Efficient Computation of the Discrete Fourier Transform (DFT)

Recall the DFT:

\[ X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \quad k = 0, 1, 2, \ldots, N-1 \]

or,

\[ X(k) = \sum_{n=0}^{N-1} x(n)W_{N}^{kn}, \quad k = 0, 1, 2, \ldots, N-1 \]

where \( W_{N} = e^{-j2\pi /N} \) and \( W_{N}^{kn} = e^{-j2\pi kn/N} \).

Note that \( W_{N}^{kn} \) are just samples on the unit circle:

\[ N = 4 \]
\[ k = 0, 1, 2, 3 \]
\[ n = 0, 1, 2, 3 \]

For example, \( W_{4}^{3(2)} = e^{-j2\pi /4}(3)(2) = e^{-j3\pi} = e^{-j\pi} = -1 \).

We note two important symmetry properties of \( W_{N}^{kn} \):

\[ W_{N}^{k+N/2} = -W_{N}^{k} \] (symmetry about the imaginary axis)
\[ W_{N}^{k+N} = W_{N}^{k} \] (periodicity)

This symmetry allows the number of computations for a DFT to be reduced significantly.
Computational Complexity

For a complex-valued sequence:

\[
X_R(k) = \sum_{n=0}^{N-1} x_R(n) \cos\left(\frac{2\pi kn}{N}\right) + x_I(n) \sin\left(\frac{2\pi kn}{N}\right)
\]

\[
X_I(k) = -\sum_{n=0}^{N-1} x_R(n) \sin\left(\frac{2\pi kn}{N}\right) - x_I(n) \cos\left(\frac{2\pi kn}{N}\right)
\]

Direct computation requires:

1. \(2N^2\) evaluations of trig functions (typically performed using table lookup — a trade-off of memory for speed)
2. \(4N^2\) real multiplications
3. \(4N(N-1)\) real additions
4. Misc. indexing and addressing operations

In general, we say that the complexity is \(O(N^2)\) — which implies it is not linearly proportional to the length of the input.

Why is this bad?
Divide and Conquer

Consider the case \( N = LM \) (N can be factored into a product of two integers):

\[
\begin{array}{cccccc}
  n=0 & n=1 & n=2 & \cdots & n=N-1 \\
  x(0) & x(1) & x(2) & \cdots & x(N-1) \\
\end{array}
\]

Consider the mapping: \( n = l + mL \):

<table>
<thead>
<tr>
<th>l/m</th>
<th>0</th>
<th>1</th>
<th>\cdots</th>
<th>M-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( x(0) )</td>
<td>( x(L) )</td>
<td>\cdots</td>
<td>( x((M-1)L) )</td>
</tr>
<tr>
<td>1</td>
<td>( x(1) )</td>
<td>( x(L+1) )</td>
<td>\cdots</td>
<td>( x((M-1)L+1) )</td>
</tr>
<tr>
<td>2</td>
<td>( x(2) )</td>
<td>( x(L+2) )</td>
<td>\cdots</td>
<td>( x((M-1)L+2) )</td>
</tr>
<tr>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>L-1</td>
<td>( x(L-1) )</td>
<td>( x(2L-1) )</td>
<td>\cdots</td>
<td>( x(ML-1) )</td>
</tr>
</tbody>
</table>

We can similarly map the DFT index \( k \) using \( k = Mp + q \) (or \( k = qL + p \)).

The DFT can be computed as:

\[
X(p, q) = \sum_{l=0}^{L-1} \left\{ W_N^{lq} \left[ \sum_{m=0}^{M-1} x(l, m) W_M^{mq} \right] \right\} W_L^{lp}
\]

The inner term represents an \( M \)-point DFT, while the outer term represents an \( L \)-point DFT. What is the advantage of this approach?

Example: \( N=1000 \)

Normal DFT (complexity \( N^2 \)): \( 10^6 \) operations

Divide and Conquer \((L = 2, \ M = 500)\): \( 5 \times 10^5 \) operations (2x reduction)

In general, the complexity of the divide and conquer approach is:

\( N(M + L + 1) \) complex multiplications
\( N(M + L - 2) \) additions

The complexity is reduced from \( O(N^2) \) to something less...
Example: Computation of a 15-Point DFT

An example of the flow of computations for a 15-point DFT decomposed into $L = 3$ and $M = 5$:

For $N$ highly composite (can be factored into a product of small prime numbers):

$$N = r_1 r_2 r_3 \ldots r_v$$

we can decompose a DFT of large order into a sequence of small DFTs.

For example,

$$N = 210 = 2 \times 3 \times 5 \times 7$$

Suppose $N$ is prime?

Can we impose additional constraints to further improve efficiency?