

Lecture 17: FFT and structured matrices



School of Mathematical Sciences, Xiamen University

1. Discrete Fourier transform and its inverse

Definition 1

The discrete Fourier transform (DFT) is a mapping on \mathbb{C}^n given by

$$[\mathcal{F}_n\{\mathbf{f}\}]_i = \sum_{j=0}^{n-1} f_j \omega_n^{ij}, \quad i = 0, 1, \dots, n-1,$$

where $\omega_n = e^{-i2\pi/n}$ and $i = \sqrt{-1}$. The inverse DFT is given by

$$[\mathcal{F}_n^{-1}\{\mathbf{g}\}]_i = \frac{1}{n} \sum_{j=0}^{n-1} g_j \omega_n^{-ij}, \quad i = 0, 1, \dots, n-1.$$

- DFT and inverse DFT as matrix-vector products:

$$\mathcal{F}_n\{\mathbf{f}\} = \mathbf{F}_n \mathbf{f}, \quad \mathcal{F}_n^{-1}\{\mathbf{g}\} = \frac{1}{n} \mathbf{F}_n^* \mathbf{g} = \frac{1}{n} \overline{\mathbf{F}_n^T} \mathbf{g}, \quad \mathbf{F}_n = [\omega_n^{ij}]_{i,j=0}^{n-1}.$$

- Discrete sine/cosine transform: DST, DCT, ...

2. The FFT algorithm

- For simplicity, we assume that $n = 2^k$ and set $m = n/2$. Obviously,

$$\omega_m = \omega_n^2 = e^{-i2\pi/m}, \quad \omega_m^m = 1, \quad \omega_n^m = -1.$$

- Given any $\mathbf{f} = [f_0 \ f_1 \ \cdots \ f_{n-1}]^\top \in \mathbb{C}^n$, for $i = 0, 1, \dots, m-1$,

$$\begin{aligned} [\mathcal{F}_n\{\mathbf{f}\}]_i &= \sum_{l=0}^{m-1} \omega_n^{i2l} f_{2l} + \sum_{l=0}^{m-1} \omega_n^{i(2l+1)} f_{2l+1} \\ &= \sum_{l=0}^{m-1} \omega_m^{il} f_{2l} + \omega_n^i \sum_{l=0}^{m-1} \omega_m^{il} f_{2l+1} \\ &= [\mathcal{F}_m\{\mathbf{f}_e\}]_i + \omega_n^i [\mathcal{F}_m\{\mathbf{f}_o\}]_i, \end{aligned}$$

where

$$\mathbf{f}_e = [f_0 \ f_2 \ \cdots \ f_{n-2}]^\top, \quad \mathbf{f}_o = [f_1 \ f_3 \ \cdots \ f_{n-1}]^\top.$$

- For $i = 0, 1, \dots, m-1$, we also have

$$\begin{aligned}
 [\mathcal{F}_n\{\mathbf{f}\}]_{m+i} &= \sum_{l=0}^{m-1} \omega_n^{(m+i)2l} f_{2l} + \sum_{l=0}^{m-1} \omega_n^{(m+i)(2l+1)} f_{2l+1} \\
 &= \sum_{l=0}^{m-1} \omega_m^{il} f_{2l} - \omega_n^i \sum_{l=0}^{m-1} \omega_m^{il} f_{2l+1} \\
 &= [\mathcal{F}_m\{\mathbf{f}_e\}]_i - \omega_n^i [\mathcal{F}_m\{\mathbf{f}_o\}]_i.
 \end{aligned}$$

- Let $\text{FFT}(n)$ denote the number of flops required to evaluate $\mathbf{F}_n \mathbf{f}$ by a recursive algorithm. Given the vectors $\mathbf{F}_m \mathbf{f}_e$ and $\mathbf{F}_m \mathbf{f}_o$, only m multiplications, m additions and m subtractions are needed to evaluate $\mathcal{F}_n\{\mathbf{f}\}$. Hence,

$$\text{FFT}(n) = 3m + 2\text{FFT}(m) = 3n/2 + 2\text{FFT}(n/2).$$

Since $\text{FFT}(1) = 0$, then

$$\text{FFT}(n) = 3n/2 \times k = \frac{3}{2}n \log n.$$

3. Flop counts for frequently used algorithms

Method	Matrix ($m \geq n$)	Operation or Factorization	Flops
MV product	$\mathbf{A} \in \mathbb{C}^{n \times n}$	$\mathbf{b} = \mathbf{A}\mathbf{x}$	$2n^2$
FFT MV product	$\mathbf{F} \in \mathbb{C}^{n \times n}$	$\mathbf{b} = \mathbf{F}\mathbf{x}$	$3n \log n / 2$
MM product	$\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n \times n}$	$\mathbf{C} = \mathbf{A}\mathbf{B}$	$2n^3$
Inverse	$\mathbf{A} \in \mathbb{C}^{n \times n}$	\mathbf{A}^{-1}	$2n^3$
LU factorization	$\mathbf{A} \in \mathbb{C}^{n \times n}$	$\mathbf{P}\mathbf{A} = \mathbf{L}\mathbf{U}$	$2n^3 / 3$
Hessenberg LU	$\mathbf{H} \in \mathbb{C}^{n \times n}$	$\mathbf{H} = \mathbf{L}\mathbf{U}$	$2n^2$
Tridiagonal LU	$\mathbf{T} \in \mathbb{C}^{n \times n}$	$\mathbf{T} = \mathbf{L}\mathbf{U}$	$3n$
Cholesky	$\mathbf{A} \in \mathbb{C}^{n \times n}$	$\mathbf{A} = \mathbf{R}^* \mathbf{R}$	$n^3 / 3$
Triangular solve	$\mathbf{L} \in \mathbb{C}^{n \times n}$	$\mathbf{L}\mathbf{x} = \mathbf{b}$	n^2
Triangular inverse	$\mathbf{L} \in \mathbb{C}^{n \times n}$	\mathbf{L}^{-1}	$2n^3 / 3$
Normal equations	$\mathbf{A} \in \mathbb{C}^{m \times n}$	$\mathbf{A}^* \mathbf{A} = \mathbf{R}^* \mathbf{R}$	$mn^2 + n^3 / 3$
Householder QR	$\mathbf{A} \in \mathbb{C}^{m \times n}$	$\mathbf{Q}^* \mathbf{A} = \mathbf{R}$	$2(mn^2 - n^3 / 3)$
MGS QR	$\mathbf{A} \in \mathbb{C}^{m \times n}$	$\mathbf{A} = \mathbf{Q}_n \mathbf{R}_n$	$2mn^2$
Bidiagonalization	$\mathbf{A} \in \mathbb{C}^{m \times n}$	$\mathbf{B} = \mathbf{U}^* \mathbf{A}\mathbf{V}$	$4(mn^2 - n^3 / 3)$
Hessenberg reduction	$\mathbf{A} \in \mathbb{C}^{n \times n}$	$\mathbf{H} = \mathbf{Q}^* \mathbf{A}\mathbf{Q}$	$10n^3 / 3$
Tridiagonal reduction	$\mathbf{A} \in \mathbb{C}^{n \times n}$	$\mathbf{T} = \mathbf{Q}^* \mathbf{A}\mathbf{Q}$	$4n^3 / 3$
SVD	$\mathbf{A} \in \mathbb{C}^{m \times n}$	$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$	$2mn^2 + 11n^3$

Remark 2

On modern computer architectures the communication costs in moving data between different levels of memory or between processors in a network can exceed the arithmetic costs by orders of magnitude.

4. Circulant matrix

Definition 3

An $n \times n$ matrix \mathbf{C} is called circulant if it has the form

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

We indicate this situation by $\mathbf{C} = \mathbf{circ}(\mathbf{c})$, where

$$\mathbf{c} = [c_0 \quad c_1 \quad \cdots \quad c_{n-1}]^\top \in \mathbb{C}^n$$

- **Discussion:** How can you generate a circulant matrix in Matlab?

Definition 4

The $n \times n$ circulant right shift matrix is given by

$$\mathbf{R} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} = \mathbf{circ} \left([0 \ 1 \ 0 \ \cdots \ 0]^\top \right).$$

- Obviously, if $\mathbf{C} = \mathbf{circ}(\mathbf{c})$, then $\mathbf{C} = \sum_{j=0}^{n-1} c_j \mathbf{R}^j$.

Lemma 5

Let $\omega_n = e^{-i2\pi/n}$. Then

$$\mathbf{R} = \frac{1}{n} \mathbf{F}_n^* \text{diag}\{1, \omega_n, \omega_n^2, \cdots, \omega_n^{n-1}\} \mathbf{F}_n.$$

Theorem 6

If $\mathbf{C} = \mathbf{circ}(\mathbf{c})$, then

$$\mathbf{C} = \mathbf{F}_n^{-1} \text{diag}\{\hat{\mathbf{c}}\} \mathbf{F}_n = \frac{1}{n} \mathbf{F}_n^* \text{diag}\{\hat{\mathbf{c}}\} \mathbf{F}_n$$

where

$$\hat{\mathbf{c}} = \mathbf{F}_n \mathbf{c}.$$

Fast algorithm 1: Circulant matrix-vector product $\mathbf{v} = \mathbf{C}\mathbf{u}$

Step 1: Compute $\hat{\mathbf{c}} = \mathbf{F}_n \mathbf{c}$ and $\hat{\mathbf{u}} = \mathbf{F}_n \mathbf{u}$ by FFTs

Step 2: Compute the component-wise vector product $\hat{\mathbf{v}} = \hat{\mathbf{c}} \cdot \hat{\mathbf{u}}$

Step 3: Compute $\mathbf{v} = \frac{1}{n} \mathbf{F}_n^* \hat{\mathbf{v}}$ by iFFT

5. Toeplitz matrix

Definition 7

A matrix is called Toeplitz if it is constant along diagonals. An $n \times n$ Toeplitz matrix \mathbf{T} has the form

$$\mathbf{T} = \begin{bmatrix} t_0 & t_{-1} & \cdots & t_{2-n} & t_{1-n} \\ t_1 & t_0 & t_{-1} & \ddots & t_{2-n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ t_{n-2} & \ddots & t_1 & t_0 & t_{-1} \\ t_{n-1} & t_{n-2} & \cdots & t_1 & t_0 \end{bmatrix}.$$

We indicate this situation by $\mathbf{T} = \mathbf{toep}(\mathbf{t})$, where

$$\mathbf{t} = [t_{1-n} \quad \cdots \quad t_{-1} \quad t_0 \quad t_1 \quad \cdots \quad t_{n-1}]^\top \in \mathbb{C}^{2n-1}.$$

- Explore `toeplitz(c,r)` in Matlab.

- Define $\mathbf{S} = \mathbf{toep}(\mathbf{s})$, where

$$\mathbf{s} = [t_1 \quad t_2 \quad \cdots \quad t_{n-1} \quad 0 \quad t_{1-n} \quad \cdots \quad t_{-2} \quad t_{-1}]^\top.$$

Then we have

$$\mathbf{T}^{\text{ce}} := \begin{bmatrix} \mathbf{T} & \mathbf{S} \\ \mathbf{S} & \mathbf{T} \end{bmatrix} = \mathbf{circ}(\mathbf{t}^{\text{ce}}),$$

where

$$\mathbf{t}^{\text{ce}} = [t_0 \quad t_1 \quad \cdots \quad t_{n-1} \quad 0 \quad t_{1-n} \quad \cdots \quad t_{-2} \quad t_{-1}]^\top \in \mathbb{C}^{2n}.$$

Note that

$$\begin{bmatrix} \mathbf{T} & \mathbf{S} \\ \mathbf{S} & \mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{T}\mathbf{u} \\ \mathbf{S}\mathbf{u} \end{bmatrix}.$$

Using the fast algorithm for a circulant matrix-vector product, we obtain the following fast algorithm for a Toeplitz matrix-vector product $\mathbf{v} = \mathbf{T}\mathbf{u}$.

Fast algorithm 2: Toeplitz matrix-vector product $\mathbf{v} = \mathbf{T}\mathbf{u}$

Step 1: Compute $\widehat{\mathbf{t}}^{\text{ce}} = \mathbf{F}_{2n}\mathbf{t}^{\text{ce}}$ and $\widehat{\mathbf{u}}^{\text{ze}} = \mathbf{F}_{2n}[\mathbf{u}^\top \mathbf{0}]^\top$ by FFTs

Step 2: Compute the component-wise vector product $\widehat{\mathbf{w}} = \widehat{\mathbf{t}}^{\text{ce}} \cdot \widehat{\mathbf{u}}^{\text{ze}}$

Step 3: Compute $\mathbf{w} = \frac{1}{2n}\mathbf{F}_{2n}^*\widehat{\mathbf{w}}$ by iFFT

Step 4: Extract the first n components of \mathbf{w} to obtain \mathbf{v} ,
i.e., $\mathbf{v} = \mathbf{w}(1:n)$

6. Hankel matrix

- A *Hankel* matrix $\mathbf{H} = [h_{ij}]$ has identical elements along all its anti-diagonals, meaning that

$$h_{ij} = h_{i+l, j-l}$$

for all relevant integers i, j , and l .

- Explore `hankel(c,r)` in Matlab.

- A Hankel matrix is symmetric by definition.
- The relation to a Toeplitz matrix: the matrix

$$\mathbf{T} = \mathbf{J}\mathbf{H}, \quad \mathbf{J} = \begin{bmatrix} & & & & 1 \\ & & & 1 & \\ & & \ddots & & \\ & 1 & & & \\ 1 & & & & \end{bmatrix}$$

is a Toeplitz matrix, where \mathbf{J} is a permutation matrix obtained by reversing the columns (or rows) of the identity.

- Fast algorithm for a Hankel matrix-vector product can be obtained easily from that of a Toeplitz matrix-vector product.
- Other issue: Discrete cosine transform: dct
symmetric Toeplitz-plus-Hankel (STH) matrix ...

7. Kronecker product and $\text{vec}(\cdot)$ operator

Definition 8

Let $\mathbf{A} \in \mathbb{C}^{m \times n}$ and $\mathbf{B} \in \mathbb{C}^{p \times q}$. Then $\mathbf{A} \otimes \mathbf{B}$, the Kronecker product of \mathbf{A} and \mathbf{B} , is the $mp \times nq$ matrix

$$\mathbf{A} \otimes \mathbf{B} := \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}.$$

Definition 9

Let $\mathbf{A} \in \mathbb{C}^{m \times n}$. Then $\text{vec}(\mathbf{A})$ is defined to be a column vector of size mn made of the columns of \mathbf{A} stacked atop one another from left to right.

- If $\mathbf{A} = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_n]$, then

$$\text{vec}(\mathbf{A}) = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_n \end{bmatrix}.$$

- Let $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{m \times n}$. Then $\text{tr}(\mathbf{A}^* \mathbf{B}) = \text{vec}(\mathbf{A})^* \text{vec}(\mathbf{B})$.

Theorem 10

Let $\mathbf{A} \in \mathbb{C}^{p \times m}$, $\mathbf{X} \in \mathbb{C}^{m \times n}$ and $\mathbf{B} \in \mathbb{C}^{n \times q}$. Then the following properties hold:

$$\text{vec}(\mathbf{A}\mathbf{X}) = (\mathbf{I}_n \otimes \mathbf{A})\text{vec}(\mathbf{X}),$$

$$\text{vec}(\mathbf{X}\mathbf{B}) = (\mathbf{B}^\top \otimes \mathbf{I}_m)\text{vec}(\mathbf{X}),$$

$$\text{vec}(\mathbf{A}\mathbf{X}\mathbf{B}) = (\mathbf{B}^\top \otimes \mathbf{A})\text{vec}(\mathbf{X}).$$

Theorem 11

The following facts about Kronecker products hold:

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}),$$

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1},$$

$$(\mathbf{A} \otimes \mathbf{B})^{\dagger} = \mathbf{A}^{\dagger} \otimes \mathbf{B}^{\dagger},$$

$$(\mathbf{A} \otimes \mathbf{B})^{\top} = \mathbf{A}^{\top} \otimes \mathbf{B}^{\top}.$$

- Exercise: For $\mathbf{A} \in \mathbb{C}^{m \times n}$, $\mathbf{B} \in \mathbb{C}^{p \times q}$ and $\mathbf{C} \in \mathbb{C}^{m \times q}$, solve

$$\min_{\mathbf{X} \in \mathbb{C}^{n \times p}} \|\mathbf{AXB} - \mathbf{C}\|_{\text{F}} = ?$$

- Exercise: Let \mathcal{T} denote the triangular truncation operator, which is a linear operator that maps a given matrix to its strictly lower triangular part. Write down the matrix form of this operator.

- Exercise: Let $\mathbf{A} \in \mathbb{C}^{m \times m}$ and $\mathbf{B} \in \mathbb{C}^{n \times n}$. What are eigenvalues of

$$\mathbf{I} \otimes \mathbf{A} + \mathbf{B} \otimes \mathbf{I}$$

and

$$\mathbf{A} \otimes \mathbf{B}?$$

8. Reference books for Toeplitz solver and FFT

- Chan, Raymond Hon-Fu and Jin, Xiao-Qing
An introduction to iterative Toeplitz solvers, SIAM, 2007
- Van Loan, Charles
Computational frameworks for the fast Fourier transform, SIAM, 1992