Introduction to CUDA Parallel Programming
CUDA平行計算導論

https://ceiba.ntu.edu.tw/1092Phys8061_CUDA

Professor Ting-Wai Chiu (趙挺偉)
Email: twchiu@phys.ntu.edu.tw
Physics Department
National Taiwan University

This lecture will cover:

- Fourier Transform
- Discrete Fourier Transform (DFT)
- Fast Fourier Transform (FFT)
- CUDA library -- cuFFT
- Physical Applications

Fourier Transform

$$k-\text{space} \qquad \tilde{\psi}(k) = \int_{-\infty}^{+\infty} dx \exp(-ikx)\, \psi(x), \qquad x-\text{space}$$

$$\psi(x) = \int_{-\infty}^{+\infty} dk \exp(+ikx)\ \tilde{\psi}(k),$$

Fourier Transform

$$k-\text{space} \qquad \tilde{\psi}(k) = \int_{-\infty}^{+\infty} dx \exp(-ikx)\, \psi(x), \qquad x-\text{space}$$

$$\psi(x) = \int_{-\infty}^{+\infty} dk \exp(+ikx)\ \tilde{\psi}(k),$$

$$\psi(x) = \int_{-\infty}^{+\infty} dk \exp(+ikx)\tilde{\psi}(k) = \int_{-\infty}^{+\infty} dk \exp(+ikx)\int_{-\infty}^{+\infty} dy \exp(-iky)\psi(y)$$

$$= \int_{-\infty}^{+\infty} dy \int_{-\infty}^{+\infty} dk \exp[+ik(x-y)]\, \psi(y) = 2\pi \int_{-\infty}^{+\infty} dy\ \delta(x-y)\psi(y)$$

$$= 2\pi\psi(x)$$

Fourier Transform

$$k-\text{space} \qquad \tilde{\psi}(k) = \int_{-\infty}^{+\infty} dx \exp(-ikx)\, \psi(x), \qquad x-\text{space}$$

$$\psi(x) = \int_{-\infty}^{+\infty} dk \exp(+ikx)\ \tilde{\psi}(k),$$

$$\psi(x) = \int_{-\infty}^{+\infty} dk \exp(+ikx)\tilde{\psi}(k) = \int_{-\infty}^{+\infty} dk \exp(+ikx)\int_{-\infty}^{+\infty} dy \exp(-iky)\psi(y)$$

$$= \int_{-\infty}^{+\infty} dy \int_{-\infty}^{+\infty} dk \exp[+ik(x-y)]\, \psi(y) = 2\pi \int_{-\infty}^{+\infty} dy\ \delta(x-y)\psi(y)$$

$$= 2\pi\psi(x)$$

Normalization $$\tilde{\psi}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} dx \exp(-ikx)\psi(x),$$

$$\psi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} dk \exp(+ikx)\tilde{\psi}(k),$$

Fourier Transform

$$k-\text{space} \qquad \tilde{\psi}(k) = \int_{-\infty}^{+\infty} dx \exp(-ikx)\, \psi(x), \qquad x-\text{space}$$

$$\psi(x) = \int_{-\infty}^{+\infty} dk \exp(+ikx)\ \tilde{\psi}(k),$$

$$\psi(x) = \int_{-\infty}^{+\infty} dk \exp(+ikx)\tilde{\psi}(k) = \int_{-\infty}^{+\infty} dk \exp(+ikx)\int_{-\infty}^{+\infty} dy \exp(-iky)\psi(y)$$

$$= \int_{-\infty}^{+\infty} dy \int_{-\infty}^{+\infty} dk \exp[+ik(x-y)]\, \psi(y) = 2\pi \int_{-\infty}^{+\infty} dy\ \delta(x-y)\psi(y)$$

$$= 2\pi\psi(x)$$

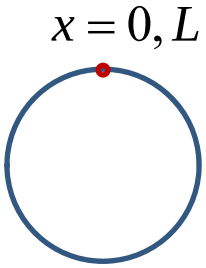
Normalization $\qquad \tilde{\psi}(k) = \int_{-\infty}^{+\infty} dx \exp(-ikx)\psi(x),$

$$\psi(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} dk \exp(+ikx)\tilde{\psi}(k),$$

Fourier Transform on a Circle

$$\tilde{\psi}(k) = \int_{-\infty}^{+\infty} dx \exp(-ikx)\psi(x),$$

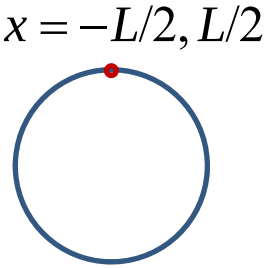
$$\psi(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} dk \exp(+ikx)\tilde{\psi}(k),$$



$$\psi(x+L) = \psi(x) \;\Rightarrow\; \exp(ikL) = 1 \;\Rightarrow\; k = \frac{2n\pi}{L},\; n = \text{integer}$$

$$\tilde{\psi}(k) = \int_{-L/2}^{+L/2} dx \exp\left(-\frac{i2\pi nx}{L}\right)\psi(x),$$

$$\psi(x) = \frac{1}{L}\sum_{n} \exp\left(\frac{i2\pi nx}{L}\right)\tilde{\psi}(k)$$



Discrete Fourier Transform (DFT)

Consider $L = Na, \quad x = -\left[\frac{N}{2} - 1\right]a, \cdots, \left[\frac{N}{2}\right]a$

For example, $\left\{N = 4, \ \frac{x}{a} = -1, 0, 1, 2\right\}; \ \left\{N = 5, \ \frac{x}{a} = -2, -1, 0, 1, 2\right\}$

$$\tilde{\psi}(k) = \int_{-L/2}^{+L/2} dx \exp\left(-\frac{i2\pi nx}{L}\right)\psi(x) \ \Rightarrow \ \tilde{\psi}(k) = \sum_x \exp\left(-\frac{i2\pi nx}{L}\right)\psi(x)$$

$$\psi(x) = \frac{1}{L}\sum_n \exp\left(\frac{i2\pi nx}{L}\right)\tilde{\psi}(k)$$

The finite number of values of x implies that the number of values of k is also finite, i.e., $k = \frac{2n\pi}{L}, \ n = -\left[\frac{N}{2} - 1\right], \cdots, \left[\frac{N}{2}\right]$

Discrete Fourier Transform (DFT)

$$\tilde{\psi}(k) = \sum_x \exp\left(-\frac{i2\pi nx}{L}\right)\psi(x), \qquad x = -\left[\frac{N}{2}-1\right]a, \cdots, \left[\frac{N}{2}\right]a$$

$$\psi(x) = \frac{1}{L}\sum_n \exp\left(\frac{i2\pi nx}{L}\right)\tilde{\psi}(k), \qquad k = \frac{2n\pi}{L},\ n = -\left[\frac{N}{2}-1\right], \cdots, \left[\frac{N}{2}\right]$$

If we choose the coordinate system such that the values of x are $\{0,\ a,\ 2a,\ \cdots, (N-1)a\}$, correspondingly, the values of k are

$$k = \frac{2n\pi}{L},\ n = 0, \cdots, N-1.$$

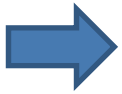
$$\tilde{\psi}(k) = \sum_{x/a=0}^{N-1} \exp\left(-\frac{i2\pi nx}{L}\right)\psi(x),$$

$$\psi(x) = \frac{1}{L}\sum_{n=0}^{N-1} \exp\left(\frac{i2\pi nx}{L}\right)\tilde{\psi}(k),$$

Discrete Fourier Transform (DFT)

$$\tilde{\psi}(k) = \sum_{x/a=0}^{N-1} \exp\left(-\frac{i2\pi nx}{L}\right)\psi(x),$$

$$\psi(x) = \frac{1}{L}\sum_{n=0}^{N-1} \exp\left(\frac{i2\pi nx}{L}\right)\tilde{\psi}(k),$$



$$\tilde{\psi}(n) = \sum_{j=0}^{N-1} \exp\left(-\frac{i2\pi nj}{N}\right)\psi(j),$$

$$\psi(j) = \frac{1}{N}\sum_{n=0}^{N-1} \exp\left(\frac{i2\pi nj}{N}\right)\tilde{\psi}(n),$$

Discrete Fourier Transform (DFT)

$$\tilde{\psi}(k) = \sum_{x/a=0}^{N-1} \exp\left(-\frac{i2\pi nx}{L}\right)\psi(x),$$

$$\psi(x) = \frac{1}{L}\sum_{n=0}^{N-1} \exp\left(\frac{i2\pi nx}{L}\right)\tilde{\psi}(k),$$

$$\tilde{\psi}(n) = \sum_{j=0}^{N-1} \exp\left(-\frac{i2\pi nj}{N}\right)\psi(j),$$

$$\psi(j) = \frac{1}{N}\sum_{n=0}^{N-1} \exp\left(\frac{i2\pi nj}{N}\right)\tilde{\psi}(n),$$

$$\begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \tilde{\psi}_2 \\ \vdots \\ \tilde{\psi}_{N-2} \\ \tilde{\psi}_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & w & w^2 & \cdots & w^{N-2} & w^{N-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(N-2)} & w^{2(N-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 1 & w^{N-2} & w^{(N-2)2} & \cdots & w^{(N-2)(N-2)} & w^{(N-2)(N-1)} \\ 1 & w^{N-1} & w^{(N-1)2} & \cdots & w^{(N-1)(N-2)} & w^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{N-2} \\ \psi_{N-1} \end{pmatrix} \equiv F \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{N-2} \\ \psi_{N-1} \end{pmatrix}$$

$w = e^{\frac{-2\pi i}{N}}$, one of the N complex roots of unity, $z = \left\{1, w, w^2, \cdots, w^{N-2}, w^{N-1}\right\}$.

Discrete Fourier Transform (DFT)

$$w = \exp\left(\frac{-2\pi i}{N}\right), \text{ one of the } N \text{ complex roots of unity, } z^N = 1,$$

$$z = \{1, w, w^2, \cdots, w^{N-2}, w^{N-1}\}.$$

$$1 + w + w^2 + \cdots + w^{N-2} + w^{N-1} = \frac{1 - w^N}{1 - w} = 0,$$

$$1 + w^m + w^{2m} + \cdots + w^{m(N-2)} + w^{m(N-1)} = \frac{1 - w^{Nm}}{1 - w^m} = 0, \quad m = 1, \cdots, (N-1).$$

Discrete Fourier Transform (DFT)

$$\begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \tilde{\psi}_2 \\ \vdots \\ \tilde{\psi}_{N-2} \\ \tilde{\psi}_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & w & w^2 & \cdots & w^{N-2} & w^{N-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(N-2)} & w^{2(N-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 1 & w^{N-2} & w^{(N-2)2} & \cdots & w^{(N-2)(N-2)} & w^{(N-2)(N-1)} \\ 1 & w^{N-1} & w^{(N-1)2} & \cdots & w^{(N-1)(N-2)} & w^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{N-2} \\ \psi_{N-1} \end{pmatrix} \equiv F \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{N-2} \\ \psi_{N-1} \end{pmatrix}$$

$$\begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{N-2} \\ \psi_{N-1} \end{pmatrix} = \frac{1}{N} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & w^{-1} & w^{-2} & \cdots & w^{-(N-2)} & w^{-(N-1)} \\ 1 & w^{-2} & w^{-4} & \cdots & w^{-2(N-2)} & w^{-2(N-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 1 & w^{-(N-2)} & w^{-(N-2)2} & \cdots & w^{-(N-2)(N-2)} & w^{-(N-2)(N-1)} \\ 1 & w^{-(N-1)} & w^{-(N-1)2} & \cdots & w^{-(N-1)(N-2)} & w^{-(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \tilde{\psi}_2 \\ \vdots \\ \tilde{\psi}_{N-2} \\ \tilde{\psi}_{N-1} \end{pmatrix} \equiv F^{-1} \begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \tilde{\psi}_2 \\ \vdots \\ \tilde{\psi}_{N-2} \\ \tilde{\psi}_{N-1} \end{pmatrix}$$

Discrete Fourier Transform (DFT)

$$\tilde{\Psi} = F\Psi, \qquad FF^* = F^*F = N1_{N\times N},$$

$$\Psi = F^{-1}\tilde{\Psi} = \frac{1}{N}F^*\tilde{\Psi}, \qquad UU^\dagger = U^\dagger U = 1_{N\times N},\ U = \frac{1}{\sqrt{N}}F,\ U^\dagger = \frac{1}{\sqrt{N}}F^*$$

Discrete Fourier Transform (DFT)

$$\tilde{\Psi} = F\Psi, \qquad FF^* = F^*F = N1_{N\times N},$$

$$\Psi = F^{-1}\tilde{\Psi} = \frac{1}{N}F^*\tilde{\Psi}, \qquad UU^\dagger = U^\dagger U = 1_{N\times N},\ U = \frac{1}{\sqrt{N}}F,\ U^\dagger = \frac{1}{\sqrt{N}}F^*$$

For $N = 2^m$,

$$\tilde{\psi}_n = \sum_{j=0}^{N-1} \exp\left(-\frac{i2\pi nj}{N}\right)\psi_j$$

$$= \sum_{j=0}^{N/2-1} \exp\left(-\frac{i2\pi n}{N}2j\right)\psi_{2j} + \sum_{j=0}^{N/2-1} \exp\left(-\frac{i2\pi n}{N}(2j+1)\right)\psi_{2j+1}$$

$$= \sum_{j=0}^{N/2-1} \exp\left(-\frac{i2\pi n}{N/2}j\right)\psi_{2j} + \exp\left(-\frac{i2\pi n}{N}\right)\sum_{j=0}^{N/2-1} \exp\left(-\frac{i2\pi n}{N/2}j\right)\psi_{2j+1}$$

$$= \sum_{j=0}^{N/2-1} (w_{N/2})^{nj}\psi_{2j} + (w_N)^n \sum_{j=0}^{N/2-1} (w_{N/2})^{nj}\psi_{2j+1} \equiv E_n + (w_N)^n O_n$$

Fast Fourier Transform (FFT)

For $N = 2^m$,

$$\tilde{\psi}_n = \sum_{j=0}^{N/2-1} (w_{N/2})^{nj} \psi_{2j} + (w_N)^n \sum_{j=0}^{N/2-1} (w_{N/2})^{nj} \psi_{2j+1} \equiv E_n + (w_N)^n O_n$$

Due to the periodicity of w_N, $\tilde{\psi}_{n+N/2}$ can be obtained from E_n and O_n

$$\tilde{\psi}_{n+N/2} = \sum_{j=0}^{N/2-1} (w_{N/2})^{(n+N/2)j} \psi_{2j} + (w_N)^{n+N/2} \sum_{j=0}^{N/2-1} (w_{N/2})^{(n+N/2)j} \psi_{2j+1}$$

$$= E_n - (w_N)^n O_n, \qquad (w_{N/2})^{(N/2)j} = e^{\frac{-2i\pi}{N/2}(N/2)j} = e^{-2i\pi j} = 1,$$

$$(w_N)^{N/2} = e^{-i\pi} = -1$$

$$\boxed{\begin{aligned} \tilde{\psi}_n &= E_n + (w_N)^n O_n, \\ \tilde{\psi}_{n+N/2} &= E_n - (w_N)^n O_n, \end{aligned}} \qquad n = 0, \cdots, N/2 - 1$$

This result, expressing the DFT of length N recursively in terms of two DFTs of size $N/2$, is the core of Fast Fourier Transform (FFT).

Fast Fourier Transform (FFT)

$$\begin{aligned}\tilde{\psi}_n &= E_n + (w_N)^n O_n, \\ \tilde{\psi}_{n+N/2} &= E_n - (w_N)^n O_n.\end{aligned} \qquad n = 0, \cdots, N/2 - 1$$

FFT can be represented as

$$\tilde{\Psi}_N = F_N \Psi_N \Rightarrow F_N = \begin{pmatrix} I_{N/2} & D_{N/2} \\ I_{N/2} & -D_{N/2} \end{pmatrix} \begin{pmatrix} F_{N/2} & 0 \\ 0 & F_{N/2} \end{pmatrix} \begin{pmatrix} \text{even-odd} \\ \text{permutations} \end{pmatrix},$$

$I_{N/2}$ is the unit matrix of size $\frac{N}{2} \times \frac{N}{2}$,

$$w_N = e^{\frac{-i2\pi}{N}}$$

$D_{N/2}$ is the diagonal matrix with diagonal elements $\left\{1, w_N, w_N^2, \cdots, w_N^{N/2-1}\right\}$.

The recursion in FFT amounts to write F_N in terms of $F_{N/2}$, then $F_{N/2}$ in terms of $F_{N/4}, \cdots,$ until F_2.

Fast Fourier Transform (FFT)

Excercise: To verify that FFT gives the correct results for $N = 4$ and $N = 8$.

$N = 4$

$$\begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \tilde{\psi}_2 \\ \tilde{\psi}_3 \end{pmatrix} = F_4 \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \end{pmatrix} = \begin{pmatrix} I_2 & D_2 \\ I_2 & -D_2 \end{pmatrix} \begin{pmatrix} F_2 & 0 \\ 0 & F_2 \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_2 \\ \psi_1 \\ \psi_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & w \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -w \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_2 \\ \psi_1 \\ \psi_3 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -i \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & i \end{pmatrix} \begin{pmatrix} \psi_0 + \psi_2 \\ \psi_0 - \psi_2 \\ \psi_1 + \psi_3 \\ \psi_1 - \psi_3 \end{pmatrix} = \begin{pmatrix} \psi_0 + \psi_2 + \psi_1 + \psi_3 \\ \psi_0 - \psi_2 - i(\psi_1 - \psi_3) \\ \psi_0 + \psi_2 - (\psi_1 + \psi_3) \\ \psi_0 - \psi_2 + i(\psi_1 - \psi_3) \end{pmatrix}$$

$$w = \exp(-i2\pi/4) = -i$$

$$w^2 = -1$$

Note that $F_2 = \begin{pmatrix} I_1 & D_1 \\ I_1 & -D_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

Fast Fourier Transform (FFT)

Excercise: To verify that FFT gives the correct results for $N = 4$ and $N = 8$.

$N = 4$

$$\begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \tilde{\psi}_2 \\ \tilde{\psi}_3 \end{pmatrix} = F_4 \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \end{pmatrix} = \begin{pmatrix} I_2 & D_2 \\ I_2 & -D_2 \end{pmatrix} \begin{pmatrix} F_2 & 0 \\ 0 & F_2 \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_2 \\ \psi_1 \\ \psi_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & w \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -w \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_2 \\ \psi_1 \\ \psi_3 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -i \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & i \end{pmatrix} \begin{pmatrix} \psi_0 + \psi_2 \\ \psi_0 - \psi_2 \\ \psi_1 + \psi_3 \\ \psi_1 - \psi_3 \end{pmatrix} = \begin{pmatrix} \psi_0 + \psi_2 + \psi_1 + \psi_3 \\ \psi_0 - \psi_2 - i(\psi_1 - \psi_3) \\ \psi_0 + \psi_2 - (\psi_1 + \psi_3) \\ \psi_0 - \psi_2 + i(\psi_1 - \psi_3) \end{pmatrix}$$

$$w = \exp(-i2\pi/4) = -i$$

$$w^2 = -1$$

$$\begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \tilde{\psi}_2 \\ \tilde{\psi}_3 \end{pmatrix} = F_4 \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 \\ 1 & w^2 & w^4 & w^6 \\ 1 & w^3 & w^6 & w^9 \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \end{pmatrix}$$

Fast Fourier Transform (FFT)

$N = 8$

$$\begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \tilde{\psi}_2 \\ \tilde{\psi}_3 \\ \tilde{\psi}_4 \\ \tilde{\psi}_5 \\ \tilde{\psi}_6 \\ \tilde{\psi}_7 \end{pmatrix} = \begin{pmatrix} I_4 & D_4 \\ I_4 & -D_4 \end{pmatrix} \begin{pmatrix} F_4 & 0 \\ 0 & F_4 \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_2 \\ \psi_4 \\ \psi_6 \\ \psi_1 \\ \psi_3 \\ \psi_5 \\ \psi_7 \end{pmatrix}$$

$$D_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & w & 0 & 0 \\ 0 & 0 & w^2 & 0 \\ 0 & 0 & 0 & w^3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & w & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & 0 & 0 & -iw \end{pmatrix}$$

$$w = \exp(-i2\pi/8) = \exp(-i\pi/4)$$

$$w^2 = -i, \quad w^3 = -iw$$

$$= \left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & +w & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -i & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -iw \\ \hline 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -w & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & +i & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & +iw \end{array}\right) \left(\begin{array}{c} \begin{pmatrix} \psi_0 + \psi_4 + \psi_2 + \psi_6 \\ \psi_0 - \psi_4 - i(\psi_2 - \psi_6) \\ \psi_0 + \psi_4 - (\psi_2 + \psi_6) \\ \psi_0 - \psi_4 + i(\psi_2 - \psi_6) \end{pmatrix} \\ \begin{pmatrix} \psi_1 + \psi_5 + \psi_3 + \psi_7 \\ \psi_1 - \psi_5 - i(\psi_3 - \psi_7) \\ \psi_1 + \psi_5 - (\psi_3 + \psi_7) \\ \psi_1 - \psi_5 + i(\psi_3 - \psi_7) \end{pmatrix} \end{array}\right)$$

Discrete Fourier Transform (DFT)

$N = 8$

$$\begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \tilde{\psi}_2 \\ \tilde{\psi}_3 \\ \tilde{\psi}_4 \\ \tilde{\psi}_5 \\ \tilde{\psi}_6 \\ \tilde{\psi}_7 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 & w^4 & w^5 & w^6 & w^7 \\ 1 & w^2 & w^4 & w^6 & w^8 & w^{10} & w^{12} & w^{14} \\ 1 & w^3 & w^6 & w^9 & w^{12} & w^{15} & w^{18} & w^{21} \\ 1 & w^4 & w^8 & w^{12} & w^{16} & w^{20} & w^{24} & w^{28} \\ 1 & w^5 & w^{10} & w^{15} & w^{20} & w^{25} & w^{30} & w^{35} \\ 1 & w^6 & w^{12} & w^{18} & w^{24} & w^{30} & w^{36} & w^{42} \\ 1 & w^7 & w^{14} & w^{21} & w^{28} & w^{35} & w^{42} & w^{49} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \\ \psi_6 \\ \psi_7 \end{pmatrix}$$

$$w = \exp(-i2\pi/8) = \exp(-i\pi/4)$$

$$w^2 = -i, \quad w^3 = -iw$$

$$w^4 = -1, \quad w^5 = -w$$

$$w^6 = i, \quad w^7 = iw$$

$$= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w & -i & -iw & -1 & -w & i & iw \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & -iw & i & w & -1 & iw & -i & -w \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -w & -i & iw & -1 & w & i & -iw \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & iw & i & -w & -1 & -iw & -i & w \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \\ \psi_6 \\ \psi_7 \end{pmatrix} = \begin{pmatrix} \psi_0 + \psi_1 + \psi_2 + \psi_3 + \psi_4 + \psi_5 + \psi_6 + \psi_7 \\ \psi_0 - \psi_4 - i(\psi_2 - \psi_6) + w(\psi_1 - \psi_5) - iw(\psi_3 - \psi_7) \\ \psi_0 + \psi_4 - (\psi_2 + \psi_6) - i(\psi_1 + \psi_5) + i(\psi_3 + \psi_7) \\ \psi_0 - \psi_4 + i(\psi_2 - \psi_6) - iw(\psi_1 - \psi_5) + w(\psi_3 - \psi_7) \\ \psi_0 - \psi_1 + \psi_2 - \psi_3 + \psi_4 - \psi_5 + \psi_6 - \psi_7 \\ \psi_0 - \psi_4 - i(\psi_2 - \psi_6) - w(\psi_1 - \psi_5) + iw(\psi_3 - \psi_7) \\ \psi_0 + \psi_4 - (\psi_2 + \psi_6) + i(\psi_1 + \psi_5) - i(\psi_3 + \psi_7) \\ \psi_0 - \psi_4 + i(\psi_2 - \psi_6) + iw(\psi_1 - \psi_5) - w(\psi_3 - \psi_7) \end{pmatrix}$$

Fast Fourier Transform (FFT)

The number of multiplications in the DFT ($\tilde{\Psi}_N = F_N \Psi_N$) is N^2.

For $N = 2^m$, the number of multiplications in FFT is $mN/2 = N \log_2 N/2$.

The reasoning is as follows. There are m levels, going from $N = 2^m$ down to $N = 2$. Each level has $N/2$ multiplications from the diagonal D's, to reassemble the half-size outputs from the lower level. This gives the final count $mN/2 = N \log_2 N/2$.

Fast Fourier Transform (FFT)

The number of multiplications in the DFT ($\tilde{\Psi}_N = F_N \Psi_N$) is N^2.

For $N = 2^m$, the number of multiplications in FFT is $mN/2 = N \log_2 N/2$.

The reasoning is as follows. There are m levels, going from $N = 2^m$ down to $N = 2$. Each level has $N/2$ multiplications from the diagonal D's, to reassemble the half-size outputs from the lower level. This gives the final count $mN/2 = N \log_2 N/2$.

$N=2^m$	N^2	$N \log_2 N /2$
64	4096	192
256	65536	1024
1024	1048576	5120
4096	16777216	24576
16384	268435456	114688
65536	4294967296	524288

A sample code – fft_cpu.cpp

https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm

```
#include <complex>
#include <cstdio>
...
void fft2 (complex<double>* X, int N) {
    if(N < 2) {
      // bottom of recursion. Do nothing here. because already X[0] = x[0]
    }
    else {
      separate(X,N);      // even-odd permutation
      fft2(X,     N/2);   // recurse even items
      fft2(X+N/2, N/2);   // recurse odd  items
      // combine results of two half recursions
      for(int k=0; k<N/2; k++) {
        complex<double> e = X[k    ];   // even
        complex<double> o = X[k+N/2];   // odd
        complex<double> w = exp( complex<double>(0,-2.*M_PI*k/N) );
        X[k    ] = e + w * o;
        X[k+N/2] = e - w * o;
      }
    }
}


// the complete code at twqcd80:/home/cuda_lecture_2021/FFT/fft_cpu.cpp
```

A sample code – fft_cpu.cpp (cont)

```
void separate (complex<double>* a, int n) {   // even-odd permutation
    complex<double>* b = new complex<double>[n/2];  // get temp heap storage
    for(int i=0; i<n/2; i++) b[i] = a[i*2+1];
    for(int i=0; i<n/2; i++) a[i] = a[i*2];
    for(int i=0; i<n/2; i++) a[i+n/2] = b[i];
    delete[] b;                                     // delete heap storage
}

int main () {
    const int nSamples = 64;
    double nSeconds = 1.0;                          // total time for sampling
    double sampleRate = nSamples / nSeconds;        // n Hz = n / second
    double freqResolution = sampleRate / nSamples;  // freq step in FFT result
    complex<double> x[nSamples];                    // storage for sample data
    complex<double> X[nSamples];                    // storage for FFT answer
    const int nFreqs = 5;
    double freq[nFreqs] = { 2, 5, 11, 17, 29 };     // known freqs for testing

    for(int i=0; i<nSamples; i++) {            // generate samples for testing
        x[i] = complex<double>(0.,0.);
        for(int j=0; j<nFreqs; j++)
          x[i] += sin( 2*M_PI*freq[j]*i/nSamples );
        X[i] = x[i];        // copy into X[] for FFT work & result
    }
    fft2(X,nSamples);       // compute fft for this data
    ...
}  // the complete code at twcp80:/home/cuda_lecture_2021/FFT/fft_cpu.cpp
```

cuFFT – GPU accelerated FFT

https://developer.nvidia.com/cufft

https://docs.nvidia.com/cuda/cufft/index.html

The NVIDIA CUDA Fast Fourier Transform library (cuFFT) provides GPU-accelerated FFT implementations that perform up to 10x faster than CPU-only alternatives. Using simple APIs, you can accelerate existing CPU-based FFT implementations in your applications with minimal code changes. As your application grows, you can use cuFFT to scale your image and signal processing applications efficiently across multiple GPUs.

cuFFT is a foundational library based on the well-known Cooley-Tukey and Bluestein algorithms. It is used for building commercial and academic applications across disciplines such as computational physics, molecular dynamics, quantum chemistry, seismic and **medical imaging**. With support for batched transforms and optimized precision arithmetic, it is widely used for building **deep-learning** based computer vision applications.

A sample code – fft_cuFFT.cu

```
#include <complex>
#include <stdio>
#include <cufft.h>
...
int main ()
{
 ...
 cufftHandle plan;
 cufftDoubleComplex *data;
 cudaMalloc((void**)&data, sizeof(cufftDoubleComplex)*nSamples*BATCH);
 cudaMemcpy(data, x, sizeof(double)*nSamples*2, cudaMemcpyHostToDevice);

 if (cufftPlan1d(&plan, nSamples, CUFFT_Z2Z, BATCH) != CUFFT_SUCCESS) {
   fprintf(stderr, "CUFFT error: Plan creation failed.\n");
   exit(1);
 }
 if (cufftExecZ2Z(plan, data, data, CUFFT_FORWARD) != CUFFT_SUCCESS) {
   fprintf(stderr, "CUFFT error: ExecZ2Z forward failed.\n");
   exit(1);
 }
 cudaMemcpy(X2, data, sizeof(double)*nSamples*2, cudaMemcpyDeviceToHost);
...
 cufftDestroy(plan);
 cudaFree(data);

}  // the complete code at twqcd80:/home/cuda_lecture_2021/FFT/fft_cuFFT.cu
```

Physical Applications

Poisson Equation $\nabla^2\phi(\vec{r}) = -4\pi\rho(\vec{r})$

$$\tilde{\phi}(\vec{k}) = \int d^3r \exp(-i\vec{k}\cdot\vec{r})\ \phi(\vec{r}), \quad \phi(\vec{r}) = \frac{1}{(2\pi)^3}\int d^3k \exp(+i\vec{k}\cdot\vec{r})\ \tilde{\phi}(\vec{k}),$$

$$\tilde{\rho}(\vec{k}) = \int d^3r \exp(-i\vec{k}\cdot\vec{r})\ \rho(\vec{r}), \quad \rho(\vec{r}) = \frac{1}{(2\pi)^3}\int d^3k \exp(+i\vec{k}\cdot\vec{r})\ \tilde{\rho}(\vec{k}),$$

$$\nabla^2\phi(\vec{r}) = -4\pi\rho(\vec{r}) \quad \Rightarrow$$

$$\frac{1}{(2\pi)^3}\int d^3k(-k^2)\exp(+i\vec{k}\cdot\vec{r})\ \tilde{\phi}(\vec{k}) = -\frac{4\pi}{(2\pi)^3}\int d^3k \exp(+i\vec{k}\cdot\vec{r})\ \tilde{\rho}(\vec{k})$$

$$\Rightarrow \tilde{\phi}(\vec{k}) = \frac{4\pi\tilde{\rho}(\vec{k})}{k^2}, \quad \phi(\vec{r}) = \frac{1}{(2\pi)^3}\int d^3k \exp(+i\vec{k}\cdot\vec{r})\frac{4\pi\tilde{\rho}(\vec{k})}{k^2}.$$

After discretization of the space to a lattice, numerical solution can be obtained by FFT. Obviosuly, the same strategy can be applied to any partial differential equations.