Tecnológico de Monterrey

Actividad 2

Divide y Vencerás:

Transformada Rápida de Fourier

Diego Andrés Figueroa Peart
A01660987

Curso:

Análisis y diseño de algoritmos avanzados
(TC2038.652)

Profesor:

Cristhian Alejandro Ávila Sánchez

Fecha de entrega:

18 de octubre de 2023

Actividad 2. Divide y Vencerás: Transformada Rápida de Fourier

Planteamiento del problema

La transformada de Fourier es un algoritmo matemático que se utiliza para transformar señales del dominio del tiempo al dominio de la frecuencia, es decir, para encontrar las frecuencias individuales que componen a una señal. El algoritmo desarrollado en esta actividad, la transformada rápida de Fourier, es un método más rápido de calcular esta misma, inventado por Carl Friedrich Gauss y redescubierto por James Cooley y John Tukey.

Datos de entrada:

- N = El tamaño del arreglo que representa a la señal.
- f = El arreglo de números entre 0.0 y 1.0 que representan la señal.

Datos de salida:

- F = El arreglo que representa las frecuencias que componen a la señal de entrada.

Funcionamiento:

1. Se define el tamaño del arreglo que representará a la señal (N).
2. Se genera un arreglo de tamaño N de números aleatorios entre 0.0 y 1.0 (f).
3. Se llama a la función calcularTransformadaRapidaFourier.
4. Se genera un arreglo de ceros de tamaño N, que podrá albergar números complejos. (F)
5. Se define $W_N = e^{-i2\pi mn/N}$ y $W = 1$.
6. Se crean subarreglos de f que contienen solo los índices pares e impares (f_par y f_impar).
7. Se calcula la transformada de Fourier recursivamente para estos subarreglos.
8. Se calcula la transformada de Fourier para el arreglo completo y se almacena en F.
9. Se devuelve F.

Pseudocódigo:

```
Espectro calcularTransformadaRapidaFourier(Señal f, int N)
{
	Espectro F;
	if (N==1)
		return(f);
	WN = e–i2π/N;
	W= 1;
	f.par= [f0, f2, ... , fN–2];
	f.impar= [f1, f3, ... , fN–1];
	A = calcularTransformadaRapidaFourier(f.par, N/2);
	B = calcularTransformadaRapidaFourier(f.impar, N/2);
	for (k=0; k<N/2; k++)
	{
		F[k]= A[k] + W*B[k];
		F[k+N/2]= A[k] - W*B[k];
		W= W * WN;
	}
	return(F);
}
```

Complejidad:

El algoritmo de transformada rápida de Fourier tiene la misma complejidad en todos los casos: O(n log n). Esto se debe a su estrategia de divide y vencerás; al calcular la transformada de Fourier recursivamente dividiendo la señal en sus partes pares e impares, se logra una eficiencia mucho mayor a la manera estándar de calcularla.

Programa:

```
import numpy

def calcularTransformadaRapidaFourier(f, N):
    F = numpy.zeros(N, dtype=complex)
    if N == 1:
        return f
    W_n = numpy.exp(-2j*numpy.pi/N)
    W = 1
    f_par = f[::2]
    f_impar = f[1::2]
    A = calcularTransformadaRapidaFourier(f_par, N//2)
    B = calcularTransformadaRapidaFourier(f_impar, N//2)
    for k in range(N//2):
        F[k] = A[k] + W*B[k]
        F[k + N//2] = A[k] - W*B[k]
        W *= W_n
    return F

N = 2048
f = numpy.random.random(N)
print(f)

print(calcularTransformadaRapidaFourier(f, N))
```