will be output to the correct text files.

The Fourier Transform function will be tested by simulating the Fourier transform shown in Question 1on the $h(t_k) = e^{it_k}$ function as these answers are already known. The Inverse Fourier Transform function will then be tested on the Fourier transformed values produced by the first test and these will be compared with the expected values of the function.

The final task will start with importing the sampling data from the file h3.txt, this will be achieved by using a two-dimensional double array of size [200][4]. Two dimensional arrays are stored in memory as shown in Figure 1.

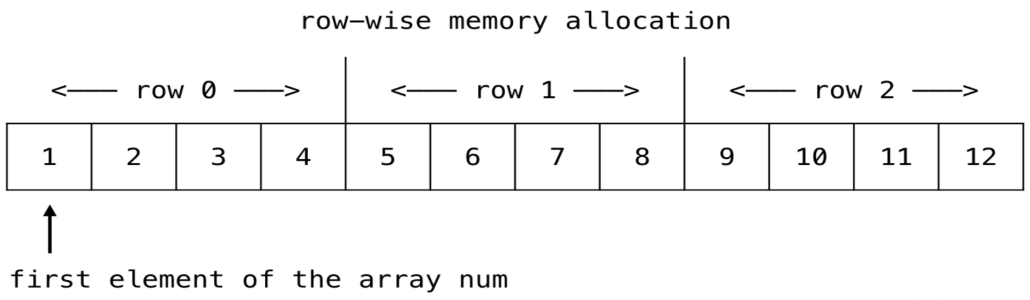

Figure 1. The row wise memory allocation used to store two dimensional arrays.

The file will be imported line by line using a for loop and a pointer to each location in the array, the for loop will go up in increments of four as each line of the file contains four separate pieces of data. The third and fourth column of the array will contain the real and imaginary parts of the sampled function, these will then be combined and stored in a separate complex array. As the equation of the function h_3 is unknown the previously defined Fourier Transform function will not be able to be used and so a new Fourier Transform function will need to be defined especially for h_3. This will work identically to the previous Fourier

Transform function except that instead of having a pointer to the function as an argument it will have the complex array with the sampled values of the function stored inside it. The h_3 Fourier Transform function will therefore also not require the Delta argument but will still retain the complex store[] argument used to store the Fourier transformed values and the integer N argument, the number of samples.

Finally the four H_3,n values with the largest amplitude will need to be found, this will be achieved by applying a bubble sort algorithm to the array storing the h_3 Fourier transformed values. The four largest values will be stored in a separate two-dimensional complex array which will store the complex number and its associated index from the original Fourier transform value array. This will be used to populate a new complex array, with the same size as the original Fourier transform value array, with only the four values with the highest amplitude in their original positions with the rest of the values in the array being set to $0 + 0i$. Using the Inverse Fourier Transform function previously explained an inverse Fourier transform will be applied to this array with the results being output to a text file (h_3'.txt).

Question 4: Interpretation

The Fourier transform of h_1 produced values of $100 + 0i$ at H_1,1 and H_1,5 with the rest of the values being $0 + 0i$. This indicated that h_1 was comprised of two superposed frequencies; one with an angular frequency $\omega = 1$ and the other with an angular frequency $\omega = 5$. As both H_1,1 and H_1,5 had equal magnitudes this showed that the amplitude of each frequency was equal, this was why the maximum amplitude of h_1 was 2, the summation of two $\sin t$ waves. These properties can be seen from the plotted values of h_1 in Figure 2.

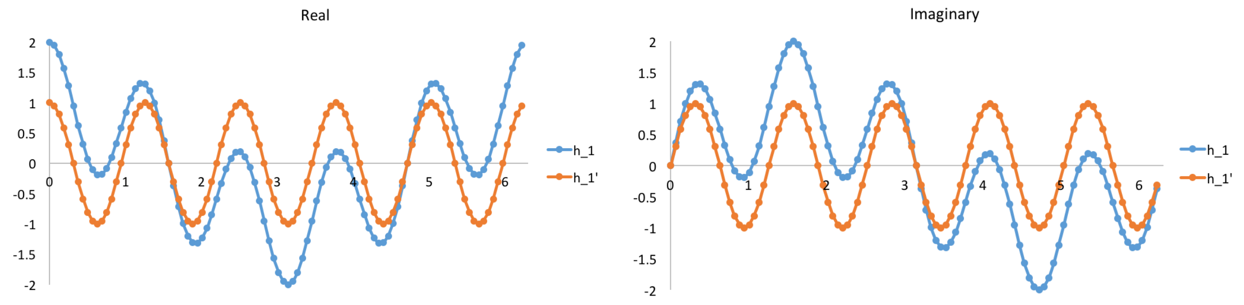

Figure 2. Two graphs showing the real and imaginary parts of the function h_1 sampled 100 times in region $0 - 2\pi$ and the inverse Fourier transformed values of the function, h_1', after the H_1,1 value was set to $0 + 0i$.

These values for the Fourier transformed values of H_1 were to be expected as $h\_1(t) = e^{it} + e^{i5t} \equiv \cos t + \cos 5t + i(\sin t + \sin 5t)$, this is two superposed frequencies one with $\omega = 1$ and the other with $\omega = 5$.

Before applying an inverse Fourier transform H_1,1 was set to $0 + 0i$, the effect of this was removing the $\omega = 1$ frequency dependence leaving only a function an $\omega = 5$ frequency dependence. This can be seen in Figure 2; the h_1' plots vary with only one frequency in both the real and the imaginary plots and the peaks from the h_1' plots line up with the peaks from the h_1 plots as you would expect from two functions with a $\omega = 5$ dependence.

The function $h\_2(t) = e^{-\frac{(t-\pi)^2}{2}}$ was a real function and therefore the imaginary plots for h_2 and h_2' were both 0 for all values of t; the inverse Fourier transform of a real function also returns a real function. This can be seen in the imaginary plot in Figure 3.

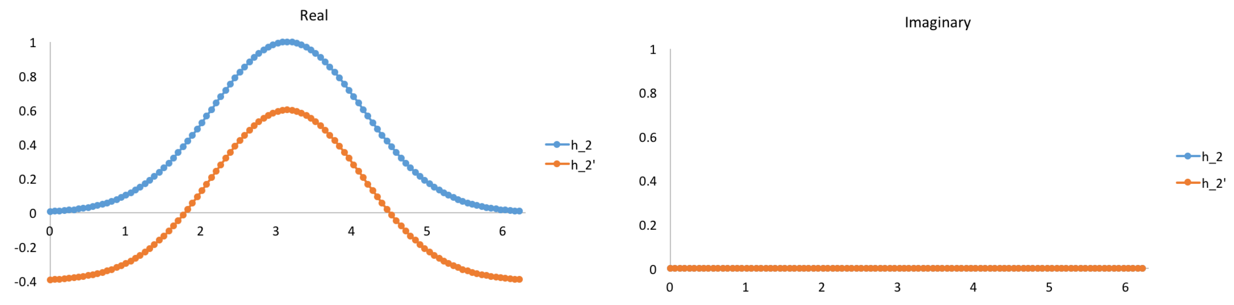

Figure 3. Two graphs showing the real and imaginary parts of the function h_2 sampled 100 times in region $0 - 2\pi$ and the inverse Fourier transformed values of the function, h_2', after the H_2,0 value was set to $0 + 0i$.

As h_2 was not a periodic function in the time domain the components in the frequency domain after the discrete Fourier transform were much more complicated than those of h_1. Every H_2,n and therefore every $\omega = n$ value contributed with the largest contributions coming from H_2,1 and H_2,99 with magnitudes of 24.3. This showed that h_2 was comprised of a sum of many different frequencies. The symmetry of h_2 was reflected in the Fourier transformed values which themselves were symmetrical about H_2,50.

Before applying an inverse Fourier transform the H_2,0 value was set to $0 + 0i$, this had the same effect as skipping this frequency component. As this occurred at $n = 0$ and therefore $\omega = 0$ this component had no frequency contribution. Removing this had the effect of reducing the sum of each h_2',k value and so translated the graph downwards while keeping the shape of the graph identical; as shown in Figure 3.

The equation for h_3 was unknown, however from the Fourier transformed values it was found that it was a periodic function comprised of ten separate frequencies. The largest contributions came from H_3,1, H_3,3, H_3,4 and H_3,13 ($\omega = 1, 3, 4, 13$) which had relative amplitudes of 360, 440, 800 and 600 respectively, the other six frequencies had relative amplitudes of 100 or 200. The combination of these ten frequencies produced a 'noisy' function as can be seen in Figure 4.

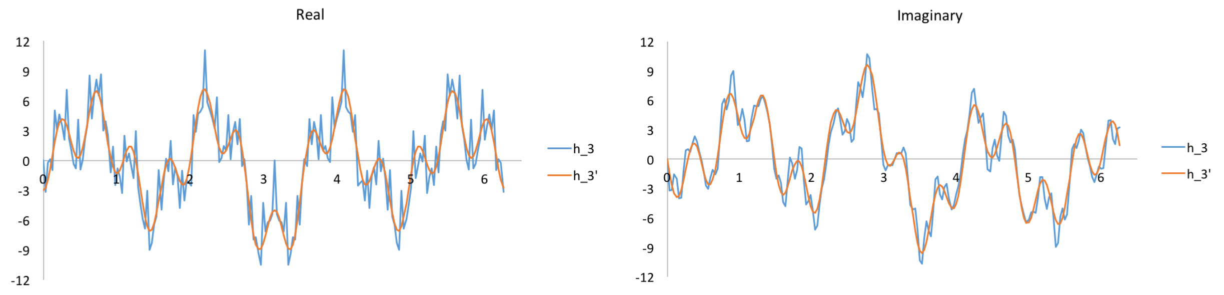

Figure 4. Two graphs showing the real and imaginary parts of the function h_3 sampled 200 times in region $0 - 2\pi$ and the inverse Fourier transformed values of the function, h_3', after the four frequency components with the largest relative amplitudes were found (H_3,1, H_3,3, H_3,4 and H_3,13) and the rest of the frequencies were set to $0 + 0i$.

The four frequencies with the largest relative amplitudes were then found using a bubble sort algorithm and the rest of the frequencies were set to $0 + 0i$ before an inverse Fourier transform was applied. This produced a smoothing affect to h_3, as can be seen in Figure 4 the plots of h_3' follow the same shape as h_3 however with much less noise on the signal. This was to be expected; keeping the four frequencies with the largest relative amplitudes meant that the inverse Fourier transform kept the same shape and removing all the smaller frequency contributions removed the excess noise.