FFT: un algoritmo *algo* importante

Introducción a la Física Computacional – Instituto Balseiro

Outline

1. Transformada de Fourier
2. Representación
3. Representación de polinomios
4. Diseño de un algoritmo
5. FFT

Transformada de Fourier 1D

$$F(u) = \int_{-\infty}^{\infty} f(x)\, e^{-i2\pi ux}\, dx$$

$$f(x) = \int_{-\infty}^{\infty} F(u)\, e^{i2\pi ux}\, du$$

Unidades: $[u] = 1/[x]$

En el ámbito de la Física

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x)\, e^{-ikx}\, dx$$

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(k)\, e^{ikx}\, dk$$

$$k = \omega/c = 2\pi\nu/c$$

Transformada de Fourier 2D

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)\, e^{-i2\pi(ux+vy)}\, dxdy$$

$$f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u,v)\, e^{i2\pi(ux+vy)}\, dudv$$

Transformada de Fourier Discreta (DFT)

$$F_u = \sum_{x=0}^{n-1} f_x \, e^{-i2\pi ux/n}$$

$$f_x = \frac{1}{n} \sum_{u=0}^{n-1} F_u \, e^{i2\pi ux/n}$$

Transformada de Fourier Discreta (DFT)

$$F_u = \sum_{x=0}^{n-1} f_x \, e^{-i2\pi ux/n}$$

$$f_x = \frac{1}{n} \sum_{u=0}^{n-1} F_u \, e^{i2\pi ux/n}$$

Premisa oculta:

Transformada de Fourier Discreta (DFT)

$$F_u = \sum_{x=0}^{n-1} f_x \, e^{-i2\pi ux/n}$$

$$f_x = \frac{1}{n} \sum_{u=0}^{n-1} F_u \, e^{i2\pi ux/n}$$

Premisa oculta: dominio periódico

Transformada de Fourier Discreta (DFT)

$$F_u = \sum_{x=0}^{n-1} f_x \, e^{-i2\pi ux/n}$$

$$f_x = \frac{1}{n} \sum_{u=0}^{n-1} F_u \, e^{i2\pi ux/n}$$

Premisa oculta: dominio periódico
Consecuencia:

Transformada de Fourier Discreta (DFT)

$$F_u = \sum_{x=0}^{n-1} f_x \, e^{-i2\pi ux/n}$$

$$f_x = \frac{1}{n} \sum_{u=0}^{n-1} F_u \, e^{i2\pi ux/n}$$

Premisa oculta: dominio periódico
Consecuencia: F_u y f_x periódicas

Transformada de Fourier Discreta (DFT)

$$F_u = \sum_{x=0}^{n-1} f_x \, e^{-i2\pi ux/n}$$

$$f_x = \frac{1}{n} \sum_{u=0}^{n-1} F_u \, e^{i2\pi ux/n}$$

Ya veremos de dónde sale ese $1/n$.
Primero, un ligero desvío...

La representación en Física

- ¿Qué significa representar?
- ¿Cuántas formas de representar algo hay?
- ¿Cuál es la verdadera?

Ejemplos clásicos

- posición (x) $\longleftrightarrow$ vector de onda (k)

Ejemplos clásicos

- posición (x) $\longleftrightarrow$ vector de onda (k)
- tiempo (t) $\longleftrightarrow$ frecuencia (ω)

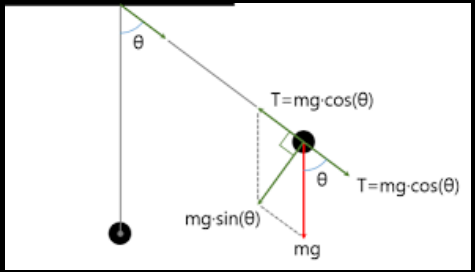
Ejemplos clásicos

- posición (x) $\longleftrightarrow$ vector de onda (k)
- tiempo (t) $\longleftrightarrow$ frecuencia (ω)
- partícula $\longleftrightarrow$ onda

Ejemplos clásicos

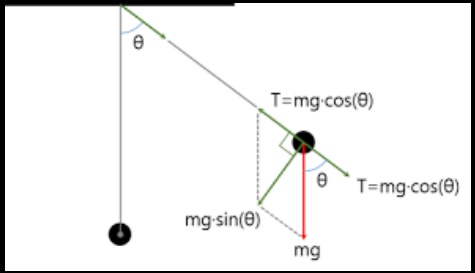
- posición (x) $\longleftrightarrow$ vector de onda (k)
- tiempo (t) $\longleftrightarrow$ frecuencia (ω)
- partícula $\longleftrightarrow$ onda
- inercial $\longleftrightarrow$ no inercial

¿Qué es el péndulo simple?



- ¿Un oscilador armónico?
- ¿Una masa colgada de un hilo?
- ¿El sistema masa - hilo - planeta?
- ¿Un gran conjunto de átomos interactuando?

¿Qué es el péndulo simple?



- ¿Un oscilador armónico?
- ¿Una masa colgada de un hilo?
- ¿El sistema masa - hilo - planeta?
- ¿Un gran conjunto de átomos interactuando?

Conceptos $\Longleftrightarrow$ Representación

Representación de polinomios

Vector de coeficientes (n coeficientes):

$$A(x) = A_0 + A_1 x + A_2 x^2 + ... + A_{n-1}^{n-1}$$
$$\equiv \quad < A_0,\ A_1,\ A_2, ...,\ A_{n-1} >$$

Representación de polinomios

Vector de coeficientes (n coeficientes):

$$\begin{aligned} A(x) &= A_0 + A_1 x + A_2 x^2 + ... + A_{n-1}^{n-1} \\ &\equiv \ < A_0,\ A_1,\ A_2, ...,\ A_{n-1} > \end{aligned}$$

Raíces (n-1 puntos especiales + 1 coeficiente):

$$\begin{aligned} A'(x) &= A_{n-1}(x - x'_0)(x - x'_1)...(x - x'_{n-2}) \\ &\equiv \ A_{n-1} < x'_0,\ x'_1,\ x'_2, ...,\ x'_{n-2} > \end{aligned}$$

Representación de polinomios

Vector de coeficientes (n coeficientes):

$$\begin{aligned} A(x) &= A_0 + A_1 x + A_2 x^2 + ... + A_{n-1}^{n-1} \\ &\equiv \ < A_0,\ A_1,\ A_2, ...,\ A_{n-1} > \end{aligned}$$

Raíces (n-1 puntos especiales + 1 coeficiente):

$$\begin{aligned} A'(x) &= A_{n-1}(x - x'_0)(x - x'_1)...(x - x'_{n-2}) \\ &\equiv \ A_{n-1} < x'_0,\ x'_1,\ x'_2, ...,\ x'_{n-2} > \end{aligned}$$

Representación por muestras (n puntos diferentes):

$$\begin{aligned} A(x_k) &= A_k^* \\ A(x) \equiv A^*(x) &= \ < (x_0,\ A_0^*), (x_1,\ A_1^*), ..., (x_{n-1},\ A_{n-1}^*) > \end{aligned}$$

Ventajas y desventajas

Rep.	Evaluación	Suma	Multiplicación
$A(x)$	$O(n)$	$O(n)$	$O(n^2)$
$A'(x)$	$O(n)$	$''\infty''$	$O(n)$
$A^*(x)$	$O(n^2)$	$O(n)$	$O(n)$

Rep.	$A(x)$	$A'(x)$	$A^*(x)$
$A(x)$	-	$''\infty''$	$O(n^2)$
$A'(x)$	$''\infty''$	-	$''\infty''$
$A^*(x)$	$O(n^{3-2})$	$''\infty''$	-

Matriz de Vandermonde

$$\underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-1} \end{pmatrix}}_{V} \cdot \underbrace{\begin{pmatrix} A_0 \\ A_1 \\ . \\ . \\ . \\ A_{n-1} \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} A_0^* \\ A_1^* \\ . \\ . \\ . \\ A_{n-1}^* \end{pmatrix}}_{A^*}$$

$$V_{kj} = x_k^j$$

Cambio de representación

$$V \cdot A = A^*$$

Coeficientes $\longrightarrow$ Muestras: $V \cdot A$ $O(n^2)$

Muestras $\longrightarrow$ Coeficientes: $V^{-1} \cdot A^*$ $O(n^3) - O(n^2)$

Cambio de representación

$$V \cdot A = A^*$$

Coeficientes $\longrightarrow$ Muestras: $V \cdot A$ $O(n^2)$

Muestras $\longrightarrow$ Coeficientes: $V^{-1} \cdot A^*$ $O(n^3) - O(n^2)$

¿Qué podemos hacer para mejorar?

¿Qué libertades tenemos?

- Explotar simetrías del problema
- Imponer restricciones que simplifiquen el problema
- Elegir los x_k que más nos convengan

¿Qué vamos a hacer?

- Elegir una simetría conveniente
- Imponer las restricciones necesarias para poder explotarla con nuestro algoritmo
- Elegir los x_k en base a lo anterior

¿Cómo lo vamos a hacer?

- Simetría por excelencia en binario: paridad
- Regla N°1 de la programación: divide y conquista
- Conjunto de x_k: colapsable

¿Cómo lo vamos a hacer?

- Simetría por excelencia en binario: paridad
- Regla N°1 de la programación: divide y conquista
- Conjunto de x_k: colapsable

¿Qué significa todo esto?

Algoritmo recursivo

$$A(x) = \underbrace{A_p(x^2)}_{< A_0,\ A_2, \ldots >} + x\ \underbrace{A_i(x^2)}_{< A_1,\ A_3, \ldots >}$$

Es decir:

$$A_p(z) = \sum_{l=0}^{n/2-1} A_{2l}\, z^l$$
$$A_i(z) = \sum_{l=0}^{n/2-1} A_{2l+1}\, z^l$$

Algoritmo recursivo

$$A(x) = \underbrace{A_p(x^2)}_{< A_0,\ A_2, \ldots >} + x\ \underbrace{A_i(x^2)}_{< A_1,\ A_3, \ldots >}$$

Es decir:

$$A_p(z) = \sum_{l=0}^{n/2-1} A_{2l}\, z^l$$
$$A_i(z) = \sum_{l=0}^{n/2-1} A_{2l+1}\, z^l$$

Condición para que esto se cumpla:

Algoritmo recursivo

$$A(x) = \underbrace{A_p(x^2)}_{< A_0,\ A_2, \ldots >} + x\ \underbrace{A_i(x^2)}_{< A_1,\ A_3, \ldots >}$$

Es decir:

$$A_p(z) = \sum_{l=0}^{n/2-1} A_{2l}\, z^l$$
$$A_i(z) = \sum_{l=0}^{n/2-1} A_{2l+1}\, z^l$$

Condición para que esto se cumpla: n potencia de 2

Conjunto X:

$$x_k \in X \iff x_k \neq x_{k'} \quad \text{si} \quad k \neq k' \ , \ k = 0, 1, 2, \ldots, n-1$$

Tamaño de $X \longrightarrow$ n

Conjunto X:

$$x_k \in X \iff x_k \neq x_{k'} \quad \text{si} \quad k \neq k' \ , \ k = 0, 1, 2, \ldots, n-1$$

Tamaño de $X \longrightarrow$ n

Conjunto X^2:

$$x_h \in X^2 \iff x_h = x_k^2 \ , \ x_k \in X$$

¿Qué tamaño tendrá el conjunto X^2?

Conjunto X:

$$x_k \in X \iff x_k \neq x_{k'} \quad \text{si} \quad k \neq k' \ , \ k = 0, 1, 2, \ldots, n-1$$

Tamaño de $X \longrightarrow$ n

Conjunto X^2:

$$x_h \in X^2 \iff x_h = x_k^2 \ , \ x_k \in X$$

¿Qué tamaño tendrá el conjunto X^2?

Depende del conjunto X elegido

Raíces n-ésimas de la unidad

Si elegimos como muestras los $x_k \quad / \quad x_k^n = 1$
$\Longrightarrow$ Conjunto X es colapsable.

i.e. $\longrightarrow$ Tamaño X^2 = (Tamaño X)/2

Raíces n-ésimas de la unidad

Si elegimos como muestras los $x_k \quad / \quad x_k^n = 1$
$\implies$ Conjunto X es colapsable.

i.e. $\longrightarrow$ Tamaño X^2 = (Tamaño X)/2

$\Longrightarrow$

$$x_k = e^{i\theta}$$

$$\theta = 0, \quad -\frac{2\pi}{n}, \quad -\frac{4\pi}{n}, \ldots, \quad -\frac{2\pi(n-1)}{n}$$

Repasemos el algoritmo

- Limitar n $=$ potencia de 2
- Elegir x_k las n-ésimas raíces de 1
- Evaluar recursivamente la función
- Listo!

¿Y si queremos pasar de A^* a A?

Mismo algoritmo, distintos coeficientes:

$$A = V^{-1} \cdot A^*$$

¿Y si queremos pasar de A^* a A?

Mismo algoritmo, distintos coeficientes:

$$A = V^{-1} \cdot A^*$$

Pero invertir una matriz es costoso, ¿no?

¿Y si queremos pasar de A^* a A?

Mismo algoritmo, distintos coeficientes:

$$A = V^{-1} \cdot A^*$$

Pero invertir una matriz es costoso, ¿no?

No si usamos las n-ésimas raíces de 1:

$$V^{-1} = \frac{1}{n}\bar{V} \qquad , \qquad \bar{V}_{kj} = e^{i2\pi kj/n}$$

Repasemos lo que hicimos

$$\underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-1} \end{pmatrix}}_{V} \cdot \underbrace{\begin{pmatrix} A_0 \\ A_1 \\ . \\ . \\ . \\ A_{n-1} \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} A_0^* \\ A_1^* \\ . \\ . \\ . \\ A_{n-1}^* \end{pmatrix}}_{A^*}$$

$$V_{kj} = x_k^j = e^{-i2\pi kj/n}$$

Repasemos lo que hicimos

$$V \cdot A = A^*$$

Función (A) $\longrightarrow$ Transformada (A^*):

$$V \cdot A \qquad O(nlog_2 n)$$

Transformada (A^*) $\longrightarrow$ Función (A):

$$V^{-1} \cdot A^* \qquad O(nlog_2 n))$$

Repasemos lo que hicimos

$$A_k^* = V_{kj} \cdot A_j = \sum_{j=0}^{n-1} A_j \, e^{-i2\pi kj/n}$$

$$A_j = \frac{1}{n}\bar{V}_{kj} \cdot A_k^* = \frac{1}{n}\sum_{k=0}^{n-1} A_k^* \, e^{i2\pi kj/n}$$

¿Les resulta familiar?

Repasemos lo que hicimos

$$F_u = \sum_{x=0}^{n-1} f_x \, e^{-i2\pi ux/n} \qquad A_k^* = \sum_{j=0}^{n-1} A_j \, e^{-i2\pi kj/n}$$

$$f_x = \frac{1}{n} \sum_{u=0}^{n-1} F_u \, e^{i2\pi ux/n} \qquad A_j = \frac{1}{n} \sum_{k=0}^{n-1} A_k^* \, e^{i2\pi kj/n}$$

Repasemos lo que hicimos

$$F_u = \sum_{x=0}^{n-1} f_x \, e^{-i2\pi ux/n} \qquad A_k^* = \sum_{j=0}^{n-1} A_j \, e^{-i2\pi kj/n}$$

$$f_x = \frac{1}{n} \sum_{u=0}^{n-1} F_u \, e^{i2\pi ux/n} \qquad A_j = \frac{1}{n} \sum_{k=0}^{n-1} A_k^* \, e^{i2\pi kj/n}$$

El algoritmo diseñado es la versión clásica de la FFT

Ejemplos de aplicar la FFT en 2D

Figuras y gifs