Computación científica LF-214

Profesor: Julio C. Marín

Departamento de Meteorología, Universidad de Valparaíso

Primer Semestre

Temas de la clase

- Transformadas de Fourier discretas
- Ejemplos y ejercicios

Transformadas de Fourier discretas

- Forma real de Series de Fourier

$$f(x) = \sum_{k=0}^{\infty} a_k \cos\left(\frac{2\pi kx}{L}\right) + \sum_{k=1}^{\infty} b_k \sin\left(\frac{2\pi kx}{L}\right)$$

$$a_k = \frac{1}{L}\int_0^L f(x)\cos\left(\frac{2\pi kx}{L}\right)dx$$

$$b_k = \frac{1}{L}\int_0^L f(x)\sin\left(\frac{2\pi kx}{L}\right)dx$$

Transformadas de Fourier discretas

- Forma compleja de series de Fourier

$$f(x) = \sum_{k=-\infty}^{\infty} \gamma_k \exp\left(i\frac{2\pi kx}{L}\right)$$

$$\gamma_k = \frac{1}{L}\int_0^L f(x)\exp\left(-i\frac{2\pi kx}{L}\right)dx$$

- Más simple algebráicamente y más simétrica. Se usa bastante en física.
- Integral de algunas funciones $f(x)$ se pueden calcular analíticamente y coeficiente se calculan exactamente
- Hay casos donde no se puede calcular integral de $f(x)$, o tenemos datos discretos (observaciones o simulaciones)

Transformadas de Fourier discretas

$$\gamma_k = \frac{1}{L}\int_0^L f(x)\exp\left(-i\frac{2\pi kx}{L}\right)dx$$

- Evaluamos expresión usando regla del trapezoide con N intervalos de tamaño $h = L/N$

$$\gamma_k = \frac{1}{L}\frac{L}{N}\left[\frac{1}{2}f(0) + \frac{1}{2}f(L) + \sum_{n=1}^{N-1} f(x_n)\exp\left(-i\frac{2\pi kx_n}{L}\right)\right]$$

- donde $x_n = \frac{n}{N}L$. Como función es periódica $=> f(0) = f(L)$

$$\gamma_k = \frac{1}{N}\sum_{n=0}^{N-1} f(x_n)\exp\left(-i\frac{2\pi kx_n}{L}\right)$$

Transformadas de Fourier discretas

$$\gamma_k = \frac{1}{N}\sum_{n=0}^{N-1} f(x_n)\exp\left(-i\frac{2\pi k x_n}{L}\right)$$

- Fórmula conveniente para aplicaciones computacionales
- Asumiendo que $y_n = f(x_n)$ son los valores de $f(x)$ en los N valores discretos x_n y haciendo uso de $x_n = \frac{n}{N}L$ llegamos a:

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

- No necesitamos saber los valores x_n ni el tamaño L del intervalo. Solo necesitamos saber y_n y el número N de valores.

Transformadas de Fourier discretas

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

- Transformada discreta de Fourier (DFT) de los valores y_n
- Hemos obtenido este resultado calculando integral usando método aproximado (Trapezoidal).
- Pudiéramos pensar que estos son resultados aproximados
- Sin embargo, DFT es en cierto sentido exacta.
- Nos permite calcular transformadas de Fourier exactas a pesar de métodos aproximados para integrales. Se puede demostrar!!!

Transformadas de Fourier discretas

- Trabajando matemáticamente con la expresión de DFT, llegamos a la siguiente expresión:

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

- Se obtiene la Transformada discreta de Fourier Inversa
- Es equivalente a DFT
- Nos permite obtener los valores y_n a partir de coeficientes c_k de manera exacta (+ errores de redondeo)

Transformadas de Fourier discretas

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

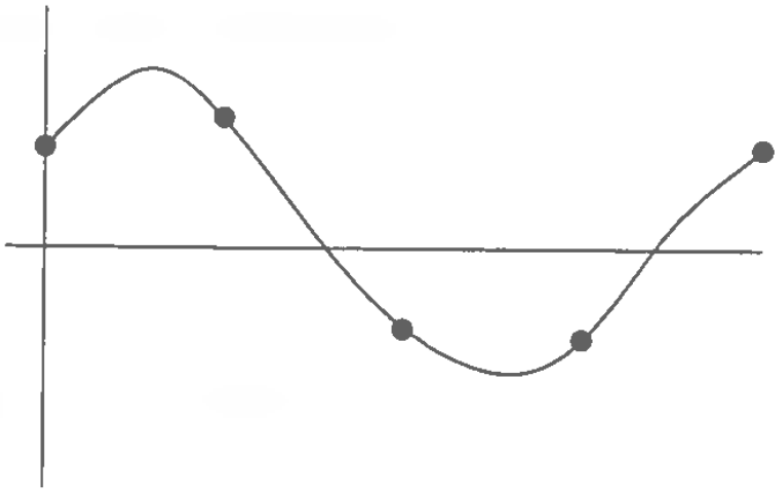
- Con estas expresiones podemos movernos libremente entre los datos y_n y los coeficientes c_k sin perder ningún detalle en datos.
- Los datos y_n y la transformada de Fourier nos dan representación completa de datos originales.
- Importante: Solo necesitamos $N-1$ coeficientes c_k para recobrar los datos y_n

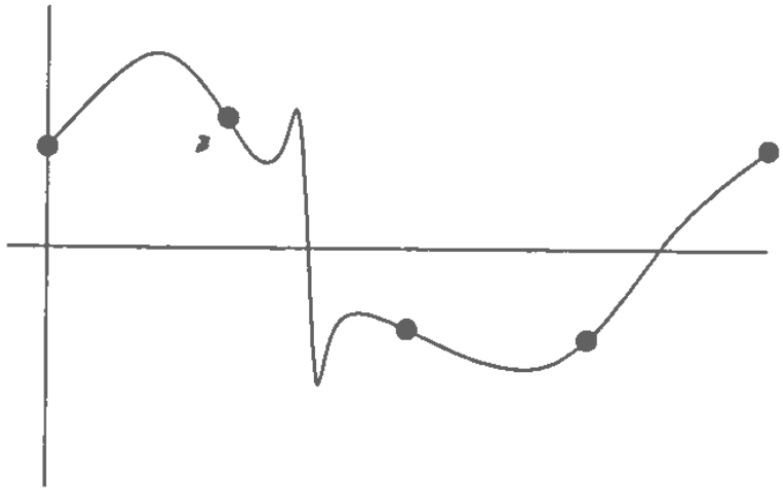
Transformadas de Fourier discretas

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

- Importante: fórmulas DFT solo entregan valores y_n
- No entregan información de $f(x)$ entre los valores x_n
- Cualquiera dos funciones que tengan los mismos puntos de muestreo (x_n), tendrán la misma DFT, no importa la forma de la función entre los puntos x_n

Transformadas de Fourier discretas





- Dos funciones tendrán la misma DFT aunque sean completamente diferentes como en ejemplo entre puntos 2 y 3.
- DFT funciona igual para funciones reales y complejas.
- En mayoría de situaciones prácticas, no s interesan las funciones reales.
- Caso de funciones reales se puede simplificar más.

Transformadas de Fourier discretas

- Si todos los valores y_n son reales, consideremos valor c_k para valor de $\frac{1}{2}N < k < N$
- Entonces $k = N - r$ donde $0 < r < \frac{1}{2}N$

$$c_{N-r} = \sum_{n=0}^{N-1} y_n \exp\left(-\mathrm{i}\frac{2\pi(N-r)n}{N}\right) = \sum_{n=0}^{N-1} y_n \exp(-\mathrm{i}2\pi n) \exp\left(\mathrm{i}\frac{2\pi rn}{N}\right)$$
$$= \sum_{n=0}^{N-1} y_n \exp\left(\mathrm{i}\frac{2\pi rn}{N}\right) = c_r^* \qquad (7.21)$$

- c_r^*: conjugado complejo de c_r y usamos expresión $e^{-i2\pi n} = 1$
- Entonces $C_{N-1} = c_1^*$, $C_{N-2} = c_2^*$, ... , etc

Si calculamos DFT para función real, solo necesitamos calcular coeficientes c_k para $0 < k < \frac{1}{2}N$

Transformadas de Fourier discretas

Si calculamos DFT para función real, solo necesitamos calcular coeficientes c_k para $0 < k < \frac{1}{2}N$

- Otra mitad de coeficientes son solo los complejos conjugados de primera mitad
- Si N es par => solo necesitamos calcular $\frac{1}{2}N + 1$ coeficientes
- Si N es impar => necesitamos calcular $\frac{1}{2}(N + 1)$ coeficientes
- Esto se escribe en python de la misma forma: "N//2+1"
- // es operador que entrega entero de la división entre 2 enteros
- Si función es compleja, tenemos que calcular los N coeficientes

Ejercicio para usar DFT

Ejercicio: Calcular los coeficientes c_k de la DFT para la función $f(x) = \sin(x)$ en el intervalo $[-\pi, \pi]$

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

Ejercicio para usar DFT

```
import numpy as np

def dft(y):
    N = len(y)
    c = np.zeros(N//2+1, complex)
    for k in range(N//2+1):
        for n in range(N):
            c[k] += y[n]*exp(-2j*np.pi*k*n/N)
    return c
xx = np.arange(-np.pi, np.pi+0.1, 0.1); y = np.sin(xx)
cc = dft(y)
print(cc)
```

Ejercicio para usar la DFT inversa

Ejercicio: Calcular la DFT inversa para recuperar la función $f(x) = \sin(x)$ en el intervalo $[-\pi, \pi]$ a partir de los coeficientes c_k. Plotear la función $f(x) = \sin(x)$ y la función recuperada de DFT inversa para comprobar el resultado.

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

Ejercicio para usar la DFT inversa

Ejercicio: Calcular la DFT inversa para recuperar la función $f(x) = \sin(x)$ en el intervalo $[-\pi, \pi]$ a partir de los coeficientes c_k. Plotear la función $f(x) = \sin(x)$ y la función recuperada de DFT inversa para comprobar el resultado.

- def idft(c):
- N = 2*(len(c) - 1); yn = np.zeros(N, complex)
- for n in range(N):
- for k in range(N//2+1):
- yn[n] += c[k]*np.exp(2j*np.pi*k*n/N)/(N//2+1)
- return yn
- yn = idft(cc)
- plt.clf(); plt.plot(xx, y, '-b', xx, yn, '-r')
- plt.hlines(0, -np.pi, np.pi+0.1, color='k', linewidth=2)
- plt.ylabel('Sin(x)'); plt.xlabel('X')
- plt.legend(('sin(x)', 'IDFT'), loc=1)