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:

```
cd ~/softw/fftw-3.3.6-pl2; mkdir ../fftw
./configure --prefix=~/softw/fftw
make
sudo make install
cd ..
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.

```
#include <fftw3.h>
#include <iostream>
#include <cmath>
#include <stdio.h>
#include <stdlib.h>

using namespace std;

// Macros for the real and imaginary parts
#define REAL 0
#define IMAG 1
#define MAX 2200

int main( int argc, char **argv )
{
```

```
16      FILE *file = fopen(argv[1], "r");
17      float arraydata[MAX];
18      int i = 0;
19      int final = 0;
20      if (file != NULL) {
21          while (!feof(file) && i < MAX) {
22              if (fscanf(file, "%f", &arraydata[i++]) != -1) {
23                  printf("%f \n", arraydata[i-1]);
24      final = i;
25              }
26          }
27          fclose(file);
28      } else {
29          printf("Unable to open file");
30          return EXIT_FAILURE;
31      }
32
33    // Define the length of te complex arrays
34    int n = final;
35    // Input array
36    fftw_complex x[n]; // This is equivalent to: double x[n][2];
37    // Output array
38    fftw_complex y[n]; //
39    // Fill the first array with some data
40    for (int i=0; i<n; i++){
41      x[i][REAL] = arraydata[i];
42      x[i][IMAG] = 0;
43    }
44
45    // Plan the FFT and execute it
46    fftw_plan plan = fftw_plan_dft_1d(n, x, y, FFTW_FORWARD, FFTW_ESTIMATE);
47    fftw_execute(plan);
48    // Do some cleaning
49    fftw_destroy_plan(plan);
50    fftw_cleanup();
51    // Display the results
52    cout << "\n\nFFT = " << endl;
53    for (int i=0; i<n; i++)
54      if (y[i][IMAG]<0)
55        cout << y[i][REAL] << " - " << abs(y[i][IMAG]) << "i" << endl;
56      else
57        cout << y[i][REAL] << " + " << y[i][IMAG] << "i" << endl;
58
59  // Write my output
60    FILE *salida;
61    salida = fopen("salida.fftw","w");
62    for (int i=0; i<n; i++){
63      fprintf( salida,"%.4f,%.4f\n",y[i][REAL], y[i][IMAG] );
64    }
65    fclose(salida);
66
67    return 0;
68  }
```

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

```
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

```
./processdata_fftw data.dat
```

3. Output

Para graficar uso

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

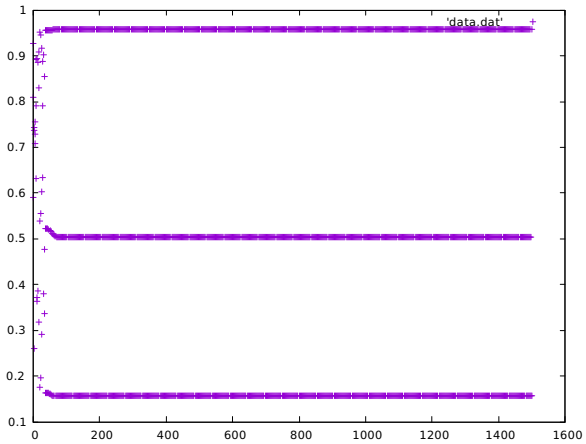

Figura 1: Ec logística $c = 2.x$.

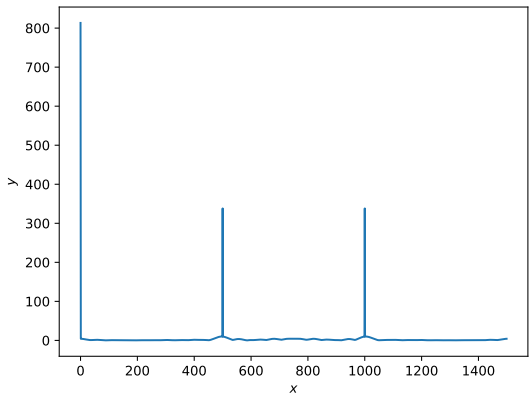

Figura 2: Después de aplicar FFTW a los datos que se graficaron en la Figura 1, se obtienen las magnitudes de la salida de FFTW (números complejos).