The FFT

What is the FFT?

It is an algorithm to compute the DFT fast.

What is the DFT?

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi i n k}{N}}, \quad 0 \le k \le N-1 \qquad (*)$$

cf. $\left(\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} dx, \quad \xi \in \mathbb{R} \right)$

Applications: Signal processing
Image compression
Approximation theory
PDEs
etc...

History

- Gauss originally invented the algorithm in 1805 to calculate the trajectories of asteroids by hand (published only posthumously).
- 1965: President Kennedy's Science Advisory Committee had a meeting to discuss how to detect nuclear weapons in the Soviet Union from outside the country
  - Seismometers would generate time series, needed fast way to compute DFT
  - Critical for ratification of nuclear test ban
  - John Tukey (Princeton) come up with idea, Ricard Garwin (IBM) saw potential & put him in touch with James Cooley (IBM)

- Cooley did not know original purpose, was told it was for crystallography

FFT computes (*) in $O(N \log N)$, instead of $O(N^2)$. (naive)

Top 10 algorithm

Assume N is a power of 2, $N = 2^l$

FFT is based on 3.5 facts

Fact 1: The DFT is polynomial evaluation at the roots of unity (in reverse)

$$X_k = \sum_{n=0}^{N-1} x_n \left(e^{\frac{-2\pi i k}{N}}\right)^n, \quad 0 \le k \le N-1$$

$$p(z) = \sum_{n=0}^{N-1} x_n z^n$$

$$\Rightarrow X_k = p(z_k) = \sum_{n=0}^{N-1} x_n z_k^n, \quad z_k = e^{\frac{-2\pi i k}{N}}, \quad 0 \le k \le N-1.$$

Makes the DFT easy to remember and understandable.

Fact 2: A polynomial can be decomposed into two parts

$$p(z) = x_0 z^0 + x_2 z^2 + x_4 z^4 + \cdots + z\left[x_1 z^0 + x_3 z^2 + \cdots x_{N-1} z^{\frac{N}{2}-1}\right]$$

$$\Rightarrow \underbrace{p(z)}_{\deg N-1} = \underbrace{p_{even}(z^2)}_{\deg \frac{N}{2}-1} + z\underbrace{p_{odd}(z^2)}_{\deg \frac{N}{2}-1}, \quad p_{even}(z) = x_0 + x_2 z + x_4 z^2 + \cdots$$

$$p_{odd}(z) = x_1 + x_3 z + x_5 z^2 + \cdots$$

$$\Rightarrow \sum_{n=0}^{N-1} x_n z_k^n = \sum_{n=0}^{N/2-1} x_{2n} z_k^{2n} + z \sum_{n=0}^{N/2-1} x_{2n+1} z_k^{2n}$$

Divide & Conquer

Fact 3: Roots of unity squared are roots of unity

$$z_k^2 = \left(e^{\frac{-2\pi i k}{N}}\right)^2 = e^{\frac{-2\pi i k}{N/2}}$$

$\uparrow$ $N/2$ roots of unity

Fact 3.5: If N is even, the first $\frac{N}{2}$ of the N roots of unity equal minus the last $\frac{N}{2}$.

$$z_k = -z_{N/2+k}, \quad 0 \le k \le \frac{N}{2} - 1$$

3.5 tells us that $p_{even}(z^2_{N/2+k}) = p_{even}(z_k^2)$

$z_{N/2+k}\, p_{odd}(z^2_{N/2+k}) = -z_k\, p_{odd}(z_k^2)$.

Given $p_{even}(z_k^2)$ & $p_{odd}(z_k^2)$, can compute $p(z_k)$ in N additions + $\frac{N}{2}$ mults = $\frac{3}{2}N$ flops

$$X_k = p(z_k) = p_{even}(z_k^2) + z_k\, p_{odd}(z_k^2), \quad 0 \le k \le N-1$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{\frac{-2\pi i k n}{N/2}} + e^{\frac{-2\pi i k}{N}} \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{\frac{-2\pi i k n}{N/2}}$$

DFT(N)

DFT$(\frac{N}{2})$ $\xrightarrow{\frac{3}{2}N \text{ ops}}$ DFT$(\frac{N}{2})$ $\}$ $\frac{3}{2}N$ ops

DFT$(\frac{N}{4})$ DFT$(\frac{N}{4})$ DFT$(\frac{N}{4})$ DFT$(\frac{N}{4})$ $\}$ $\frac{3}{2}\frac{N}{2} + \frac{3}{2}\frac{N}{2} = \frac{3}{2}N$ ops

$\downarrow$ $\log_2 N$ levels

So total cost is $\frac{3}{2}N \log_2 N = O(N \log N)$!

Can write this in matrix form as:

$$\begin{bmatrix} p(z_0) \\ \vdots \\ p(z_{N/2-1}) \\ p(z_{N/2}) \\ \vdots \\ p(z_{N-1}) \end{bmatrix} = \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \begin{bmatrix} p_{even}(z_0^2) \\ \vdots \\ p_{even}(z^2_{N/2-1}) \\ z_0\, p_{odd}(z_0^2) \\ \vdots \\ z_{N/2-1}\, p_{odd}(z^2_{N/2-1}) \end{bmatrix}$$

F = DFT(N) matrix is given by $F_{kn} = e^{\frac{-2\pi i n k}{N}} \in \mathbb{C}^{N \times N}$

$N=4$: $F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$

$\begin{bmatrix} X_0 \\ \vdots \\ X_{N-1} \end{bmatrix} = F_N \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix}$ Don't actually construct this in practice, or it's already $O(N^2)$

$F^T = F$

$F^{-1} = \frac{1}{N^2} F^3 \Rightarrow$ ifft(x) = fft(fft(fft(x)))/N^2 (slower than MATLAB)

$O(N \log N)$

Fast matrix-vector multiplication using the FFT ($N \times N$)

Idea: Exploit structure of matrix to compute mat-vec in $O(N \log N)$, not $O(N^2)$

Don't construct matrix or you're already doomed to $O(N^2)$.

Circulant:

$$C = \begin{bmatrix} c_0 & c_{N-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{N-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{N-2} & & \ddots & \ddots & c_{N-1} \\ c_{N-1} & c_{N-2} & \cdots & c_1 & c_0 \end{bmatrix}$$

(Only need 1st column to know C)

This is, in fact, diagonalized by F = DFT(N) matrix

$FCF^{-1} = \Lambda$ (diagonal),

$\Rightarrow C = F^{-1} \Lambda F$

What is Λ? $\begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_N \end{pmatrix} = \Lambda \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = FCF^{-1} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = FC \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = Fc$

Cx

① Compute Fx = fft(x)

② Compute ΛFx = fft(c) .* Fx $O(N \log N)$

③ Compute $F^{-1} \Lambda Fx$ = ifft(ΛFx)

Toeplitz:

$$T = \begin{bmatrix} a_0 & a_{-1} & \cdots & a_{-N+1} \\ a_1 & a_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{-1} \\ a_{N-1} & \cdots & a_1 & a_0 \end{bmatrix}$$

(Only need 1st column (c) & row (r) to know T)

Trick: embed T into a circulant matrix of size $2N$.

$$C = \left[\begin{array}{cc|cc} a_0 & a_{-1} \cdots & a_{-N+1} & 0 \; a_{N-1} \cdots a_2 \; a_1 \\ a_1 & a_0 \; \ddots & \vdots & a_{-N+1} \; \ddots \; a_2 \\ \vdots & \ddots \; a_{-1} & & \vdots \; \ddots \; a_{N-1} \\ a_{N-1} \cdots & a_1 \; a_0 & & a_{-2} \; a_{-1} \cdots a_{-N+1} \; 0 \\ \hline 0 \; a_{N-1} \cdots & a_1 & a_0 \; a_{-1} \cdots & a_{-N+1} \\ a_{-N+1} \; \ddots & a_2 & a_1 \; \ddots & \vdots \\ \vdots \; \ddots & a_{N-1} & \vdots \; \ddots & a_{-1} \\ a_{-2} & & a_{N-1} \cdots & a_1 \; a_0 \\ a_{-1} \; a_{-2} \cdots & a_{-N+1} \; 0 & & \end{array}\right]$$

0's don't matter

Compute Tx by $Tx = \underset{n \times 2N}{[I_N \; 0]} \; \underset{2N \times 2N}{C} \begin{bmatrix} x \\ 0 \end{bmatrix}_{2N \times 1}$

Hankel:

$$H = \begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_{N-1} \\ b_1 & b_2 & \iddots & b_{-1} & \\ b_2 & \iddots & \iddots & b_{-2} & \\ \vdots & \iddots & b_{-2} & \iddots & \\ b_{N-1} & b_{-1} & b_{-2} & \cdots & b_{-N+1} \end{bmatrix}$$

(Only need 1st column (c) & last row (r) to know H)

fliplr(H) is Toeplitz

$$Hx = \underbrace{\text{fliplr}(H)}_{\text{Toeplitz}} \text{flipud}(x)$$

Can also do $(T+H)$ fast since $(T+H)x = Tx + Hx$.

Can use this to solve equations too, e.g. $Cx = b$

$$\Rightarrow F^{-1}\Lambda F x = b \quad O(N \log N)$$

Higher dimensional problems lead to block-structured matrices
Can apply same ideas at each level of structure.

Example: Circulant matrix with circulant blocks

$$C = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{13} & C_{11} & C_{12} \\ C_{12} & C_{13} & C_{11} \end{bmatrix}$$

C_{11}, C_{12}, C_{13} are circulant, $m \times m$

$C_{11} = F^{-1}\Lambda_{11}F$

$\Lambda_{11}\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = FC_{11}(:,1)$

Every block is circulant

$$\underbrace{\begin{bmatrix} F & & \\ & F & \\ & & F \end{bmatrix}}_{I \otimes F} C \underbrace{\begin{bmatrix} F^{-1} & & \\ & F^{-1} & \\ & & F^{-1} \end{bmatrix}}_{I \otimes F^{-1}} = \begin{bmatrix} FC_{11}F^{-1} & FC_{12}F^{-1} & FC_{13}F^{-1} \\ FC_{13}F^{-1} & FC_{11}F^{-1} & FC_{12}F^{-1} \\ FC_{12}F^{-1} & FC_{13}F^{-1} & FC_{11}F^{-1} \end{bmatrix} = \underbrace{\begin{bmatrix} \Lambda_{11} & \Lambda_{12} & \Lambda_{13} \\ \Lambda_{13} & \Lambda_{11} & \Lambda_{12} \\ \Lambda_{12} & \Lambda_{13} & \Lambda_{11} \end{bmatrix}}_{\Lambda}$$

Suppose C ($mn \times mn$) was a block circulant ($n \times n$) with circulant ($m \times m$) blocks.

Naive cost of Cx? $O((mn)^2)$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \} m \quad (3m \times 1)$$

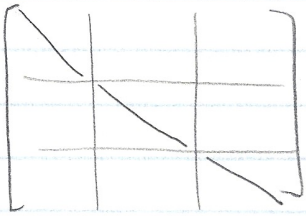
① Compute $y = \begin{bmatrix} Fx_1 \\ Fx_2 \\ Fx_3 \end{bmatrix}$ $O(mn \log n)$

② Compute $\Delta y = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \} m$ $O(n^2 m)$

③ Compute $\begin{bmatrix} F^{-1}z_1 \\ F^{-1}z_2 \\ F^{-1}z_3 \end{bmatrix}$ $O(mn \log n)$

Using only circulant block structure: $O(mn \log n)$.

Now exploit the fact that the blocks are circulant:

$(F \otimes I)\,\Lambda\,(F^{-1} \otimes I) =$  $= \Lambda_{nice}$

① Compute $\underline{y} = \begin{bmatrix} F\underline{x}_1 \\ F\underline{x}_2 \\ F\underline{x}_3 \end{bmatrix}$ $O(mn \log n)$

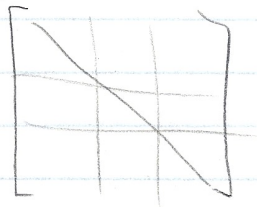
② (i) Compute $\underline{v} = (F \otimes I)\,\underline{y}$ $O(mn \log m)$

(ii) Compute $\underline{w} = \Lambda_{nice}\,\underline{v}$ $O(mn)$

(iii) Compute $(F^{-1} \otimes I)\,\underline{w} = \begin{bmatrix} \underline{z}_1 \\ \underline{z}_2 \\ \underline{z}_3 \end{bmatrix}$ $O(mn \log m)$

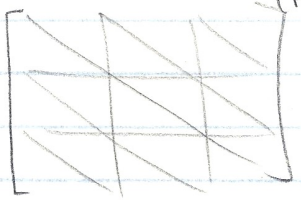
③ Compute $\begin{bmatrix} F^{-1}\underline{z}_1 \\ F^{-1}\underline{z}_2 \\ F^{-1}\underline{z}_3 \end{bmatrix}$ $O(mn \log n)$

Using both levels of structure: $O(mn \log n)$!

$(F \otimes I)(I \otimes F)\,C\,(I \otimes F^{-1})(F^{-1} \otimes I) =$ 

$(F \otimes F)$ (= $(F^{-1} \otimes F^{-1})$)

2D FFT



To do block Toeplitz, do embedding to make it block circulant with circulant blocks!