The Fourier transform and the FFT algorithm

Steffen Haug

The Fourier transform

$$\hat{f}(\xi) = \int\limits_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt \tag{1}$$

The Fourier transform

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt \tag{1}$$

Analytic evaluation generally not feasible.

$$\hat{f}(\xi) = \int\limits_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

Most engineering applications use its discrete counterpart – the DFT:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i k n}{N}} \qquad (2)$$

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i k n}{N}}$$

We want a way to compute this *efficiently*.

Preliminaries

Eulers formula and the roots of unity

Eulers identity

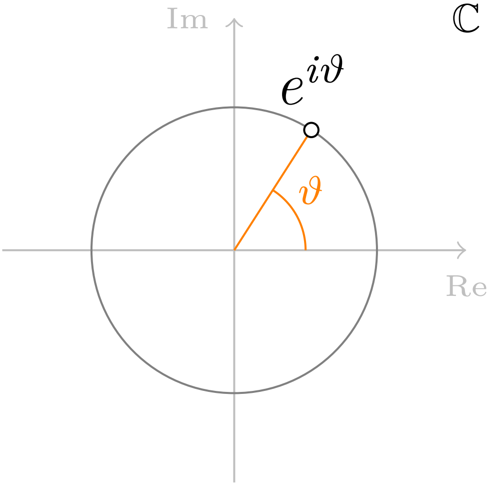
$$e^{i\pi} = -1 \tag{3}$$

$$e^{i\pi} = -1$$

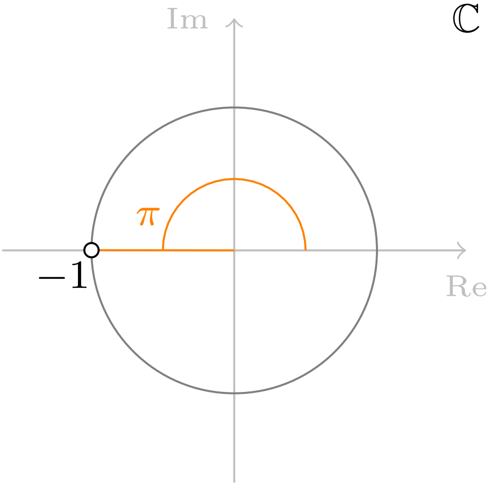
Just a special case of $e^{i\vartheta}$ when $\vartheta = \pi$.

$$e^{i\vartheta} = \cos\vartheta + i\sin\vartheta \tag{4}$$

$$e^{i\vartheta} = \cos\vartheta + i\sin\vartheta$$



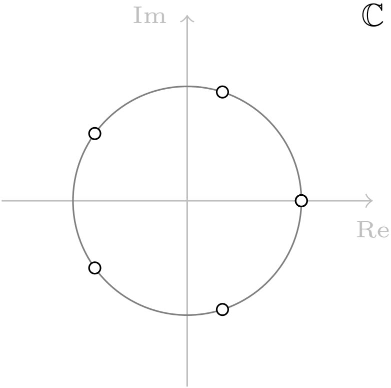
$$e^{i\pi} = -1$$



The roots of unity

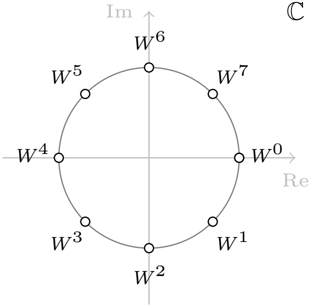
$$z^N = 1$$

$$z^N = 1$$



SOLUTIONS: $\left(e^{\frac{2\pi i}{N}}\right)^n$ for n in $0, \ldots, N-1$.

Let $W = e^{\frac{-2\pi i}{N}}$ (notation from Cooley-Tukey).



We can restate the DFT (4) as

$$X_k = \sum_{n=0}^{N-1} x_n W^{nk} \tag{5}$$

Foreshadowing

The goal is to recursively decompse the DFT in such a way that *intermediate calculations* are *overlapping* or *symmetrically opposite* on the unit circle.

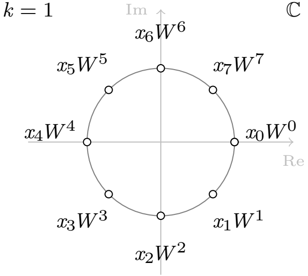
Thus, they may be reused directly or by simply a change of sign.

$$X_k = \sum_{n=0}^{N-1} x_n \, W^{nk}$$

So what is the DFT *actually* doing?

$$X_k = \sum_{n=0}^{N-1} x_n W^{nk}$$

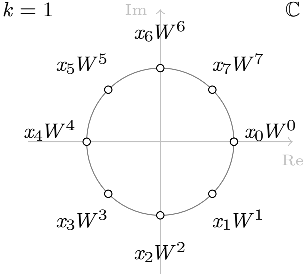
So what is the DFT *actually* doing?

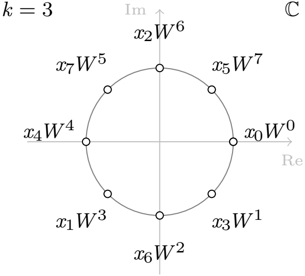


Just multiplying our N array values by the Nth roots of unity!

$$X_k = \sum_{n=0}^{N-1} x_n \, W^{nk}$$

So what is the DFT *actually* doing?





k is the "stride" along the roots of unity – the "frequency".

(If k is larger, we go faster around the circle)

The key idea

Decomposing the DFT

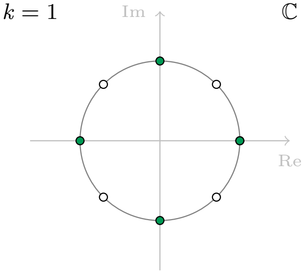
We can decompose (5) into a a sum of sums:

$$
\begin{aligned}
X_k &= \sum_{n=0}^{N-1} x_n W^{nk} \\
&= \underbrace{\sum_{n=0}^{N/2-1} x_{2n} W^{(2n)k}}_{\text{Even indices}} + \underbrace{\sum_{n=0}^{N/2-1} x_{2n+1} W^{(2n+1)k}}_{\text{Odd indices}}
\end{aligned}
$$

We can just as well decompose it in three sums that sum every 3rd value and so on.

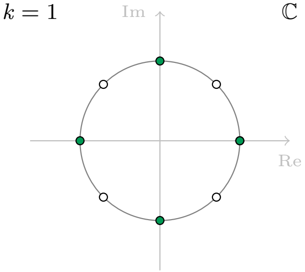
Remember

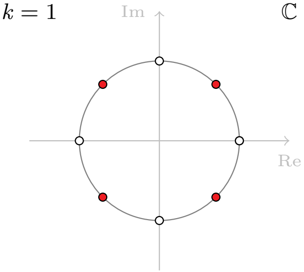
An N-point DFT uses the Nth roots of unity!



$$\sum_{n=0}^{N/2-1} x_{2n}\, W^{2nk} \qquad \sum_{n=0}^{N/2-1} x_{2n+1}\, W^{(2n+1)k}$$

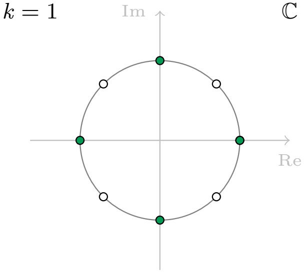
The odd-index sum does not use the N/2th roots of unity.

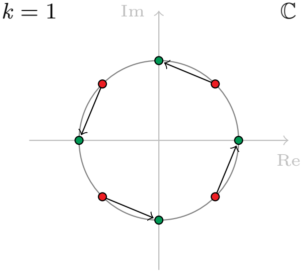




$$\sum_{n=0}^{N/2-1} x_{2n} \, {\color{green}W^{2n}}{}^{k} \qquad \sum_{n=0}^{N/2-1} x_{2n+1} \, {\color{red}W^{(2n+1)}}{}^{k}$$

It is clearly not a proper N/2-point DFT!





$$\sum_{n=0}^{N/2-1} x_{2n}\, {\color{green}W^{2nk}} \qquad {\color{red}W^k} \sum_{n=0}^{N/2-1} x_{2n+1}\, {\color{green}W^{2nk}}$$

Factor out W^k to get what seems like a proper DFT.

Correct decomposition of the DFT

$$X_k = \sum_{n=0}^{N/2-1} x_{2n} W^{2nk} + W^k \sum_{n=0}^{N/2-1} x_{2n+1} W^{2nk} \qquad (6)$$

Correct decomposition of the DFT

$$X_k = \sum_{n=0}^{N/2-1} x_{2n} \underset{!}{W^{2nk}} + \underset{!}{W^k} \sum_{n=0}^{N/2-1} x_{2n+1} \underset{!}{W^{2nk}} \tag{6}$$

But wait...

E_k, O_k is supposedly N/2-point DFTs, but $0 \leq k < N$?

Almost correct
~~Correct~~ decomposition of the DFT

$$X_k = \sum_{n=0}^{N/2-1} x_{2n}\, W^{2nk} + W^k \sum_{n=0}^{N/2-1} x_{2n+1}\, W^{2nk} \tag{6}$$

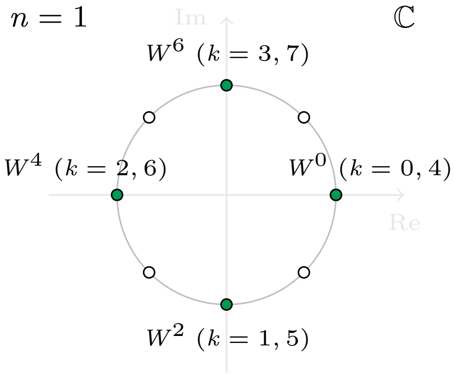
! ! !

IN FACT, the sums yield the correct result if evaluated directly for every k, but fails to take advantage of any symmetry, and thus gives no performance benefit.

Gaining speed

Reusing intermediate results

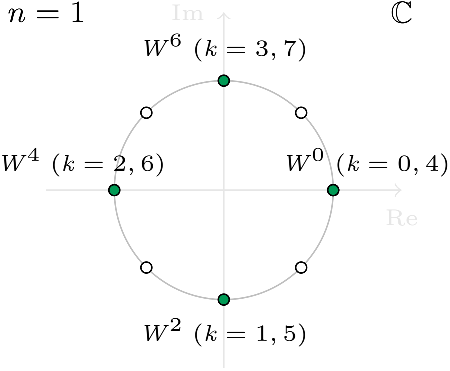
Let's look closer at W^{2nk}



Upper half range of W^{2nk} overlaps with lower half for $N = 1$

$n = 1$

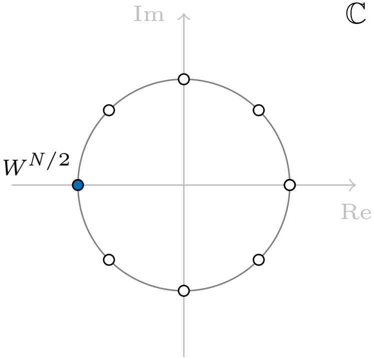
$\mathbb{C}$



This holds in general:

$$W^{2n(k+N/2)} = W^{2nk} \underbrace{W^{\frac{\cancel{2}nN}{\cancel{2}}}}_{=1} = W^{2nk}$$

Now, let's look closer at W^k



$$W^{k+N/2} = W^k W^{N/2} = -W^k$$

Upper range of W^k symmetrically opposite to lower range!

$$X_k = \sum_{n=0}^{N/2-1} x_{2n} \underbrace{W^{2nk}}_{\text{Overlapping}} + \underbrace{W^k}_{\text{Symmetrically opposite}} \sum_{n=0}^{N/2-1} x_{2n+1} \underbrace{W^{2nk}}_{\text{Overlapping}} \tag{6}$$

$$X_k = \sum_{n=0}^{N/2-1} x_{2n} \underbrace{W^{2nk}}_{\text{Overlapping}} + \underbrace{W^k}_{\text{Symmetrically opposite}} \sum_{n=0}^{N/2-1} x_{2n+1} \underbrace{W^{2nk}}_{\text{Overlapping}} \tag{6}$$

Decomposition

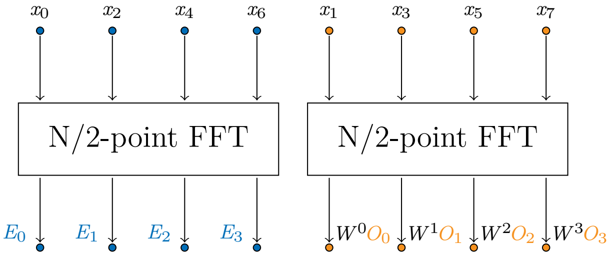
$$\begin{cases} X_k & = \quad E_k + W^k O_k \\ X_{k+N/2} & = \quad E_k - W^k O_k \end{cases} \tag{7}$$

$k \in [0, N/2-1]$

Algorithm

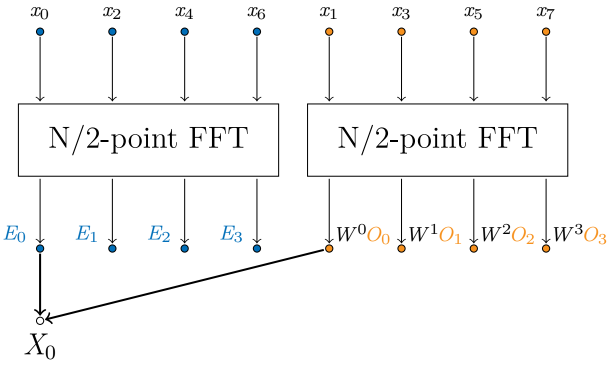
$\{x_n\}$ ordered by index parity.

Algorithm



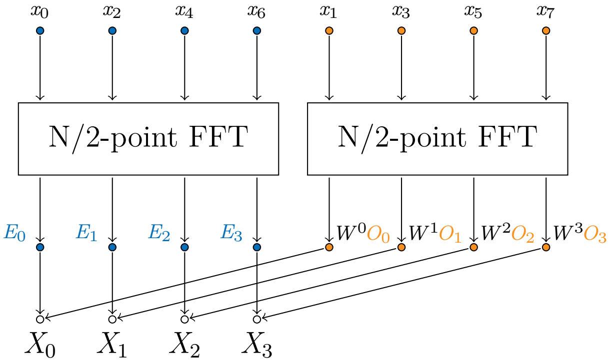
Recursively compute the N/2-point FFTs E and O.

Algorithm



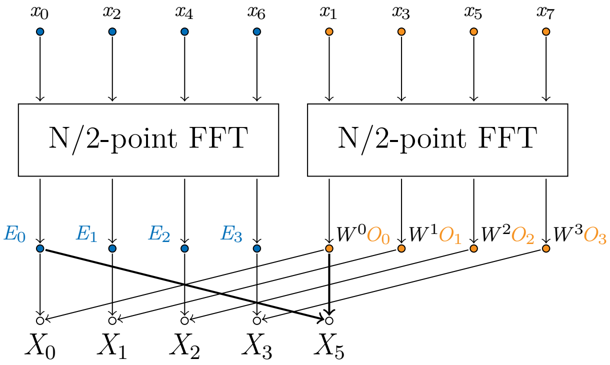
$$X_0 = E_0 + W^0O_0...$$

Algorithm



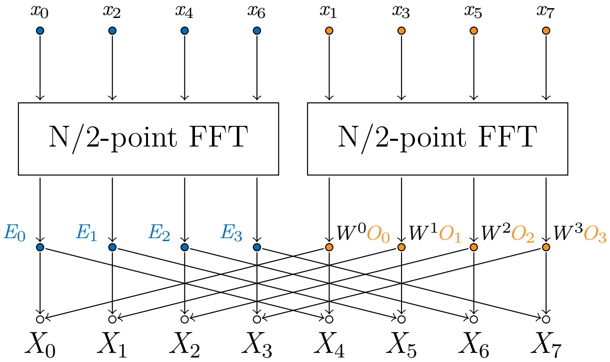
...and so on for all $k < N/2$.

Algorithm



$$X_5 = E_0 - W^0 O_0 ...$$

Algorithm



...and so on for all $k \geq N/2$.

Complexity analysis

Just for the record

$$T(n) = \begin{cases} 2\,T(n/2) + n & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

This is not very deep – straight forward application of the Master Theorem:

$$\log_2 2 = 1 \implies T(n) = O(n \log n)$$

References

Cooley & Tukey: *An algorithm for the machine calculation of complex Fourier series*
Johnson & Frigo: *Implementing FFTs in practice*
Falch: *The Discrete and Fast Fourier Transforms*
Bogart & Stein: *Discrete Math in Computer Science* (Section 5.2: The Master theorem)