

Logbook: FFTW3

Pedraza-Espitia S.

< git.io/salvador >

1. Instalación

Siguiendo las instrucciones en http://www.fftw.org/fftw3_doc/Installation-on-Unix.html#Installation-on-Unix

Descargar la última versión de fftw3, descomprimir (extraer en ~/softw, en mi caso se creo el directorio fftw-3.3.6-pl2),

En la terminal:

```
1 cd ~/softw/fftw-3.3.6-pl2; mkdir ../fftw
2 ./configure --prefix=~/softw/fftw
3 make
4 sudo make install
5 cd ..
6 mv fftw-3.3.6-pl2 ~/sources
```

2. Programa en C

Hay tres partes esenciales para remarcar, la primera es la lectura de los datos de un archivo de datos que debe contener una columna de datos reales (float), la segunda parte usa las funciones de la biblioteca `fftw.3` para obtener la transformada de fourier y la tercera parte guarda los datos de la salida a un archivo que automáticamente nombrará `salida.fftw`.

```
1 #include <fftw3.h>
2 #include <iostream>
3 #include <cmath>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7
8 using namespace std;
9
10 // Define some constants
11 #define REAL 0
12 #define IMAG 1
13 #define MAX 2200
14
15 int main( int argc, char *argv[] )
```

```

16 {
17     // Read the data from argv[1]
18     FILE *file = fopen(argv[1], "r");
19     float arraydata[MAX];
20     int i = 0;
21     int final = 0;
22     if (file != NULL) {
23         while (!feof(file) && i < MAX) {
24             if (fscanf(file, "%f", &arraydata[i++]) != -1) {
25                 printf("%f \n", arraydata[i-1]);
26                 final = i;
27             }
28         }
29         fclose(file);
30     } else {
31         printf("Unable to open file");
32         return EXIT_FAILURE;
33     }
34
35     // Define the length of the complex arrays
36     int n = final;
37     // Input array
38     fftw_complex x[n]; // This is equivalent to: double x[n][2];
39     // Output array
40     fftw_complex y[n]; //
41     // Fill the first array with some data
42     for (int i=0; i<n; i++){
43         x[i][REAL] = arraydata[i];
44         x[i][IMAG] = 0;
45     }
46
47     // Plan the FFT and execute it
48     fftw_plan plan = fftw_plan_dft_1d(n, x, y, FFTW_FORWARD, FFTW_ESTIMATE);
49     fftw_execute(plan);
50     // Do some cleaning
51     fftw_destroy_plan(plan);
52     fftw_cleanup();
53     // Display the results
54     cout << "\n\nFFT = " << endl;
55     for (int i=0; i<n; i++)
56         if (y[i][IMAG]<0)
57             cout << y[i][REAL] << " - " << abs(y[i][IMAG]) << "i" << endl;
58         else
59             cout << y[i][REAL] << " + " << y[i][IMAG] << "i" << endl;
60
61     // Write my output
62     FILE *salida;
63     if (argc == 3){
64         salida = fopen(strcat (argv[2], ".fftw"), "w");
65     }
66     else{
67         salida = fopen( "salida.fftw" , "w");
68     }

```

```

69  for (int i=0; i<n; i++){
70      fprintf( salida, "%4f,%4f\n", y[i][REAL], y[i][IMAG] );
71  }
72  fclose(salida);
73
74  return 0;
75 }

```

El programa se guarda en un archivo `processdata_fftw.cpp` y se compila con

```

1  g++ -o processdata_fftw processdata_fftw.cpp -lfftw3 -I/home/salva/softw/fftw
    -3.3.6-pl2/fftw/include -L/home/salva/softw/fftw-3.3.6-pl2/fftw/lib

```

Se ejecuta con

```

1  ./processdata_fftw input.dat outputName

```

3. Output

Para graficar uso

```

1  gnuplot -p -e "plot 'data.dat'"

```

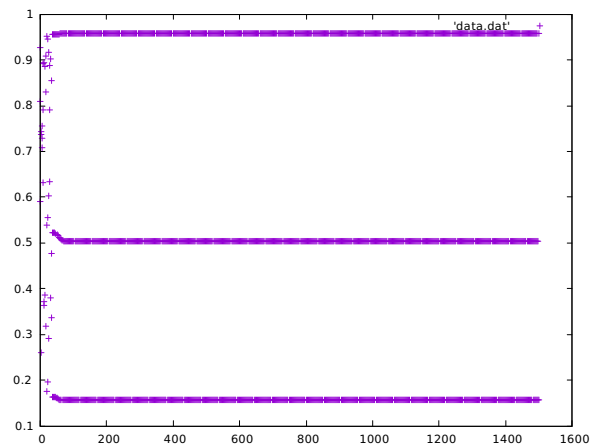


Figura 1: Ec logística $c = 2.x$.

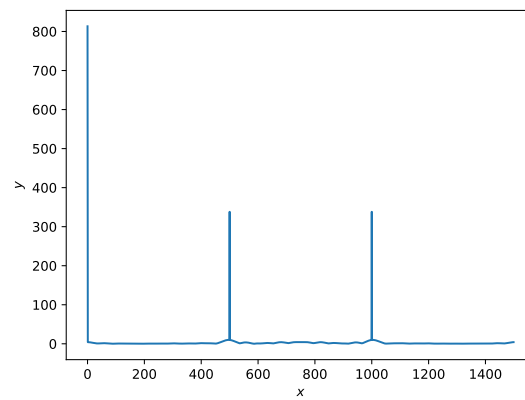


Figura 2: Después de aplicar FFTW a los datos que se graficaron en la ??, se obtienen las magnitudes de la salida de FFTW (números complejos).

4. Transformada de Fourier en Python