Linear transforms

compute

$$y = Tx$$

where x is the input vector, y the output vector and T the transform matrix.

Example: DFT

1. form (standard in signal processing): given $x_0, \dots, x_{n-1}$

$$\text{compute } y_k = \sum_{\ell=0}^{n-1} e^{-2\pi i k\ell/n} x_\ell, \text{ for } k = 0..n-1 \qquad \boxed{i = \sqrt{-1}}$$

$$= \sum_{\ell=0}^{n-1} \omega_n^{k\ell} x_\ell, \text{ for } k = 0..n-1, \quad \omega_n = e^{-\frac{2\pi i}{n}}$$

primitive nth root of 1

2. form (we will use): $x = (x_0, \dots, x_{n-1})^T$ is given

$$\text{compute } y = DFT_n x, \qquad DFT_n = [\omega_n^{k\ell}]_{0 \le k, \ell < n}$$

$[y = (y_0, \dots, y_{n-1})^T$ is the output$]$

Examples:

$$DFT_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad DFT_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

How many complex ops for DFT_4?

Transform algorithms

an algorithm for $y = Tx$ is given by a factorization

$$T = T_1 T_2 \cdots T_m$$

Namely, instead of $y = Tx$ you can compute

$$\left.\begin{aligned} t_1 &= T_m x \\ t_2 &= T_{m-1} t_1 \\ &\dots \\ y &= T_1 \cdot t_{m-1} \end{aligned}\right\} m \text{ steps}$$

This reduces the op count only if

- the T_i are sparse
- m is not too large

Note: For generic T, $y = Tx$ is $\Theta(n^2)$

Example: Cooley-Tukey FFT for n=4

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & i \end{bmatrix} \begin{bmatrix} 1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1 \\ & & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Compare (complex) cost:

by definition: 12 adds, 4 mults by i

using FFT: 8 adds, 1 mult by i

The sparse matrices are structured:

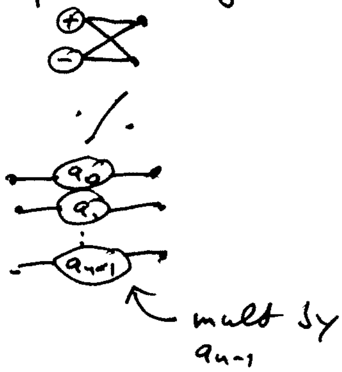
$$DFT_4 = (DFT_2 \otimes I_2)\, \mathrm{diag}(1,1,1,i)\, (I_2 \otimes DFT_2)\, L^4_2$$

(explained next)

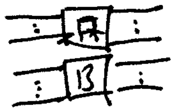
$\rightarrow$ back to slides

Structured matrices

data flow (right to left)



- $DFT_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- $I_n = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}$
- $\mathrm{diag}(a_0, \dots, a_{n-1}) = \begin{pmatrix} a_0 & & \\ & \ddots & \\ & & a_{n-1} \end{pmatrix}$
- $A \oplus B = \begin{pmatrix} A & \\ & B \end{pmatrix}$



- $A \otimes B = [a_{k,l} \cdot B]_{k,l}$ where $A = [a_{k,l}]_{k,l}$

most important:

$$I_n \otimes A = \begin{pmatrix} A & & \\ & \ddots & \\ & & A \end{pmatrix}$$ contains n A's

$$A \otimes I_n = \begin{pmatrix} \ddots & \ddots & \underbrace{\ddots}_{a_{k,l} \cdot I_n} & \cdots \\ \ddots & \ddots & \ddots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

contains n A's
e.g., all bullets together constitute one A

- L^n_k: stride permutation matrix
more later

General radix, recursive Cooley-Tukey FFT

assume $n = km$

$$DFT_{km} = (\underset{\uparrow \text{ radix}}{DFT_k} \otimes I_m)\, \underbrace{T^n_m}_{\text{diagonal matrix}}\, (I_k \otimes DFT_m)\, L^n_k$$

3 key structures: $I_k \otimes A_m$, $A_k \otimes I_m$, L^n_k

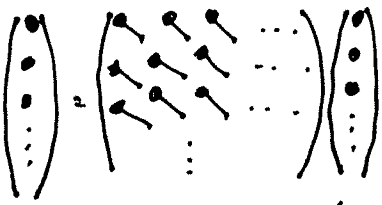
1.) $y = (I_k \otimes A_m)\, x$

$$\begin{pmatrix} | \end{pmatrix} = \begin{pmatrix} A & & & \\ & A & & \\ & & \ddots & \\ & & & A \end{pmatrix} \begin{pmatrix} | \end{pmatrix}$$

k A's at stride 1

for $i = 0 : k-1$
$\quad y[im : 1 : im+m-1]$
$= A \cdot x[\quad " \quad]$

2.) $y = (A_k \otimes I_m)\, x$



m A's at stride m

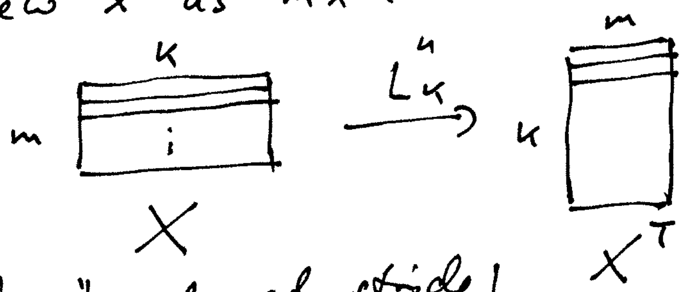
for $i = 0 : m-1$
$\quad y[i : m : i+(k-1)m]$
$= A \cdot x[\quad " \quad]$

Permutes x to obtain y

3.) $y = L^n_k\, x$: different ways of viewing it

```
for i = 0:k-1
  for j = 0:m-1 // m = n/k
    y[i*m+j] = x[i+k*j]
```

a.) view x as $m \times k$ matrix:



transposition!

b.) L^n_k "reads at stride 1 and writes at stride m"

stride 1 -> m is the same as stride k -> 1

c.) L^n_k performs permutation $im + j \to jk + i$ $\quad 0 \le i < k$, $0 \le j < m$

FFT again:

$$DFT_{km} = \underbrace{(DFT_k \otimes I_m)}_{\substack{\text{stride} \\ m \to m}}\, T^n_m\, \underbrace{(I_k \otimes DFT_m)}_{\substack{\text{stride} \\ 1 \to 1}}\, \underbrace{L^n_k}_{\substack{\text{stride} \\ 1 \to m}}$$

stride 1 -> m is the same as stride k -> 1

this is the "decimation-in-time" version

Decimation in frequency: transpose!

Use:
- DFT is symmetric
- $(L^n_k)^T = L^n_m$
- $(A \otimes B)^T = A^T \otimes B^T$

Gives:

$$DFT_{kn} = L^n_m (I_k \otimes DFT_m) T^n_m (DFT_k \otimes I_m)$$

Cost analysis: (was in exam for radix 2), assume $n=2$

Measure: (complex adds, complex mults)

Cost: independent of radix $(n \log_2(n), \frac{1}{2} n \log_2(n))$

complex add = 2 real adds
" mult ≤ 4 real mults
2 real adds

$\Rightarrow$ real cost $\leq 2n \log_2(n) + 3n \log_2(n) = 5n \log_2(n)$

Iterative radix-2 FFT

$$DFT_{2^t} = \underbrace{R_{2^t}}_{\substack{\text{bit-reversal} \\ \text{permutation}}} \prod_{i=1}^{t} \underbrace{(I_{2^{t-i}} \otimes T^{2^i}_{2^{i-1}})}_{\substack{\text{diagonal} \\ \text{matrix}}} \underbrace{(I_{2^{t-i}} \otimes DFT_2 \otimes I_{2^{i-1}})}_{\substack{2^{t-1}\ DFT_2\text{'s} \\ \text{at varying strides}}}$$

Most people consider this "the FFT"