Supplement to FFT

The divide and conquer approach used in the FFT appears in many computer science approaches.

Suppose that an algorithm \( X \) applied to data of length \( N \) has runtime \( t(N) \). Suppose further that \( X(N) \) can be computed as follows:

1) Partition the data into two halves. (Divide.)
2) Conquer — run \( X(N/2) \) twice.
3) Merge the results.

If the merge takes \( C_1 \cdot N \) time, where \( C_1 \) is some constant, then

\[
t(N) \leq C_2 \cdot N \cdot \log_2(N).
\]

Let us prove this using induction on \( N \).

**Basis.** Take \( N = 4 \) or \( N = 2 \) or some other small value.

\( t(N) = C_3 \) in this case, and there exists some \( C_2 \) such that

\[
c_2 \cdot N \cdot \log_2(N) \geq C_3.
\]

**Inductive Step.** Assume for some \( N \) that

\[
t(n) \leq C_2 \cdot N \cdot \log_2(N).
\]

We will prove it for \( 2N \).
we need to show
\[ t(2^N) \leq C_2 \cdot 2^N \log_2 (2^N) \]
Recall from our divide and conquer approach that
\[ t(2^N) \leq 2 \cdot t(N) + C_1 \cdot N \]
\[ = 2 \cdot C_2 \cdot N \cdot \log_2 (N) + C_1 \cdot N \]
Suppose \( C_2 \geq C_1 \)
\[ \leq C_2 \cdot N \cdot [2 \cdot \log_2 (N) + 1] \]
\[ \leq C_2 \cdot N \cdot 2[\log_2 (C_2) + 1] \]
\[ = C_2 \cdot 2^N \cdot \log_2 (2^N). \]

Basis case of FFT
It is convenient to use DFT (2) or DFT (4) as the basis cases in our recursive implementation. My implementation from 2013 uses DFT(2), but DFT(4) is fine.

DFT(2):
\[
\begin{pmatrix}
X(0) \\
X(1)
\end{pmatrix} =
\begin{pmatrix}
w_2^0 & w_2^{0 \cdot 1} \\
w_2^{1 \cdot 0} & w_2^{1 \cdot 1}
\end{pmatrix}
\begin{pmatrix}
x(0) \\
x(1)
\end{pmatrix}
\]
\[ = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \end{pmatrix} \]
\[ = \begin{pmatrix} x(0) + x(1) \\ x(0) - x(1) \end{pmatrix}. \]
\[ DFT(4) \]

\[
\begin{bmatrix}
X(0) \\
X(1) \\
X(2) \\
X(3)
\end{bmatrix} =
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & -j & -1 & j \\
1 & -1 & 1 & -1 \\
1 & j & -1 & -j
\end{bmatrix}
\begin{bmatrix}
x(0) \\
x(1) \\
x(2) \\
x(3)
\end{bmatrix}
\]