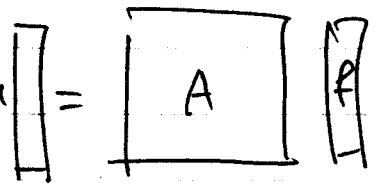
1. Introduction.

$A \in \mathbb{R}^{n\times n}, \mathbb{C}^{n\times n}$ $\quad f$ — n vector

want $u = Af$ $\quad u$ — n vector



(*)

Total # of operations in order to compute (*)

n^2 multiplication + n^2 additions

$= 2n^2$

Notation. Complexity of (*) is of order $O(n^2)$

Questions: How to speed it up ?

Goal: Compute (*) in $O(n)$ or $O(n \log n)$

Idea: Exploit the structure inside A.

Ex1. Discrete Fourier transform.

Given $f_0, f_1 \dots f_{n-1}$

want to compute

$$u_j := \sum_{k=0}^{n-1} e^{-2\pi i \frac{j\cdot k}{n}} \cdot f_k \quad \text{for } j = -\tfrac{n}{2}, \dots \tfrac{n}{2}-1.$$

u_j = freq modes. $\quad f_k$ = time samples.

$$u_{j+n} := \sum_{k=0}^{n-1} e^{-2\pi i \frac{(j+n)\cdot k}{n}} \cdot f_k$$

$$= \sum_{k=0}^{n-1} e^{-2\pi i \frac{j\cdot k}{n}} \cdot \underbrace{e^{-2\pi i j}}_{=1} \cdot f_k = u_j$$

Instead of computing

u_j for $j = -\frac{n}{2} \cdots \frac{n}{2}-1$

we can compute

u_j for $j = 0, 1, 2 \cdots, n-1$.

and wrap the result.

DFT:

$$u_j = \sum_{k=0}^{n-1} e^{-\frac{2\pi i \cdot j \cdot k}{n}} \cdot f_k \quad \text{for } j = 0, 1, 2 \cdots n-1$$

Matrix form

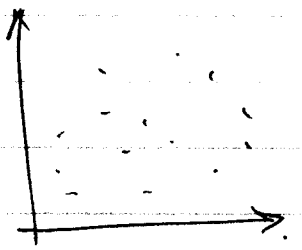
$$u = F_n \cdot f \qquad u = \begin{bmatrix} u_0 \\ \vdots \\ u_{n-1} \end{bmatrix} \qquad f = \begin{bmatrix} f_0 \\ \vdots \\ f_{n-1} \end{bmatrix}$$

$$F_n = \left(e^{-\frac{2\pi i}{n} \cdot j \cdot k} \right)_{0 \le j,k \le n-1}$$

▷ Ex2. Evaluation of Gravitational potential

Given $p_0, p_1 \cdots p_{n-1}$ (n particle) $\subseteq \mathbb{R}^2, \mathbb{R}^3$

$f_0, f_1 \cdots f_{n-1}$ mass at $p_0, p_1 \cdots p_{n-1}$.



Gravitational potential between x, y

$$G(x,y) = \begin{cases} \ln|x-y| & \text{in 2D} \\ \frac{1}{|x-y|} & \text{in 3D} \end{cases}$$

Gravitational potential at P_i

$$u_i = \sum_{\substack{j=0 \\ j \neq i}}^{n-1} G(P_i, P_j) \cdot f_j$$

In the matrix form $u = G \cdot f$

$$u = \begin{bmatrix} u_0 \\ \vdots \\ u_{n-1} \end{bmatrix} \quad f = \begin{bmatrix} f_0 \\ \vdots \\ f_{n-1} \end{bmatrix} \quad G = [G(P_i, P_j)]_{0 \le i,j < n}$$

except at $i=j$

$G_{i,j} = 0$

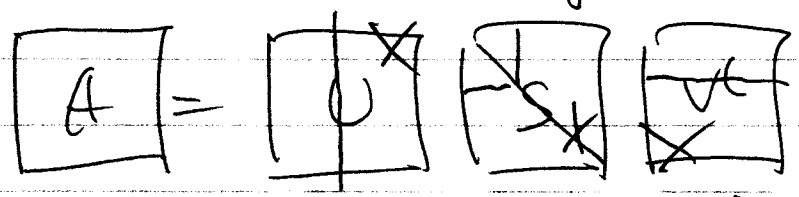
Outline

1. Study two basic tools
   - low rank approximation
   - Fast Fourier transform (Ex 1)
2. Extentions of Fast Fourier transfom
   Cyclic matrices, non-uniform F.T.
3. Fast multipole method (Ex 2).
4. Butterfly algorithm. (Ex 1, Ex 2)

2. Low rank appoximation.

▷ A is approximately low rank if $\exists \tilde{A}$ with $rk(\tilde{A}) = r_\varepsilon = O(1)$ and $\|A - \tilde{A}\| \leq \varepsilon$.

Ex: $A = USV^t$ (singular value decomposition)



U: orthogonal matrix. $U = [u_1, u_2 \cdots u_n]$

V: orthogonal matrix $V = [v_1 v_2 \cdots v_n]$.

S: diagonal matrix with ~~positive~~ non-neg diagonal entries,

$$s_{11} \geq s_{22} \geq \cdots \geq s_{rr} \geq \varepsilon \geq s_{r+1,r+1} \cdots \geq s_{n,n} \geq 0.$$

$$\tilde{A} = \tilde{U} \cdot \tilde{S} \cdot \tilde{V}^t$$

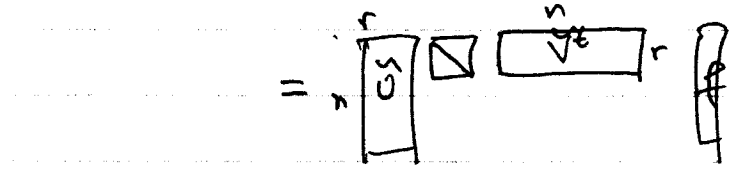
$$\tilde{U} = [u_1 u_2 \cdots u_r] \qquad \tilde{S} = \begin{bmatrix} s_{11} & & \\ & \ddots & \\ & & s_{rr} \end{bmatrix}$$

$$\tilde{V} = [v_1 \cdots v_r]$$

$$\Rightarrow \|A - \tilde{A}\| \leq \varepsilon$$

Therefore $u = Af \approx \tilde{A}f$

$$= \tilde{U} \cdot \tilde{S} \cdot \tilde{V}^t \cdot f$$



Computationally:

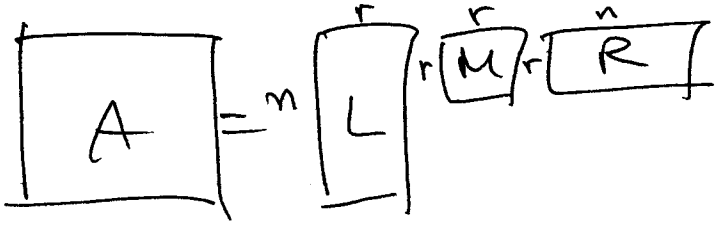
i)	$\hat{V}^t$.	$O(n \cdot r)$
ii)	$\hat{S}$.	$O(r)$
iii)	$\hat{U}$	$O(nr)$
+)		
		$O(nr)$

When r is very small (A is approx. low rank)

$$O(nr) \ll O(n^2)$$

SVD is not the only choice.

$$A = LMR$$



The same process can be repeated $\Rightarrow O(nr)$.

3. Fast Fourier transform.

$$u = F_n \cdot f \quad \text{where} \quad F_n = \left(e^{-\frac{2\pi i \cdot j \cdot k}{n}}\right)_{0 \le j,k \le n-1}$$

$$u = \begin{bmatrix} u_0 \\ \vdots \\ u_{n-1} \end{bmatrix} \qquad f = \begin{bmatrix} f_0 \\ \vdots \\ f_{n-1} \end{bmatrix}.$$

▷ We order $f \Rightarrow \begin{bmatrix} f^e \\ f^o \end{bmatrix}$ ← even, ← odd

$$f^e = \begin{bmatrix} f_0 \\ f_2 \\ \vdots \\ f_{n-2} \end{bmatrix} \qquad f^o = \begin{bmatrix} f_1 \\ f_3 \\ \vdots \end{bmatrix}.$$

Assumption: n is an integer power of 2

$$u = F_n(:, [\text{even}, \text{odd}]) \begin{pmatrix} f^e \\ f^o \end{pmatrix}$$

▷ $$F_n(:, [\text{even}, \text{odd}]) = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

$(j,k) \to$ index the entries in F_n. $(j',k') \to$ index the entries in M_{11}

+ M_{11}

$$(M_{11})_{j',k'} = (F_n)_{jk} \quad \text{with} \quad j = j', \quad k = 2k'$$

$$= e^{-\frac{2\pi i j \cdot k}{n}} = e^{-\frac{2\pi i \cdot j'(2k')}{n}}$$

$$= e^{-2\pi i \frac{j'k'}{(n/2)}} \Rightarrow \quad M_{11} = F_{n/2}$$

M_{21} $\quad (M_{21})_{j',k'} = (F_n)_{jk}$ $\quad j = j' + \frac{n}{2}$, $k = 2k'$

$$= e^{-2\pi i \frac{j \cdot k}{n}}$$

$$= e^{-2\pi i \frac{(j' + \frac{n}{2}) \cdot 2k'}{n}}$$

$$= e^{-2\pi i \frac{j' \cdot 2k'}{n}} \cdot e^{-2\pi i \frac{\frac{n}{2} \cdot 2k'}{n}} \quad (= 1)$$

$$= e^{-2\pi i \frac{j' \cdot k'}{(n/2)}} \Rightarrow F_{n/2}$$

M_{12} $\quad (M_{12})_{j',k'} = (F_n)_{jk}$ $\quad j = j'$, $k = 2k'+1$

$$= e^{-2\pi i \frac{j' \cdot (2k'+1)}{n}}$$

$$= e^{-2\pi i \frac{j' \cdot 2k'}{n}} \cdot e^{-2\pi i \frac{j'}{n}}$$

$$= e^{-2\pi i j'/n} \cdot e^{-2\pi i \frac{j' \cdot 2k'}{n}}$$

$$(M_{12}) = \left(e^{-2\pi i \frac{j'}{n}} \right) F_{n/2}$$

$$\Omega_{n/2}$$

$M_{12} = \Omega_{n/2} \cdot F_{n/2}$

M_{22}.

$$(M_{22})_{j'k'} = (F_n)_{jk} \qquad j = j' + \frac{n}{2}, \quad k = 2k'+1$$

$$= e^{-\frac{2\pi i}{n}(j'+\frac{n}{2})(2k'+1)}$$

$$= -e^{-\frac{2\pi i}{n}\cdot j'} \cdot e^{-2\pi i \frac{j'k'}{n/2}} \quad \curvearrowright \cdots$$

$$M_{22} = -\Omega_{n/2} \cdot F_{n/2}$$

$$F_n(:, [\text{even}, \text{odd}]) = \begin{bmatrix} F_{n/2} & \Omega_{n/2} \cdot F_{n/2} \\ F_{n/2} & -\Omega_{n/2} \cdot F_{n/2} \end{bmatrix} \qquad (*)$$

$$u = F_n \cdot f = F_n(:, [\text{even}, \text{odd}]) \begin{pmatrix} f^e \\ f^o \end{pmatrix}$$

$$= \begin{bmatrix} F_{n/2} & \Omega_{n/2} F_{n/2} \\ F_{n/2} & -\Omega_{n/2} F_{n/2} \end{bmatrix} \begin{pmatrix} f^e \\ f^o \end{pmatrix}$$

$$= \begin{bmatrix} F_{n/2} \cdot f^e + \Omega_{n/2} \cdot F_{n/2} \cdot f^o \\ F_{n/2} \cdot f^e - \Omega_{n/2} \cdot F_{n/2} \cdot f^o \end{bmatrix}$$

Rmk. one size-n DFT

$\Rightarrow$ two size-$n/2$ DFT + ... simple (diagonal mul).

Idea: Recursion, Divide and Conquer.

Algo.

func $u = FFT(f, n)$

If $n = 1$,

$u = f$

else

$f \Rightarrow \begin{bmatrix} f^e \\ f^o \end{bmatrix}$

$x \leftarrow FFT(f^e, \frac{n}{2})$

$y \leftarrow FFT(f^o, \frac{n}{2})$

$u = \begin{bmatrix} x + \Omega_{n/2}\, y \\ x - \Omega_{n/2}\, y \end{bmatrix}$;

end

Total # of operations ?

Let $T(n)$ = the ~~total~~ # of operation used in $FFT(f, n)$.

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \underline{\underline{C \cdot n}} \quad \Big| \quad T\left(\frac{n}{2}\right) = 2 \cdot T\left(\frac{n}{4}\right) + C \cdot \frac{n}{2}$$

$$= 2\left(2 \cdot T\left(\frac{n}{4}\right) + C \cdot \frac{n}{2}\right) + \underline{\underline{Cn}}$$

$$= 4 \cdot T\left(\frac{n}{4}\right) + C \cdot n + Cn \quad \Big|$$

$$= \vdots$$

$$= n \cdot T\left(\frac{n}{n}\right) + \underbrace{Cn + \cdots + Cn}_{\log_2 n}$$

In the matrix form

$$\begin{bmatrix} z_0 \\ \vdots \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 & y_{n-1} & \cdots & y_1 \\ y_1 & \ddots & \ddots & \\ y_2 & & & y_{n-1} \\ y_{n-1} & y_2 & y_1 & y_0 \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ x_{n-1} \end{bmatrix}$$

Naive algo $\Rightarrow$ Complexity $O(n^2)$.

Use FFT to speed it up.

Idea:

$$\hat{z}_\xi := \sum_j e^{-2\pi i \frac{j\cdot\xi}{N}} \cdot z_j$$

$$= \sum_{j=0}^{n-1} e^{-2\pi i \frac{j\cdot\xi}{N}} \cdot \sum_{k=0}^{n-1} x_k \cdot y_{j-k}$$

$$\frac{j\cdot\xi}{N} = \frac{k\cdot\xi}{N} + \frac{(j-k)\cdot\xi}{N}$$

$$= \sum_j \sum_k e^{-2\pi i \frac{k\cdot\xi}{N}} \cdot e^{-2\pi i \frac{(j-k)\cdot\xi}{N}} \cdot x_k \cdot y_{j-k}$$

$$= \sum_k e^{-2\pi i \frac{k\xi}{N}} \cdot x_k \left(\sum_j e^{-2\pi i \frac{(j-k)\xi}{N}} \cdot y_{j-k} \right)$$

$\Downarrow$ Change of var $\ell = j-k$

$$\left(\sum_\ell e^{-2\pi i \frac{\ell\cdot\xi}{N}} \cdot y_\ell \right)$$

$\Downarrow$

$$= \sum_k e^{-2\pi i \frac{k\xi}{N}} \cdot x_k \quad \left(\hat{y}_\xi \right)$$

$$= \hat{x}_\xi \cdot \hat{y}_\xi$$

$$= n \cdot 1 + C \cdot n \log n = O(n \log n)$$

▷ Rmk: Reduction from $O(n^2) \Rightarrow O(n \log n)$ using careful ordering + algebraic structure of the matrix.

▷ $F_n = \left(e^{-\frac{2\pi i}{n} \cdot j \cdot k} \right)_{0 \le j,k < n}$.

F_n^* adjoint $\left(e^{\frac{2\pi i}{n} j \cdot k} \right)_{0 \le j,k < n}$.

F_n^{-1} inverse $\left(\frac{1}{n} e^{\frac{2\pi i}{n} j \cdot k} \right)_{0 \le j,k < n}$.

▷ Application.

Circulant matrix.

Given $x = \begin{bmatrix} x_0 \\ \vdots \\ x_{n-1} \end{bmatrix}$ $y = \begin{bmatrix} y_0 \\ \vdots \\ y_{n-1} \end{bmatrix}$

Want to compute $z = \begin{bmatrix} z_0 \\ \vdots \\ z_{n-1} \end{bmatrix}$ s.t

$$z_j = \sum_{k=0}^{n-1} x_k \cdot y_{j-k}$$

↑ algebra is modulus n.

Idea: discrete Convolution is multiplication in Fourier.

Algo.
1. $\hat{x} = FFT(x)$
2. $\hat{y} = FFT(y)$
3. $\hat{z}_\xi = \hat{x}_\xi \cdot \hat{y}_\xi$
4. $z = IFFT(\hat{z}_\xi)$

---

4. Non-uniform F.T.

▷ Given the freq data α_k with $k = -\frac{N}{2} \cdots \frac{N}{2}-1$, the function f with α_k as its Fourier coefficents is defined by

$$f(x) = \sum_{k=-N/2}^{N/2-1} e^{2\pi i \cdot x \cdot k} \cdot \alpha_k \qquad x \in [0,1]$$

periodic.

Suppose $x_j \in [0,1]$ for $j = 0, 1, \cdots N-1$.

Want to evaluate

$$f_j \equiv f(x_j) = \sum_{k=-N/2}^{N/2-1} e^{2\pi i \, x_j \cdot k} \cdot \alpha_k$$

If x_j were $x_j = \frac{j}{N}$, then

$$f_j = \sum_{-k=N/2}^{N/2-1} e^{2\pi i \cdot \frac{j}{N} \cdot k} \cdot \alpha_k$$

Rmk: $\{x_j\} = \{\frac{j}{N}\} \Rightarrow$ use FFT

Otherwise, FFT cannot be used directly.

$$f = \begin{pmatrix} f_0 \\ \vdots \\ f_{N-1} \end{pmatrix} = \left(e^{2\pi i \frac{x_j \cdot k}{N}}\right)_{j,k} \begin{pmatrix} \alpha_{-N/2} \\ \vdots \\ \alpha_{N/2-1} \end{pmatrix}$$

↑ Fourier coefficients

▷ ~~Algo~~. Idea: { ① Evaluate $f(x)$ on a refined Cartesian grid
② Approx $f_j = f(x_j)$ using Taylor expansion at nearby Cartesian grid pt.

▷ Algo:

① Let $M = q \cdot N$. q - TBD.

Define a grid

$$\{x_j^M = \frac{j}{M}\}_{0 \le j < M}$$ equally spaced

② Define $\alpha_k^M = \begin{cases} \alpha_k & -N/2 \le k \le N/2-1 \\ 0 & -M/2 \le k \le -\frac{N}{2} \text{ and} \\ & N/2 \le k < M/2 \end{cases}$

(3) Compute

$$f(\vec{x}_j) = \sum_{k=-M/2}^{M/2} e^{2\pi i \vec{k}\cdot\vec{x}_j} \alpha_j$$

$$= \sum_{k=-M/2}^{M/2} e^{2\pi i \vec{k}\cdot\vec{x}_j} \cdot \vec{\alpha}_j$$ ← extension by zero-padding.

$$= \sum_{k=-M/2}^{M/2} e^{2\pi i \cdot k \cdot \frac{j}{M}} \cdot \vec{\alpha}_j$$

Fourier transform of size M

$$= O(M \log M)$$

(4) Compute $f^{(1)}(\vec{x}_j) \ldots f^{(p-1)}(\vec{x}_j)$ $j = 0 \ldots M-1$

$$f(x) = \sum_{k=-M/2}^{M/2-1} e^{2\pi i x \cdot k} \alpha_k$$

$$f^{(1)}(x) = \sum e^{2\pi i x k} (2\pi i k) \alpha_k$$

$$f^{(p)}(x) = \sum e^{2\pi i x k} (2\pi i k)^p \cdot \alpha_k$$

$$f^{(p-1)}(\vec{x}_j) = \sum_{k=-M/2}^{M/2-1} e^{2\pi i \vec{x}_j \cdot k} (2\pi i k)^{p-1} \cdot \alpha_k$$

$$= \sum_{k=-M/2}^{M/2-1} e^{2\pi i \cdot \frac{j}{M} \cdot k} (2\pi i k)^{p-1} \cdot \alpha_k$$

$$= \sum_{k=-M/2}^{M/2-1} e^{2\pi i \frac{j \cdot k}{M}} \cdot \vec{(2\pi i k)^{p-1} \cdot \alpha_k}$$

Fourier transform of $(2\pi i k)^{p+1} \alpha_k$

with ~~$k=0,1,\dots,N$~~

$-N/2 \le k < N/2$

$O(N \log N) \times (p-1)$

(5) For each x_j, determine the closest grid pt

Say $\tilde{x}_m$, use Taylor expansion

$$f_j \equiv f(x_j) \approx \left[f(\tilde{x}_m) + f^{(1)}(\tilde{x}_m)\cdot(x_j - \tilde{x}_m) + \dots + \frac{f^{(p)}(\tilde{x}_m)}{(p-1)!}\cdot(x_j - \tilde{x}_m)^{p-1} \right] + \dots$$

Error

$$\left| f_j - [\quad] \right| \le \frac{1}{p!} \cdot \underbrace{f^{(p)}(x^*)}_{(A)} \cdot \underbrace{|x_j - \tilde{x}_m|^p}_{(B)} \qquad (\square)$$

$$(A) = f^{(p)}(x^*) = \sum_{k=-N/2}^{N/2-1} e^{2\pi i \cdot x^* \cdot k} \cdot (2\pi i \cdot k)^p \cdot \alpha_k$$

$$\le (2\pi \cdot \tfrac{N}{2})^p \cdot \sum_k |\alpha_k| \le (\pi N)^p \cdot \sum_k |\alpha_k|$$

$$(B) \quad |x_j - \tilde{x}_m|^p \le \left(\frac{1}{2M}\right)^p = \left(\frac{1}{2Nq}\right)^p$$

$P! \geq \left(\frac{P}{e}\right)^P$

$$\textcircled{\Pi} \leq \frac{1}{P!} (\pi N)^P \sum_k |\alpha_k| \cdot \left(\frac{1}{2Nq}\right)^P$$

$$\leq \left(\frac{e}{P}\right)^P \cdot \pi^P \left(\frac{1}{2q}\right)^P = \left(\frac{e\pi}{2Pq}\right)^P \cdot \sum_k |\alpha_k|.$$

If I want $\textcircled{\Pi} \leq \varepsilon$, all we need to do is to make sure

$$\left(\frac{e\pi}{2Pq}\right)^P \leq \varepsilon$$

E.g. If $P=8$, $q=2$ $\Rightarrow$ $\varepsilon = 10^{-5}$

▷ Complexity

$$\underline{M \log M} + (P-1) \cdot M \log M \qquad (M = qN)$$

$$= P \cdot M \log M = P \cdot q \cdot N \log q \cdot N$$

$$\leq O(P \cdot q\, N \log N)$$