

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=$HOME/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 2000000
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, *y;
39     x = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * MAX);
40     y = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * MAX);
41     fftw_plan plan;
42     //fftw_complex x[n]; // This is equivalent to: double x[n][2];
43     // Output array
44     //fftw_complex y[n]; //
45     // Fill the first array with some data
46     for (int i=0; i<n; i++){
47         x[i][REAL] = arraydata[i];
48         x[i][IMAG] = 0;
49     }
50
51     // Plan the FFT and execute it
52     plan = fftw_plan_dft_1d(n, x, y, FFTW_FORWARD, FFTW_ESTIMATE);
53     fftw_execute(plan);
54     // Do some cleaning
55     fftw_destroy_plan(plan);
56     fftw_cleanup();
57     // Display the results
58     cout << "\n\nFFT = " << endl;
59     //for (int i=0; i<n; i++)
60     //    if (y[i][IMAG]<0)
61     //        cout << y[i][REAL] << " - " << abs(y[i][IMAG]) << "i" << endl;
62     //    else
63     //        cout << y[i][REAL] << " + " << y[i][IMAG] << "i" << endl;
64
65     // Write my output
66     FILE *salida;
67     if (argc == 3){
68         salida = fopen(strcat (argv[2], ".fftw"), "w");

```

```

69 }
70 else{
71     salida = fopen( "salida.fftw" ,"w");
72 }
73 for (int i=0; i<n; i++){
74     fprintf( salida, "%4f,%4f\n",y[i][REAL], y[i][IMAG] );
75 }
76 fclose(salida);
77 //fftw_destroy_plan(plan);
78 fftw_free(x); fftw_free(y);
79
80 return 0;
81 }

```

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 fftw

Para graficar en gnuplot uso

```

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

```

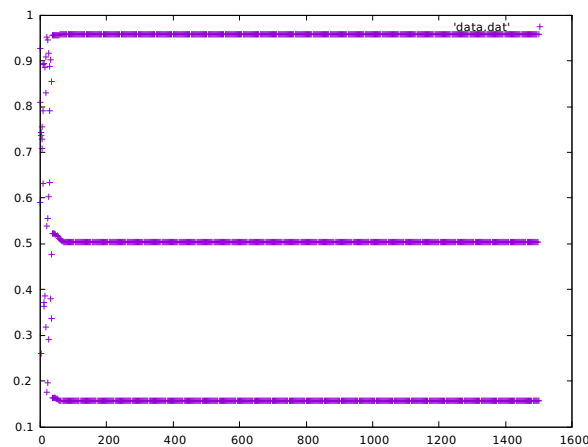


Figura 1: Ec logística $c = 2.x$.

A continuación se ejecuta el código presentado en la [Sección 2](#) y se obtiene la gráfica de la [Figura 2](#)

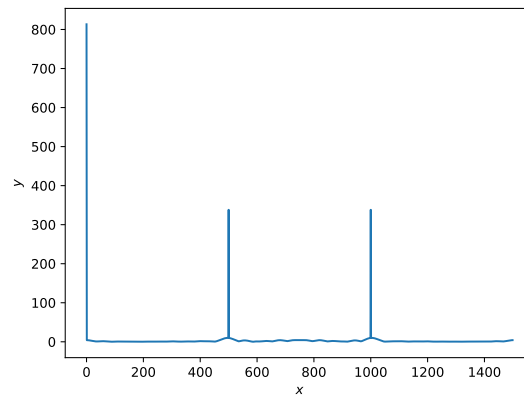


Figura 2: Después de aplicar FFTW a los datos que se graficaron en la Figura 1, se obtienen las magnitudes de la salida de FFTW (números complejos).

4. Transformada de Fourier Discreta en Python

$$X(F) = \int x(t)e^{-j2\pi Ft} dt \quad (1)$$

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j2\pi kn}{N}} \quad (2)$$

Un objetivo es programar la transformada de Fourier discreta en Python y obtener el mismo resultado que los algoritmos implementados en FFTW

```

1 from matplotlib import pyplot as plt
2 import numpy as np
3
4 x_ = np.loadtxt('data.dat')
5 #x_ = np.array( [0., 0.707, 1., 0.707, 0., -0.707, -1., -0.707] )
6
7 N = len( x_ )
8 X_ = np.zeros( (N,2) )
9 for k in np.arange( N ): # frequency bins ( Fourier coefficients )
10     real = 0. ; cmplx = 0.
11     for n in range( N ):
12         bn = -2*np.pi * k * n/N
13         real += x_[n] * np.cos( bn )
14         cmplx += x_[n] * np.sin( bn )
15     X_[k,0] = real
16     X_[k,1] = cmplx
17
18 magnitud = np.sqrt( X_[:,0]**2 +X_[:,1]**2 )
19 fig = plt.figure(1)
20 ax = fig.add_subplot(111)
21 ax.set_xlabel(r'freq')
22 ax.set_ylabel(r'$\alpha$*ampl')
```

```
23 ax.plot(magnitud)
24 plt.show()
25 #print( X_ )
26 # numpy.savetxt(pathFile+'salida.dat', mydata[0:1000], \
27 #     delimiter='', fmt='%3.5f')
```