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=$HOME/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>
#include <string.h>

using namespace std;

// Define some constants
#define REAL 0
#define IMAG 1
#define MAX 2200

int main( int argc, char *argv[] )
```

```
{
      // Read the data from argv[1]
      FILE *file = fopen(argv[1], "r");
      float arraydata[MAX];
      int i = 0;
      int final = 0;
      if (file != NULL) {
          while (!feof(file) && i < MAX) {
                if (fscanf(file, "%f", &arraydata[i++]) != -1) {
                    printf("%f \n", arraydata[i-1]);
        final = i;
            }
        }
        fclose(file);
      } else {
        printf("Unable to open file");
        return EXIT_FAILURE;
      }

  // Define the length of te complex arrays
  int n = final;
  // Input array
  fftw_complex x[n]; // This is equivalent to: double x[n][2];
  // Output array
  fftw_complex y[n]; //
  // Fill the first array with some data
  for (int i=0; i<n; i++){
    x[i][REAL] = arraydata[i];
    x[i][IMAG] = 0;
  }

  // Plan the FFT and execute it
  fftw_plan plan = fftw_plan_dft_1d(n, x, y, FFTW_FORWARD, FFTW_ESTIMATE);
  fftw_execute(plan);
  // Do some cleaning
  fftw_destroy_plan(plan);
  fftw_cleanup();
  // Display the results
  cout << "\n\nFFT = " << endl;
  for (int i=0; i<n; i++)
    if (y[i][IMAG]<0)
      cout << y[i][REAL] << " - " << abs(y[i][IMAG]) << "i" << endl;
    else
      cout << y[i][REAL] << " + " << y[i][IMAG] << "i" << endl;

  // Write my output
  FILE *salida;
  if (argc == 3){
  salida = fopen(strcat (argv[2],".fftw"),"w");
  }
  else{
  salida = fopen( "salida.fftw" ,"w");
  }
```

```
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

```
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 input.dat outputName
```

3. Output fftw

Para graficar en gnuplot uso

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

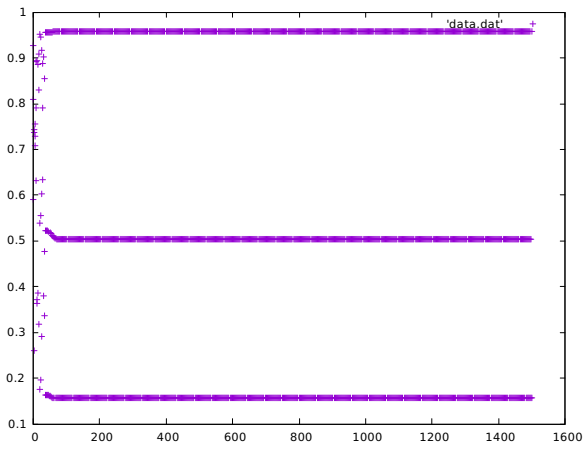

Figura 1: Ec logística $c = 2.x$.

A continuación se ejecuta el código presentado en la Sección 2 y se obtiene la gráfica de la Figura 2

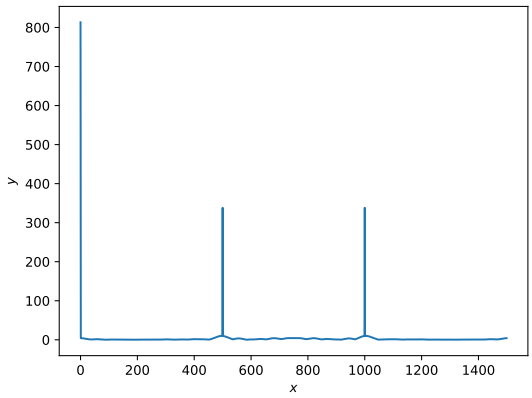

Figura 2: Después de aplicar FFTW a los datos que se graficaron en la Figura 1, se obtienen las magnitudes de la salida de FFTW (números complejos).

4. Transformada de Fourier Discreta en Python

$$X(F) = \int x(t) e^{-j2\pi Ft} \mathrm{d}t \tag{1}$$

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

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

```
from matplotlib import pyplot as plt
import numpy as np

x_ = np.loadtxt('data.dat')
#x_ = np.array( [0., 0.707, 1., 0.707, 0., -0.707, -1., -0.707] )

N = len( x_ )
X_ = np.zeros( (N,2) )
for k in np.arange( N ): # frequency bins ( Fourier coefficients )
  real = 0. ; cmplx = 0.
  for n in range ( N ):
    bn = -2*np.pi * k * n/N
    real += x_[n] * np.cos( bn )
    cmplx += x_[n] * np.sin( bn )
  X_[k,0] = real
  X_[k,1] = cmplx

magnitud = np.sqrt( X_[:,0]**2 +X_[:,1]**2 )
fig = plt.figure(1)
ax = fig.add_subplot(111)
ax.set_xlabel(r'freq')
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')
```