Universidad Católica
San Pablo

Ciencia de la Computación

Análisis y Diseño de Algoritmos

Transformada de Fourier

Roberto Cayro Cuadros

Luis Arroyo Pinto

2022-1

"Los alumnos declaran haber realizado el presente trabajo de acuerdo a las normas de la Universidad Católica San Pablo"

_____ _____

Contents

1 Introducción

En este informe se explica el algoritmo de la Transformación Rápida de Fourier, mostrando su definición, funcionamiento para el caso de un audio
La Transformada de Fourier, en el campo de la programación, es un conocido algoritmo que permite la descomposicion de una señal en las señales puras que las descomponen. Esto permite el analisis de audio mostrando el espectro de las frecuencias que componen a dicho audio.

2 Funciones periódicas

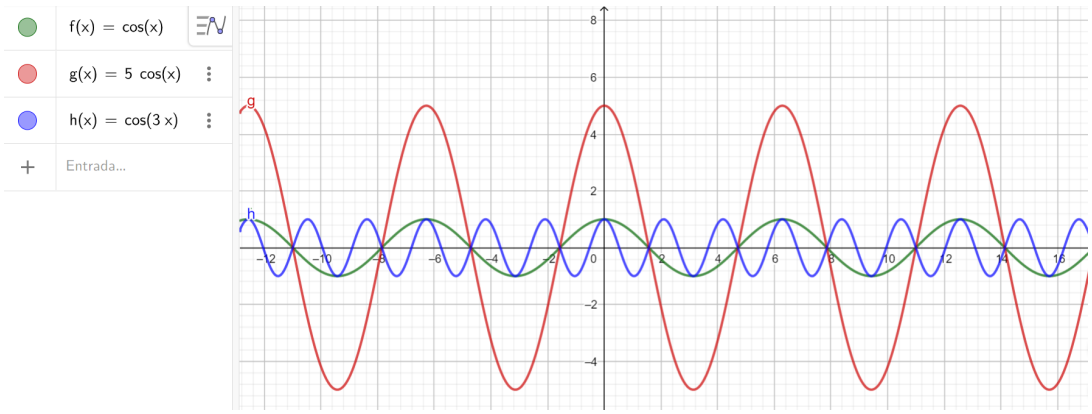
Seno y Coseno son aquellas funciones periódicas que son usadas para representar a las señales de audio de la forma:

$$f(x) = A\ cos(wx) \qquad ; \qquad f(x) = A\ sen(wx)$$

Donde A va a ser la amplitud y determinara la altura máxima a la pueden llegar las curvas de igual forma esta determinaria el rango de la función, *f(x) = 3 cos(x) Rango = [-3, 3]*

Y donde w será igual al periodo, y este determinara la longitud de intervalo que toma a la gráfica repetirse. Estará dado por:

$$P{=}2\pi\frac{1}{w}$$



3 Fourier-Series

Nosotros seremos capaces de descomponer las señales de audio tomando en cuenta estas funciones periódicas. Sea p(t) una funcion en L1 y asuma que esta es periódica de periodo l, al igual que un vector r en un plano R2, nosotros podemos descomponer el vector de forma ortogonal dado una bases b1 y b2, donde r=r1b1+r2b2, de la misma forma nosotros podemos descomponer la funcion p como por ejemplo en:

$$x_k = \sum_{n=0}^{\infty} a_n cos(\tfrac{n\pi}{l}) + b_n sen(\tfrac{n}{l})$$

La serie de Fourier representa la sintesis de una funcion peródica s(x) al sumar sinusoides relacionados armonicamente llamados harmonics. Estas series son representadas en: amplitude-phase form, sine-cosine form, exponential form.[Fou22b]

1) Amplitude-phase form:

$$s(x) = \sum_{n=1}^{N} A_n cos(\tfrac{2\pi}{P} nx - \phi_n) + (\tfrac{A_0}{2})$$

A) Donde el armonico nth *será igual a* :

$Cos(2\pi \frac{1}{P} nx - \phi_n)$

B) A_n será la amplitud del harmonic y phi(n) su cambio de fase

2) Sine-Cosine-phase form:

$$x_k = (\sum_{n=0}^{\infty} a_n cos(\frac{2nx\pi}{P}) + b_n sen(\frac{2nx}{P})) + \frac{a_0}{2}$$

A) Equivalencia de la forma polar y cartesiana

$$Cos(2\pi \frac{1}{P} nx - \phi_n) = cos(\phi_n) * cos(2\pi \frac{1}{P} nx) + sen(\phi_n) * sen(2\pi \frac{1}{P} nx)$$

B) Dado $cos(\phi_n) = a_n/A_n$ *y* $sen(\phi_n) = b_n/B_n$*tomando en cuenta los puntos A, B y reemplazandolos en la Amplitude-phase form se obtiene esta nueva forma Sine-Cosine-phase form*

3) Exponential form:

$$x_k = \sum_{n=-N}^{N-1} x_n e^{i2\pi nx/P}$$

A) Tomando en cuenta la formula de Euler: $e^{ix} = cos(x) + isen(x)$

$$Cos(2\pi \frac{1}{P} nx - \phi_n) = 1/2e^{i(2\pi nx/P - \phi_n)} + 1/2e^{-i(2\pi nx/P - \phi_n)}$$

4 Funcionamiento-fourier

Fourier transforma una señal desde su dominio original a una representacion en la frecuencia dominio, a diferencia de la transformación discreta que tiende a ser lenta en la práctica, la fft consigue esta descomposición al factorizar la DFT-matrix en un reducido grupo de factores en su mayoría 0, como resultado su complejidad es reducida de[Fou22a]:

$O(N^2)$ *a* $O(NlogN)$

Definición-DFT:

$$x_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}$$

Un algoritmo fft es cualquier algoritmo que pueda computar los resutados de la transformación en un tiempo *NlogN*

Cooley tukey algorithm:
Modifica la DFT de un tamaño N=N1, N2 en terminos de N1 detamaños DFT N2, forma recursiva, de esta forma se consigue el tiempo *NlogN*. El algoritmo divide el DFT original en DFTs más pequeños[Coo22]

El caso Radix-2 DIT

Divide un DFT de tamaño N en 2 intervalos DFTs de tamaño N/2 cada uno, de forma recursiva. Radix-2 DIT resuelve primero las entradas pares: *x0,x2...xN-2*, las impares *x1,x3,...xN-1*, para luego combinar ambos y obtener los resultados de toda la secuencia. Radix-2 DIT conseguira esta division separando el algoritmo DFT en 2 partes de acuerdo al indice par n=2m o impar n=2m+1:

$$x_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-i2\pi(2m)k/N} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-i2\pi(2m+1)n/N}$$

Esta claro entonces que esta division será la clave para acelerar nuestro algoritmo. *Establecemos que la suma de los indices pares será igual a* E_k*, mientras que la suma de los indices impares sera* O_k

$$X_k = E_k + e^{-i2\pi kn/N} O_k$$

5 Diagrama de Mariposa

Cuando se realiza el algoritmo de Cooley-Tukey para poder realizar la transformación rápida de Fourier, este algoritmo descompone recursivamente una DFT's de tamaño n, luego estas DFT's se combinan. Estas DFT's seran multiplicadas por la raiz de unidad w_n^k que es igual a $e^{-\frac{2\pi ik}{n}}$.
Para el caso de un radix-2 se tendra que el diagrama de mariposa tendra dos inputs(x_0, x_1) y dos outputs(y_0, y_1) teniendo la siguiente formula.
$y_0 = x_0 + x_1 w_n^k$
$y_1 = x_0 + x_1 w_n^k$

6 Código

Listing 1: Example C++

```
1
2 //
3 void fft1(int* t, complex<double>* muestras_1, int tam) {
4
5
6   for (int i = 0; i < tam; i++) {
7     muestras_1[i] = complex<double>(t[i], 0);
8     muestras_1[i] *= 1;
9   }
10
11
12   fft2(muestras_1, tam);
13 }
```

Listing 2: Example C++

```
void fft2(complex<double>* m1, int t) {

  if (t <= 1) {
    return;
  }

  //cout << "t=" << t << endl;
  complex<double>* par = new complex<double>[t / 2];
  complex<double>* imp = new complex<double>[t / 2];

  for (int i = 0; i < t; i++) {
    par[i] = m1[i * 2];
    imp[i] = m1[i * 2 + 1];
  }


  fft2(par, t / 2);
  fft2(imp, t / 2);

  for (int i = 0; i < t / 2; i++) {
    complex<double> x = exp(complex<double>(0, -2 * 3.141592 * i / t)) * imp[i];
    m1[i] = par[i] + x;
    m1[t / 2 + i] = par[i] - x;
  }

}
```

7 Funcionamiento-fourier-inversa

La inversa de la transformada de fourier consiste en que una vez obtenida la funcion de fourier es posible obtener la inversa de esta. Esto es debido a que una vez conocida todas las frencuencias que componen un audio u onda, es posible reconstruir con precision el audio u onda original[FFT22].

Por lo tanto esto nos permite convertir de una señal en el dominio de frecuencia en una en el dominio del tiempo. Para ello se basa en la siguiente formula:

$f_t = \frac{1}{2*\pi} \int_{-\infty}^{\infty} F(W) * e^{i*w*t}\, dw$

References

[Coo22] *Cooley–tukey fft algorithm. https://en.wikipedia.org/wiki/Cooley–Tukey_FFT_algorithm, June 2022.*

[FFT22] *Understanding audio data, fourier transform, fft and spectrogram features for a speech recognition system. https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520, June 2022.*

[Fou22a] *Fast fourier transform. https://en.wikipedia.org/wiki/Fast_Fourier_transform, June 2022.*

[Fou22b] *Fourier series. https://en.wikipedia.org/wiki/Fourier_series, June 2022.*