Discrete Fourier Transform: W = W_N

$$F_n = \frac{1}{N}\sum_{k=0}^{N-1} f_k e^{-i\frac{2\pi nk}{N}} = \frac{1}{N}\sum_{k=0}^{N-1} f_k W^{-kn}$$

(transformed data)

$$f_n = \sum_{k=0}^{N-1} F_k e^{i\frac{2\pi nk}{N}} = \sum_{k=0}^{N-1} F_k W^{nk}$$

(original data - inverse transformation)

Example: Consider the data

$$(0,1),\ (\tfrac{1}{6},\tfrac{1}{2}),\ (\tfrac{2}{6},\tfrac{-1}{2}),\ (\tfrac{3}{6},-1),\ (\tfrac{4}{6},\tfrac{-1}{2}),\ (\tfrac{5}{6},\tfrac{1}{2})$$

- $T = 1$, $t_n = (1/6)n$, $f = 1, \frac{1}{2}, -\frac{1}{2}, -1, -\frac{1}{2}, \frac{1}{2})$
- $W = e^{i\pi/3} = \cos\left(\frac{\pi}{3}\right) + i\sin\left(\frac{\pi}{3}\right) = \frac{1}{2} + i\frac{\sqrt{3}}{2}$
- $W^2 = e^{i2\pi/3} = \cos\left(\frac{2\pi}{3}\right) + i\sin\left(\frac{2\pi}{3}\right) = -\frac{1}{2} + i\frac{\sqrt{3}}{2}$
- $W^3 = e^{i3\pi/3} = \cos(\pi) + i\sin(\pi) = -1 + i0$
- $W^4 = e^{i4\pi/3} = \cos\left(\frac{4\pi}{3}\right) + i\sin\left(\frac{4\pi}{3}\right) = -\frac{1}{2} - i\frac{\sqrt{3}}{2}$
- $W^5 = e^{i5\pi/3} = \cos\left(\frac{5\pi}{3}\right) + i\sin\left(\frac{5\pi}{3}\right) = \frac{1}{2} - i\frac{\sqrt{3}}{2}$
- $W^6 = W^0 = 1$

$$F_n = \frac{1}{N}\sum_{k=0}^{N-1} f_k W^{-kn}\text{, for n=0:5}$$

- $F_0 = \frac{1}{6}(f_0 W^0 + f_1 W^{0\cdot 1} + f_2 W^{0\cdot 2} + f_3 W^{0\cdot 3} + f_4 W^{0\cdot 4} + f_4 W^{0\cdot 5}) \quad = 0$
- $F_1 = \frac{1}{6}(f_0 W^0 + f_1 W^{-1\cdot 1} + f_2 W^{-1\cdot 2} + f_3 W^{-1\cdot 3} + f_4 W^{-1\cdot 4} + f_4 W^{-1\cdot 5}) = \frac{1}{2}$
- $F_2 = \frac{1}{6}(f_0 W^0 + f_1 W^{-2\cdot 1} + f_2 W^{-2\cdot 2} + f_3 W^{-2\cdot 3} + f_4 W^{-2\cdot 4} + f_4 W^{-2\cdot 5}) = 0$
- $F_3 = \frac{1}{6}(f_0 W^0 + f_1 W^{-3\cdot 1} + f_2 W^{-3\cdot 2} + f_3 W^{-3\cdot 3} + f_4 W^{-3\cdot 4} + f_4 W^{-3\cdot 5}) = 0$
- $F_4 = \frac{1}{6}(f_0 W^0 + f_1 W^{-4\cdot 1} + f_2 W^{-4\cdot 2} + f_3 W^{-4\cdot 3} + f_4 W^{-4\cdot 4} + f_4 W^{-4\cdot 5}) = 0$
- $F_5 = \frac{1}{6}(f_0 W^0 + f_1 W^{-5\cdot 1} + f_2 W^{-5\cdot 2} + f_3 W^{-5\cdot 3} + f_4 W^{-5\cdot 4} + f_4 W^{-5\cdot 5}) = \frac{1}{2}$

$$F_n = \frac{1}{N}\sum_{k=0}^{N-1} f_k W^{-kn}$$, for n=0:5

Interpolating function with period T=1:

$$f(t) = \sum_{k=0}^{N-1} F_k e^{i2\pi kt}$$

$$= F_1 e^{i\mathbf{2\pi t}} + F_5 e^{i\mathbf{10\pi t}}$$

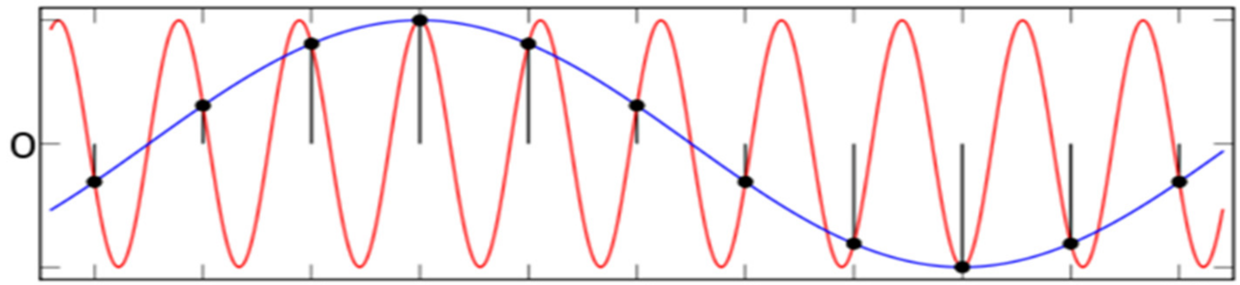
$$= F_1 e^{i\mathbf{2\pi t}} + F_{-1} e^{-\boldsymbol{i2\pi t}}$$

$$= \cos 2\pi t$$

Since $F_5 e^{i10\pi t}$ = $F_{-1} e^{-i2\pi t}$ by previous analysis.

Aliasing

- Consider the following situation – multiple harmonics "fit" the observations



- The coefficients in the DFT may be affected by some higher order frequencies from the CFT (frequencies above N/(2T) – Nyquist frequency)
- May cause poor digital images or "echos" on radio signals.
- Solution: Sample at a higher rate (i.e. more often) or filter the data before digitizing.

More on the DFT

- The DFT (and its Inverse) allow us to more between original data (time) and transformed data (frequency).
- We've seen how we can truncate some of the signals (and we will see more later, e.g. image compression).
- The transforms are really only useful if they can be performed quickly → Fast Fourier Transform (FFT) algorithm.

FFT Algorithm

$$F_n = \frac{1}{N}\sum_{k=0}^{N-1} f_k W^{-kn} \text{ and } f_n = \sum_{k=0}^{N-1} F_k W^{nk}$$

- We can calculate all F_n in $O(N^2)$ steps.
- We will see how to do it in O(N log n) steps.

FFT Algorithm: $O(n^2)$ vs $O(n \log n)$

- Is this difference actually significant?
  - Assume flops/s ≈ 5e12 (5 Terra-flops/s)
  - Problem of size $N = 2^{20}$:
    - N^2 steps require 0.22 seconds
    - N*log(N) steps require 0.0000042 seconds
  - How big a problem can be solved in 1s by:
    - $O(N^2)$ algorithm:
    - O(N*log(N)) algorithm:
  - By 1960s, computing power sufficient to make algorithm worthwhile