

Fourieranalyse

Fast Fourier Transform

Team G



Overzicht

- ① Inleiding
- ② Waarom werkt DFT zoveel trager dan FFT?
- ③ FFT
- ④ Besluit

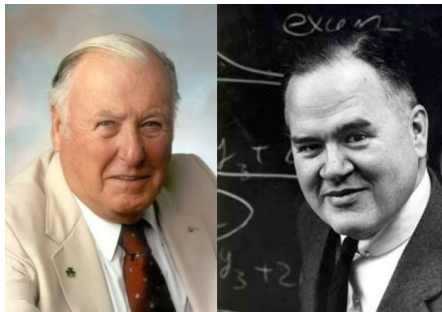
Overzicht

- ① Inleiding
- ② Waarom werkt DFT zoveel trager dan FFT?
- ③ FFT
- ④ Besluit

Inleiding

- ▶ James Cooley & John Tukey
- ▶ DFT = traag!
- ▶ Oplossing: FFT = snel!
- ▶ Toepassingen: noise cancelling, complexe DVGL, ...

Figuur: James Cooley & John Tukey



Overzicht

- ① Inleiding
- ② Waarom werkt DFT zoveel trager dan FFT?
- ③ FFT
- ④ Besluit

Waarom werkt DFT zoveel trager dan FFT?

DFT

- ▶ formules:

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

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{i2\pi kn/N}$$

- ▶ efficiëntie: $\mathcal{O}(N^2)$
- ▶ vermenigvuldigingen: N^2
- ▶ optellingen: $N(N-1)$

FFT

- ▶ formules: zie verder
- ▶ efficiëntie: $\mathcal{O}(N \log_2 N)$
- ▶ vermenigvuldigingen: $\frac{N}{2} \log_2 N$
- ▶ optellingen: $N \log_2 N$

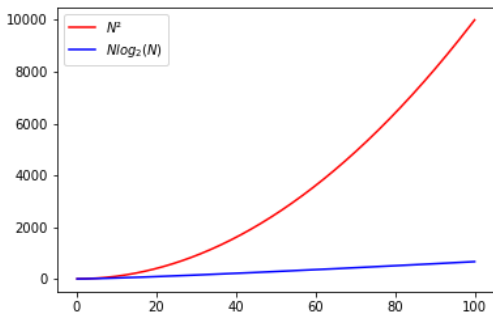
Complexiteit van DFT vs. FFT

Figuur: Aantal wiskundige operaties voor DFT en FFT

N	DFT			Radix-2		
	CM N^2	CA $N(N-1)$	Total	CM $(N/2)\log_2 N$	CA $N\log_2 N$	Total
2	4	2	6	1	2	3
4	16	12	28	4	8	12
8	64	56	120	12	24	36
16	256	240	496	32	64	96
32	1,024	992	2,016	80	160	240
64	4,096	4,032	8,128	192	384	576
128	16,384	16,256	32,640	448	896	1,344
256	65,536	65,280	130,816	1,024	2,048	3,072
512	262,144	261,632	523,776	2,304	4,608	6,912
1,024	1,048,576	1,047,552	2,096,128	5,120	10,240	15,360
2,048	4,194,304	4,192,256	8,386,560	11,264	22,528	33,792
4,096	16,777,216	16,773,120	33,550,336	24,576	49,152	73,728

Complexiteit van DFT vs. FFT

Figuur: Complexiteit van beide algoritmes



Voorbeeld:

Geluidsbestand van enkele seconden
262.144 datapunten ($N = 262.144$)

$$\frac{\mathcal{O}(N^2)}{\mathcal{O}(N \log_2 N)} \approx 12.000$$

\Rightarrow 12.000 keer meer berekeningen

Overzicht

- ① Inleiding
- ② Waarom werkt DFT zoveel trager dan FFT?
- ③ FFT**
- ④ Besluit

Hoe dit wordt opgelost in de FFT

- ▶ Periodiciteit en symmetrie van sinus- en cosinusfuncties
- ▶ Datapunten met zelfde informatie
- ▶ Grote problemen opsplitsen
- ▶ Machten van 2
- ▶ Matrixproduct

DFT-algoritme

$$\begin{Bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{Bmatrix} = \begin{Bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 \\ \omega^0 & \omega^2 & \omega^4 & \omega^6 \\ \omega^0 & \omega^3 & \omega^6 & \omega^9 \end{Bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} \quad (1)$$

A_n = getransformeerde functiewaarde

$\omega = e^{-2\pi i/n}$

a_n = datapunten

Working FFT

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{pmatrix} = \begin{pmatrix} I_2 & -D_2 \\ I_2 & -D_2 \end{pmatrix} \cdot \begin{pmatrix} F_2 & 0_2 \\ 0_2 & F_2 \end{pmatrix} \cdot \begin{pmatrix} a_{\text{even}} \\ a_{\text{oneven}} \end{pmatrix} \quad (2)$$

Werking FFT 2

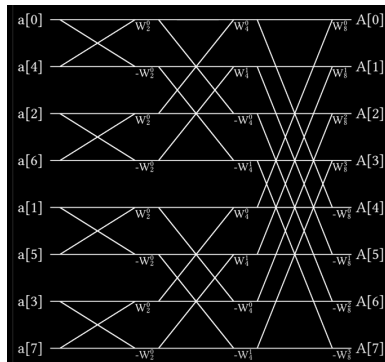
$$\begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_6 \\ A_7 \end{pmatrix} = \begin{pmatrix} I_4 & -D_4 \\ I_4 & -D_4 \end{pmatrix} \cdot \begin{pmatrix} Q_4 & 0_4 \\ 0_4 & Q_4 \end{pmatrix} \cdot \begin{pmatrix} a_{\text{even}} \\ a_{\text{oneven}} \end{pmatrix} \quad (3)$$

$$Q_4 = \begin{pmatrix} F_2 & 0_2 \\ 0_2 & F_2 \end{pmatrix} \quad (4)$$

Het vlinderdiagram

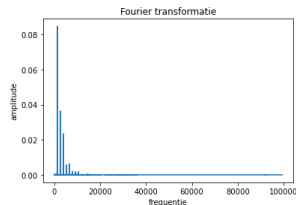
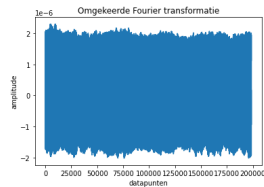
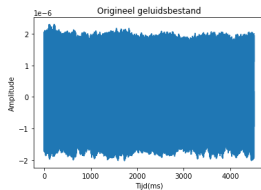
- ▶ 'array' van elementen \rightarrow tactisch door elkaar halen
- ▶ elementen terug combineren
- ▶ 1 'vlinder' = 1 DFT
- ▶ $\omega_n^k = e^{-2\pi i k/n}$
- ▶ n datapunten $\rightarrow n - 1$ simpele DFT-berekeningen

Figuur: Vlinderdiagram



Implementatie

- ▶ Geluidsbestand openen
- ▶ FFT toepassen
- ▶ Manipulatie
- ▶ RFFT toepassen
- ▶ Geluidsbestand maken



Overzicht

- ① Inleiding
- ② Waarom werkt DFT zoveel trager dan FFT?
- ③ FFT
- ④ Besluit

Besluit

- ▶ Dagelijks leven
- ▶ 2 soorten fouriertransformaties
- ▶ Verschil in uitvoeringssnelheid
- ▶ FFT gebaseerd op DFT

Referentielijst I



Todd Mateer.

Fast fourier transform algorithms with applications.
2008.



Paul S. Heckbert.








Fourier transforms and the fast fourier transform (fft) algorithm.
1998.



Steven L Brunton and J Nathan Kutz.

Data-driven science and engineering: Machine learning, dynamical systems, and control.
Cambridge University Press, 2019.

Referentielijst II

-  Michael A Peimani.
Pitch correction for the human voice.
PhD thesis, PhD. Thesis, University of California, Santa Cruz, 2009.
-  Understanding the fft algorithm.
-  Discrete fourier transform (dft).
-  Fast fourier transform (fft).
-  What makes a fourier transform fast?
-  Dft vs fft.
-  Fast fourier transform.

Referentielijst III



Discrete fouriertransformatie.