PH20105 -> Coursework 2020 - 2021

Question 1

$$H_n(\omega_n) = \sum_{k=0}^{N-1} h_k(t_k)\, e^{-i 2\pi n k / N}$$

$$h(t) = \exp[i \omega_1 t] \quad \text{with } \omega_1 = 1$$

$$N = 4$$

$$H_0(\omega_0) = \sum_{k=0}^{4-1} \exp[i t_k] \exp[-i 2\pi n k / 4]$$

H_0 so $n = 0$

$$= \sum_{k=0}^{3} \exp[i t_k] \exp[\cancel{-i 0}\ 0]$$

$$= \cancel{\sum_{k=0}^{3} e^{i} e^{0} +}$$

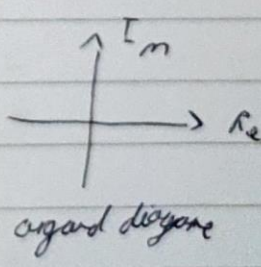
$$= e^{i t_0} + e^{i t_1} + e^{i t_2} + e^{i t_3}$$

We know that $t_k = k\Delta$ and $\Delta = \frac{T}{N}$ and $T = 2\pi$

So $t_k = k\frac{2\pi}{N} = k\frac{2\pi}{4} = \frac{k\pi}{2}$

$$= e^{i0} + e^{i\pi/2} + e^{i\pi} + e^{i\frac{3\pi}{2}}$$

$$= 1 + i + -1 + -i = 0$$



argand diagram

$$H_1(w_1) = \sum_{k=0}^{3} e^{it_k} e^{-i2\pi nk/4} \qquad t_k = \frac{k\pi}{2}$$

$n = 1$

$$= e^{it_0 - 0} + e^{(it_1 - i\,2\pi/4)} + e^{it_2 - i2\pi\cdot\frac{2}{4}}$$

$$+ e^{it_3 - i2\pi\cdot\frac{3}{4}}$$

$$t_0 = 0, \quad t_1 = \frac{\pi}{2}, \quad t_2 = \pi, \quad t_3 = \frac{3\pi}{2}$$

$$= e^{0} + e^{i\frac{\pi}{2} - i\frac{\pi}{2}} + e^{i\pi - i\pi} + e^{\frac{3\pi}{2}i - i\pi\frac{3}{2}}$$

$$= e^{0} + e^{0} + e^{0} + e^{0} = 4$$

$$H_2(w_2) = \sum_{k=0}^{3} e^{ik\frac{\pi}{2}} \cancel{e^{-ik\frac{\pi}{2}}}$$

$$H_2(w_2) = \sum_{k=0}^{3} \exp\left[ik\tfrac{\pi}{2}\right] \exp\left[-i\cdot 2\pi\cdot 2\cdot\tfrac{k}{4}\right]$$

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

$$= e^{0} + e^{i\frac{\pi}{2} - i\pi} + e^{i\pi - i2\pi} + e^{i\frac{3\pi}{2} - i3\pi}$$

$$= 1 + e^{-\frac{\pi}{2}i} + e^{-i\pi} + e^{-\frac{3}{2}\pi i}$$

$$= 1 - i - 1 + i = 0$$

$$H_3(\omega_3) = \sum_{k=0}^{3} \exp\left[ik\frac{\pi}{2}\right] \exp\left[i\frac{k}{4}\cdot 3\cdot 2\pi\right]$$

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

$$= e^{0} + e^{i\frac{\pi}{2} - i\frac{3\pi}{2}} + e^{i\pi - i3\pi}$$

$$+ e^{i\frac{3\pi}{2} - i\frac{4\pi}{2}}$$

$$= 1 + e^{-i\pi} + e^{-2\pi i} + e^{-3\pi i}$$

$$= 1 - 1 + 1 - 1 = 0$$

Question 1 – interpretation

- The fourier transform shifts our complex function from the time domain to the frequency domain.
- Doing the calculation we can see that H_0, H_2, H_3 all equal 0. This is because the function exp[iωt] is orthogonal to the exponential in the fourier transform equation and therefore has no contribution to the function.
- H_1 equals 4. This makes sense since the function has an angular frequency of 1 and so the transform has a peak at n = 1. The amplitude of the peak is equal to 4 since the function has an amplitude of 1 and this is summed over the number of samples N = 4. When doing the inverse we must discount for this by using the normalising factor of 1/N.

Question 2 – Planning the code for question 3

a. Define two C functions that respectively implement two mathematical functions, h1(t) = exp[it] + exp[i5t], and h2(t) = exp[(t − π) 2/2]. Both mathematical function returns are complex numbers (even though h2 only has a real part), so you will need to find a way to define your C functions such that, when provided with a real number t, they are able to provide you with what amounts to a complex number (i.e. two real numbers worth of information rather than one). You are not allowed to use any complex number functionality from an existing library routine.

   Firstly we need to find a way to split h1 and h2 into real and imaginary parts. For h1, I will use Euler's formula to split up the exponential to cosines and sines so the real and imaginary parts are separated. Then form a function which has time as a function variable. I will define the real and imaginary part as type pointers in the function so that both variables can be accessed outside the function. Repeat this method for h2, which is a completely real function so it does not need to be split up as the imaginary part is simply equal to 0. Use type double to store all variables to make use of its double precision compared to using a float.

b. Sample these functions N = 100 times each over a time period T = 2π, and write the results to two text files for plotting. There is no need to include the text files themselves in your report.

   If N = 100 and T = 2π, then the sampling interval will be 2π/100. Therefore the function should be sampled using t = 2πk/100, where k is an integer extending from k = 0 to k = 99. This can be done using a for loop. The contents should be printed into an appropriately named open file using a pointer to the file. The real and imaginary parts of the function can be accessed by pointers as shown above. fprintf will be used to print contents into the file. fopenf and fclosef should be outside of the loop.

c. (outside of your C program) generate plots for both functions and include these in your report. These should be combined with the plots requested in question 3h below.

   Use Excel for this. Columns: k, t, h1_real, h1_imaginary, h2_real, h_2 imaginary.

   Make plots of the functions against time.

d. Use a discrete Fourier transform to obtain H1(ω) and H2(ω) from h1(t) and h2(t) respectively.

The equation for the fourier transform is: $H_n(\omega_n) = \sum_{k=0}^{N-1} h_k(t_k) e^{-i2\pi nk/N}$

Convert the exponential part of the fourier transform into sines and cosines. Since these variables will be needed repeatedly it will be useful to define this in a function of it's own and utilise pointers to access these variables. Multiply out the exponential with the function, noting that the two real parts multiply to give a real part, two imaginary parts multiply to give a real part (remember the -1), and an imaginary and a real part multiply to give an imaginary part. Now the final real and the final imaginary part needs to be summated over k. This summation should then be repeated from n = 0 to n = 99.
A nested for loop should work well for this. The inner loop will carry out the summation and the outer loop will repeat the summation for varying values of n.

e. Print the results for H1 and H2 onto the screen.

Add a print statement in the outer loop to print the results from n = 0 to n = 99. Additionally put the results into an array which can later be accessed for the next question. This can be done by inputting the result from the loop into an array of length 100 using index notation. The index will be the same as n in the for loop.

f. Apply an Inverse Fourier transform to H1 and H2 to obtain h 0 1 and h 0 2 . But for H1 skip n = 1, and for H2 skip n = 0.

The equation for the inverse is $h_k(t_k) = \frac{1}{N}\sum_{n=0}^{N-1} H_n(t_n) e^{i2\pi nk/N}$

The equation is similar to that of the transform so the same mathematical approach can be used. The values from the array for H1 and H2 will be extracted and multiplied with the pointers for the exponential part of the equation. It should be noted that in the inverse the imaginary part will be positive rather than negative so this should be accounted for. Make use of a Boolean If statement within the nested loop so that the necessary value which needs to be skipped is skipped. For example: if n != 1 {carry out instructions}. Also note that this time n should iterate through the inner loop and k is I the outer loop.

g. Write the outcomes of the inverse Fourier transforms to text files for plotting. Again, no need to include the text files themselves in your report.
Around the nested for loop open a file with a pointer with an appropriate name. Print the results into the file using fprintf and close the file at the end.

h. (outside of your C program) generate plots for both h 0 1 and h 0 2 . Please overplot these on the plots for h1 and h2 respectively.
Import the text file from part g into excel and plot h1' and h2' against time.

i. Load the sampling data from the file h3.txt (provided on Moodle) into memory. This file contains N = 200 samplings of a function h3.
Define a 200 by 3, 2D array to store all the data. Open a file in writing mode and scan in the data. The commas in the file should be ignored. k should be read in as type integer and the other 3 variables as long float in standard form since the data is written in standard form. It should also be noted that the commas need to be ignored and this can be done using the format specifier %*c.

j. Apply a discrete Fourier transform to h3 in order to obtain H3.

The same method as used for H1 and H2 should be used but this time the values from the array for h3 should be inserted in.

k. Apply an inverse discrete Fourier transform to H3 in order to obtain h 0 3 . But only include the four terms out of the N terms for H3(ωi) with the largest amplitude, |H3(ωi)|.

Use an if statement within the nested for loop so that only the four terms of n with the largest amplitude is chosen.

l. l Write the outcome for h 0 3 to a text file. No need to include your text file in the report.

Same method as part g.

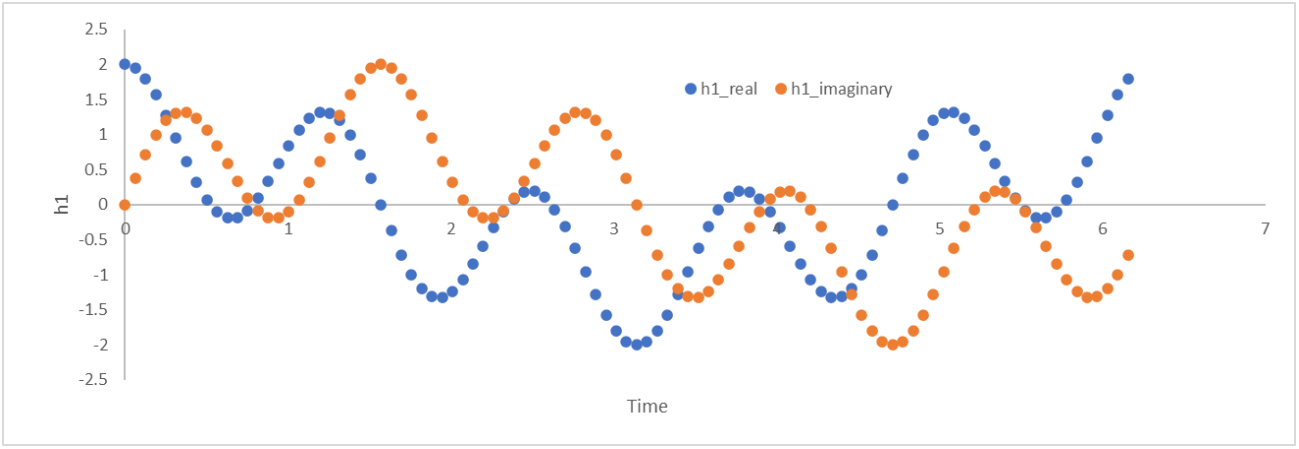
m. m (outside of your C program) overplot h 0 3 (from your own text file) and h3 (from the provided file).

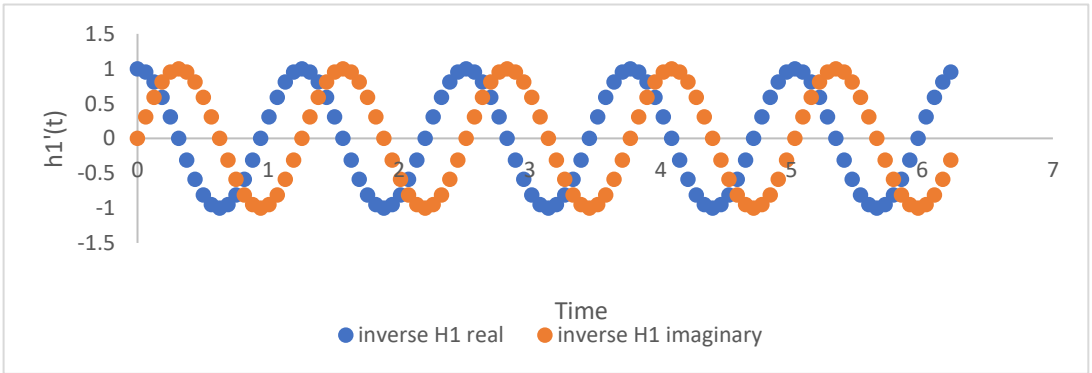
Import data and use excel.

Question 3 + 4 – Graphs interpretation and source code

GRAPHS

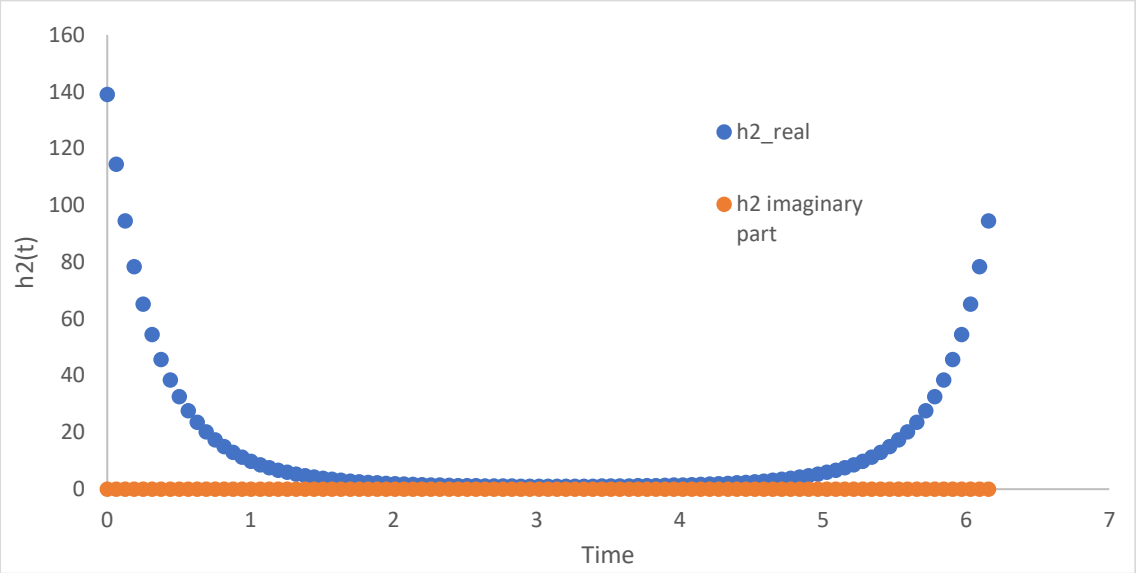
h1

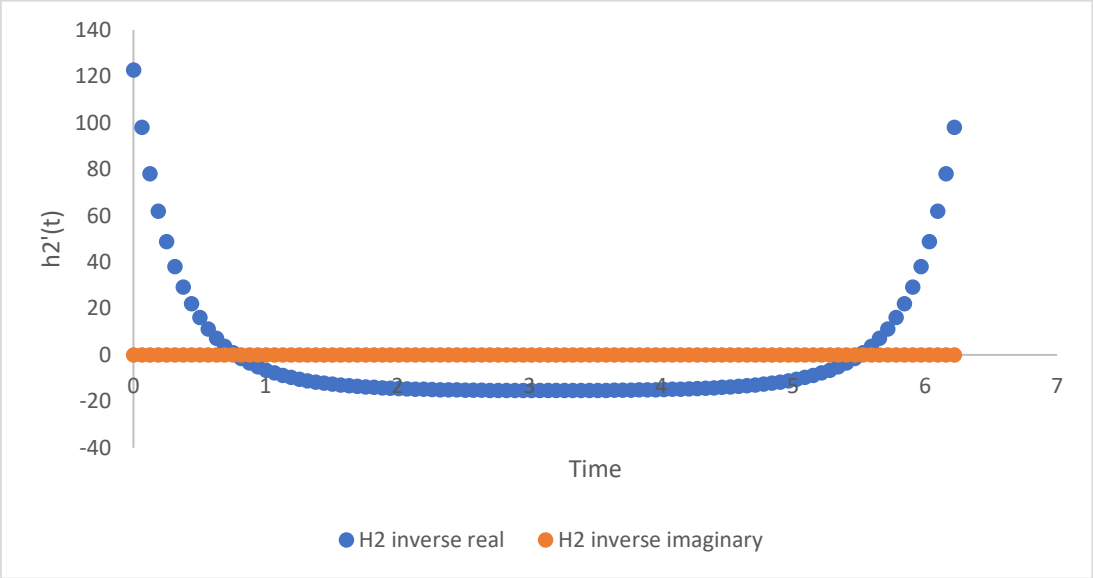




- The first graph is produced from h1 spit into it's real and imaginary components. The function is a sum of 2 complex sinusoids with a frequency of 1 radian and 5 radians.
- We then calculate its fourier transform (H1) we obtain two peaks at n = 1 and n = 5 as expected with peaks of magnitude 100 which corresponds to the number of samples in the period of 2π.
- Following this we apply the inverse transform but we skip n = 1. This discards the frequency of 1 rad and therefore we are left with a complex sinusoid of where the angular frequency is 5. This is confirmed in the second graph as shown in the second graph.

h2

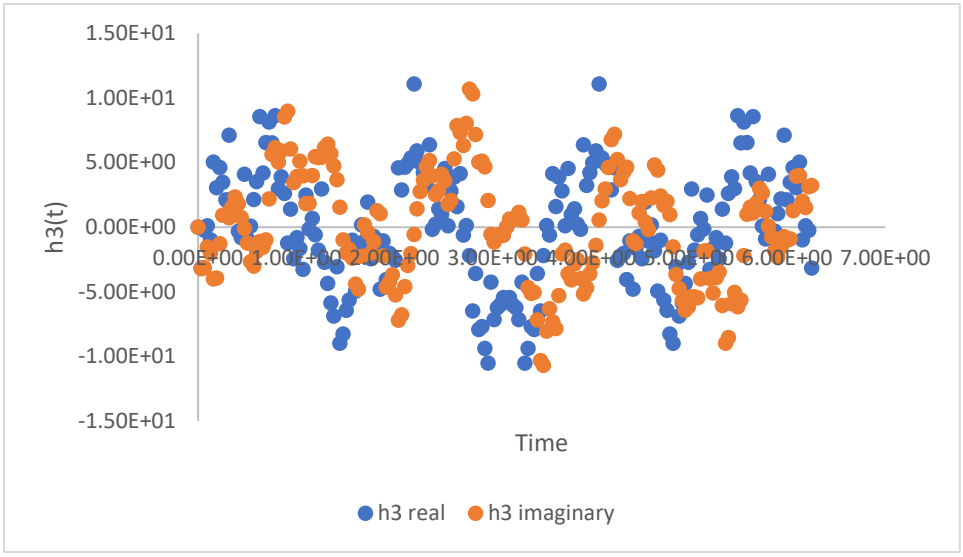


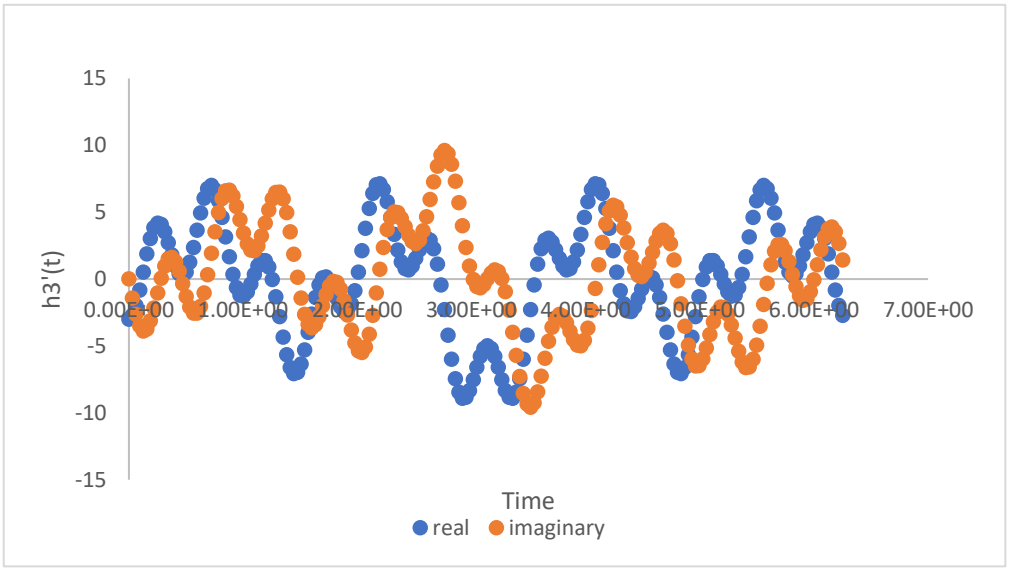


- h2 is a real function as shown above (the imaginary part is equal to 0)
- Applying the inverse to the fourier transform yields the components of sinusoids in the function at the particular sampling points.
- When doing the inverse but omitting n = 0 , we yields the same function but the real part is shifted down by approximately 17. This is because we have removed the DC term which is equivalent to the average. If we calculate the average by taking the integral of the function over a period and dividing by the period 2π this is equal to 17. Since this is removed the graph moves down by this amount.

h3

- data from h3 is quite noisy.
- Doing the transform we obtain the components of the frequencies of sinuisoids and their proportions represented by the magnitudes.
- When doing the inverse we use the complex sinusoids with the largest contribution (magnitude) to the data and the second graph is a sum of these components.





SOURCE CODE:

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415926535  //defining a global variable for the value of pi

//start ----------------------------------------------------------------------------------------------------------//

//defining 2 functionf for h1 and h2

void h1(double t, double * h1_Re, double * h1_Im)  //Makes use of pointers
 {

   * h1_Re = cos(t) + cos(5*t);                //real part of h1
   * h1_Im = sin(t) + sin(5*t);               //imaginary part of h1

   //printf("h1: %f + %f i\n", *h1_Re, *h1_Im);
 }

void h2(double t, double * h2_Re, double * h2_Im )
 {
   double part = (t - PI)*(t - PI);
   double power = part / 2 ;
   * h2_Re = exp(power);    //real part of h2
   * h2_Im  = 0;          //imaginary part of h2

   //printf("h2: %f + %f i\n", *h2_Re, *h2_Im);
 }

 //function for the exponential part of fourier transforms
 void exponential(double k, double n, double * t, double * exp_real, double * exp_im)
```

```
  {
     *t = k * ((2 * PI) /100);
     * exp_real = cos ((*t) * (n));
     * exp_im = - sin ((*t) * (n));
  }

//-----------------------------------------------------------------------------------------------------------------------//

// writing the sampled results (with N = 100) into files named 'h1_data.txt' and 'h2_data.txt'

void h1_txt_file()
 {
   FILE *fp; // declare a pointer to FILE fp
   fp = fopen("h1_data.txt", "w");
   int k = 0;

   for (k = 0; k < 100; k++)
   {
      double h1_Re, h1_Im;
      double t = k * ((2 * PI) /100);    // t is time
      h1(t, &h1_Re, &h1_Im);
      fprintf(fp, "%f %f %f\n", t, h1_Re, h1_Im);
   }
   fclose(fp);
 }


 void h2_txt_file()
 {
   FILE *fp; // declare a pointer to FILE fp
   fp = fopen("h2_data.txt", "w");
```

```
   int k = 0;

   for (k = 0; k < 100; k++)
   {
      double h2_Re, h2_Im;
      double t = k * ((2 * PI) /100);
      h2(t, &h2_Re, &h2_Im);
      fprintf(fp, "%f %f %f\n", t, h2_Re, h2_Im);
   }
   fclose(fp);
 }

// ------------------------------------------------------------------------------------------------------------------------
-//

// fourier analysis for function h1 and h2
void fourier_analysis ()
 {
   // H1 - fourier transform for h1

   printf("\n");
   printf("Printing fourier transform of h1:\n");
   printf("\n");

   //define all variables
   double t, Real_1, Real_2, Imaginary_1, Imaginary_2;
   double  H1_Re, H1_Im;

   // Real1 and Imaginary1 refer to the real/ imaginary part of the function.
   // Real2 and Imaginary2 refer to the real/ imaginary part of the exponential in the fourier series.
```

```
double real, imaginary;                 //for summations
double array_H1_Re[100], array_H1_Im[100]; //store results in an array
int k, n;                               // for iterations

for(n = 0; n < 100; n ++)
{
   real = 0;                            //set the summation back to 0 after each iteration of n
   imaginary = 0;

   for(k = 0; k < 100; k++)
   {

      exponential(k, n, &t, &Real_2, &Imaginary_2);
      h1(t, &Real_1, &Imaginary_1);

      H1_Re = (Real_1 * Real_2) - (Imaginary_1 * Imaginary_2);
      H1_Im = (Real_1 * Imaginary_2) + (Imaginary_1 * Real_2);
      real += H1_Re;
      imaginary += H1_Im;
   }
   array_H1_Re[n] = real;                   //iterate results to an array to access the values
   array_H1_Im[n] = imaginary;
   printf("for n = %d, H1_Re = %f, H1_Im = %f\n", n, real, imaginary);

}

//H2 - fourier transform for h2

printf("\n");
printf("Printing fourier transform of h2:\n");
printf("\n");
```

```
double H2_Re, H2_Im, h2_Re, h2_Im;
double array_H2_Re[100], array_H2_Im[100];

for(n = 0; n < 100; n ++)
{
   real = 0;
   imaginary = 0;

   for(k = 0; k < 100; k++)
   {
      exponential(k, n, &t, &Real_2, &Imaginary_2);
      h2(t, &h2_Re, &h2_Im);

      H2_Re = h2_Re * Real_2;
      H2_Im = h2_Im * Imaginary_2;
      real += H2_Re;
      imaginary = H2_Im ;
   }

   array_H2_Re[n] = real;
   array_H2_Im[n] = imaginary;
   printf("for n = %d, H2_Re = %f, H2_Im = %f\n", n, real, imaginary);
}
//for(int loop = 0; loop < 100; loop++)
 //{printf("%f\n", array_1_Re[loop]);}         //code for checking the array is done correctly

//-------------------------------------------------------------------------------------------//

// inverse transform
// h1'
```

```
FILE *fp; // declare a pointer to FILE fp
fp = fopen("h1'_data.txt", "w");
fprintf(fp, "k t Real Imaginary\n");

 for(k = 0; k < 100; k ++)
  {
    real = 0;
    imaginary = 0;

    for(n = 0; n < 100  ; n ++)
    {
       if (n !=  1)
       {
          double H1_Re_inv, H1_Im_inv;   //Real_1, Imaginary1,
          exponential(k, n, &t, &Real_2, &Imaginary_2);
          Real_1 = array_H1_Re[n];
          Imaginary_1 = array_H1_Im[n];

          H1_Re_inv = (Real_1 * Real_2) - (Imaginary_1 *  - Imaginary_2);
          H1_Im_inv = (Real_1 * - Imaginary_2) + (Imaginary_1 * Real_2);
          real += H1_Re_inv;
          imaginary += H1_Im_inv;
       }
    }
    fprintf(fp, "%d %f %f %f\n", k, t, real/100, imaginary/100);
    //printf("\nfor k = %d, t = %f: H1_Re_inv = %f, H1_Im_inv = %f", k, t, real/100, imaginary/100);
  }

 fclose(fp);
```

```
// h2'

FILE *fp_h2; // declare a pointer to FILE fp
fp_h2 = fopen("h2'_data.txt", "w");
fprintf(fp_h2, "k t Real Imaginary\n");

 for(k = 0; k < 100; k ++)
  {
    real = 0;
    imaginary = 0;

    for(n = 1; n < 100  ; n ++)
    {
          double H1_Re_inv, H1_Im_inv;

          exponential(k, n, &t, &Real_2, &Imaginary_2);

          Real_1 = array_H2_Re[n];
          Imaginary_1 = array_H2_Im[n];
          H1_Re_inv = (Real_1 * Real_2) - (Imaginary_1 * - Imaginary_2);
          H1_Im_inv = (Real_1 * - Imaginary_2) + (Imaginary_1 * Real_2);
          real += H1_Re_inv;
          imaginary += H1_Im_inv;
    }

    fprintf(fp_h2, "%d %f %f %f\n", k, t, real/100, imaginary/100);
    //printf("for k = %d, t = %f: H2_Re_inv = %f, H2_Im_inv = %f\n", k, t, real/100, imaginary/100);

  }

 fclose(fp_h2);
```

```
 }

//-----------------------------------------------------------------------------------------------------------------------------

void h3_analysis()
 {
   FILE *fp;
   double t, re, im;
   double h3_array[200][3];  //define a 2D array which stores the real and imaginary parts
   fp = fopen("h3.txt", "r");
   int n, k = 0;
   while(fscanf(fp, "%d%*c  %le%*c %le%*c %le", &n, &t, &re, &im) != EOF)    //*c tells the scan statement to omit the comma

   {
      h3_array[k][0] = t;
      h3_array[k][1] = re;
      h3_array[k][2] = im;

      //printf("%le, %le, %le\n", h3_array[k][0], h3_array[k][1], h3_array[k][2]);
      k ++;
   }
   fclose(fp);

//-------------------------------------------------------------------------------------------
   //H3
   printf("\n");

   double H1_Re, H1_Im, Real_1, Real_2, Imaginary_1, Imaginary_2;
   double real, imaginary;
   double array_1_Re[200], array_1_Im[200];
```

```
 for(n = 0; n < 200; n ++)
 {
    real = 0;
    imaginary = 0;

    for(k = 0; k < 200; k++)
    {
       //exponential(k, n, &t, &Real_2, &Imaginary_2);
       double t = h3_array[k][0];
       Real_1 = h3_array[k][1];
       Imaginary_1 = h3_array[k][2];
       Real_2 = cos(t * n);
       Imaginary_2 = -sin(t * n);
       H1_Re = (Real_1 * Real_2) - (Imaginary_1 * Imaginary_2);
       H1_Im = (Real_1 * Imaginary_2) + (Imaginary_1 * Real_2);
       real += H1_Re;
       imaginary += H1_Im;
    }
    array_1_Re[n] = real;
    array_1_Im[n] = imaginary;
    //printf("for n = %d, H3_Re = %f, H3_Im = %f\n", n, real, imaginary);
 }

//-------------------------------------------------------------------------------
//inverse of H3
 FILE *inverse_fp; // declare a pointer to FILE inverse_fp
 inverse_fp = fopen("h3'_data.txt", "w");
 fprintf(inverse_fp, "k Real Imaginary\n");

  for(k = 0; k < 200; k ++)
```

```
    {
      real = 0;
      imaginary = 0;

      for(n = 0; n < 200; n++)
      {
        if (n ==  1 || n == 3 || n== 4 || n == 13 )
         {

            double t = h3_array[k][0];
            Real_2 = cos(t *n );
            Imaginary_2 = sin(t *n );
            double Real_1 = array_1_Re[n];
            double Imaginary_1 = array_1_Im[n];
            double H3_Re_inv = (Real_1 * Real_2) - (Imaginary_1 * - Imaginary_2);
            double H3_Im_inv = (Real_1 * - Imaginary_2) + (Imaginary_1 * Real_2);
            real += H3_Re_inv;
            imaginary += H3_Im_inv;
         }
      }
      //printf("for k = %d, H3_Re_inv = %f, H3_Im_inv = %f\n", k, real/200, imaginary/200);
      fprintf(inverse_fp, "%d %f %f\n", k, real/200, imaginary/200);
    }

    fclose(inverse_fp);
  }

int main()
  {
    h1_txt_file();
    h2_txt_file();
```

```
    fourier_analysis();
    h3_analysis();
}
```