Calculating DFT

\[ X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} = \sum_{n=0}^{N-1} x(n)(e^{-j2\pi/N})^{nk} \]

Denote \( W_N = e^{-j2\pi/N} \), then \( X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk} \)

- Each \( X(k) \) requires \( N \) complex multiplications and \( N \) complex additions.
- Each complex multiplication needs 4 real multiplications and 2 real addition because \((a+jb)\ast(c+jd) = (ac-bd)+j(bc+ad)\)
- There are \( N \) different \( X(k) \), so we need a total of
  - \( 4N^2 \) real multiplications
  - \( 4N^2 \) real additions

However, by taking advantages of some properties of \( W_N^m \), we can significantly speed out the calculations ⇒ Fast Fourier Transform

Interesting Properties of \( W_N^m \):

1. \( W_N^0 = (e^{-j2\pi/N})^0 = e^0 = 1 \), \( W_N^N = e^{-j2\pi} = 1 \)
2. \( W_N^{N+m} = W_N^m \) (periodic)
3. Assume \( N \) is a multiple of 4
   \( W_N^{N/2} = e^{-j2\pi/(N/2)/N} = e^{-j\pi/2} = -j \)
   \( W_N^{N/4} = e^{-j2\pi/(N/4)/N} = e^{-j\pi/4} = -j^{1/2} \)
   \( W_N^{3N/4} = e^{-j2\pi/(3N/4)/N} = e^{-j3\pi/4} = j^{1/2} \)
4. Assume \( N \) is even
   \( W_N^{2k} = (e^{-j2\pi/N})^{2k} = (e^{-j2\pi/(N/2)})^{2k} = W_N^{kr} \)
Example 10-3: Two-Point DFT

\[ x(0), x(1): \quad X(k) = \sum_{n=0}^{1} x(n)W_2^{nk} \quad k = 0, 1 \]

\[ X(0) = \sum_{n=0}^{1} x(n)W_2^{0n} = \sum_{n=0}^{1} x(n) = x(0) + x(1) \]

\[ X(1) = \sum_{n=0}^{1} x(n)W_2^{n1} = \sum_{n=0}^{1} x(n)W_2^{n} \]

\[ = x(0)W_2^{0} + x(1)W_2^{1} \]

\[ = x(0) + x(1)(-1) \]

\[ = x(0) - x(1) \]

Example 10-4: Four-point DFT of \( x(0), x(1), x(2), x(3) \)

\[ X(k) = \sum_{n=0}^{3} x(n)W_4^{nk} \quad k = 0, 1, 2, 3, \]

\[ X(0) = \sum_{n=0}^{3} x(n)W_4^{0n} = \sum_{n=0}^{3} x(n) = x(0) + x(1) + x(2) + x(3) \]

\[ X(1) = \sum_{n=0}^{3} x(n)W_4^{n1} = x(0)W_4^{0} + x(1)W_4^{1} + x(2)W_4^{2} + x(3)W_4^{3} \]

\[ = x(0) - jx(1) - x(2) + jx(3) \]

\[ X(2) = \sum_{n=0}^{3} x(n)W_4^{2n} = x(0)W_4^{0} + x(1)W_4^{2} + x(2)W_4^{4} + x(3)W_4^{6} \]

\[ = x(0) + x(1)(-1) + x(2)(1) + x(3)W_4^{2} \]

\[ = x(0) - x(1) + x(2) - x(3) \]
\[
X(3) = \sum_{n=0}^{3} x(n)W_4^{3n} = x(0)W_4^0 + x(1)W_4^1 + x(2)W_4^2 + x(3)W_4^3 \\
= x(0) + x(1)W_4^1 + x(2)W_4^2 + x(3)W_4^3 \\
= x(0) + jx(1) + (-1)x(2) + (-j)x(3) \\
= x(0) + jx(1) - x(2) - jx(3)
\]

\[
X(0) = [x(0) + x(2)] + [x(1) + x(3)] \\
X(1) = [x(0) - x(2)] + (-j)[x(1) - x(3)] \\
X(2) = [x(0) + x(2)] - [x(1) + x(3)] \\
X(3) = [x(0) - x(2)] + j[x(1) - x(3)]
\]

If we denote \(z(0) = x(0), \ z(1) = x(2)\) then \(Z(0) = z(0) + z(1) = x(0) + x(2)\)
\(Z(1) = z(0) - z(1) = x(0) - x(2)\)

\(v(0) = x(1), \ v(1) = x(3)\) then \(V(0) = v(0) + v(1) = x(1) + x(3)\)
\(V(1) = v(0) - v(1) = x(1) - x(3)\)

Our four-point DFT:
\[
X(0) = Z(0) + V(0) \\
X(1) = Z(1) + (-j)V(1) \\
X(2) = Z(0) - V(0) \\
X(3) = Z(1) + jV(1)
\]

Key point: We compute 4-point DFT based on two 2-point DFTs
Decimation-in-Time FFT Algorithm

\[ x(0), x(1), \ldots, x(N-1) \quad N = 2^m \]

Separate into even and odd samples:

\[ g(r) = x(2r) \quad h(r) = x(2n + 1) \]

\[ X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn} \]

\[ = \sum_{r=0}^{N/2-1} g(r)W_N^{k(2r)} + \sum_{r=0}^{N/2-1} h(r)W_N^{k(2r+1)} \quad (k = 0,1,\ldots,N-1) \]

\[ = \sum_{r=0}^{N/2-1} g(r)W_N^{2kr} + W_N^{k} \sum_{r=0}^{N/2-1} h(r)W_N^{2kr} \]

Using property 4:

\[ W_N^{2kr} = \left( e^{-j2\pi/N} \right)^{2kr} = \left( e^{-j2\pi/(N/2)} \right)^{kr} = W_N^{kr} \]

\[ \Rightarrow X(k) = \sum_{r=0}^{N/2-1} g(r)W_{N/2}^{kr} + W_N^{k} \sum_{r=0}^{N/2-1} h(r)W_{N/2}^{kr} \]

\[ = G(k) + W_N^{k} H(k) \]

where \( G(k) \) and \( H(k) \) are the N/2 point DFT of \( g(r) \) and \( h(r) \) respectively.

\[ X(k) = G(k) + W_N^{k} H(k) \quad k = 0,1,\ldots,N-1 \]

\[ G(k) = \sum_{r=0}^{N/2-1} g(r)W_{N/2}^{kr} = \sum_{r=0}^{N/2-1} x(2r)W_{N/2}^{kr} \]

\[ H(k) = \sum_{r=0}^{N/2-1} h(r)W_{N/2}^{kr} = \sum_{r=0}^{N/2-1} x(2r+1)W_{N/2}^{kr} \]

Question: \( X(k) \) needs \( G(k), H(k), \quad k=0 \ldots N-1 \)

How do we obtain \( G(k), H(k) \), for \( k > N/2-1 \)?

\[ G(k) = G(N/2+k) \quad k \leq N/2-1 \]

\[ H(k) = H(N/2+k) \quad k \leq N/2-1 \]
Such recursion can be carried on by considering the even and odd samples of the sequence at each stage. For a 8-point FFT, the resulting decomposition looks like:

What is the big deal?
(1) Very regular structure $\Rightarrow$ easy hardware (and software) implementation
(2) Big complexity savings!!

Why? Let’s say you want to compute 1024-point DFT.

Using direct computation, we need $4N^2 \sim 4$ million operations

Using the above strategy:

Let $C(N) =$ total operations for 1024-pt DFT

\[
C(1024) = 2\times1024 + 2C(512) : \text{One stage butterfly + Two 512-DFT} \\
= 2\times1024 + 2[2\times512+2C(256)] \\
= 4\times1024 + 4[2\times256+2C(128)] \\
= 6\times1024 + 8[2\times128+2C(64)] \\
= 8\times1024 + 16[2\times64+2C(32)] \\
=10\times1024+32[2\times32+2C(16)] \\
=12\times1024+64[2\times16+2C(8)] \\
=14\times1024+128[2\times8+2C(4)] \\
=16\times1024+256\times C(2) \\
=16\times1024+256\times4 = 17408 \text{ operations} \sim A \text{ factor of 100 savings!}
\]

In general, $C(N) \sim N(\log_2 N)$. The great savings come from the reduction of a factor $N$ to $(\log_2 N)$, thanks to the recursion.