Computational Symmetry: Accelerating Physical Systems through Fast Fourier Transform

Leveraging Fast Fourier Transform (FFT) sparse matrix factorization to achieve exponential speeds up in quantum phase estimation (QPE) and phonon dispersion analysis.

Representation of Polynomial Multiplication

1) Coefficient Representation

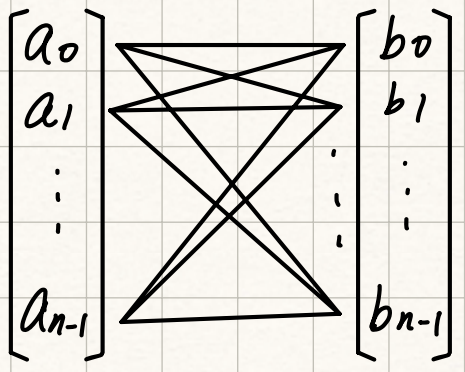
for multiplying two polynomials of degree n,

$$A(x)=\sum_{j=0}^{n-1} a_j x^j \quad , \quad B(x)=\sum_{j=0}^{n-1} b_j x^j$$

$$\Rightarrow a=(a_0, a_1, \cdots, a_{n-1}) \quad , \quad b=(b_0, b_1, \cdots, b_{n-1})$$

$$\text{and } A(x)\times B(x)=\sum_{j=0}^{2n-2} c_j x^j \quad , \quad c=(c_0, c_1, \cdots, c_{2n-2})$$

$$\Rightarrow c = a \otimes b$$



Time Complexity

$\Rightarrow O(n^2)$

↳ Big O notation

⇒ Qualitative description of the trend of algorithm execution time as the amount of input data (n) increases

for adding two polynomials ⇒ Time Complexity $O(n)$

2) Point Value Representation

We choose n three different, $x_0, x_1, \ldots, x_{n-1}$, and calculate the values of these n points

e.g. $A(x)=\{(x_0,y_0),(x_1,y_1),\cdots,(x_{n-1},y_{n-1})\}$

$B(x)=\{(x_0,y_0'),(x_1,y_1'),\cdots,(x_{n-1},y_{n-1}')\}$

$\Rightarrow C(x) = A(x) \times B(x) = \{(x_0, y_0 y_0'), (x_1, y_1, y_1'), \cdots, (x_{n-1}, y_{n-1} y_{n-1}')\}$

$\Rightarrow$ Time Complexity $O(n)$

Fast Fourier Transform (FFT)

- Converting two polynomials from "Coefficient Representation" to "Point Value Representation".
  Time Complexity $O(n^2) \longrightarrow O(n \log n)$
- Doing polynomial multiplication by using "Point Value Representation".
  Time Complexity $O(n)$
- The result is converted back to "Coefficient Representation" using inverse fast fourier transform
  Time Complexity $O(n \log n)$

1) Discrete Fourier Transform (DFT)

For a discrete signal / signal $x[n]$ with period N

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} \underbrace{x[n]}_{a_n} \underbrace{e^{-iwkn}}_{W_N^k = e^{iwk}}, \quad k \in [0, N-1], \quad W \equiv \frac{2\pi}{N}$$

2) Fast Fourier Transform (DFT)

set $X_k = W_N^k = e^{i\frac{2\pi k}{N}}$, for "Coefficient Representation"

$$\Rightarrow y_k = A(X_k) = A(W_N^k) = \sum_{j=0}^{N-1} a_j (W_N^k)^j$$

$$\Rightarrow A(W_N^k) = A^{[0]}\left((W_N^k)^2\right) + W_N^k A^{[1]}\left((W_N^k)^2\right)$$

where

$$\begin{cases} A^{[0]}\left((W_N^k)^2\right) = a_0 + a_2(W_N^k)^2 + a_4(W_N^k)^4 + \cdots + a_{N-2}(W_N^k)^{N-2} \\ A^{[1]}\left((W_N^k)^2\right) = a_1 + a_3(W_N^k)^2 + a_5(W_N^k)^4 + \cdots + a_{N-1}(W_N^k)^{N-2} \end{cases}$$

by lemma 1. $(W_N^k)^2 = W_{\frac{N}{2}}^k$ (The proof is in the appendix)

$$\Rightarrow A(W_N^k) = A^{[0]}\left(W_{\frac{N}{2}}^k\right) + W_N^k A^{[1]}\left(W_{\frac{N}{2}}^k\right)$$

3) Butterfly Operation

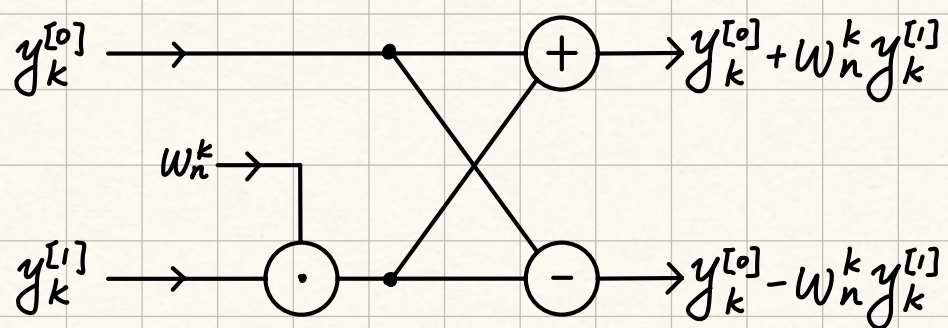
By symmetry, we don't to calculate $k \in [0, N-1]$, we only need to calculate the first half $k \in [0, \frac{N}{2}-1]$, and the result of the second half can be derived from the first half.

for the second half, $k+\frac{N}{2}$

$$W_N^{k+\frac{N}{2}} = e^{i\frac{2\pi(k+\frac{N}{2})}{N}} = e^{i\frac{2\pi k}{N}} \cdot e^{i\pi} = -W_N^k$$

$$W_{\frac{N}{2}}^{k+\frac{N}{2}} = e^{i\frac{2\pi(k+\frac{N}{2})}{N/2}} = e^{i\frac{4\pi(k+\frac{N}{2})}{N}} = e^{i\frac{2\pi k}{N/2}} \cdot e^{i2\pi} = W_{\frac{N}{2}}^k$$

$$\Rightarrow \begin{cases} y_k = y_k^{[0]} + W_N^k y_k^{[1]} \\ y_{k+\frac{N}{2}} = y_k^{[0]} - W_N^k y_k^{[1]} \end{cases}, \text{ where } \begin{cases} y_k^{[0]} = A^{[0]}\left(W_{\frac{N}{2}}^k\right) \\ y_k^{[1]} = A^{[1]}\left(W_{\frac{N}{2}}^k\right) \end{cases}$$

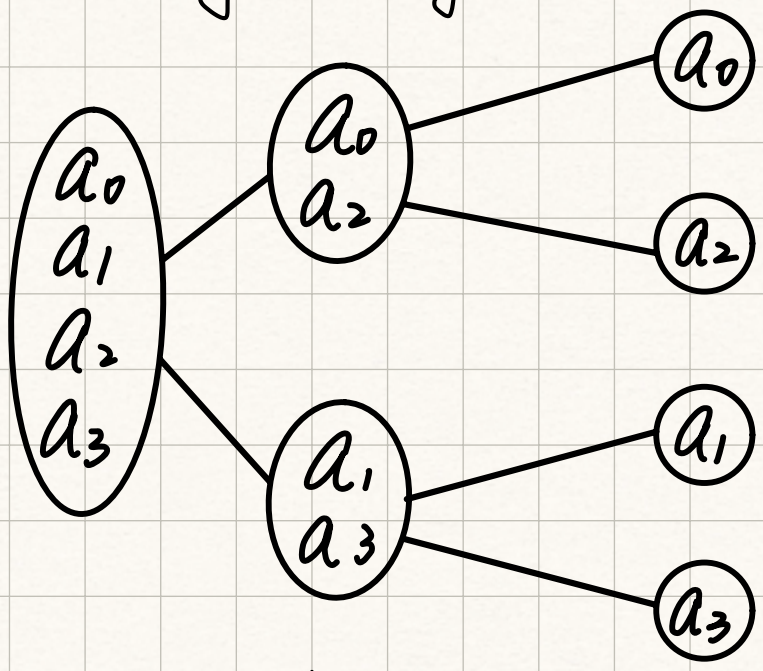


4) Bit-Reversal Permutation (位元反轉排列)

Bit-Reversal Permutation adresses the issues of space (空間) and access (存取).

During recursive calls, in order to protect each layer of data, the computer will constantly open up new space in the memory stack to store the coefficients of sub-problems. This will lead to additional space, overhead, and frequent memory configuration will cause delays.

take $N=4$ as example, we first observe the call structure (呼叫結構) of the original recursive form, we can draw the following tree diagram:



original index (i)	0	1	2	3
binary (3-bit)	000	001	010	011
⇒ reversed binary	000	010	001	011
target index (j)	0	2	1	3

if we reorder the input to the order after bit reversion ($\{a_0, a_1, a_2 a_3\} \longrightarrow \{a_0, a_2, a_1, a_3\}$), we can perform "in-place" operations. The result of butterfly operations in each stage can be directly covered back to the original position of the same array.

* "in-place" operation: With bit-reversal permutation, the data will be reshuffled into a "perfect" operation sequence". In the butterfly operation of each layer, you take out the data of two positions for addition, substraction and multiplication, and the new results calculated directly cover these two positions.
  ⇒ The old data disappears, and the new data is replenished.

Time Complexity : $O(N \log N)$

Space Complexity: $O(N \log N) \longrightarrow O(1)$

Sparse Matrix Factorization Method of FFT
快速傅立葉轉換的稀疏矩陣分解方法

"Sparse" means that most elements in a matrix are 0, only a few positions having values.

$$X_k = \sum_{n=0}^{N-1} x[n] e^{-iwkn} = \sum_{n=0}^{N-1} x[n] (W_N^k)^n, \quad W_N^k \equiv e^{-iwk}, \quad w \equiv \frac{2\pi}{N}$$

$$\Rightarrow X = F_N x$$

$$\begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^2 \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix}}_{F_N} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

* Time Complexity
$O(N^2)$

"Cooley Tukey FFT"

Seperating odd-order terms and even order terms by FFT.

take $N = 2^m$ as example,

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \left(W_N^k\right)^{2n} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \left(W_N^k\right)^{2n+1}$$

by lemma 1. $(W_N^k)^2 = W_{\frac{N}{2}}^k$

$$\Rightarrow X_k = E_k + W_N^k O_k \quad , E_k = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \left(W_N^k\right)^{2n}$$

$$X_{k+\frac{N}{2}} = E_k - W_N^k O_k \quad O_k = \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \left(W_N^k\right)^{2n}$$

by some mathematical calculation

$$\Rightarrow F_N = \begin{bmatrix} I & D_N \\ -I & D_N \end{bmatrix} \begin{bmatrix} F_{\frac{N}{2}} & 0 \\ 0 & F_{\frac{N}{2}} \end{bmatrix} P_N \quad , \text{where } D_N = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & W_N & 0 & \cdots & 0 \\ 0 & 0 & W_N^2 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & & W_N^{\frac{N}{2}-1} \end{bmatrix}$$

e.g. $N = 2^3 = 8$, $X = F_8 x$

$$\begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^2 \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^7 & W_N^{2\times 7} & \cdots & W_N^{7\times 7} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_7 \end{bmatrix}$$

$$\Rightarrow F_8 = \begin{bmatrix} I & D_8 \\ -I & D_8 \end{bmatrix} \begin{bmatrix} F_4 & 0 \\ 0 & F_4 \end{bmatrix} P_8$$

$$= \begin{bmatrix} \begin{pmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1 \end{pmatrix} \begin{pmatrix} 1&0&0&0\\0&w_8&0&0\\0&0&w_8^2&0\\0&0&0&w_8^3 \end{pmatrix} \\ \begin{pmatrix} -1&0&0&0\\0&-1&0&0\\0&0&-1&0\\0&0&0&-1 \end{pmatrix} \begin{pmatrix} 1&0&0&0\\0&w_8&0&0\\0&0&w_8^2&0\\0&0&0&w_8^3 \end{pmatrix} \end{bmatrix} \begin{bmatrix} \begin{pmatrix} 1&1&1&1\\1&w_8&w_8^2&w_8^3\\1&w_8^2&w_8^4&w_8^6\\1&w_8^3&w_8^6&w_8^9 \end{pmatrix} \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix} \\ \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix} \begin{pmatrix} 1&1&1&1\\1&w_8&w_8^2&w_8^3\\1&w_8^2&w_8^4&w_8^6\\1&w_8^3&w_8^6&w_8^9 \end{pmatrix} \end{bmatrix} P_8$$

where $P_8 \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \\ x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix}$

$$\Rightarrow X = F_8 x$$

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} = \underbrace{\begin{bmatrix} \begin{pmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1 \end{pmatrix} \begin{pmatrix} 1&0&0&0\\0&w_8&0&0\\0&0&w_8^2&0\\0&0&0&w_8^3 \end{pmatrix} \\ \begin{pmatrix} -1&0&0&0\\0&-1&0&0\\0&0&-1&0\\0&0&0&-1 \end{pmatrix} \begin{pmatrix} 1&0&0&0\\0&w_8&0&0\\0&0&w_8^2&0\\0&0&0&w_8^3 \end{pmatrix} \end{bmatrix}}_{\begin{bmatrix} I & D_8 \\ -I & D_8 \end{bmatrix}} \underbrace{\begin{bmatrix} \begin{pmatrix} 1&1&1&1\\1&w_8&w_8^2&w_8^3\\1&w_8^2&w_8^4&w_8^6\\1&w_8^3&w_8^6&w_8^9 \end{pmatrix} \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix} \\ \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix} \begin{pmatrix} 1&1&1&1\\1&w_8&w_8^2&w_8^3\\1&w_8^2&w_8^4&w_8^6\\1&w_8^3&w_8^6&w_8^9 \end{pmatrix} \end{bmatrix}}_{\begin{bmatrix} F_{\frac{N}{2}} & 0 \\ 0 & F_{\frac{N}{2}} \end{bmatrix}} P_8 \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

⇒ Combined into a complete DFT

⇒ Perform DFT on the smaller matrices respectively

$$= \begin{bmatrix} E_0 + w_8^0 O_0 \\ E_1 + w_8^1 O_1 \\ E_2 + w_8^2 O_2 \\ E_3 + w_8^3 O_3 \\ E_0 - w_8^0 O_0 \\ E_1 - w_8^1 O_1 \\ E_2 - w_8^2 O_2 \\ E_3 - w_8^3 O_3 \end{bmatrix}$$

Corresponding to

$$X_k = E_k + W_N^k O_k$$

$$X_{k+\frac{N}{2}} = E_k - W_N^k O_k$$

Time complexity $O(N^2) \to O(N \log N)$

$\Rightarrow F(N) = F(2^m) = 2F(2^{m-1}) + \frac{N}{2}$

Number of operations

$= 2\left[2F(2^{m-2}) + \frac{N}{4}\right] + \frac{N}{2}$

$= 4F(2^{m-2}) + \frac{N}{2} + \frac{N}{2}$

$\vdots$

$= \underbrace{2^m F(2^0)}_{= N \times 1 = N} + \overbrace{\underbrace{\frac{N}{2} + \frac{N}{2} + \cdots + \frac{N}{2}}_{\frac{N}{2} \times m = \frac{N}{2}\log N}}^{\text{※} = m}$

$\Rightarrow O\left(N + \frac{N}{2}\log N\right) = O(N \log N)$

Application 1 : Quantum Phase Estimation (QPE)

Now we have unitary operator U and its eigenstate $|u\rangle$

$\Rightarrow U|u\rangle = e^{i2\pi\phi}|u\rangle$, ϕ (unknown)

Using 3 qubits as phase registers $|000\rangle$

* phase register (相位暫存器) : a set of qubits specifically designed to "store" and "display" the phase value we want to measure.

$\Rightarrow$ initial state $|000\rangle|u\rangle$

Doing Hadamard tranform : Convert the computational basis into an "equal-weighted superposition basis"

$$H^{\otimes 3}|000\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)$$

$$= \frac{1}{\sqrt{8}}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)$$

$$\Rightarrow H^{\otimes 3}|000\rangle = \frac{1}{\sqrt{8}}\sum_{y=0}^{7}|y\rangle = \frac{1}{\sqrt{8}}(|0\rangle+|1\rangle+|2\rangle+|3\rangle+|4\rangle+|5\rangle+|6\rangle+|7\rangle)$$

$$= \frac{1}{\sqrt{8}}(|000\rangle+|001\rangle+|010\rangle+|011\rangle+|100\rangle+|101\rangle+|110\rangle+|111\rangle)$$

$$\Rightarrow |000\rangle|u\rangle \longrightarrow (H^{\otimes 3}|000\rangle)|u\rangle$$

Controlled-$U^y \Rightarrow C(U): |y\rangle|u\rangle \mapsto |y\rangle U^y|u\rangle$

$$U^y\left(\frac{1}{\sqrt{8}}\sum_{y=0}^{7}|y\rangle\right)|u\rangle = \left(\frac{1}{\sqrt{8}}\sum_{y=0}^{7}|y\rangle\right)U^y|u\rangle = \frac{1}{\sqrt{8}}\sum_{y=0}^{7}e^{i2\pi\phi y}|y\rangle|u\rangle$$

(phase pattern)

Inverse QFT: Decode the phase pattern into a bit string
(相位模式) (位元串)

$$*QFT_N|k\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{n-1}e^{i2\pi ky/N}|y\rangle$$

$$QFT_N^{-1}|y\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1}e^{-i2\pi ky/N}|k\rangle$$

$$\Rightarrow QFT^{-1}\left(\frac{1}{\sqrt{8}}\sum_{y=0}^{7}e^{i2\pi\phi y}|y\rangle\right)|u\rangle = \frac{1}{8}\sum_{k=0}^{7}\sum_{y=0}^{7}e^{i2\pi\phi y}e^{-i2\pi ky/8}|k\rangle|u\rangle$$

$$= \frac{1}{8}\sum_{k=0}^{7}\left(\sum_{y=0}^{7}e^{i2\pi(\phi-\frac{k}{8})y}\right)|k\rangle|u\rangle$$

$\Rightarrow$ It only exhibits "constructive interference" when $\phi \approx \frac{k}{8}$. Otherwise, the contributions cancel out through destructive interference.

Set $\theta_y = 2\pi(\phi-\frac{k}{8})y$, $\Delta\theta = 2\pi(\phi-\frac{k}{8})$

We can see if $\phi = \frac{3}{8}$

k	$\phi-\frac{k}{8}$	$\theta_y = 2\pi(\phi-\frac{k}{8})y \ (y=0,\cdots,7)$	total
0	$\frac{3}{8}$	→ ↖ ↓ ↗ ← ↘ ↑ ↙	0
1	$\frac{2}{8}$	→ ↑ ← ↓ → ↑ ← ↓	0
2	$\frac{1}{8}$	→ ↗ ↑ ↖ ← ↙ ↓ ↘	0
3	0	→ → → → → → → →	⟶
4	$\frac{-1}{8}$	→ ↘ ↓ ↙ ← ↖ ↑ ↗	0
5	$\frac{-2}{8}$	→ ↓ ← ↑ → ↓ ← ↑	0
6	$\frac{-3}{8}$	→ ↙ ↑ ↘ ← ↗ ↓ ↖	0
7	$\frac{-4}{8}$	→ ← → ← → ← → ←	0

$$\Rightarrow QFT_8|3\rangle = \frac{1}{\sqrt{8}}\sum_{y=0}^{7} e^{i2\pi\frac{3y}{8}}|y\rangle$$

$$QFT_8^{-1}\left(\frac{1}{\sqrt{8}}\sum_{y=0}^{7} e^{i2\pi\frac{3y}{8}}|y\rangle\right) = |3\rangle = |011\rangle$$

Since

$$QFT_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 & w^4 & w^5 & w^6 & w^7 \\ 1 & w^2 & w^4 & w^6 & w^8 & w^{10} & w^{12} & w^{14} \\ \vdots & \vdots & & & & & & \vdots \\ 1 & w^7 & & & & & & w^{49} \end{bmatrix}, \quad w = e^{i\frac{2\pi}{8}}$$

QFT can decomposed into many sparse small matrices

$$QFT = SWAPs \cdot \text{controlled phases} \cdot Hadamards$$

$$\Rightarrow QFT_8 = SWAP_{1,3}[H_3 \cdot CR_2 \cdot CR_3 \cdot H_2 \cdot CR_2 \cdot H_1]$$

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad CR_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}, \quad SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Classical FFT	Quantum QFT
permutation matrix	SWAP gates
twiddle factor W_N^k	controlled phase R_k
butterfly matrix	Hadamard gate
sparse matrix product	sparse quantum gate product

Application 2: Dispersion Analysis of Phonons

for a n-point crystal lattice $u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$

, $u_n(t)$: the displacement of the n^{th} atom.

$$m\ddot{u}_n = K(u_{n+1} - u_n) - K(u_n - u_{n-1})$$

$$\Rightarrow m\ddot{u}_n = K(u_{n+1} + u_{n-1} - 2u_n)$$

$$\Rightarrow m\begin{pmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \vdots \\ \ddot{u}_n \end{pmatrix} = -k\begin{pmatrix} 2 & -1 & 0 & \cdots & -1 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & \cdots & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$$

Due to lattice translation symmetry, we guess

$u_n(t) = Ae^{i(q_k n - \omega t)} \Rightarrow u_{n\pm1} = u_n e^{\pm i q_k}$, $q_k = \frac{2\pi k}{N}$

$$\Rightarrow m(-\omega^2)u_n = K(u_{n+1} + u_{n-1} - 2u_n)$$

$$\Rightarrow -m\omega^2 u_n = K(e^{iq_k} + e^{-iq_k} - 2)u_n$$

$$\Rightarrow m\omega^2 = K(2 - 2\cos q_k) = K4\sin^2\left(\frac{q_k}{2}\right)$$

$$\Rightarrow \omega(q_k) = 2\sqrt{\frac{K}{m}}\left|\sin\frac{q_k}{2}\right|$$

Define the discrete Fourier transform

$$\tilde{u}_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} u_n e^{-i\theta_k n}$$

and inverse transform

$$u_n = \frac{1}{\sqrt{N}} \sum_k \tilde{u}_k e^{i\theta_k n}$$

Substitute into the equation of motion:

$$m\ddot{u}_n = K(u_{n+1} + u_{n-1} - 2u_n)$$

Since $u_{n\pm1} = u_n e^{\pm i\theta_k}$, by IDFT

$$\Rightarrow u_{n\pm1} = \frac{1}{\sqrt{N}} \sum_k \tilde{u}_k e^{i\theta_k n} e^{\pm i\theta_k}$$

$$\Rightarrow m\ddot{u}_n = \frac{K}{\sqrt{N}} \sum_k (\tilde{u}_k e^{i\theta_k(n+1)} + \tilde{u}_k e^{i\theta_k(n-1)} - 2\tilde{u}_k e^{i\theta_k n})$$

$$= \frac{K}{\sqrt{N}} \sum_k \tilde{u}_k e^{i\theta_k n}(e^{i\theta_k} + e^{-i\theta_k} - 2) = \frac{K}{\sqrt{N}} \sum_k \tilde{u}_k e^{i\theta_k n}(2\cos\theta_k - 2)$$

$$\Rightarrow \frac{m}{\sqrt{N}} \sum_k \ddot{\tilde{u}}_k e^{i\theta_k n} = \frac{K}{\sqrt{N}} \sum_k \tilde{u}_k e^{i\theta_k n}(2\cos\theta_k - 2)$$

$$\Rightarrow m\ddot{\tilde{u}}_k = -K\tilde{u}_k(2 - 2\cos\theta_k)$$

The DFT can be written as $\tilde{u}_k = F_N u_n$, where F_N is the DFT matrix.

for example, take $N = 4$

$$F_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} = B_2 B_1 P \quad \text{(sparse butterfly factorization)}$$

where $P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, $Pu_n = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_3 \\ u \\ u_4 \end{pmatrix}$

and $B_1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}$, $B_2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -i \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & i \end{pmatrix}$

$$\Rightarrow F_4 U_n = B_2 B_1 P U_n = \begin{pmatrix} (x_0+x_2) + (x_1+x_3) \\ (x_0-x_2) - i(x_1-x_3) \\ (x_0+x_2) - (x_1+x_3) \\ (x_0-x_2) + i(x_1-x_3) \end{pmatrix}$$

Time complexity $O(N^2) \to O(N \log N)$

Appendix

- Proposition 1: n-th roots of Unity
  For any $k \in \mathbb{Z}_n$, we have $W_n^{nk} = 1$, where $W_n = e^{i\frac{2\pi}{n}}$
  pf. by $e^{i\theta} = \cos\theta + i\sin\theta$, now $\theta = \frac{2\pi}{n} \times n k = 2\pi k$
  $\Rightarrow e^{i 2\pi k} = 1$

- Lemma 1: Cancellation Lemma
  For any intergers $n, k, d > 0$, we have $W_{dn}^{dk} = W_n^k$
  pf. $W_{dn}^{dk} = \left(e^{\frac{2\pi i}{dn}}\right)^{dk} = e^{i\frac{2\pi k}{n}} = W_n^k$

- Lemma 2: Symmetry Lemma
  If n is even (where $n = 2k$), then $W_n^{\frac{n}{2}} = W_2 = -1$
  pf. $W_n^{\frac{n}{2}} = \left(e^{i\frac{2\pi}{n}}\right)^{\frac{n}{2}} = e^{i\pi} = W_2 = -1$

- Lemma 3: Halving Lemma
  If $n > 0$ is even, then the set of squares of the n complex

n-th roots of unit is identical to the set of the $\frac{n}{2}$ complex $(\frac{n}{2})$-th roots of unity.

pf. by lemma 1, we have $(W_n^k)^2 = W_{\frac{n}{2}}^k$ for $k \in \mathbb{Z}$

$\Rightarrow (W_n^{k+\frac{n}{2}})^2 = (W_n^k)^2$

- Lemma 4 : Summation Lemma

  $\forall n \geq 1$ and nonzero integer k not divisible by n: $\sum_{j=0}^{n-1} (W_n^k)^j = 0$

  pf. $\sum_{j=0}^{n-1} (W_n^k)^j = \frac{W_n^{kn} - 1}{W_n^k - 1} = \frac{1 - 1}{W_n^k - 1} = 0$

Reference

Huang, B. (Ed.). (2026, April 19). 快速傅立葉轉換（*FFT*). hackmd. https://hackmd.io/@8dSak6oVTweMeAe9fXWCPA/H1y3L57Yd