CHAPTER 8

Computation of the Discrete Fourier Transform

Tutorial Problems

1. Solution:
   The resulting trend in the computational complexity of the direct DFT computations is of power 2 of the number of points N.

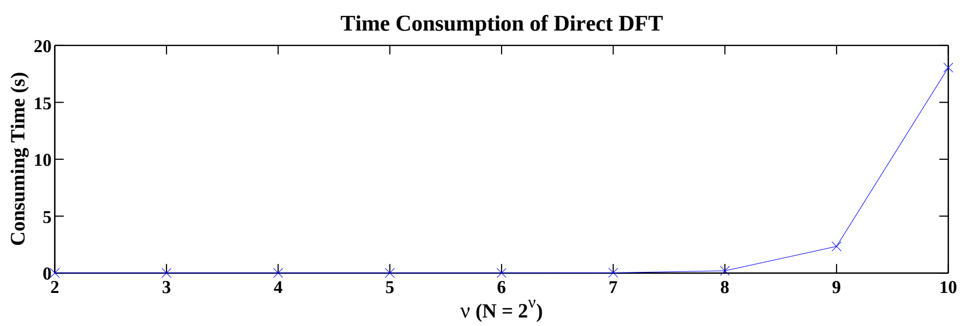


FIGURE 8.1: Plot of computation time for the `dftdirect` function for $N = 2^{\nu}$ where $2 \leq \nu \leq 10$.

MATLAB script:

```
% P0801: Investigate time consumption using direct DFT
close all; clc
nu = 2:10;
N = 2.^nu;
Ni = length(N);
t = zeros(1,Ni);
```

```
for ii = 1:Ni
    x = randn(1,N(ii)) + j*randn(1,N(ii));
    tic
    X = dftdirect(x);
    t(ii) = toc;
end
% Plot:
hfa = figconfg('P0801a','long');
plot(nu,t,'x-','markersize',12)
xlabel('\nu (N = 2^{\nu})','fontsize',LFS)
ylabel('Consuming Time (s)','fontsize',LFS)
title('Time Consumption of Direct DFT','fontsize',TFS)
```

2. (a) Solution:

The 4-point DIT matrix algorithm is:

$$\begin{bmatrix} X[0] \\ X[1] \\ \hline X[2] \\ X[3] \end{bmatrix} = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & W_4^2 & W_4^1 & W_4^3 \\ \hline 1 & 1 & W_4^2 & W_4^2 \\ 1 & W_4^2 & W_4^3 & W_4^1 \end{array}\right] \begin{bmatrix} x[0] \\ x[2] \\ \hline x[1] \\ x[3] \end{bmatrix}$$

which can be simplified as:

$$\begin{bmatrix} \boldsymbol{X}_T \\ \hline \boldsymbol{X}_B \end{bmatrix} = \left[\begin{array}{c|c} \boldsymbol{W}_2 & \boldsymbol{D}_4\boldsymbol{W}_2 \\ \hline \boldsymbol{W}_2 & -\boldsymbol{D}_4\boldsymbol{W}_2 \end{array}\right] \begin{bmatrix} \boldsymbol{x}_E \\ \hline \boldsymbol{x}_O \end{bmatrix}$$

where

$$\boldsymbol{W}_2 = \begin{bmatrix} 1 & 1 \\ 1 & W_4^2 \end{bmatrix}, \qquad \boldsymbol{D}_4 = \begin{bmatrix} 1 & 0 \\ 0 & W_4^1 \end{bmatrix}$$

Hence, we conclude as follows:

$$\begin{cases} \boldsymbol{X}_E = \boldsymbol{W}_2 \cdot \boldsymbol{x}_E \\ \boldsymbol{X}_O = \boldsymbol{W}_2 \cdot \boldsymbol{x}_O \end{cases}$$

and

$$\begin{cases} \boldsymbol{X}_T = \boldsymbol{X}_E + \boldsymbol{D}_4\boldsymbol{X}_O \\ \boldsymbol{X}_B = \boldsymbol{X}_E - \boldsymbol{D}_4\boldsymbol{X}_O \end{cases}$$

(b) Solution:

The 4-point DIF matrix algorithm is:

$$\begin{bmatrix} X[0] \\ X[2] \\ \hline X[1] \\ X[3] \end{bmatrix} = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & W_4^2 & 1 & W_4^2 \\ \hline 1 & W_4^1 & W_4^2 & W_4^3 \\ 1 & W_4^3 & W_4^2 & W_4^1 \end{array}\right] \begin{bmatrix} x[0] \\ x[1] \\ \hline x[2] \\ x[3] \end{bmatrix}$$

which can be simplified as:

$$\left[\begin{array}{c}\boldsymbol{X}_E\\ \hline \boldsymbol{X}_O\end{array}\right] = \left[\begin{array}{c|c}\boldsymbol{W}_2 & \boldsymbol{W}_2\\ \hline \boldsymbol{W}_2\boldsymbol{D}_4 & -\boldsymbol{W}_2\boldsymbol{D}_4\end{array}\right]\left[\begin{array}{c}\boldsymbol{x}_T\\ \hline \boldsymbol{x}_B\end{array}\right]$$

where

$$\boldsymbol{W}_2 = \begin{bmatrix} 1 & 1\\ 1 & W_4^2\end{bmatrix}, \qquad \boldsymbol{D}_4 = \begin{bmatrix} 1 & 0\\ 0 & W_4^1\end{bmatrix}$$

Hence, we conclude as follows:

$$\begin{cases}\boldsymbol{\nu} = \boldsymbol{x}_T + \boldsymbol{x}_B\\ \boldsymbol{z} = \boldsymbol{D}_4(\boldsymbol{x}_T - \boldsymbol{x}_B)\end{cases}$$

and

$$\begin{cases}\boldsymbol{X}_E = \boldsymbol{W}_2 \cdot \boldsymbol{\nu}\\ \boldsymbol{X}_O = \boldsymbol{W}_2 \cdot \boldsymbol{z}\end{cases}$$

3. (a) Solution:
Stage I:
The 8-point DFT $X[k]$ can be divided according to even and odd index k, we have

$$\begin{cases}\boldsymbol{v} = \boldsymbol{x}_T + \boldsymbol{x}_B\\ \boldsymbol{w} = \boldsymbol{D}_8(\boldsymbol{x}_T - \boldsymbol{x}_B)\end{cases} \implies \begin{cases}\boldsymbol{X}_E = \boldsymbol{W}_4\boldsymbol{v}\\ \boldsymbol{X}_O = \boldsymbol{W}_4\boldsymbol{w}\end{cases}$$

Stage II:
Each 4-point DFT $Y[k]$ can be divided according to even and odd index k, we have

$$\begin{cases}\boldsymbol{p} = \boldsymbol{y}_T + \boldsymbol{y}_B\\ \boldsymbol{q} = \boldsymbol{D}_4(\boldsymbol{y}_T - \boldsymbol{y}_B)\end{cases} \implies \begin{cases}\boldsymbol{Y}_E = \boldsymbol{W}_2\boldsymbol{p}\\ \boldsymbol{Y}_O = \boldsymbol{W}_4\boldsymbol{q}\end{cases}$$

Stage III:
Each 2-point DFT $Z[k]$ can be divided according to even and odd index k, we have

$$\begin{cases}m = z[0] + z[1]\\ n = \boldsymbol{D}_2(z[0] - z[1])\end{cases} \implies \begin{cases}Z[0] = m\\ Z[1] = n\end{cases}$$

(b) MATLAB function:

```
function Xdft = difrecur(x)
% Recursive computation of the DFT using divide & conquer
% N should be a power of 2
N = length(x);
Xdft = zeros(1,N);
if N ==1
  Xdft = x;
else
     m = N/2;
     D = exp(-2*pi*sqrt(-1)/N).^(0:m-1);
     v = x(1:N/2)+x(N/2+1:end);
     z = D.*(x(1:N/2)-x(N/2+1:end));
     Xdft(1:2:N) = difrecur(v);
     Xdft(2:2:N) = difrecur(z);
end
```

(c) MATLAB script:

```
% P0803: Testing DIF-FFT function 'difrecur'
close all; clc
x = [1,2,3,4,5,4,3,2];
Xdft = difrecur(x);
X_ref = fft(x);
```

4. MATLAB script:

```
% P0804: Investigate Decimation-in-time procedure
close all; clc
x = [1 2 3 4 5 4 3 2];
N = length(x);
%% Part (a):
a = x(1:2:N);
A = fft(a);
%% Part (b):
b = x(2:2:N);
B = fft(b);
%% Part (c):
W = exp(-j*2*pi/N).^(0:N/2-1);
temp = W.*B;
Xdft = zeros(1,N);
Xdft(1:N/2) = A + temp;
```

```
Xdft(N/2+1:N) = A - temp;
%% Part (d):
X_ref = fft(x);
```

5. (a) Solution:
   Since, $q = 1$, we have

$$W_N^{q\ell} = W_N^{\ell}, \qquad 0 \le \ell \le 8$$

   The number of complex multiplications is:

$$1 + 2 + \cdots + 7 = 28$$

   (b) Solution:
   Using the recursion formula, the number of complex multiplications is 7.

6. Proof:
   The two equations are repeated as follows:

$$X[2k] = \sum_{n=0}^{N/2-1} \left(x[n] + x[n+\frac{N}{2}] \right) W_{\frac{N}{2}}^{kn}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \quad (8.36)$$

$$X[2k+1] = \sum_{n=0}^{N/2-1} \left(x[n] - x[n+\frac{N}{2}] \right) W_N^n W_{\frac{N}{2}}^{kn}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \quad (8.37)$$

   Equation (8.37) can be derived as:

$$\begin{aligned}
X[2k+1] &= \sum_{n=0}^{N-1} x[n] W_N^{(2k+1)n} \\
&= \sum_{n=0}^{\frac{N}{2}-1} x[n] W_N^{(2k+1)n} + \sum_{n=0}^{\frac{N}{2}-1} x[n+\frac{N}{2}] W_N^{(2k+1)(n+\frac{N}{2})} \\
&= \sum_{n=0}^{\frac{N}{2}-1} x[n] W_N^n W_N^{2kn} + \sum_{n=0}^{\frac{N}{2}-1} x[n+\frac{N}{2}](-W_N^n) W_N^{2kn} \\
&= \sum_{n=0}^{N/2-1} \left(x[n] - x[n+\frac{N}{2}] \right) W_N^n W_{\frac{N}{2}}^{kn}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1
\end{aligned}$$

7. (a) Solution:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \qquad k = 0, 1, \ldots, N-1 \tag{8.1}$$

$$\begin{aligned} X[k] &= \sum_{n=0}^{N/2-1} x[n] W_N^{kn} + \sum_{n=0}^{N/2-1} x[n+\frac{N}{2}] W_N^{k(n+\frac{N}{2})} \\ &= \sum_{n=0}^{N/2-1} \left(x[n] + x[n+\frac{N}{2}] W_2^k \right) W_N^{kn} \end{aligned}$$

(b) Solution:
If $k = 2m,\ m = 0, 1, \ldots, \frac{N}{2} - 1$, we have

$$X[k] = X[2m] = \sum_{n=0}^{N/2-1} \left(x[n] + x[n+\frac{N}{2}] \right) W_{\frac{N}{2}}^{mn}$$

If $k = 2m+1,\ m = 0, 1, \ldots, \frac{N}{2} - 1$, we have

$$X[k] = X[2m+1] = \sum_{n=0}^{N/2-1} \left(x[n] - x[n+\frac{N}{2}] \right) W_N^n W_{\frac{N}{2}}^{mn}$$

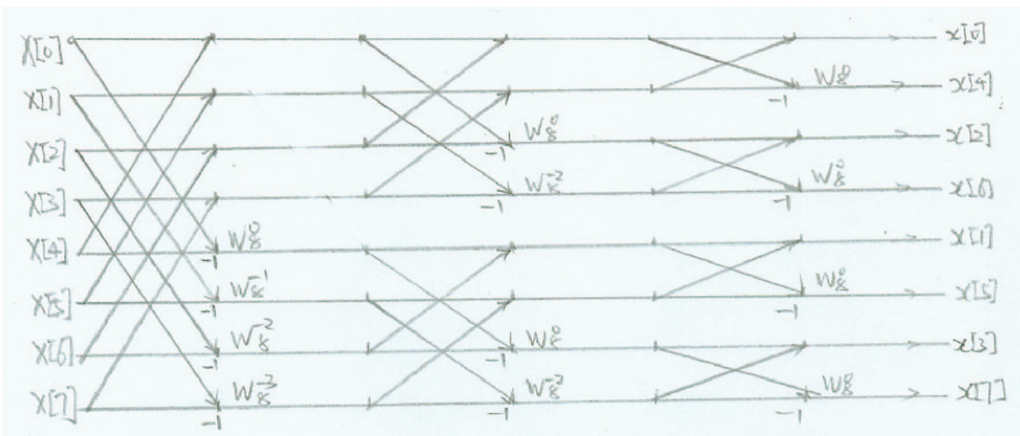
(c) Solution:
The above equations are exactly the same to the DIF FFT algorithm described in the context if we replace the variable m by k.

8. MATLAB function:

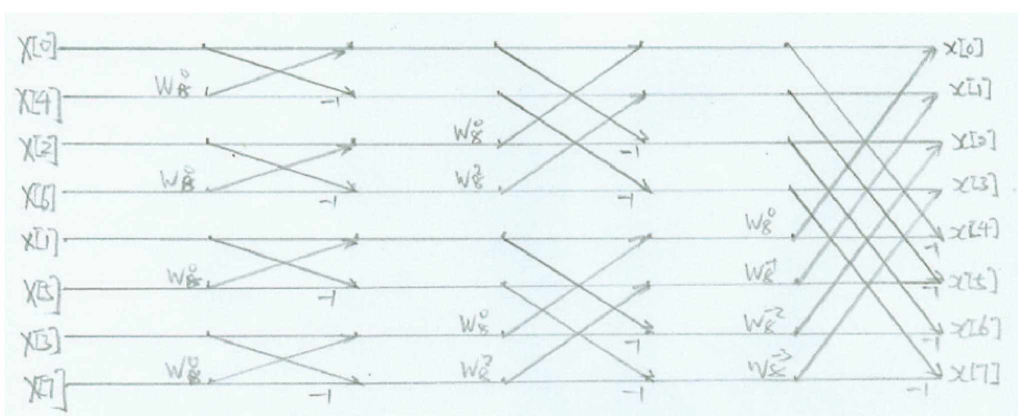
```
function x=fftdifr2(x)
% DIF Radix-2 FFT Algorithm
N=length(x); nu=log2(N);
for m=nu:-1:1;
    L=2^m;
    L2=L/2;
    for ir=1:L2;
        W=exp(-i*2*pi*(ir-1)/L);
        for it=ir:L:N;
            ib=it+L2;
            temp=x(it)+x(ib);
```

```
            x(ib)=x(it)-x(ib);
            x(ib)=x(ib)*W;
            x(it)=temp;
        end
    end
end
x = bitrevorder(x);
```

9. (a) See graph below.



(b) See graph below.



10. (a) Solution:

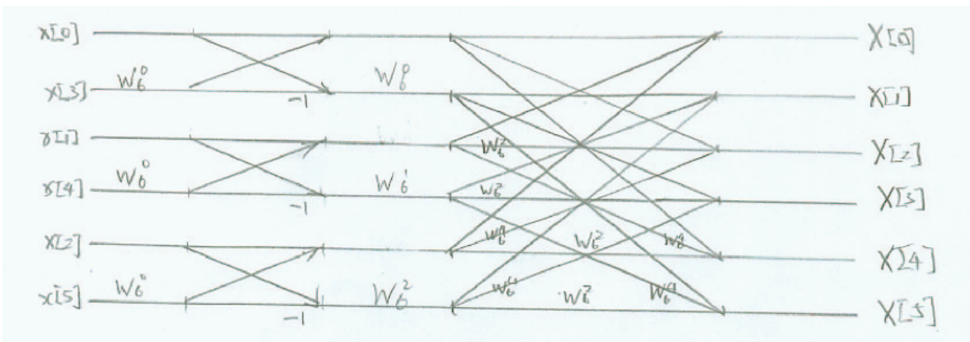
The number of complex multiplications is:

$$3 + 2 \times 6 = 15$$

The number of complex addition is:

$$6 + 2 \times 6 = 18$$

Hence, the number of real multiplication is 60 and the number of real addition is 54.



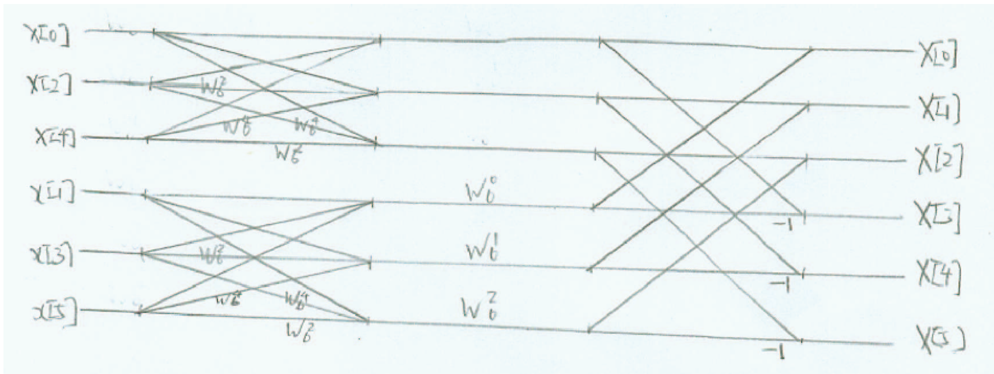
(b) Solution:
The number of complex multiplications is:

$$2 \times 2 \times 3 + 3 = 15$$

The number of complex addition is:

$$2 \times 6 + 6 = 18$$

Hence, the number of real multiplication is 60 and the number of real addition is 54.



11. (a) Proof:

$$X[2k] = \sum_{n=0}^{N/2-1} \left(x[n] + x[n + \frac{N}{2}] \right) W_{\frac{N}{2}}^{kn}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{8.36}$$

(b) Proof:

$$X[2k+1] = \sum_{n=0}^{N/2-1} \left(x[n] - x[n + \frac{N}{2}] \right) W_N^n W_{\frac{N}{2}}^{kn}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{8.37}$$

12. (a) Solution:

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{14} x[n]W_{15}^{kn} = \sum_{m=0}^{2} x[5m]W_{15}^{5mk} + \sum_{m=0}^{2} x[5m+1]W_{15}^{(5m+1)k} \\
&= \sum_{m=0}^{2} x[5m+2]W_{15}^{(5m+2)k} + \sum_{m=0}^{2} x[5m+3]W_{15}^{(5m+3)k} \\
&\quad + \sum_{m=0}^{2} x[5m+4]W_{15}^{(5m+4)k} \\
&= \left(\sum_{m=0}^{2} x[5m]W_{3}^{km}\right) + \left(\sum_{m=0}^{2} x[5m+1]W_{3}^{km}\right) W_{15}^{k} \\
&\quad + \left(\sum_{m=0}^{2} x[5m+2]W_{3}^{km}\right) W_{15}^{2k} + \left(\sum_{m=0}^{2} x[5m+3]W_{3}^{km}\right) W_{15}^{3k} \\
&\quad + \left(\sum_{m=0}^{2} x[5m+4]W_{3}^{km}\right) W_{15}^{4k}
\end{aligned}
$$

If we define that

$$
\begin{cases}
A[k] = \sum_{m=0}^{2} x[5m]W_3^{km}, \\
B[k] = \sum_{m=0}^{2} x[5m+1]W_3^{km}, \\
C[k] = \sum_{m=0}^{2} x[5m+2]W_3^{km}, \quad k = 0,1,2 \\
D[k] = \sum_{m=0}^{2} x[5m+3]W_3^{km}, \\
E[k] = \sum_{m=0}^{2} x[5m+4]W_3^{km}.
\end{cases}
$$

We have

$$
\begin{aligned}
X[k] &= A[k] + B[k]W_{15}^{k} + C[k]W_{15}^{2k} + D[k]W_{15}^{3k} + E[k]W_{15}^{4k} \\
X[k+3] &= A[k] + B[k]W_{15}^{k}W_{15}^{3} + C[k]W_{15}^{2k}W_{15}^{6} + D[k]W_{15}^{3k}W_{15}^{9} + E[k]W_{15}^{4k}W_{15}^{12} \\
X[k+6] &= A[k] + B[k]W_{15}^{k}W_{15}^{6} + C[k]W_{15}^{2k}W_{15}^{12} + D[k]W_{15}^{3k}W_{15}^{3} + E[k]W_{15}^{4k}W_{15}^{9} \\
X[k+9] &= A[k] + B[k]W_{15}^{k}W_{15}^{9} + C[k]W_{15}^{2k}W_{15}^{3} + D[k]W_{15}^{3k}W_{15}^{12} + E[k]W_{15}^{4k}W_{15}^{6} \\
X[k+12] &= A[k] + B[k]W_{15}^{k}W_{15}^{12} + C[k]W_{15}^{2k}W_{15}^{9} + D[k]W_{15}^{3k}W_{15}^{6} + E[k]W_{15}^{4k}W_{15}^{3}
\end{aligned}
$$

(b) Solution:

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{14} x[n]W_{15}^{nk} \\
&= \sum_{m=0}^{4} x[3m]W_{15}^{(3m)k} + \sum_{m=0}^{4} x[3m+1]W_{15}^{(3m+1)k} + \sum_{m=0}^{4} x[3m+2]W_{15}^{(3m+2)k} \\
&= \left(\sum_{m=0}^{4} x[3m]W_5^{km}\right) + \left(\sum_{m=0}^{4} x[3m+1]W_5^{km}\right) W_{15}^{k} \\
&\quad + \left(\sum_{m=0}^{4} x[3m+2]W_5^{km}\right) W_{15}^{2k}
\end{aligned}
$$

If we define that

$$
\begin{cases}
A[k] = \sum_{m=0}^{4} x[3m]W_5^{km}, \\
B[k] = \sum_{m=0}^{4} x[3m+1]W_5^{km}, \quad k = 0, 1, 2, 3, 4 \\
C[k] = \sum_{m=0}^{4} x[3m+2]W_5^{km}.
\end{cases}
$$

We conclude

$$
\begin{aligned}
X[k] &= A[k] + B[k]W_{15}^{k} + C[k]W_{15}^{2k} \\
X[k+5] &= A[k] + B[k]W_{15}^{k}W_{15}^{5} + C[k]W_{15}^{2k}W_{15}^{10} \\
X[k+10] &= A[k] + B[k]W_{15}^{k}W_{15}^{10} + C[k]W_{15}^{2k}W_{15}^{5}
\end{aligned}
$$

(c) Solution:

For part (a), the number of complex multiplication is:

$$5 \times 2 \times 3 + 4 \times 15 = 90$$

The number of complex addition is:

$$2 \times 15 + 4 \times 15 = 90$$

For part (b), the number of complex multiplication is:

$$3 \times 4 \times 5 + 2 \times 15 = 90$$

The number of complex addition is:

$$4 \times 15 + 2 \times 15 = 90$$

13. (a) Solution:

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{15} x[n]W_{16}^{kn} = \sum_{m=0}^{3} x[4m]W_{16}^{k(4m)} + \sum_{m=0}^{3} x[4m+1]W_{16}^{k(4m+1)} \\
&\quad + \sum_{m=0}^{3} x[4m+2]W_{16}^{k(4m+2)} + \sum_{m=0}^{3} x[4m+3]W_{16}^{k(4m+3)} \\
&= \left(\sum_{m=0}^{4} x[4m]W_4^{km}\right) + \left(\sum_{m=0}^{4} x[4m+1]W_4^{km}\right) W_{16}^{k} \\
&\quad + \left(\sum_{m=0}^{4} x[4m+2]W_4^{km}\right) W_{16}^{2k} + \left(\sum_{m=0}^{4} x[4m+3]W_4^{km}\right) W_{16}^{3k}
\end{aligned}
$$

If we define that

$$
\begin{cases}
A[k] = \sum_{m=0}^{3} x[4m]W_4^{km}, & k = 0,1,2,3 \\
B[k] = \sum_{m=0}^{3} x[4m+1]W_4^{km}, & k = 0,1,2,3 \\
C[k] = \sum_{m=0}^{3} x[4m+2]W_4^{km}, & k = 0,1,2,3 \\
D[k] = \sum_{m=0}^{3} x[4m+3]W_4^{km}, & k = 0,1,2,3
\end{cases}
$$

We conclude that

$$
\begin{aligned}
X[k] &= A[k] + B[k]W_{16}^{k} + C[k]W_{16}^{2k} + D[k]W_{16}^{3k}, \quad k = 0,1,2,3 \\
X[k+4] &= A[k] + B[k]W_{16}^{k}W_{16}^{4} + C[k]W_{16}^{2k}W_{16}^{8} + D[k]W_{16}^{3k}W_{16}^{12}, \quad k = 0,1,2,3 \\
X[k+8] &= A[k] + B[k]W_{16}^{k}W_{16}^{8} + C[k]W_{16}^{2k}W_{16}^{0} + D[k]W_{16}^{3k}W_{16}^{8}, \quad k = 0,1,2,3 \\
X[k+12] &= A[k] + B[k]W_{16}^{k}W_{16}^{12} + C[k]W_{16}^{2k}W_{16}^{8} + D[k]W_{16}^{3k}W_{16}^{4}, \quad k = 0,1,2,3
\end{aligned}
$$

(b) Solution:

The total number of complex multiplications to implement the radix-4 FFT is:

$$2 \times 16 + 2 \times 16 = 64$$

The total number of complex additions to implement the radix-4 FFT is:

$$3 \times 16 + 3 \times 16 = 96$$

(c) Solution:

The number of complex multiplications to implement the radix-2 FFT is:

$$4 \times 8 = 32$$

which is two times the number of complex multiplications in radix-4 FFT.
Since, in the radix-4 algorithm, each complex multiplication only requires two real multiplication while in general the complex multiplication in radix-2 requires four real multiplications, the number of multiplications are reduced by half.

14. (a) Proof:

$$X[n] = X(\mathrm{e}^{\mathrm{j}\omega_n}) \triangleq \sum_{k=0}^{N-1} g[k]W^{nk} \tag{8.67}$$

$$g[n] \triangleq x[n]\mathrm{e}^{-\mathrm{j}\omega_L n}, \quad \text{and} \quad W = \mathrm{e}^{-\mathrm{j}\delta\omega} \tag{8.68}$$

$$\begin{cases} \mathrm{e}^{\mathrm{j}\omega_L} \to R\mathrm{e}^{\mathrm{j}\omega_L} & \Longrightarrow \quad g[n] = x[n]\left(\frac{1}{R}\mathrm{e}^{-\mathrm{j}\omega_L}\right)^n \\ \mathrm{e}^{\mathrm{j}\delta\omega} \to r\mathrm{e}^{\mathrm{j}\delta\omega} & \Longrightarrow \quad W = \frac{1}{r}\mathrm{e}^{-\mathrm{j}\delta\omega} \end{cases} \tag{8.70}$$

$$z_n = \left(R\mathrm{e}^{\mathrm{j}\omega_L}\right)\left(r\mathrm{e}^{\mathrm{j}\delta\omega}\right)^n, \quad 0 \le n \le M \tag{8.71}$$

$$X(z_n) = \left\{\left(g[n]W^{n^2/2}\right) * W^{-n^2/2}\right\} W^{n^2/2} \tag{8.72}$$

$$\begin{aligned}
X[z_n] &= \sum_{k=0}^{N-1} x[k](z_n)^{-k} = \sum_{k=0}^{N-1} x[k]\left[(R\mathrm{e}^{\mathrm{j}\omega_L})(r\mathrm{e}^{\mathrm{j}\delta\omega})^n\right]^{-k} \\
&= \sum_{k=0}^{N-1}\left[x[k](R\mathrm{e}^{\mathrm{j}\omega_L})^{-k}\right]\left[(r\mathrm{e}^{\mathrm{j}\delta\omega})^{-1}\right]^{nk} \\
&= \sum_{k=0}^{N-1}\left[x[k](R\mathrm{e}^{\mathrm{j}\omega_L})^{-k}\right]\left[(r\mathrm{e}^{\mathrm{j}\delta\omega})^{-1}\right]^{\frac{k^2}{2}}\left[(r\mathrm{e}^{\mathrm{j}\delta\omega})^{-1}\right]^{-\frac{(n-k)^2}{2}}\left[(r\mathrm{e}^{\mathrm{j}\delta\omega})^{-1}\right]^{\frac{n^2}{2}} \\
&= \left[\sum_{k=0}^{N-1}\left(g[k]W^{\frac{k^2}{2}}\right)W^{-\frac{(n-k)^2}{2}}\right]W^{\frac{n^2}{2}} \\
&= \left\{\left(g[n]W^{n^2/2}\right) * W^{-n^2/2}\right\}W^{n^2/2}
\end{aligned}$$

(b) MATLAB function:

```
function [X,w] = czta(x,M,wL,wH,R,r)
% Chirp z-Transform Algorithm (CTA)
%   Given x[n] CZTA computes M z-transform values
%   on the spiral line over wL <= w <= wH
% [X,w] = czta(x,M,wL,wH,R,r)
```

```
Dw = wH-wL; dw = Dw/(M-1); W = exp(-1j*dw)/r;
N = length(x); nx = 0:N-1;
K = max(M,N); n = 0:K; Wn2 = W.^(n.*n/2);
g = x.*R.^(-nx).*exp(-1j*wL*nx).*Wn2(1:N);
nh = -(N-1):M-1; h = W.^(-nh.*nh/2);
y = conv(g,h);
X = y(N:N+M-1).*Wn2(1:M); w = wL:dw:wH;
```

15. (a) Proof:

$$X_n[k] = \sum_{m=0}^{N-1} x_n[m] W_N^{mk} = \sum_{m=0}^{N-1} x_n[n-N+1+m] W_N^{mk}, \quad \begin{cases} n \geq N-1, \\ 0 \leq k \leq N-1 \end{cases} \tag{8.85}$$

$$X_n[k] = \{X_{n-1}[k] + x[n] - x[n-N]\} W_N^{-k} W_N^{-k}, \quad \begin{cases} n \geq N-1, \\ 0 \leq k \leq N-1 \end{cases} \tag{8.86}$$

$$\begin{aligned} X_{n-1}[k] &= \sum_{m=0}^{N-1} x_{n-1}[m] W_N^{mk} = \sum_{m=0}^{N-1} x[n-1-N+1+m] W_N^{mk} \\ &= \sum_{m=0}^{N-1} x[n-N+m] W_N^{mk} = x[n-N] + \sum_{m=1}^{N} x[n-N+m] W_N^{mk} - x[n] \end{aligned}$$

Hence, we can conclude that

$$\begin{aligned} &\sum_{m=1}^{N} x[n-N+m] W_N^{mk} W_N^{-k} = \sum_{m=1}^{N} x[n-N+m] W_N^{(m-1)k} \\ &= \sum_{m=0}^{N-1} x[n-N+m+1] W_N^{mk} = X_n[k] \end{aligned}$$

(b) Solution:

$$x_n[k] = \sum_{m=0}^{N-1} w_{\mathrm{e}}[m] x[n-N+1+m] W_N^{mk} = \sum_{m=0}^{N-1} \lambda^{N-1-m} x[n-N+1+m] W_N^{mk}$$

$$x_{n-1}[k] = \sum_{m=0}^{N-1} \lambda^{N-1-m} x[n-N+m] W_N^{mk}$$

We can summarize a recursive SDFT algorithm, that is

$$X_n[k] = \left\{\lambda^{-1} X_{n-1}[k] + x[n] - \lambda^{N-2} x[n-N]\right\} W_N^{-k}$$

Basic Problems

16. (a) Proof:

$$x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X[k]W_N^{-kn}, \quad n = 0, 1, \ldots, N-1 \qquad (8.2)$$

$$\begin{aligned} x[n] &= \frac{1}{N}\sum_{n=0}^{N-1} X[k]W_N^{-kn} = \frac{1}{N}\mathrm{j}\left(\sum_{n=0}^{N-1}(-\mathrm{j}X[k])W_N^{-kn}\right) \\ &= \frac{1}{N}\mathrm{j}\left\{\sum_{k=0}^{N-1}(\mathrm{j}X^*[k])W_N^{kn}\right\}^* \end{aligned}$$

(b) tba.

(c) MATLAB function:

```
function x = idft_0816(X,N)
% Compute idft using fft function according to (8.90)
X = X(:).';
Nx = length(X);
if nargin == 1
    N = Nx;
elseif N <= Nx
    X = X(1:N);
else
    X = [X zeros(1,N-Nx)];
end
x = fft(conj(X)*j);
x = conj(x)*j/N;
```

17. Solution:
Direct computation:

$$(a + \mathrm{j}b)(c + \mathrm{j}d) = ac + \mathrm{j}bc + \mathrm{j}ad - bd = (ac - bd) + \mathrm{j}(bc + ad)$$

which contains four real multiplications and two real additions.
If we define

$$\begin{aligned} k_1 &= c \cdot (a + b) \\ k_2 &= a \cdot (d - c) \\ k_3 &= b \cdot (c + d) \end{aligned}$$

Hence, we can conclude that

$$ac - bd = k_1 - k_3, \qquad bc + ad = k_1 + k_2$$

which contains 3 real multiplications and 5 real additions.

18. Solution:
The resulting trend in the computational complexity of recursive DFT computations is much more efficient and close to linear than direct computation.

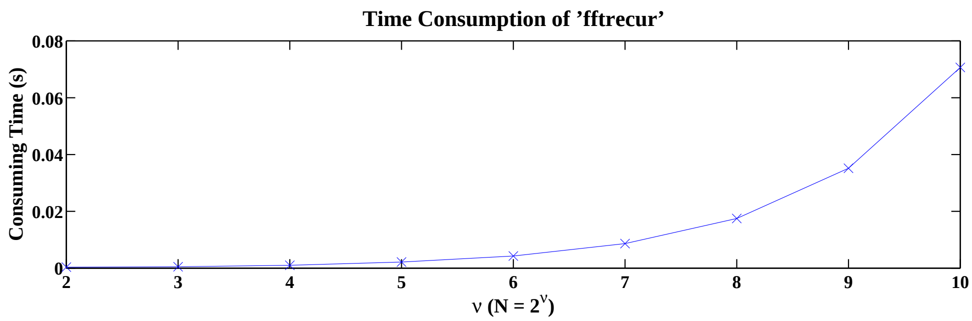


FIGURE 8.2: Plot of computation time for the `fftrecur` function for $N = 2^{\nu}$ where $2 \leq \nu \leq 10$.

MATLAB script:

```
% P0818: Investigate time consumption using fftrecur
close all; clc
nu = 2:10;
N = 2.^nu;
Ni = length(N);
t = zeros(1,Ni);
for ii = 1:Ni
    x = randn(1,N(ii)) + j*randn(1,N(ii));
    tic
    X = fftrecur(x);
    t(ii) = toc;
end
% Plot:
hfa = figconfg('P0818a','long');
plot(nu,t,'x-','markersize',12)
xlabel('\nu (N = 2^{\nu})','fontsize',LFS)
```

```
ylabel('Consuming Time (s)','fontsize',LFS)
title('Time Consumption of ''fftrecur''','fontsize',TFS)
```

19. (a) Solution:

$$\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{nk} = \sum_{m=0}^{\frac{N}{3}-1} x[3m]W_N^{k(3m)} \\
&\quad + \sum_{m=0}^{\frac{N}{3}-1} x[3m+1]W_N^{k(3m+1)} + \sum_{m=0}^{\frac{N}{3}-1} x[3m+2]W_N^{k(3m+2)} \\
&= \left(\sum_{m=0}^{\frac{N}{3}-1} x[3m]W_{\frac{N}{3}}^{km}\right) + \left(\sum_{m=0}^{\frac{N}{3}-1} x[3m+1]W_{\frac{N}{3}}^{km}\right) W_N^k \\
&\quad + \left(\sum_{m=0}^{\frac{N}{3}-1} x[3m+2]W_{\frac{N}{3}}^{km}\right) W_N^{2k}
\end{aligned}$$

If we define the following,

$$\begin{cases}
A[k] = \sum_{m=0}^{\frac{N}{3}-1} x[3m]W_{\frac{N}{3}}^{km}, & k = 0, 1, \ldots, \frac{N}{3}-1 \\
B[k] = \sum_{m=0}^{\frac{N}{3}-1} x[3m+1]W_{\frac{N}{3}}^{km}, & k = 0, 1, \ldots, \frac{N}{3}-1 \\
C[k] = \sum_{m=0}^{\frac{N}{3}-1} x[3m+2]W_{\frac{N}{3}}^{km}, & k = 0, 1, \ldots, \frac{N}{3}-1
\end{cases}$$

We conclude that:

$$\begin{aligned}
X[k] &= A[k] + B[k]W_N^k + C[k]W_N^{2k} \\
X[k+N/3] &= A[k] + B[k]W_N^k W_N^{\frac{N}{3}} + C[k]W_N^{2k}W_N^{\frac{2N}{3}} \\
X[k+2N/3] &= A[k] + B[k]W_N^k W_N^{\frac{2N}{3}} + C[k]W_N^{2k}W_N^{\frac{N}{3}}
\end{aligned}$$

(b) tba

(c) Solution:
The total number of complex multiplications needed to implement is:

$$2 \times 27 \times 3 = 162$$

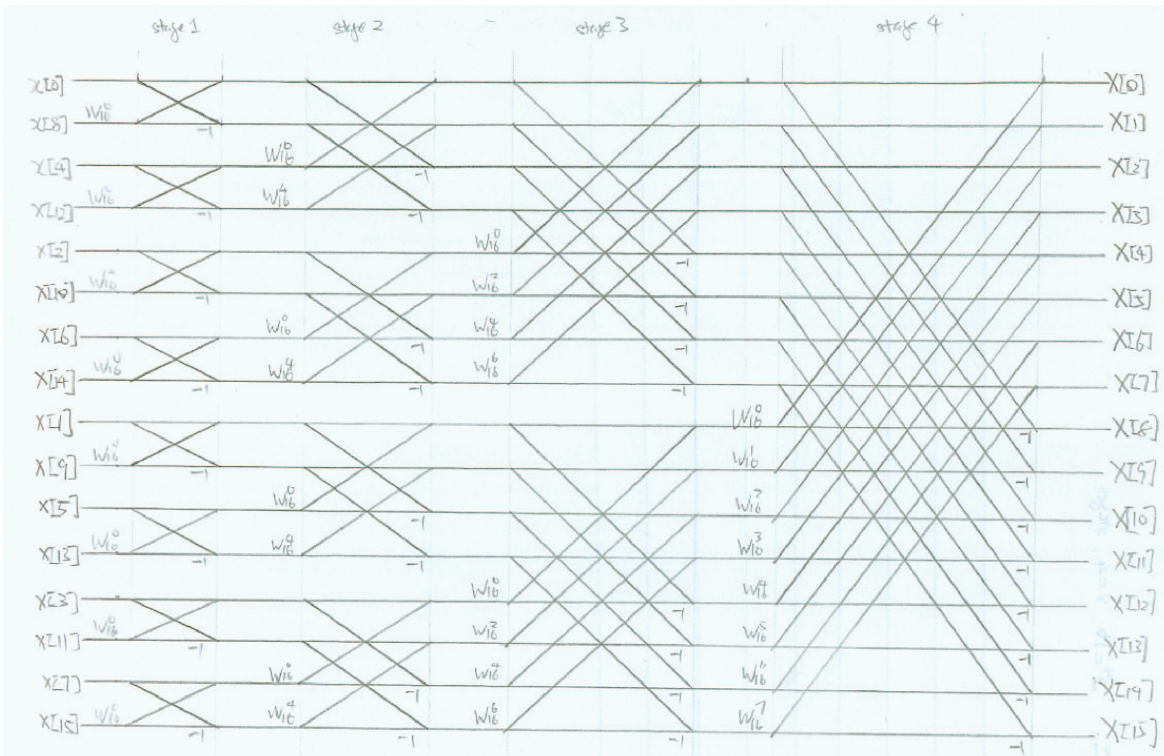
20. Solution:
The total number of complex multiplications is:

$$8 \times 4 = 32$$

The total number of complex additions is:

$$16 \times 4 = 64$$

Hence, the total number of real multiplications is 128 and the total number of real additions is 192.



21. (a) Proof:

$$
\begin{aligned}
y[Ln] &= \frac{1}{LN}\sum_{k=0}^{LN-1} Y[k]W_{LN}^{-k(LN)} \\
&= \frac{1}{LN}\left(\sum_{k=0}^{k_0-1} X[k]W_{LN}^{-k(Ln)} + \sum_{k=LN-k_0+1}^{LN-1} X[k+N-LN]W_{LN}^{-k(Ln)}\right) \\
&= \frac{1}{LN}\left(\sum_{k=0}^{k_0-1} X[k]W_{N}^{-kn} + \sum_{k=LN-k_0+1}^{LN-1} X[k+N-LN]W_{N}^{-kn}\right) \\
&= \frac{1}{LN}\left(\sum_{k=0}^{k_0-1} X[k]W_{N}^{-kn} + \sum_{k=N-k_0+1}^{N-1} X[k]W_{N}^{-kn}\right) \\
&= \frac{1}{L}\left(\frac{1}{N}\sum_{k=0}^{N-1} X[k]W_{N}^{-kn}\right) \\
&= \frac{1}{L}x[n]
\end{aligned}
$$

(b) Solution:

$$y[k] = [2, 0, 0, 0, 0, 0, 0, 2]$$

$$
\begin{aligned}
y[n] &= \frac{1}{8}\sum_{k=0}^{7} Y[k]W_8^{nk} = \frac{1}{8}(2W_8^0 + 2W_8^{7n}) \\
&= \frac{1}{4}(1 + W_8^{7n})
\end{aligned}
$$

22. Solution:

$$
\begin{cases}
a[n] = x[2n], & n = 0, 1, \dots, \frac{N}{2} - 1 \\
b[n] = x[2n+1], & n = 0, 1, \dots, \frac{N}{2} - 1
\end{cases}
\tag{8.19}
$$

$$
\begin{cases}
X[k] = A[k] + W_N^k B[k], & n = 0, 1, \dots, \frac{N}{2} - 1 \\
X[k+N/2] = A[k] - W_N^k B[k], & n = 0, 1, \dots, \frac{N}{2} - 1
\end{cases}
\tag{8.23}
$$

If we mistakenly assign $a[n] = x[2n+1]$ and $b[n] = x[2n]$, we can recover the DFT $X[k]$ as:

$$
\begin{aligned}
X'[k] &= \frac{X[k] - X[k+\frac{N}{2}]}{2}W_N^{-k} + \frac{X[k] + X[k+\frac{N}{2}]}{2}W_N^{k} \\
X'[k+\frac{N}{2}] &= \frac{X[k] - X[k+\frac{N}{2}]}{2}W_N^{-k} - \frac{X[k] + X[k+\frac{N}{2}]}{2}W_N^{k}
\end{aligned}
$$

23. Solution:
It is a DIT approach.

24. (a) Solution:
There is only one path in the flow-graph begin at the input node $x[1]$ and terminate on the output node $X[2]$.
There is only one path in the flow-graph begin at the input node $x[1]$ and terminate on the output node $x[4]$ to $X[7]$.
The conclusion is that there is only one path from every input node to every output node.

(b) Solution:
The total gain from input node $x[1]$ to output node $X[2]$ is W_8^2.
The total gain from input node $x[4]$ to output node $X[7]$ is $W_8^{28} = -1$.

(c) Solution:
$X[4] = \sum_{n=0}^{7} x[n]W_8^{4n}$ can be verified.

25. Solution:

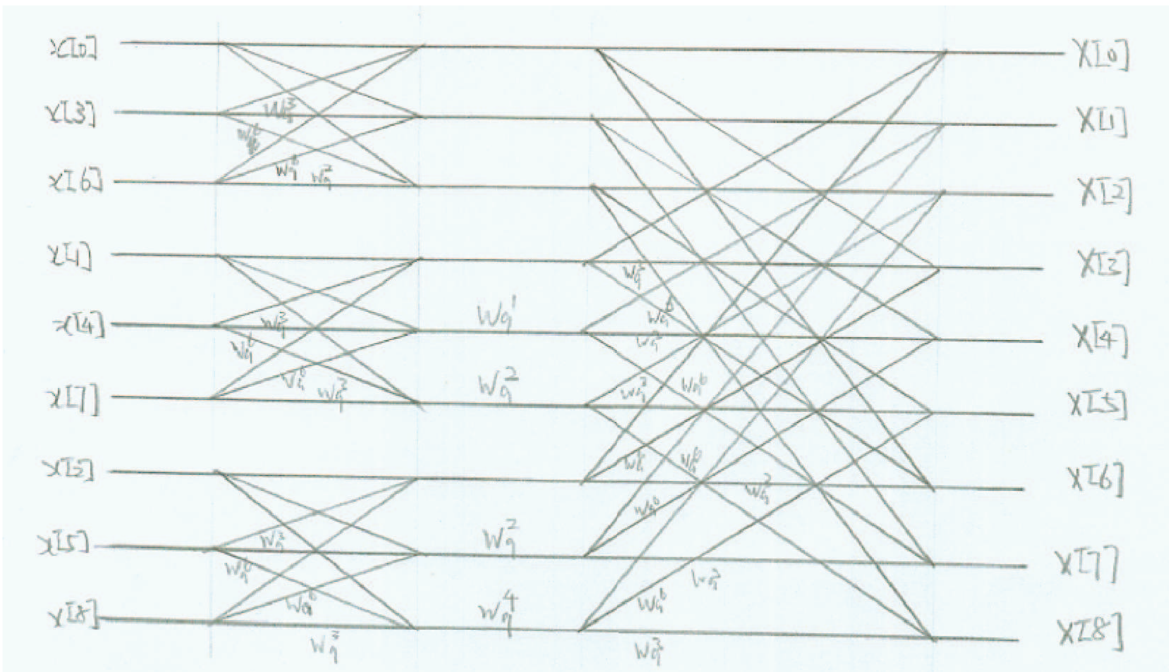
$$\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{nk} = \sum_{m=0}^{\frac{N}{3}-1} x[3m]W_N^{k(3m)} \\
&\quad + \sum_{m=0}^{\frac{N}{3}-1} x[3m+1]W_N^{k(3m+1)} + \sum_{m=0}^{\frac{N}{3}-1} x[3m+2]W_N^{k(3m+2)} \\
&= \left(\sum_{m=0}^{\frac{N}{3}-1} x[3m]W_{\frac{N}{3}}^{km} \right) + \left(\sum_{m=0}^{\frac{N}{3}-1} x[3m+1]W_{\frac{N}{3}}^{km} \right) W_N^k \\
&\quad + \left(\sum_{m=0}^{\frac{N}{3}-1} x[3m+2]W_{\frac{N}{3}}^{km} \right) W_N^{2k}
\end{aligned}$$

If we define the following,

$$\begin{cases}
A[k] = \sum_{m=0}^{\frac{N}{3}-1} x[3m]W_{\frac{N}{3}}^{km}, & k = 0, 1, \ldots, \frac{N}{3} - 1 \\
B[k] = \sum_{m=0}^{\frac{N}{3}-1} x[3m+1]W_{\frac{N}{3}}^{km}, & k = 0, 1, \ldots, \frac{N}{3} - 1 \\
C[k] = \sum_{m=0}^{\frac{N}{3}-1} x[3m+2]W_{\frac{N}{3}}^{km}, & k = 0, 1, \ldots, \frac{N}{3} - 1
\end{cases}$$

We conclude that:

$$X[k] = A[k] + B[k]W_N^k + C[k]W_N^{2k}$$
$$X[k+N/3] = A[k] + B[k]W_N^k W_N^{\frac{N}{3}} + C[k]W_N^{2k}W_N^{\frac{2N}{3}}$$
$$X[k+2N/3] = A[k] + B[k]W_N^k W_N^{\frac{2N}{3}} + C[k]W_N^{2k}W_N^{\frac{N}{3}}$$



26. (a) Solution:

$$\begin{cases} s_1[k] = s_0[k] + s_0[k+4]W_8^0 \\ s_1[k+4] = s_0[k] - s_0[k+4]W_8^0 \end{cases} \qquad k = 0,1,2,3$$

$$\begin{cases} s_2[k] = s_1[k] + s_1[k+2]W_8^0, & k = 0,1 \\ s_2[k] = s_1[k+2] + s_1[k+4]W_8^2, & k = 2,3 \\ s_2[k+4] = s_1[k] - s_1[k+2]W_8^0, & k = 0,1 \\ s_2[k+4] = s_1[k+2] - s_1[k+4]W_8^2, & k = 2,3 \end{cases}$$

$$\begin{cases} s_3[k] = s_2[2k] + s_2[2k+1]W_8^k \\ s_3[k+4] = s_2[2k] - s_2[2k+1]W_8^k \end{cases} \qquad k = 0,1,2,3$$

(b) MATLAB function:

```
function X = fftalt8(x)
if length(x)~=8
    error('bad input, illegal length')
end
N = 8;
s = x;
w = exp(-j*2*pi/N).^(0:N-1);
% Stage I:
temp = s;
s(1:4) = temp(1:4)+temp(5:8);
s(5:8) = temp(1:4)-temp(5:8);
% Stage II:
temp = s;
s(1:2) = temp(1:2)+temp(3:4);
s(3:4) = temp(5:6)+temp(7:8)*w(3);
s(5:6) = temp(1:2)-temp(3:4);
s(7:8) = temp(5:6)-temp(7:8)*w(3);
% Stage III:
temp = s;
s(1:4) = temp(1:2:end)+temp(2:2:end).*w(1:4);
s(5:8) = temp(1:2:end)-temp(2:2:end).*w(1:4);

X = s;
```

(c) Solution:
The coding complexity of the above function is much larger than that of the `fftditr2` function since the equations are not recursive.

27. Proof:

$$\begin{cases} X[k] = A[k] + W_N^k B[k] \\ X[k+\frac{N}{2}] = A[k] - W_N^k B[k] \end{cases} \quad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{8.23}$$

$$X_{n_2}[k_1] = \sum_{n_1=0}^{N_1-1} x_{n_2}[n_1] W_{N_1}^{k_1 n_1} \tag{8.52}$$

$$x_{k_1}[n_2] \triangleq W_N^{k_1 n_2} X_{n_2}[k_1] \tag{8.54}$$

$$X[k_1 + N_1 k_2] = \sum_{n_2=0}^{N_2-1} x_{k_1}[n_2] W_{N_2}^{k_2 n_2} \tag{8.55}$$

If we have $N_1 = \frac{N}{2}$, and $N_2 = 2$, thus (8.55) can be written as

$$\begin{cases} X[k_1] = x_{k_1}[0] + x_{k_1}[1] \\ X[k_1 + \frac{N}{2}] = x_{k_1}[0] - x_{k_1}[1] \end{cases}$$

Then, (8.54) can be written as

$$\begin{cases} x_{k_1}[0] = X_0[k_1] \\ x_{k_1}[1] = W_N^{k_1} X_1[k_1] \end{cases} \qquad k_1 = 0, 1, \ldots, \frac{N}{2}$$

Hence, (8.52) can be written as

$$X_0[k_1] = \sum_{n_1=0}^{\frac{N}{2}-1} x_0[n_1] W_{\frac{N}{2}}^{k_1 n_1} = \sum_{n_1=0}^{\frac{N}{2}-1} x[2n_1] W_{\frac{N}{2}}^{k_1 n_1}$$
$$X_1[k_1] = \sum_{n_1=0}^{\frac{N}{2}-1} x_1[n_1] W_{\frac{N}{2}}^{k_1 n_1} = \sum_{n_1=0}^{\frac{N}{2}-1} x[2n_1+1] W_{\frac{N}{2}}^{k_1 n_1}$$

which is the same as the DIT-FFT algorithm of (8.23).

28. (a) Solution:

$$X[5k] = \sum_{n=0}^{2} \left(x[n] + x[n+3] + x[n+6] + x[n+9] + x[n+12]\right) W_3^{nk}$$
$$X[5k+1] = \sum_{n=0}^{2} \big(x[n] + x[n+3]W_{15}^{3} + x[n+6]W_{15}^{6} + x[n+9]W_{15}^{9} + x[n+12]W_{15}^{12}\big) W_{15}^{n} W_3^{nk}$$
$$X[5k+2] = \sum_{n=0}^{2} \big(x[n] + x[n+3]W_{15}^{6} + x[n+6]W_{15}^{12} + x[n+9]W_{15}^{3} + x[n+12]W_{15}^{9}\big) W_{15}^{2n} W_3^{nk}$$
$$X[5k+3] = \sum_{n=0}^{2} \big(x[n] + x[n+3]W_{15}^{9} + x[n+6]W_{15}^{3} + x[n+9]W_{15}^{12} + x[n+12]W_{15}^{6}\big) W_{15}^{3n} W_3^{nk}$$
$$X[5k+4] = \sum_{n=0}^{2} \big(x[n] + x[n+3]W_{15}^{12} + x[n+6]W_{15}^{9} + x[n+9]W_{15}^{6} + x[n+12]W_{15}^{3}\big) W_{15}^{4n} W_3^{nk}$$

If we define the following that

$$A[k] = \sum_{n=0}^{2} x[n]W_3^{nk}, \qquad B[k] = \sum_{n=0}^{2} x[n+3]W_3^{nk}$$
$$C[k] = \sum_{n=0}^{2} x[n+6]W_3^{nk}, \qquad D[k] = \sum_{n=0}^{2} x[n+9]W_3^{nk}$$
$$E[k] = \sum_{n=0}^{2} x[n+12]W_3^{nk}$$

Hence, we can conclude that

$$\begin{aligned}
X[5k] &= A[k] + B[k] + C[k] + D[k] + E[k] \\
X[5k+1] &= (A[k] + B[k]W_{15}^{3} + C[k]W_{15}^{6} + D[k]W_{15}^{9} + E[k]W_{15}^{12})W_{15}^{n} \\
X[5k+2] &= (A[k] + B[k]W_{15}^{6} + C[k]W_{15}^{12} + D[k]W_{15}^{3} + E[k]W_{15}^{9})W_{15}^{2n} \\
X[5k+3] &= (A[k] + B[k]W_{15}^{9} + C[k]W_{15}^{3} + D[k]W_{15}^{12} + E[k]W_{15}^{6})W_{15}^{3n} \\
X[5k+4] &= (A[k] + B[k]W_{15}^{12} + C[k]W_{15}^{9} + D[k]W_{15}^{6} + E[k]W_{15}^{3})W_{15}^{4n}
\end{aligned}$$

(b) Solution:

$$\begin{aligned}
X[3k] &= \sum_{n=0}^{4} \left(x[n] + x[n+5] + x[n+10]\right) W_5^{nk} \\
X[3k+1] &= \sum_{n=0}^{4} \left(x[n] + x[n+5]W_{15}^{5} + x[n+10]W_{15}^{10}\right) W_{15}^{n} W_5^{nk} \\
X[3k+2] &= \sum_{n=0}^{4} \left(x[n] + x[n+5]W_{15}^{10} + x[n+10]W_{15}^{5}\right) W_{15}^{2n} W_5^{nk}
\end{aligned}$$

If we define the following that

$$A[k] = \sum_{n=0}^{4} x[n]W_5^{nk}, \qquad B[k] = \sum_{n=0}^{4} x[n+5]W_5^{nk}$$
$$C[k] = \sum_{n=0}^{4} x[n+10]W_5^{nk},$$

Hence, we can conclude that

$$\begin{aligned}
X[3k] &= A[k] + B[k] + C[k] \\
X[3k+1] &= (A[k] + B[k]W_{15}^{5} + C[k]W_{15}^{10})W_{15}^{n} \\
X[3k+2] &= (A[k] + B[k]W_{15}^{10} + C[k]W_{15}^{5})W_{15}^{2n}
\end{aligned}$$

(c) Solution:
For part (a), the total number of complex multiplications is:

$$2 \times 15 + 4 \times 15 = 90$$

The total number of complex additions is:

$$2 \times 15 + 4 \times 15 = 90$$

For part (b), the total number of complex multiplications is:

$$4 \times 15 + 2 \times 15 = 90$$

The total number of complex additions is:

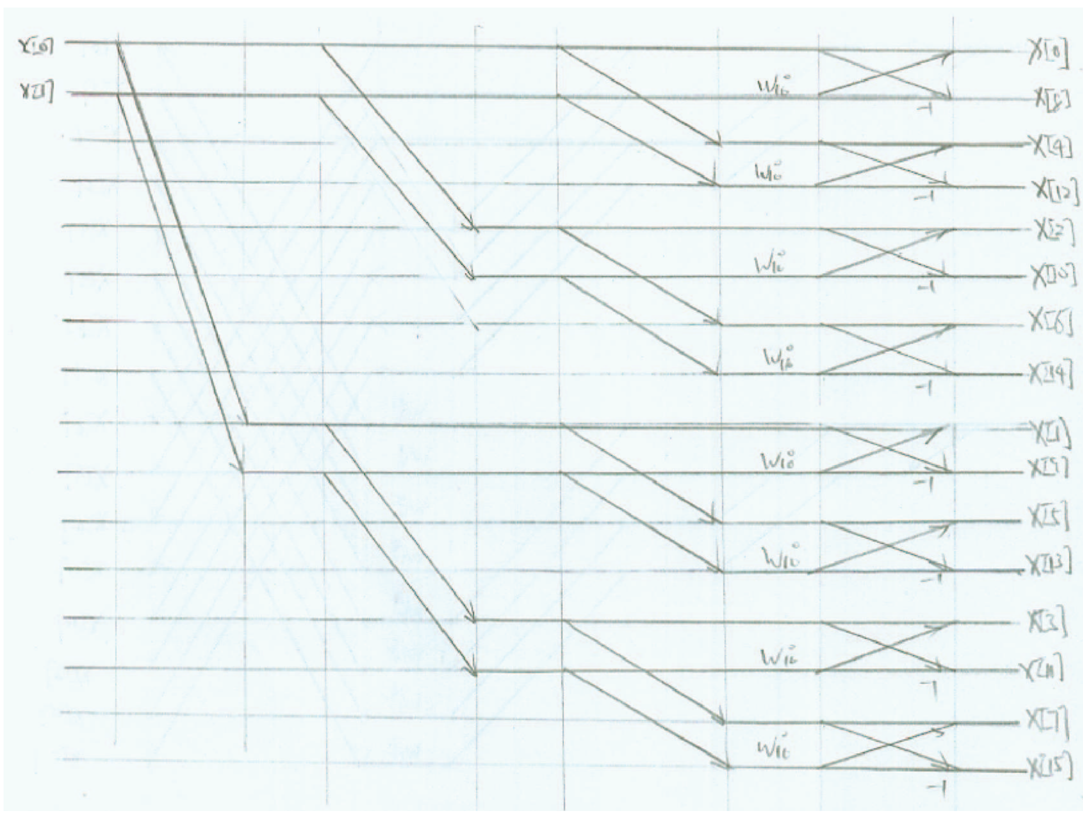
$$4 \times 15 + 2 \times 15 = 90$$

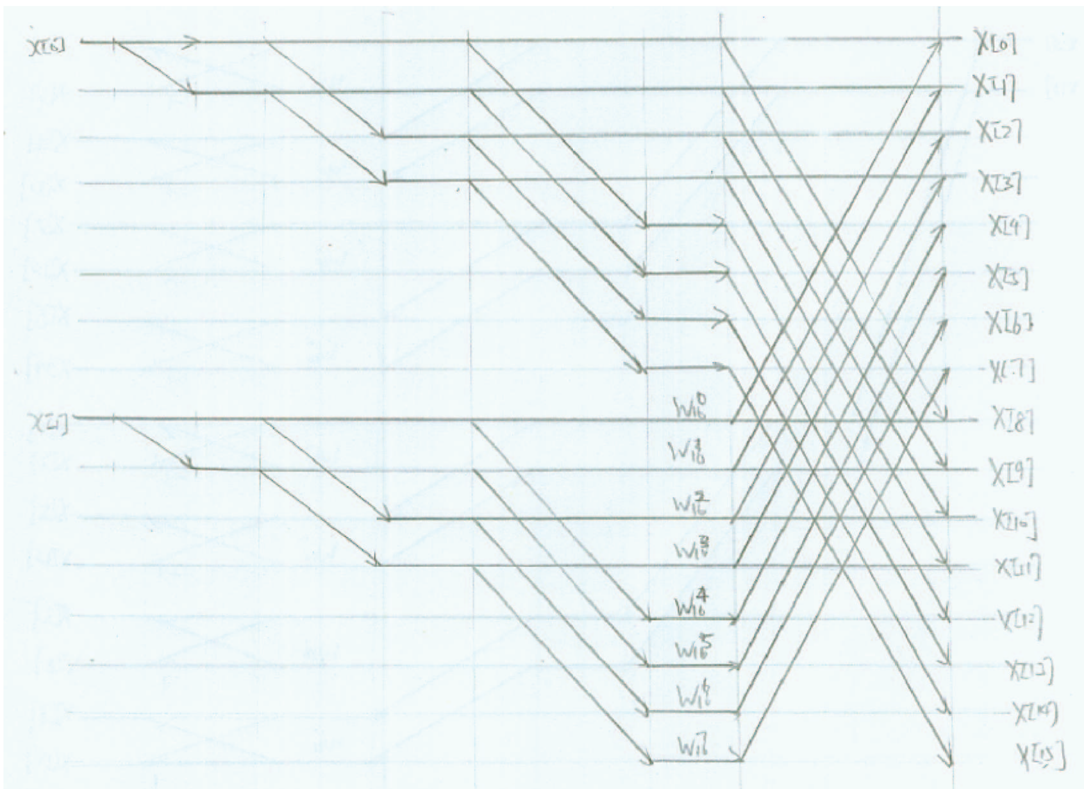
29. (a) See graph below.

(b) See graph below.

(c) Solution:
For part (a), the total number of complex multiplications is 0.
For part (b), the total number of complex multiplications is 7.

(d) tba


x[0]
x[1]
X[0]
X[8]
X[4]
X[12]
X[2]
X[10]
X[6]
X[14]
X[1]
X[9]
X[5]
X[13]
X[3]
X[11]
X[7]
X[15]
W_{16}^{0}
-1


X[0]
X[1]
X[2]
X[3]
X[4]
X[5]
X[6]
X[7]
X[8]
X[9]
X[10]
X[11]
X[12]
X[13]
X[14]
X[15]
W_{16}^0
W_{16}^1
W_{16}^2
W_{16}^3
W_{16}^4
W_{16}^5
W_{16}^6
W_{16}^7

30. (a) tba.

(b) MATLAB function:

```
function X = gafft_vec(x,N,k)
% Goertzel's algorithm
% X = gafft(x,N,k)
% Computes k-th sample of an N-point DFT X[k] of x[n]
% using Goertzel Algorithm
% k is a vector
L = length(x); x = [reshape(x,1,L),zeros(1,N-L+1)];
K = length(k); X = zeros(1,K);
for ii = 1:K
   v = filter(1,[1,-2*cos(2*pi*k(ii)/N),1],x);
   X(ii) = v(N+1)-exp(-1j*2*pi*k(ii)/N)*v(N);
end
```

(c) MATLAB script:

```
% P0830: Single Tone Detection
close all; clc
Fd = [490 1280 2730 3120];
Fs = 8e3;
N = 100;
k = round(Fd/Fs*N);
T = 1/Fs;
nT = 0:T:1;
xn = cos(2*pi*Fd'*nT);
Xd = zeros(length(Fd));
for ii = 1:length(Fd)
Xd(ii,:) = gafft_vec(xn(ii,:),N,k);
end
abs(Xd)
```

31. MATLAB function:

```
function [X,w] = ctafft(x,M,wL,wH)
% Chirp Transform Algorithm (CTA)
%   Given x[n] CTA computes M equispaced DTFT values X[k]
%   on the unit circle over wL <= w <= wH
% [X,w] = cta(x,M,wL,wH)
Dw = wH-wL; dw = Dw/(M-1); W = exp(-1j*dw);
N = length(x); nx = 0:N-1;
```

```
K = max(M,N); n = 0:K; Wn2 = W.^(n.*n/2);
g = x.*exp(-1j*wL*nx).*Wn2(1:N);
nh = -(N-1):M-1; h = W.^(-nh.*nh/2);
L = M + N;
G = fft(g,L);
H = fft(h,L);
Y = G.*H;
y = ifft(Y);
X = y(N:N+M-1).*Wn2(1:M); w = wL:dw:wH;
```

32. (a) See plot below.

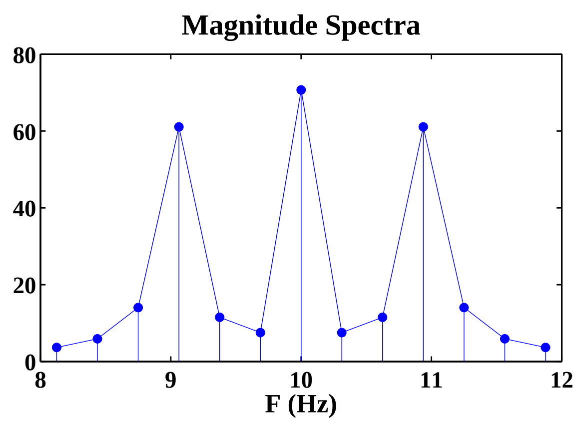


FIGURE 8.3: Magnitude spectra of 128-point FFT of $x[n]$ over $8 \leq F \leq 12$ Hz.

(b) See plot below.

(c) See plot below.

(d) Solution:
Using `cta` function has the smallest number of computations with a better display.

MATLAB script:

```
% P0832: Illustration of using cta algorithm
close all; clc
Fs = 40;
n = 0:127;
T = 1/Fs;
nT = n*T;
```

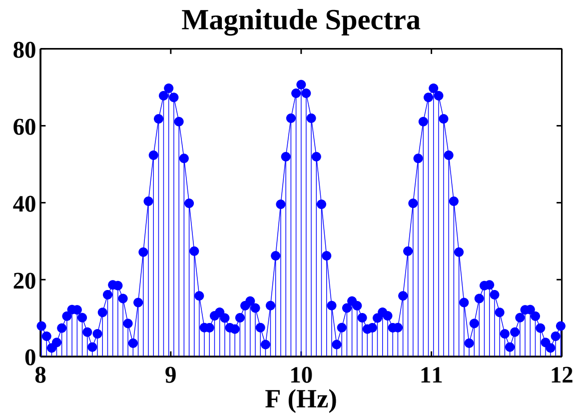


FIGURE 8.4: Magnitude spectra of 1024-point FFT of $x[n]$ over $8 \leq F \leq 12$ Hz.

```
dt = 0.01;
t = 0:dt:n(end)*T;
xc = cos(20*pi*t) + cos(18*pi*t) + cos(22*pi*t);
xn = cos(20*pi*nT) + cos(18*pi*nT) + cos(22*pi*nT);
L = 128; % Part (a)
% L = 1024; % Part (b)
k = 0:L-1;
X = fft(xn,L);
Fp = Fs*k/L;
indk = Fp >= 8 & Fp <= 12;
kp = k(indk);
%% Part (c):
[Xcta,w] = cta(xn,length(kp),8*2*pi*T,12*2*pi*T);
%% Plot:
hfa = figconfg('P0832a','small');
plot(Fp(indk),abs(X(indk))); hold on
stem(Fp(indk),abs(X(indk)),'filled')
xlim([8 12])
xlabel('F (Hz)','fontsize',LFS)
title('Magnitude Spectra','fontsize',TFS)

hfb = figconfg('P0832b','small');
plot(w*Fs/2/pi,abs(Xcta)); hold on
stem(w*Fs/2/pi,abs(Xcta),'filled')
```

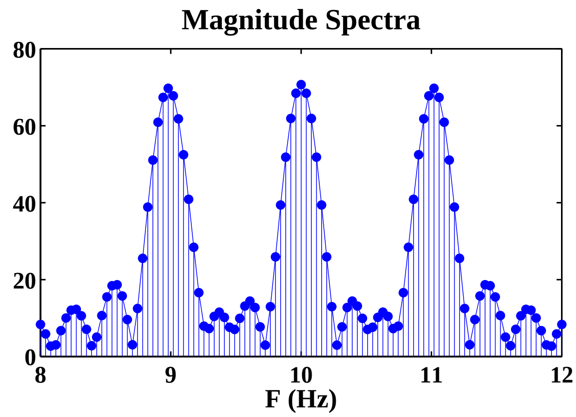


FIGURE 8.5: Magnitude spectra of DFT of $x[n]$ over $8 \leq F \leq 12$ Hz using `cta` function.

```
xlim([8 12])
xlabel('F (Hz)','fontsize',LFS)
title('Magnitude Spectra','fontsize',TFS)
```

Assessment Problems

33. (a) Proof:

We first repeat (8.2) as follow:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-kn}, \quad n = 0, 1, \ldots, N-1. \tag{8.2}$$

The derivation procedure is:

$$\begin{aligned} x[n] &= \frac{1}{N}\sum_{k=0}^{N-1} X[\langle -k\rangle_N]W_N^{-n\langle -k\rangle_N} \\ &= \frac{1}{N}\sum_{k=0}^{N-1} X[\langle -k\rangle_N]W_N^{-n(N-k)} = \frac{1}{N}\sum_{k=0}^{N-1} X[\langle -k\rangle_N]W_N^{nk} \\ &= \text{DFT}\left\{X[\langle -k\rangle_N]\right\} \end{aligned}$$

(b) MATLAB function:

```
function x = IDFT_0833(X,N)
X = [X(1) X(end:-1:2)];
x = fft(X)/N;
```

MATLAB script:

```
% P0833: Testing function IDFT
close all; clc
xn = 1:8;
X = fft(xn);
xr = IDFT_0833(X,length(xn));
```

34. Solution:

The resulting trend in the computational complexity is much simplified than direct computation.

MATLAB script:

```
% P0834: Investigate time consumption using fftditr2
close all; clc
nu = 2:10;
N = 2.^nu;
Ni = length(N);
t = zeros(1,Ni);
```

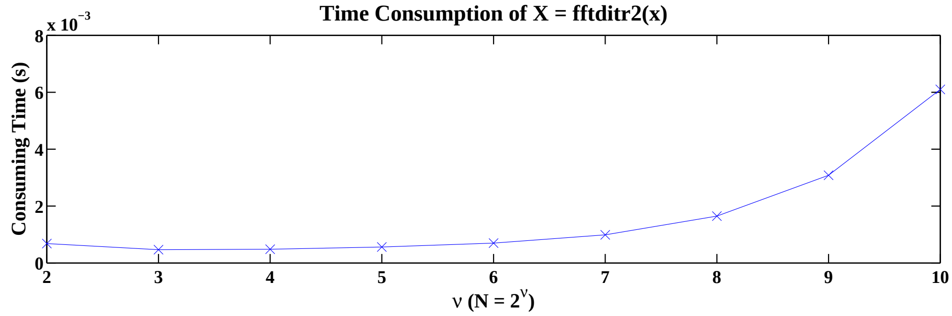


FIGURE 8.6: Plot of computation time for the fftditr2 function for $N = 2^{\nu}$ where $2 \leq \nu \leq 10$.

```
for ii = 1:Ni
    x = randn(1,N(ii)) + j*randn(1,N(ii));
    tic
    X = fftditr2(x);
    t(ii) = toc;
end
% Plot:
hfa = figconfg('P0834a','long');
plot(nu,t,'x-','markersize',12)
xlabel('\nu (N = 2^{\nu})','fontsize',LFS)
ylabel('Consuming Time (s)','fontsize',LFS)
title('Time Consumption of X = fftditr2(x)','fontsize',TFS)
```

35. Solution:
Compare the computation complexity by the number of complex multiplications. The direct form needs N^2 complex multiplications while the radix-2 DIT-FFT algorithm requires $N \log_2 N$ complex multiplications. Hence, the minimum values K is defined as follow:

$$\min K = \log_2 N$$

Thus, for $N = 128$, 1024, and 8192, the minimum of K is 7, 10, and 13, correspondingly.

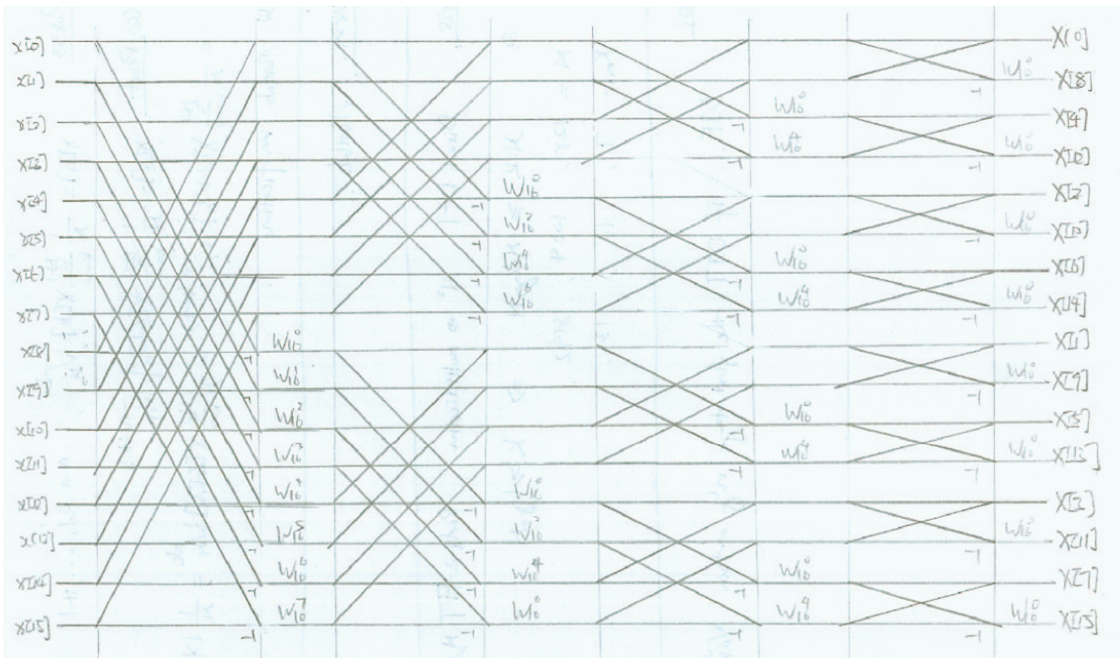
36. Solution:
The total number of complex multiplications is:

$$8 \times 4 = 32$$

The total number of complex additions is:

$$16 \times 4 = 64$$

Hence, the total number of real multiplications is 128 and the total number of real additions is 192.



37. Solution:
    This algorithm implements DIF approach since DIT approach only contains W_{64}^{0} in the first stage.

38. (a) Solution:

$$X[3k] = \sum_{n=0}^{1} \left(x[n] + x[n+2] + x[n+4]\right) W_2^{nk}$$

$$X[3k+1] = \sum_{n=0}^{1} \left(x[n] + x[n+2]W_6^2 + x[n+4]W_6^4\right) W_6^n W_2^{nk}$$

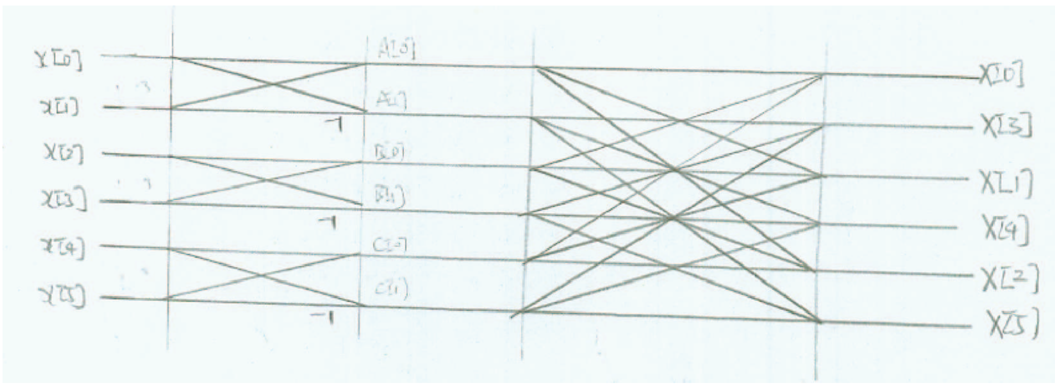
$$X[3k+2] = \sum_{n=0}^{1} \left(x[n] + x[n+2]W_6^4 + x[n+4]W_6^2\right) W_6^{2n} W_2^{nk}$$

The total number of real multiplications is:

$$4 \times (3 + 2 \times 6) = 60$$

The total number of real additions is:

$$3 \times (6 + 2 \times 6) = 36$$



(b) Solution:

$$X[2k] = \sum_{n=0}^{2} (x[n] + x[n+3])\, W_3^{nk}$$

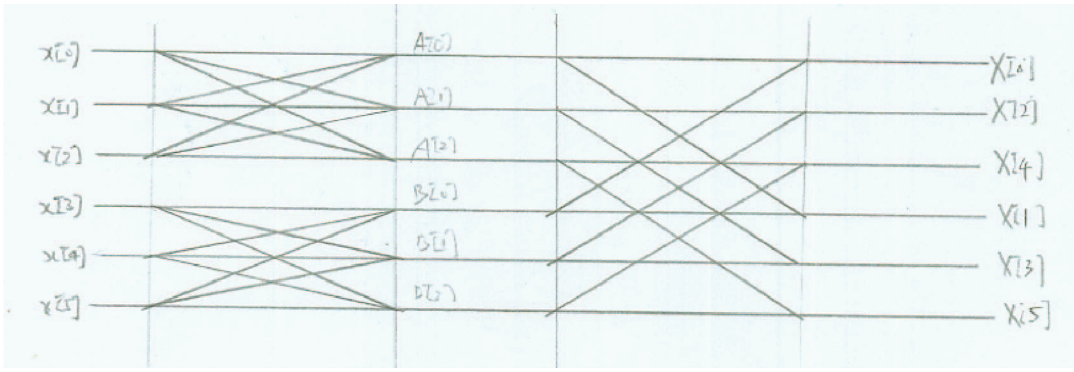
$$X[2k+1] = \sum_{n=0}^{2} \left(x[n] + x[n+3] W_6^3\right) W_6^n W_3^{nk}$$

The total number of real multiplications is:

$$4 \times (2 \times 6 + 3) = 60$$

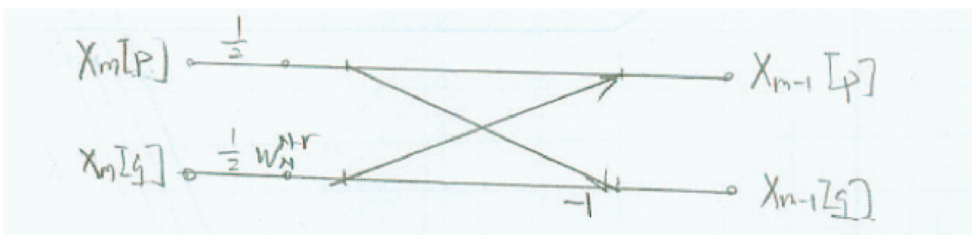
The total number of real additions is:

$$3 \times (2 \times 6 + 6) = 36$$



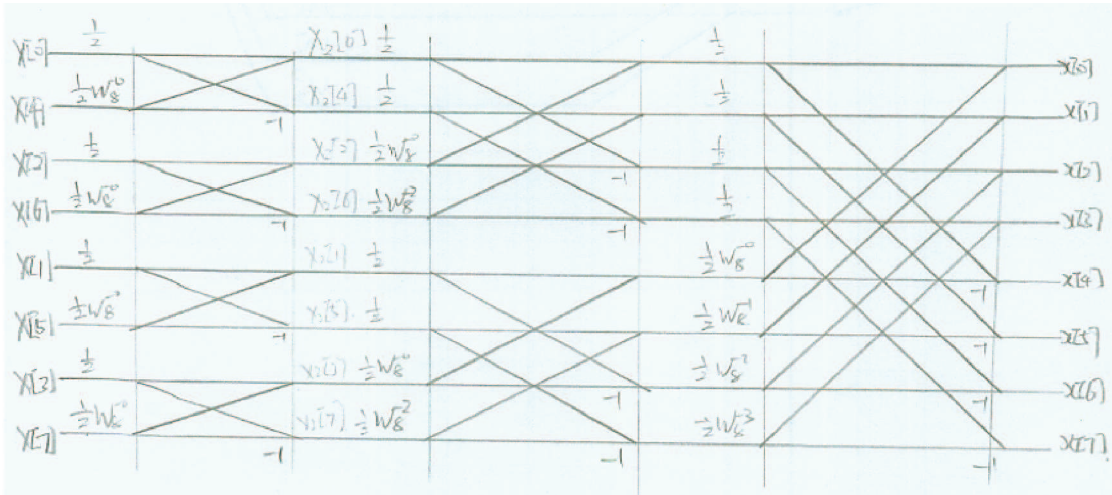
39. (a) Solution:

$$X_{m-1}[p] = \frac{1}{2}\left(X_m[p] + X_m[q]W_N^{-r}\right)$$
$$X_{m-1}[q] = \frac{1}{2}\left(X_m[p] - X_m[q]W_N^{-r}\right)$$

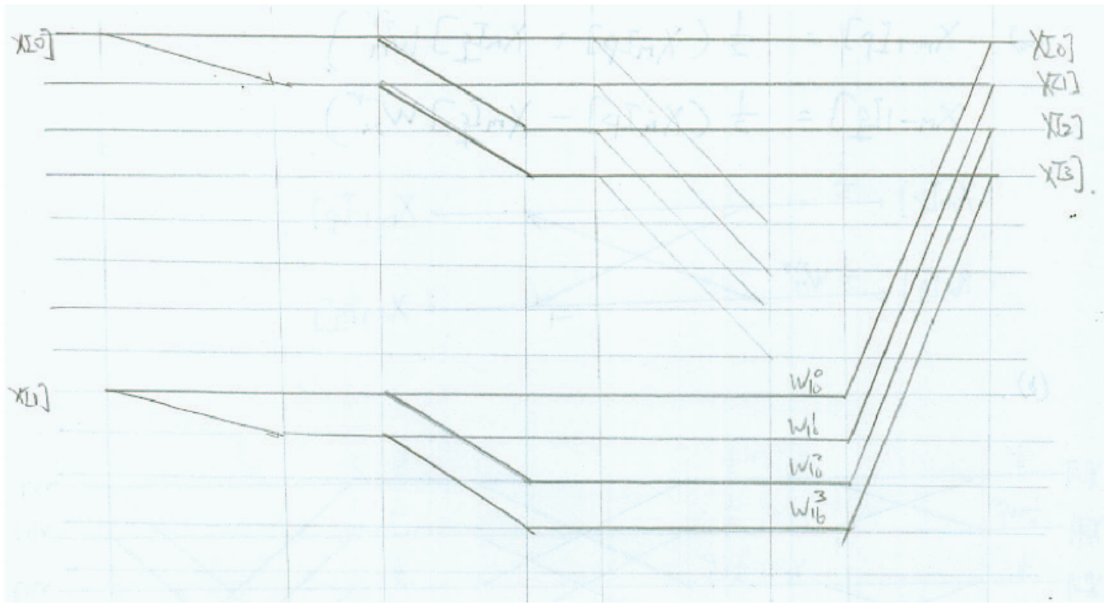


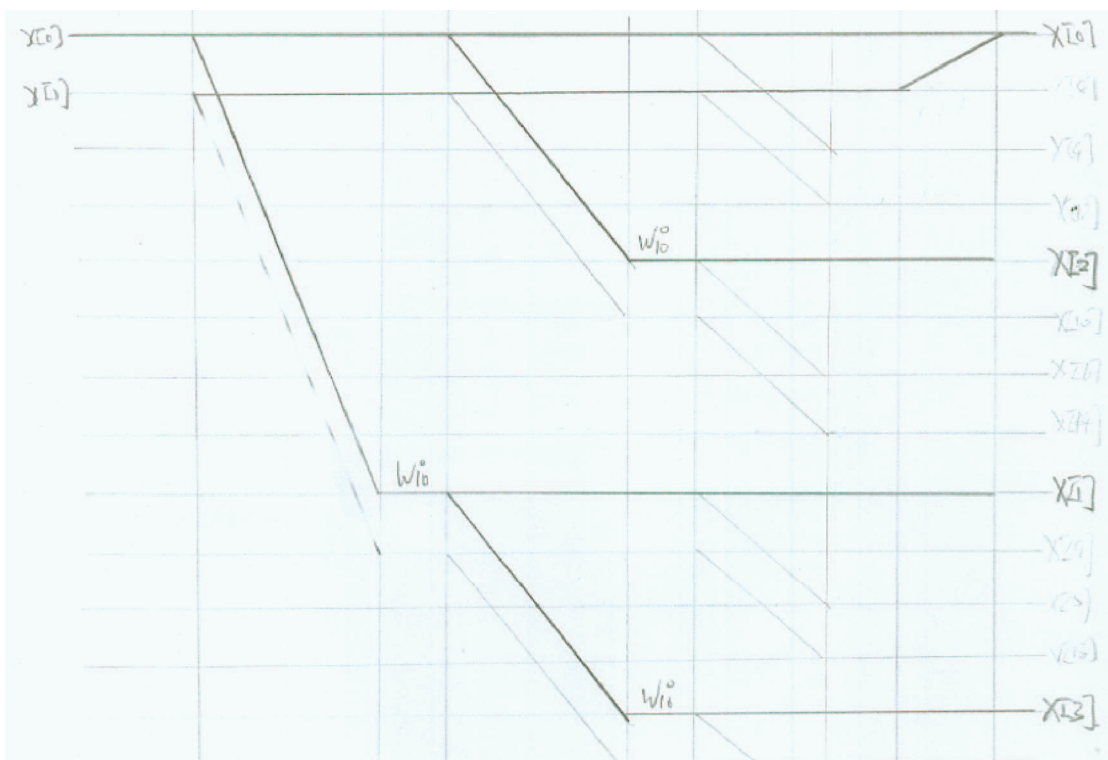
(b) See graph below.

(c) Comments:
The two flow-graphs are the same.



40. (a) See graph below.
    (b) See graph below.
    (c) Solution:
    Part (a) requires 3 complex multiplications while part (b) requires none complex multiplication.
    (d) tba


x[0]
x[1]
X[0]
X[1]
X[2]
X[3]
W_{16}^0
W_{16}^1
W_{16}^2
W_{16}^3


y[0]
y[1]
X[0]
X[2]
X[1]
X[3]
W_{16}^0
W_{16}^0
W_{16}^0

41. The problem is incomplete.

42. Proof:

$$\begin{cases} a[n] \triangleq x[n] + x[n+\frac{N}{2}] \\ b[n] \triangleq (x[n] - x[n+\frac{N}{2}])W_N^n \end{cases} \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{8.38}$$

$$A[k] = \sum_{n=0}^{\frac{N}{2}-1} a[n] W_{\frac{N}{2}}^{kn} \tag{8.39a}$$

$$B[k] = \sum_{n=0}^{\frac{N}{2}-1} b[n] W_{\frac{N}{2}}^{kn} \tag{8.39b}$$

$$\begin{cases} X[2k] = A[k] \\ X[2k+1] = B[k] \end{cases} \quad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{8.40}$$

$$X_{n_2}[k_1] = \sum_{n_1=0}^{N_1-1} x_{n_2}[n_1] W_{N_1}^{k_1 n_1} \tag{8.52}$$

$$x_{k_1}[n_2] \triangleq W_N^{k_1 n_2} X_{n_2}[k_1] \tag{8.54}$$

$$X[k_1 + N_1 k_2] = \sum_{n_2=0}^{N_2-1} x_{k_1}[n_2] W_{N_2}^{k_2 n_2} \tag{8.55}$$

If we have $N_1 = 2$ and $N_2 = \frac{N}{2}$, (8.55) can be written as

$$X[2k_2] = \sum_{n_2=0}^{\frac{N}{2}-1} x_0[n_2] W_{\frac{N}{2}}^{k_2 n_2}$$

$$X[2k_2+1] = \sum_{n_2=0}^{\frac{N}{2}-1} x_1[n_2] W_{\frac{N}{2}}^{k_2 n_2}$$

Thus, (8.54) can be written as

$$\begin{cases} x_0[n_2] = X_{n_2}[0] \\ x_1[n_2] = W_N^{n_2} X_{n_2}[1] \end{cases}$$

Then, (8.52) can be written as

$$\begin{cases} X_{n_2}[0] = x_{n_2}[0] + x_{n_2}[1] = x[n_2] + x[n_2 + \frac{N}{2}] \\ X_{n_2}[1] = x_{n_2}[0] + x_{n_2}[1]W_2^1 = x[n_2] - x[n_2 + \frac{N}{2}] \end{cases}$$

which is of the same form as the DIT-FFT algorithm of (8.38) – (8.40).

43. (a) MATLAB function:

```
function X = gafft_vec(x,N,k)
% Goertzel's algorithm
% X = gafft(x,N,k)
% Computes k-th sample of an N-point DFT X[k] of x[n]
% using Goertzel Algorithm
% k is a vector
L = length(x); x = [reshape(x,1,L),zeros(1,N-L+1)];
K = length(k); X = zeros(1,K);
for ii = 1:K
   v = filter(1,[1,-2*cos(2*pi*k(ii)/N),1],x);
   X(ii) = v(N+1)-exp(-1j*2*pi*k(ii)/N)*v(N);
end
```

(b) MATLAB script:

```
% P0843: verify gafft_vec
close all; clc
ii = 2;
N = [8 16 32];
n = 0:N(ii)-1;
% k = 0:4;
k = [1 4 9 8];
xn = cos(0.5*pi*n);
X = gafft_vec(xn,N(ii),k);
%% Verification:
X_ref = fft(xn);
X_ref = X_ref(k+1);
```

44. (a) See plot below.
(b) See plot below.
(c) See plot below.

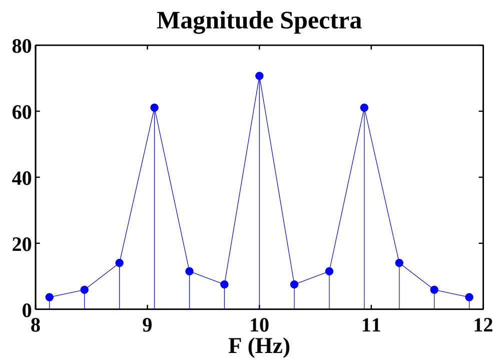


FIGURE 8.7: Magnitude spectra of 128-point FFT of $x[n]$ over $8 \leq F \leq 12$ Hz.

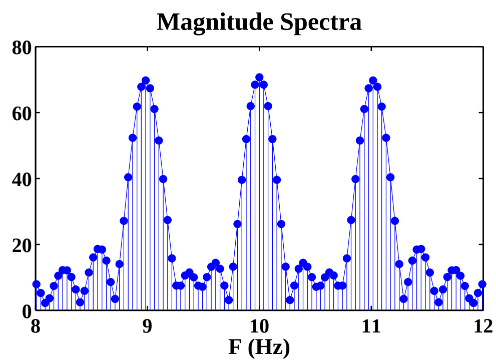


FIGURE 8.8: Magnitude spectra of 1024-point FFT of $x[n]$ over $8 \leq F \leq 12$ Hz.

(d) Solution:
Using `zfa` function has the smallest number of computations with a better display.

MATLAB script:

```
% P0844: Illustration of using cza algorithm
close all; clc
Fs = 40;
n = 0:127;
T = 1/Fs;
nT = n*T;
dt = 0.01;
t = 0:dt:n(end)*T;
```

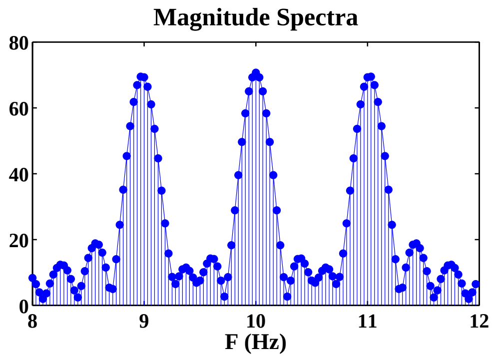


FIGURE 8.9: Magnitude spectra of $x[n]$ over $8 \leq F \leq 12$ Hz using the zfa function.

```
xc = cos(20*pi*t) + cos(18*pi*t) + cos(22*pi*t);
xn = cos(20*pi*nT) + cos(18*pi*nT) + cos(22*pi*nT);
% L = 128; % Part (a)
L = 1024; % Part (b)
k = 0:L-1;
X = fft(xn,L);
Fp = Fs*k/L;
indk = Fp >= 8 & Fp <= 12;
kp = k(indk);
%% Part (c):
[Xcta,w] = zfa(xn,length(kp),8*2*pi*T,12*2*pi*T);
%% Plot:
hfa = figconfg('P0844a','small');
plot(Fp(indk),abs(X(indk))); hold on
stem(Fp(indk),abs(X(indk)),'filled')
xlim([8 12])
xlabel('F (Hz)','fontsize',LFS)
title('Magnitude Spectra','fontsize',TFS)

hfb = figconfg('P0844b','small');
plot(w*Fs/2/pi,abs(Xcta)); hold on
stem(w*Fs/2/pi,abs(Xcta),'filled')
xlim([8 12])
xlabel('F (Hz)','fontsize',LFS)
title('Magnitude Spectra','fontsize',TFS)
```

Review Problems

45. (a) Solution:
    Use MATLAB function "`W = dftmtx(N)`".

    (b) See plot below.

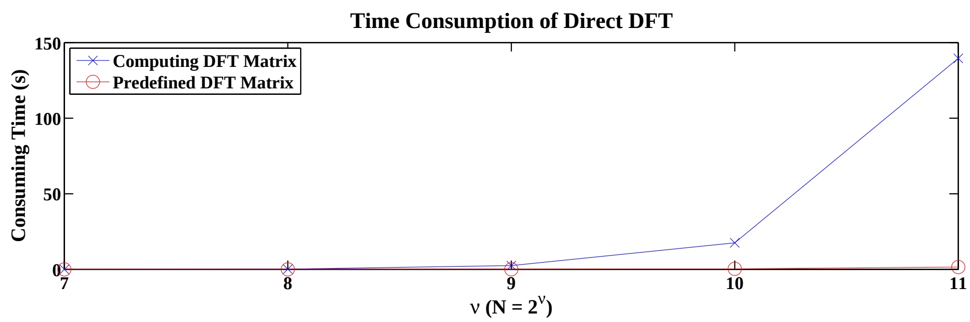


FIGURE 8.10: Plot of computation time for the `dftdirect` function and the `dftdirect_m` function.

MATLAB function:

```
function Xdft=dftdirect_m(x,W)
% Direct computation of the DFT
N=length(x);
Xdft = zeros(1,N);
for k=1:N
     S=0;
     for n=1:N
          S=S+W(k,n)*x(n);
     end
     Xdft(k)=S;
end
```

(c) See plot below.
MATLAB function:

```
function Xdft = fftrecur_m(x,W)
% Recursive computation of the DFT using divide & conquer
% N should be a power of 2
N = length(x);
if N ==1
  Xdft = x;
```

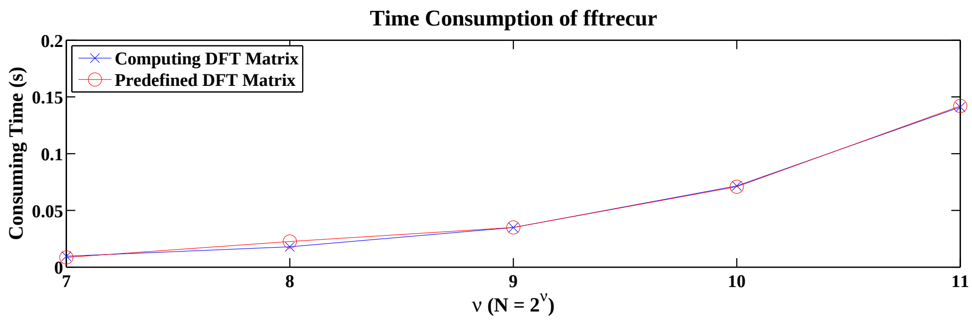


FIGURE 8.11: Plot of computation time for the fftrecur function and the fftrecur_m function.

```
    else
        m = N/2;
        XE = fftrecur(x(1:2:N));
        XO = fftrecur(x(2:2:N));
        temp = W(1:m,2).*XO;
        Xdft = [ XE+temp ; XO-temp ];
    end
```

MATLAB script:

```
% P0845: Investigate efficiency improvement
%         when the DFT matrix is precalculated
close all; clc
N = [128 256 512 1024 2048];
nu = log2(N);
Ni = length(N);
t_dir = zeros(1,Ni);
t_dirm = zeros(1,Ni);
t_recur = zeros(1,Ni);
t_recurm = zeros(1,Ni);
for ii = 1:Ni
    x = randn(1,N(ii)) + j*randn(1,N(ii));
    tic
    X_dir = dftdirect(x);
    t_dir(ii) = toc;
    W = dftmtx(N(ii));
    tic
```

```
    X_dirm =dftdirect_m(x,W);
    t_dirm(ii) = toc;

    tic
    X_recur = fftrecur(x);
    t_recur(ii) = toc;
    tic
    X_recurm = fftrecur_m(x,W);
    t_recurm(ii) = toc;
    ii
end
% Plot:
hfa = figconfg('P0845a','long');
plot(nu,t_dir,'x-','markersize',12); hold
plot(nu,t_dirm,'o-','color','red','markersize',12)
set(gca,'XTick',nu)
xlabel('\nu (N = 2^{\nu})','fontsize',LFS)
ylabel('Consuming Time (s)','fontsize',LFS)
title('Time Consumption of Direct DFT','fontsize',TFS)
legend('Computing DFT Matrix','Predefined DFT Matrix',...
    'location','northwest')

hfb = figconfg('P0845b','long');
plot(nu,t_recur,'x-','markersize',12); hold
plot(nu,t_recurm,'o-','color','red','markersize',12)
set(gca,'XTick',nu)
xlabel('\nu (N = 2^{\nu})','fontsize',LFS)
ylabel('Consuming Time (s)','fontsize',LFS)
title('Time Consumption of fftrecur','fontsize',TFS)
legend('Computing DFT Matrix','Predefined DFT Matrix',...
    'location','northwest')
```

46. (a) Proof:

$$x_N[n] = x_N(nT) = x(n \cdot \frac{T}{2N+1}) = \sum_{k=-N}^{N} c_k \mathrm{e}^{\mathrm{j}2\pi k \frac{T}{2N+1}\frac{n}{T}}$$
$$= \sum_{k=-N}^{N} c_k \mathrm{e}^{\mathrm{j}\frac{2\pi}{2N+1}nk} = \sum_{k=-N}^{N} c_k W_{2N+1}^{kn}$$

Hence, we proved the CTFS coefficients $\{c_k\}$ can be interpreted as $2N+1$-point DFT coefficients of the data values.

(b) Scheme:
Step I:
Compute the CTFS coefficients $\{c_k\}$ using FFT.
Step II:
Properly padding $L-2N-1$ zeros to $\{c_k\}$ and applying IFFT to construct interpolation signal.

(c) See plot below.

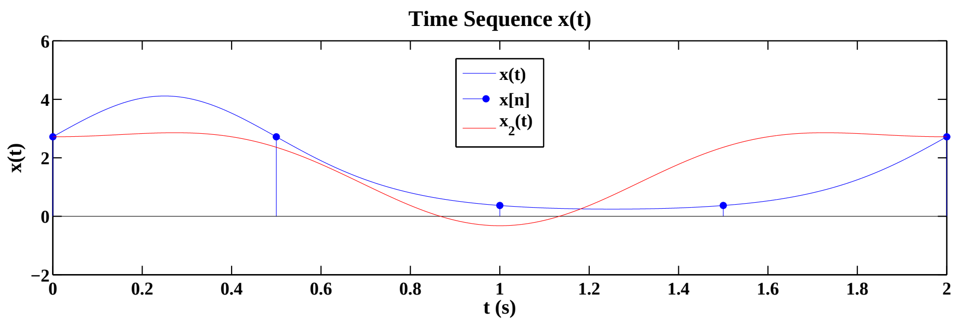


FIGURE 8.12: Plot of interpolated signal $x_2(t)$ when $N=2$ compared to the original signal $x(t)$.

(d) Comments:
With more CTFS coefficients, the interpolation is closer to the original signal.

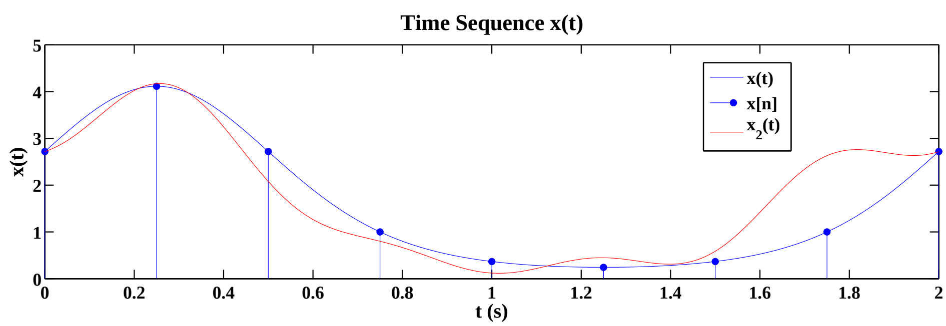


FIGURE 8.13: Plot of interpolated signal $x_2(t)$ when $N=4$ compared to the original signal $x(t)$.

MATLAB script:

```
% P0846: Illustrate using FFT to perform fast interpolation
close all; clc
t1 = 0; t2 = 2*pi;
dt = 0.001;
t = t1:dt:t2;
xt = exp(sin(t)+cos(t));
N = 2; % Part (c)
% N = 4; % Part (d)
L = 2*N+1;
nT = linspace(t1,t2,L);
xn = exp(sin(nT)+cos(nT));
ck = fft(xn)/L;
NN = length(t);
yk = [ck(1:N+1),zeros(1,NN-L),ck(N+2:end)];
xr = real(ifft(yk))*NN;
%% Plot:
hfa = figconfg('P0846a','long');
plot(t/pi,xt); hold on
stem(nT/pi,xn,'filled')
plot(t/pi,xr,'color','red')
xlim([0 2])
xlabel('t (s)','fontsize',LFS)
ylabel('x(t)','fontsize',LFS)
title('Time Sequence x(t)','fontsize',TFS)
legend('x(t)','x[n]','x_2(t)','location','best')
```

47. (a) Proof:

$$\begin{aligned}
X[2k] &= \sum_{n=0}^{N-1} x[n] W_N^{n(2k)} = \sum_{n=0}^{N-1} x[n] W_{\frac{N}{2}}^{nk} \\
&= \sum_{n=0}^{\frac{N}{2}-1} x[n] W_{\frac{N}{2}}^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x[n+\frac{N}{2}] W_{\frac{N}{2}}^{(n+\frac{N}{2})k} \\
&= \sum_{n=0}^{\frac{N}{2}-1} \left(x[n] + x[n+\frac{N}{2}] \right) W_{\frac{N}{2}}^{nk}
\end{aligned}$$

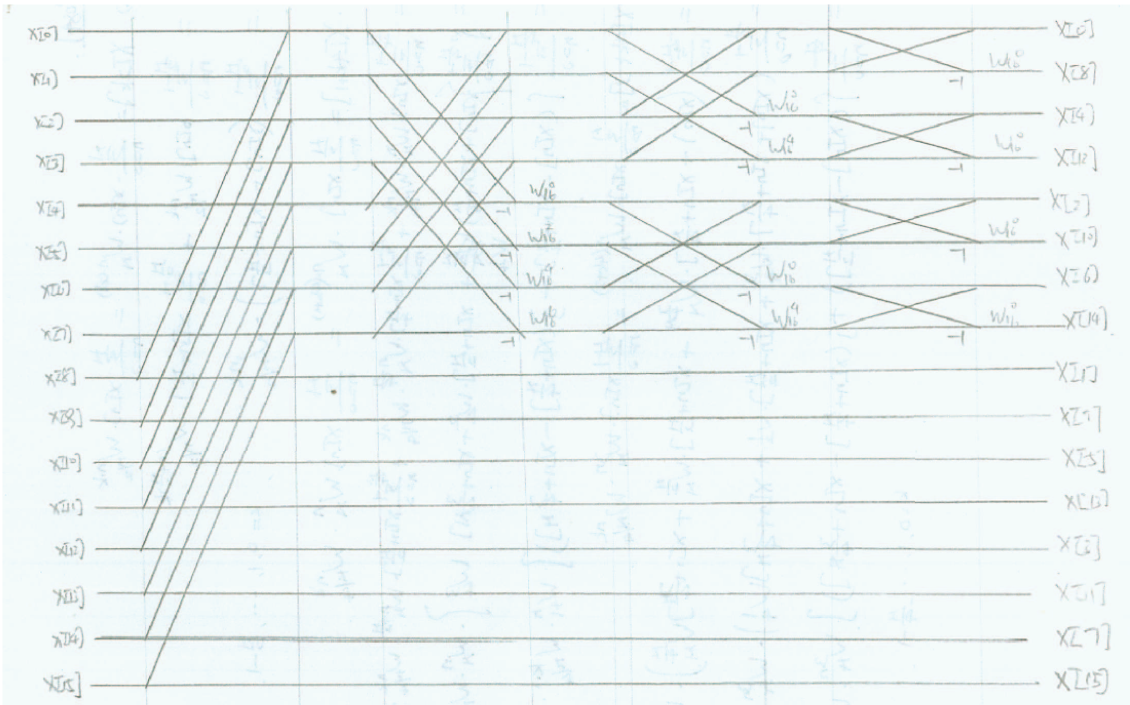
(b) Proof:

$$
\begin{aligned}
X[4k+1] &= \sum_{n=0}^{N-1} x[n] W_N^{n(4k+1)} = \sum_{n=0}^{N-1} x[n] W_N^n W_{\frac{N}{4}}^{nk} \\
&= \sum_{n=1}^{\frac{N}{4}-1} x[n] W_N^n W_{\frac{N}{4}}^{nk} + \sum_{n=1}^{\frac{N}{4}-1} x[n+\frac{N}{4}] W_N^{n+\frac{N}{4}} W_{\frac{N}{4}}^{nk} \\
&\quad + \sum_{n=1}^{\frac{N}{4}-1} x[n+\frac{N}{2}] W_N^{n+\frac{N}{2}} W_{\frac{N}{4}}^{nk} + \sum_{n=1}^{\frac{N}{4}-1} x[n+\frac{3N}{4}] W_N^{n+\frac{3N}{4}} W_{\frac{N}{4}}^{nk} \\
&= \sum_{n=1}^{\frac{N}{4}-1} \left(x[n] + x[n+\frac{N}{4}] W_4^1 + x[n+\frac{N}{2}] W_4^2 + x[n+\frac{3N}{4}] W_4^3 \right) W_N^n W_{\frac{N}{4}}^{nk} \\
&= \sum_{n=1}^{\frac{N}{4}-1} \left\{ \left(x[n] - x[n+\frac{N}{2}] \right) - \mathrm{j} \left(x[n+\frac{N}{4}] - x[n+\frac{3N}{4}] \right) \right\} W_N^n W_{\frac{N}{4}}^{nk}
\end{aligned}
$$

$$
\begin{aligned}
X[4k+3] &= \sum_{n=0}^{N-1} x[n] W_N^{n(4k+3)} = \sum_{n=0}^{N-1} x[n] W_N^{3n} W_{\frac{N}{4}}^{nk} \\
&= \sum_{n=1}^{\frac{N}{4}-1} \left(x[n] + x[n+\frac{N}{4}] W_N^{\frac{3N}{4}} + x[n+\frac{N}{2}] W_N^{\frac{N}{2}} + x[n+\frac{3N}{4}] W_N^{\frac{N}{4}} \right) W_N^{3n} W_{\frac{N}{4}}^{nk} \\
&= \sum_{n=1}^{\frac{N}{4}-1} \left(x[n] + x[n+\frac{N}{4}] W_4^3 + x[n+\frac{N}{2}] W_4^2 + x[n+\frac{3N}{4}] W_4^1 \right) W_N^{3n} W_{\frac{N}{4}}^{nk} \\
&= \sum_{n=1}^{\frac{N}{4}-1} \left\{ \left(x[n] - x[n+\frac{N}{2}] \right) + \mathrm{j} \left(x[n+\frac{N}{4}] - x[n+\frac{3N}{4}] \right) \right\} W_N^{3n} W_{\frac{N}{4}}^{nk}
\end{aligned}
$$

(c) tba

(d) tba



48. (a) MATLAB function:

```
function x = sym2TT(S)
% Generate 0.5 second samples of DTMF signal of symbol s
FH = [1209 1336 1477 1633];
FL = [697 770 852 941];
switch S
    case '1'
        F1 = FL(1); F2 = FH(1);
    case '2'
        F1 = FL(1); F2 = FH(2);
    case '3'
        F1 = FL(1); F2 = FH(3);
    case '4'
        F1 = FL(2); F2 = FH(1);
    case '5'
        F1 = FL(2); F2 = FH(2);
    case '6'
        F1 = FL(2); F2 = FH(3);
```

```
        case '7'
            F1 = FL(3); F2 = FH(1);
        case '8'
            F1 = FL(3); F2 = FH(2);
        case '9'
            F1 = FL(3); F2 = FH(3);
        case '*'
            F1 = FL(4); F2 = FH(1);
        case '0'
            F1 = FL(4); F2 = FH(2);
        case '#'
            F1 = FL(4); F2 = FH(3);
        otherwise
            error('Illegal Input')
    end
    Fs = 8e3; t1 = 0; t2 = 0.5;
    T = 1/Fs;
    nT = t1:T:t2;
    x = cos(2*pi*F1*nT)+cos(2*pi*F2*nT);
```

(b) MATLAB function:

```
function X = gafft_vec(x,N,k)
% Goertzel's algorithm
% X = gafft(x,N,k)
% Computes k-th sample of an N-point DFT X[k] of x[n]
% using Goertzel Algorithm
% k is a vector
L = length(x); x = [reshape(x,1,L),zeros(1,N-L+1)];
K = length(k); X = zeros(1,K);
for ii = 1:K
   v = filter(1,[1,-2*cos(2*pi*k(ii)/N),1],x);
   X(ii) = v(N+1)-exp(-1j*2*pi*k(ii)/N)*v(N);
end
```

(c) MATLAB function:

```
function S = TT2sym(x)
% Detect DTMF symbol
N = 200;
FH = [1209 1336 1477 1633];
FL = [697 770 852 941];
```

```
Fs = 8e3;
k = round([FL FH(1:3)]/Fs*N);
X = gafft_vec(x,N,k);
thresh = 50;
V = abs(X) > thresh;
switch bin2dec(num2str(V))
    case bin2dec('1000100')
        S = '1';
    case bin2dec('1000010')
        S = '2';
    case bin2dec('1000001')
        S = '3';
    case bin2dec('0100100')
        S = '4';
    case bin2dec('0100010')
        S = '5';
    case bin2dec('0100001')
        S = '6';
    case bin2dec('0010100')
        S = '7';
    case bin2dec('0010010')
        S = '8';
    case bin2dec('0010001')
        S = '9';
    case bin2dec('0001100')
        S = '*';
    case bin2dec('0001010')
        S = '0';
    case bin2dec('0001001')
        S = '#';
    otherwise
        error('Bad Input')
end
```

MATLAB script:

```
% P0848: DTMF Generation and Detection
close all; clc
ii = 1;
S = ['1','2','3','4','5','6','7','8','9','*','0','#'];
```

```
x = sym2TT(S(ii));
Sr = TT2sym(x);
%% Testing:
% for ii = 1:length(S)
%     x = sym2TT(S(ii));
%     Sr = TT2sym(x)
% end
```