Discrete Fourier Transform (DFT)

Recall complex form:

$$f(x) = \sum_{n=-\infty}^{\infty} F_n e^{i\pi n x/L}, \quad -L \le x \le L$$

$$F_n = \frac{1}{2L}\int_{-L}^{L} f(x) e^{-i\pi n x/L}\, dx = \frac{1}{2L}\int_{0}^{2L} f(x) e^{-i\pi n x/L}\, dx \quad 0 \le x \le 2L$$

How do programming language like Matlab or Python implement these? → discrete

Take $L = 1/2$, and let $x \to x + 1/2$:

$$f(x) = \sum_{n=-\infty}^{\infty} F_n e^{2\pi i n x}, \quad 0 \le x \le 1$$

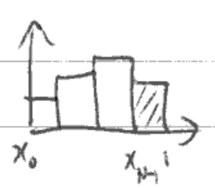
$$F_n = \int_0^1 f(x) e^{-2\pi i n x}\, dx$$

Discretize integral: $x_j = j\Delta x$, $j = 0, \cdots, N-1$

$\Delta x = \frac{1}{N}$

$x_0 = 0$, $x_{N-1} = \frac{N-1}{N}$

Recall Riemann sum: $\int_0^1 g(x)\, dx = \sum_{j=0}^{N-1} g(x_j)\Delta x_j$ as $N \to \infty$.



$$F_n = \sum_{j=0}^{N-1} f(x_j)\, e^{-2\pi i n x_j} \Delta x$$

$$= \frac{1}{N}\sum_{j=0}^{N-1} f(x_j)\, e^{-2\pi i n j/N}$$

But now something strange happens:

$$F_{n+N} = \frac{1}{N}\sum_{j=0}^{N-1} f(x_j)\, e^{-2\pi i (n+N) j/N} \qquad e^{-2\pi i j} = 1$$

$$= F_n \ !$$ So F_n is periodic in "Fourier space"!

$\Rightarrow$ For N sample points of $f(x)$, we get N Fourier coefficients.

N even: $\quad -\frac{N}{2} \ \cdots \ -1 \ 0 \ 1 \ \cdots \ \frac{N}{2}$

$F_{-N/2} = F_{N/2}$, so drop one.

Computer language list coeff as: $\quad 0 \ 1 \ 2 \cdots \frac{N}{2}-1 \ -\frac{N}{2} \cdots -1$

$$f(x) = \sum_{n=-\infty}^{\infty} F_n\, e^{2 i n \pi x}$$

$$f(x_j) = \sum_{n=-N/2}^{N/2-1} F_n\, e^{2\pi i n x_j}, \qquad x_j = j\Delta x$$

$$= \sum_{n=0}^{N-1} F_n\, e^{2\pi i n x_j}$$ since $F_n e^{2\pi i n x_j}$ has period N.

To summarize:

$$f(x_j) = \sum_{n=0}^{N-1} F_n \, e^{2\pi i n x_j} \quad , \quad j = 0, \cdots, N-1$$
$$x_j = j\Delta x$$

$$F_n = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j)\, e^{-2\pi i n x_j}$$

Let's check that these are inverses of each other:

$$f(x_j) = \sum_{n=0}^{N-1} \left[\frac{1}{N} \sum_{k=0}^{N-1} f(x_k)\, e^{-2\pi i n x_k} \right] e^{2\pi i n x_j}$$

$$= \sum_{k=0}^{N-1} \frac{f(x_k)}{N} \sum_{n=0}^{N-1} e^{2\pi i n (x_j - x_k)}$$

$$\sum_{n=0}^{N-1} x^n = \frac{x^N - 1}{x - 1}$$

<u>Claim:</u> $S_N = \frac{1}{N} \sum_{n=0}^{N-1} e^{2\pi i n (x_j - x_k)} = \delta_{j,k}$.

If <u>$j = k$</u>: $S_N = \frac{1}{N} \sum_{n=0}^{N-1} 1 = 1$

<u>$j \neq k$</u>: $S_N = \dfrac{\left(e^{2\pi i (x_j - x_k)}\right)^N - 1}{\left(e^{2\pi i (x_j - x_k)} - 1\right)} = 0,$

since $e^{2\pi i (x_j - x_k) N} = e^{2\pi i (j-k)} = 1$

Most programming languages drop the $1/N$ from F_n:

$$f(x_j) = \frac{1}{N}\sum_{n=0}^{N-1} F_n e^{2\pi i n x_j}$$

$$F_n = \sum_{j=0}^{N-1} f(x_j)\, e^{-2\pi i n x_j}$$

example in Matlab: `x = linspace(0,1,11);`
`x(end) = [];` % drop last point ($x=1$)
So $N=10$

$F = \text{fft}(\underbrace{\sin(2*\text{pi}*x)}_{f})$

← see below why

$F_n \rightarrow$ 0 $\boxed{-5i}$ 0 0 0 0 0 0 0 $\boxed{5i}$

$n =$ 0 1 2 3 4 −5 −4 −3 −2 −1

Note that $F_1 = \overline{F_{-1}}$ (reality condition)

Inverse transform: $\text{norm}(\text{ifft}(F) - f) \approx 10^{-16}$
("machine eps")

$$f(x) = \sin(2\pi x) = \frac{1}{2i}\left(e^{2\pi i x} - e^{-2\pi i x}\right)$$

$$= \frac{1}{N} F_1 e^{2\pi i x} - \frac{1}{N} F_{-1} e^{-2\pi i x}$$

So: $\frac{F_1}{N} = \frac{1}{2i}$

$F_1 = \frac{N}{2i} = -\frac{10}{2} i = \boxed{-5i}$

as above