Computación científica LF-214

Profesor: Julio C. Marín

Departamento de Meteorología, Universidad de Valparaíso

Primer Semestre

Temas de la clase

- Transformada rápida de Fourier (FFT)
- Ejemplos y ejercicios

Transformada rápida de Fourier (FFT)

- Fórmula para DFT es:

$$c_k = \sum_{n=0}^{N-1} y_n \exp\left(-i\frac{2\pi kn}{N}\right)$$

- Programa para calcular DFT tiene 2 ciclos
- Para cada coeficiente c_k se adicionan N términos
- Luego hay que agregar todos los términos para cada uno de los $\frac{1}{2}N + 1$ coeficientes c_k
- Debemos evaluar finalmente $N(\frac{1}{2}N + 1)$ términos
- Computador tendrá que evaluar aproximadamente $\frac{1}{2}N^2$ operaciones aritméticas para calcular DFT

Transformada rápida de Fourier (FFT)

- En curso anterior vimos que el mayor número de operaciones que puede hacer un PC para hacer cálculos en tiempos razonables es un billón de operaciones
- Aplicando para DFT si tenemos $\frac{1}{2}N^2 = 10^9 => N \simeq 45000$ muestras
- Este número no es muy grande en muchas aplicaciones prácticas
- Por ejemplo, 45000 muestras es solo 1 segundo de música u otras señales de audio
- Muchas veces queremos aplicar DFT a muestras mucho más grandes

Transformada rápida de Fourier (FFT)

- Carl F. Gauss desarrolló la Transformada rápida de Fourier (FFT) en 1805
- Desarrolló cuadratura Gaussiana para calcular integrales numéricas
- Desarrolló eliminación Gaussiana para resolver sist. ecs. lineales
- Desarrolló método Gauss-Newton para sist. de ecuaciones no-lineales
- Gran parte de métodos numéricos usados se deben a Gauss más de 1 siglo antes de que se inventaran las computadoras

Transformada rápida de Fourier (FFT)

Los paquetes numpy y scipy incluyen funciones para calcular transformadas de Fourier, particularmente FFT

- Función fft: Calcula la DFT en 1-D usando el algoritmo FFT
- Función ifft: Calcula la transformada inversa usando FFT.

1-D discrete Fourier transforms

The FFT *y[k]* of length N of the length-N sequence *x[n]* is defined as

$$y[k] = \sum_{n=0}^{N-1} e^{-2\pi j \frac{kn}{N}} x[n] \,,$$

and the inverse transform is defined as follows

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi j \frac{kn}{N}} y[k] \,.$$

Transformada rápida de Fourier (FFT)

- Si datos son reales, se usa automáticamente un algoritmo más eficiente de FFT
- Pero podemos mejorar más la eficiencia usando función rfft

Función rfft: Calcula el FFT cuando los datos son reales. Solo entrega la mitad del espectro de coeficientes simétricos

- Si los datos son reales y simétricos, podemos aumentar aún más la eficiencia usando función dct

Función dct: Calcula la serie discreta de cosenos de Fourier.

Función fft2: Calcula la FFT en 2 dimensiones

Transformada rápida de Fourier (FFT)

Ejemplo sencillo con funciones fft, rfft y ifft

- import numpy as np
- from scipy.fft import fft, rfft, ifft
- x = np.array([1.0, 2.0, 1.0, -1.0, 1.5, 1.0])
- y1 = fft(x); print(y1)
- y2 = rfft(x); print(y2)
- yinv = ifft(y1); print(yinv)

Ejercicio 7.2

Exercise 7.2: Detecting periodicity

In the on-line resources there is a file called `sunspots.txt`, which contains the observed number of sunspots on the Sun for each month since January 1749. The file contains two columns of numbers, the first representing the month and the second being the sunspot number.

a) Write a program that reads the data in the file and makes a graph of sunspots as a function of time. You should see that the number of sunspots has fluctuated on a regular cycle for as long as observations have been recorded. Make an estimate of the length of the cycle in months.

b) Modify your program to calculate the Fourier transform of the sunspot data and then make a graph of the magnitude squared $|c_k|^2$ of the Fourier coefficients as a function of k—also called the *power spectrum* of the sunspot signal. You should see that there is a noticeable peak in the power spectrum at a nonzero value of k. The appearance of this peak tells us that there is one frequency in the Fourier series that has a higher amplitude than the others around it—meaning that there is a large sine-wave term with this frequency, which corresponds to the periodic wave you can see in the original data.

Transformada rápida de Fourier (FFT)

Ejemplo con suma de dos funciones seno

- import numpy as np
- import matplotlib.pyplot as plt
- from scipy.fft import fft, fftfreq
-
- N = 600 # Number of sample points
- T = 1.0/1000.0 # sample spacing
- x = np.linspace(0.0, N*T, N)
- y = np.sin(50*2.0*np.pi*x) + 0.5*np.sin(80*2.0*np.pi*x)
- yf = fft(y)
- xf = fftfreq(N, T)[:N//2] # Entrega frecuencias de muestreo
- plt.clf(); plt.plot(xf, np.abs(yf[0:N//2])); plt.grid()

Transformada rápida de Fourier (FFT)

Ejemplo: Para muchas señales solo los primeros coeficientes de la Transformada Discreta de Cosenos (DCT) tienen magnitud significativa. Se pueden convertir a cero los demás coeficientes y esto no produce errores significativos a la hora de reconstruir la señal. Esto se utiliza por ejemplo a la hora de comprimir archivos JPEG.

- Veremos ejemplo de recontruir señal con los primeros 10, 15, 20 y con los primeros 25 coeficientes de la transformada DCT. Veremos qué error se obtiene de estas reconstrucciones.

Transformada rápida de Fourier (FFT)

- import numpy as np
- N = 100; t = np.linspace(0,20,N)
- x = np.exp(-t/3)*np.cos(2*t)
- y = dct(x, norm='ortho')
- # Reconstruir con primeros 10 elementos
- window = np.zeros(N)
- window[:10] = 1
- yr10 = idct(y*window, norm='ortho')
- print("Error Relativo con 10 elem = ",sum(abs(x-yr10)**2)/sum(abs(x)**2)*100," %")