CMPE 352
Signal Processing & Algorithms

Spring 2019

Sedat Ölçer

6 May, 2019

Review Questions (1)

- Why is the Discrete Fourier Transform (DFT) algorithm needed / useful?

  Allows computing the Fourier Transform numerically (on a processor).

- In the DFT algorithm, what are the quantities that are being discretized?

  The signal in the time domain: $g(t) \rightarrow g(kT_s)$

  The signal spectrum in the frequency domain: $G(\omega) \rightarrow G(r\omega_0) = G_r$

  → Hence T_s is the discretization interval on the time axis and ω_0 the discretization interval on the frequency axis.

- What can you say about the <u>number of samples</u> taken in the <u>time domain</u> and those taken in the <u>frequency domain</u>?

  They are equal (say, *N* samples).

  Time: N samples, spaced by T_s. (We have $NT_s = T_0$: time-domain period.)

  Frequency: N samples, spaced by ω_0. (We have $Nf_0 = f_s$: freq.-domain period.)

  ($f_0 = 1/T_0$)

Discrete Fourier Transform (DFT)

$$\boxed{g_k = \frac{1}{N}\sum_{r=0}^{N-1} G_r e^{jk\Omega_0 r}}$$

$$g_k = T_s g(kT_s)$$

$$\Omega_0 = \omega_0 T_s = \frac{2\pi}{N}$$

$$N = \frac{T_0}{T_s} = \frac{\omega_s}{\omega_0} = \frac{f_s}{f_0}$$

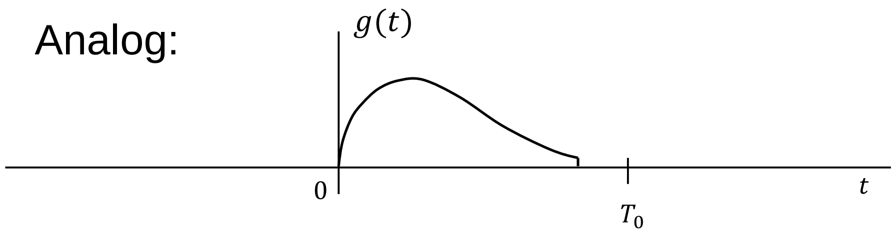
$$\boxed{G_r = \sum_{k=0}^{N-1} g_k e^{-jr\Omega_0 k}}$$

$$G_r = G(r\omega_0)$$

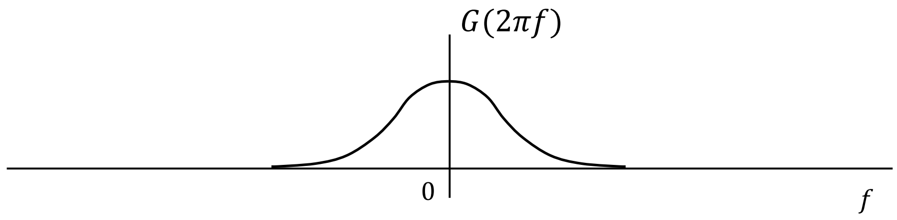
$$\omega_s = \frac{2\pi}{T_s} = 2\pi f_s$$

$$\omega_0 = \frac{2\pi}{T_0} = 2\pi f_0$$

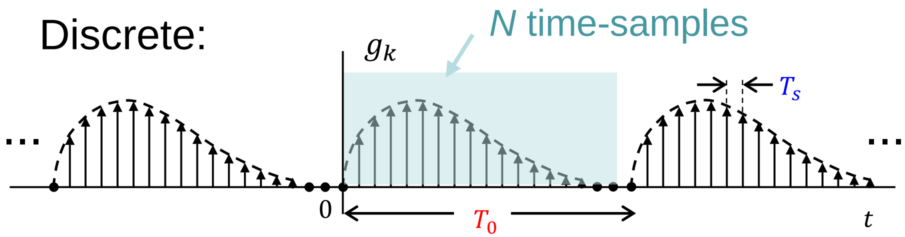
Analog:



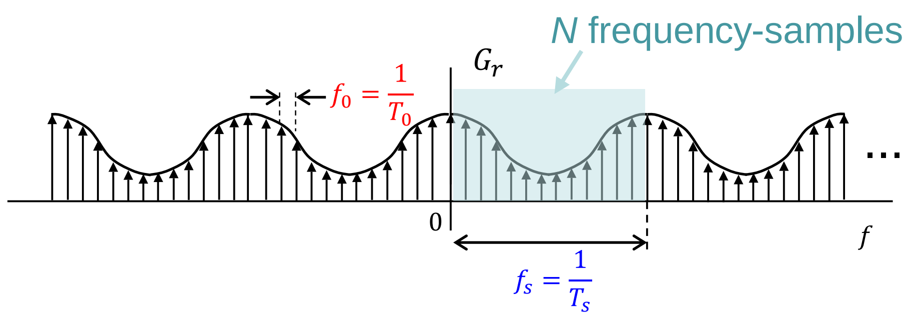
$\Leftrightarrow$



Discrete:



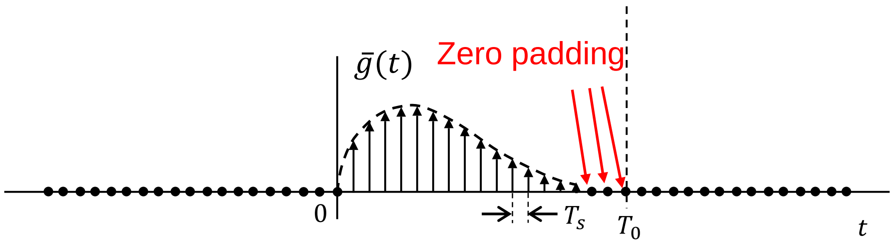
$\Leftrightarrow$



Review Questions (2)

- What is called zero-padding in the DFT algorithm?

  It is the extension of the original sample-set obtained by adding zero-valued samples to it.



- What is the purpose of zero-padding?

  To increase spectral resolution, because in the frequency domain the samples are spaced by $1/T_0$.

DFT Example

Problem: Use the DFT algorithm to compute the Fourier transform of

$$x(t) = e^{-2t}\, u(t).$$

Plot the Fourier spectra obtained analytically and those obtained by DFT.

1. Determine the Fourier transform analytically

$$X(j\omega) = \frac{1}{j\omega + 2}$$

2. Determine the sampling period T_s

The signal $x(t)$ has a low-pass characteristic but is not bandlimited.

We therefore need a criterion to define its ″essential bandwidth.″

For example we can take the ″essential bandwidth″ B to be the frequency at which $|X(j\omega)|$ drops to 1% of its peak value.

→ What is the value of B?

→ Then how do you choose T_s?

DFT Example (cntd)

3. Determine the time interval T_0

The signal $x(t)$ is not time-limited, so we have to truncate it at T_0 such that $x(T_0) \ll 1$.

A reasonable choice could be $T_0 = 4$, because $x(4) = e^{-8} = 0.000335 \ll 1$.

→What is the resulting DFT size $N = \frac{T_0}{T_s}$?

Note: you can slightly change the value of, for example, T_s so that N is an integer (or a power of 2).

4. Compute the DFT in Matlab

Note 1: you can compute the DFT by writing your own Matlab routine for the algorithm:

$$G_r = \sum_{k=0}^{N-1} g_k \, e^{-jr\Omega_0 k} \qquad\qquad g_k = \frac{1}{N}\sum_{r=0}^{N-1} G_r e^{jk\Omega_0 r}$$

or you can directly use the Matlab FFT routine

Note 2: because the signal has a jump discontinuity at $t = 0$, the first sample (at $t = 0$) can be taken as 0.5, the average of the values on the two sides of the discontinuity.

→ Plot the magnitude and phase spectra computed by Matlab

→ Plot (in Matlab) the magnitude and phase spectra computed analytically

compare

DFT – Computational Complexity - 1

$$G_r = \sum_{k=0}^{N-1} g_k e^{-jr\Omega_0 k}$$

- Requires N complex multiplications and $N-1$ complex additions

- Since G_r is to be computed for $r = 0, \dots, N-1$, there is in total

$$\left.\begin{array}{c} N^2 \text{ complex multiplications} \\ N(N-1) \text{ complex additions} \end{array}\right\} \text{Complexity is } O(N^2)$$

- The **Fast Fourier Transform (FFT)** brings complexity down to $O(N \log N)$

DFT – Computational Complexity - 2

N	N^2	$N \log N$	$N^2/(N \log N)$
$2^4 = 16$	256	64	4
$2^5 = 32$	1024	160	6
$2^6 = 64$	4096	384	11
$2^7 = 128$	16384	896	18
$2^8 = 256$	65536	2048	32
$2^9 = 512$	262144	4608	57
$2^{10} = 1024$	1048576	10240	102
$2^{11} = 2048$	...	...	186
$2^{13} = 8192$	...	...	630
$2^{14} = 16384$	...	...	1170

https://www.youtube.com/watch?v=aqa6vyGSdos

The Fast Fourier Transform (FFT)

[Cooley- Tukey (1965)]

- We want to evaluate efficiently (we use a slightly different notation):

$$g[n] = \frac{1}{N}\sum_{k=0}^{N-1} G[k]e^{jk\Omega_0 n} \qquad G[k] = \sum_{n=0}^{N-1} g[n]e^{-jk\Omega_0 n} \qquad (\Omega_0 = \frac{2\pi}{N})$$

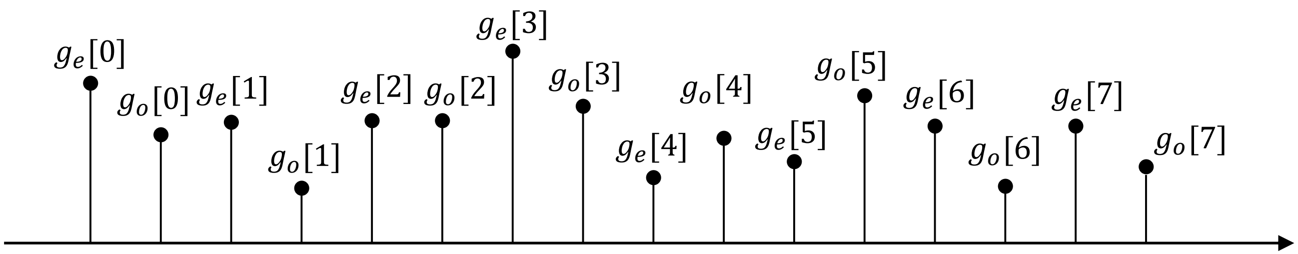
- Note that both equations are very similar, hence if we develop an efficient algorithm for one, we also have an efficient algorithm for the other.
- Let us concentrate on the equation for $G[k]$.

The Fast Fourier Transform (FFT) - 2

- $G[k] = \sum_{n=0}^{N-1} g[n] e^{-jk\Omega_0 n}$ → assume N to be even ($\Omega_0 = \frac{2\pi}{N}$)
- Split $g[n]$ into even/odd indexed signals:

Let $N' = \frac{N}{2}$ $\rightarrow$
$$g_e[n] = g[2n] \qquad 0 \le n \le N'-1$$
$$g_o[n] = g[2n+1] \qquad 0 \le n \le N'-1$$

Example: $N = 16 \Rightarrow N' = 8$



The Fast Fourier Transform (FFT) - 2

- $G[k] = \sum_{n=0}^{N-1} g[n]e^{-jk\Omega_0 n}$ → assume N to be even $\quad (\Omega_0 = \frac{2\pi}{N})$
- Split $g[n]$ into even/odd indexed signals:

Let $N' = \frac{N}{2}$ $\rightarrow$ $\quad g_e[n] = g[2n] \quad 0 \le n \le N' - 1$

$$g_o[n] = g[2n+1] \quad 0 \le n \le N' - 1$$

and let:

$$\left.\begin{aligned} g_e[n] &\overset{\text{DFT}}{\leftrightarrow} G_e[k] \\ g_o[n] &\overset{\text{DFT}}{\leftrightarrow} G_o[k] \end{aligned}\right\}$$

with Ω_0 replaced by $\Omega_0' = \frac{2\pi}{N'}$ $(= 2\,\Omega_0)$

- Now express $G[k]$ as

$$G[k] = \sum_{n=0}^{N-1} g[n]e^{-jk\Omega_0 n} = \sum_{n\ even} g[n]e^{-jk\Omega_0 n} + \sum_{n\ odd} g[n]e^{-jk\Omega_0 n}$$

- Write the time index n for even and odd streams as $2m$ and $2m+1$

$$G[k] = \sum_{m=0}^{N'-1} \underbrace{g[2m]}_{g_e[m]} e^{-jm\overbrace{2\Omega_0}^{\Omega_0'}k} + \sum_{m=0}^{N'-1} \underbrace{g[2m+1]}_{g_o[m]} e^{-j(m\overbrace{2\Omega_0}^{\Omega_0'}k + \Omega_0 k)}$$

The Fast Fourier Transform (FFT) - 3

$$G[k] = \underbrace{\sum_{m=0}^{N'-1} g_e[m] e^{-jm\Omega_0' k}}_{G_e[k]} + e^{-j\Omega_0 k} \underbrace{\sum_{m=0}^{N'-1} g_o[m] e^{-jm\Omega_0' k}}_{G_o[k]}$$

(obtained from the even samples) (obtained from the odd samples)

$$\boxed{G[k] = G_e[k] + e^{-j\Omega_0 k} G_o[k] \qquad \text{for } 0 \le k \le N-1}$$

$\rightarrow G[k]$ is a weighted combination of $G_e[k]$ and $G_o[k]$.

Note that $G_e[k]$ is periodic with period N': $G_e[k+N'] = G_e[k]$

$$G_e[k+N'] = \sum_{m=0}^{N'-1} g_e[m] e^{-jm\Omega_0'(k+N')} = \sum_{m=0}^{N'-1} g_e[m] e^{-jm\Omega_0' k} e^{-jm\Omega_0' N'} = \sum_{m=0}^{N'-1} g_e[m] e^{-jm\Omega_0' k} \underbrace{e^{-jm2\pi}}_{1} = G_e[k]$$

Similarly $G_o[k]$ is periodic with period N': $G_o[k+N'] = G_o[k]$

Note that $e^{-j(k+N')\Omega_0} = e^{-jk\Omega_0} e^{-jN'\Omega_0} = e^{-jk\Omega_0} e^{-j\pi} = -e^{-jk\Omega_0}$

Example for $N = 8;\ N' = N/2 = 4$

$G[k] = G_e[k] + e^{-j\Omega_0 k} G_o[k]$ for $0 \le k \le 7$ $\quad (\Omega_0 = \frac{2\pi}{N} = \frac{2\pi}{8} = \frac{\pi}{4})$

$G_e[k + N'] = G_e[k]$

$G_e[k + 4] = G_e[k], k = 0,1,2,3$

$G_o[k + N'] = G_o[k]$

$G_o[k + 4] = G_o[k], k = 0,1,2,3$

$e^{-j(k+N')\Omega_0} = -e^{-jk\Omega_0}, k = 0,1,2,3$

$e^{-j(k+4)\Omega_0} = -e^{-jk\Omega_0}, k = 0,1,2,3$

$G[0] = G_e[0] + G_o[0]$

$G[1] = G_e[1] + e^{-j\Omega_0} G_o[1] \quad = G_e[1] + e^{-j\pi/4} G_o[1]$

$G[2] = G_e[2] + e^{-j\Omega_0 2} G_o[2] \quad = G_e[2] + e^{-j\pi/2} G_o[2]$

$G[3] = G_e[3] + e^{-j\Omega_0 3} G_o[3] \quad = G_e[3] + e^{-j3\pi/4} G_o[3]$

$G[4] = G_e[4] + e^{-j\Omega_0 4} G_o[4] \quad = G_e[0] + e^{-j\Omega_0 4} G_o[0] \quad = G_e[0] - G_o[0]$

$G[5] = G_e[5] + e^{-j\Omega_0 5} G_o[5] \quad = G_e[1] + e^{-j\Omega_0 5} G_o[1] \quad = G_e[1] - e^{-j\Omega_0} G_o[1] \quad = G_e[1] - e^{-j\pi/4} G_o[1]$

$G[6] = G_e[6] + e^{-j\Omega_0 6} G_o[6] \quad = G_e[2] + e^{-j\Omega_0 6} G_o[2] \quad = G_e[2] - e^{-j\Omega_0 2} G_o[2] \quad = G_e[2] - e^{-j\pi/2} G_o[2]$

$G[7] = G_e[7] + e^{-j\Omega_0 7} G_o[7] \quad = G_e[3] + e^{-j\Omega_0 7} G_o[3] \quad = G_e[3] - e^{-j\Omega_0 3} G_o[3] \quad = G_e[3] - e^{-j3\pi/4} G_o[3]$

Example for $N = 8;\ N' = N/2 = 4$

$$G[k] = G_e[k] + e^{-j\Omega_0 k} G_o[k] \quad \text{for} \quad 0 \le k \le 7 \qquad \left(\Omega_0 = \frac{2\pi}{N} = \frac{2\pi}{8} = \frac{\pi}{4}\right)$$

$G[0] = G_e[0] + G_o[0]$

$G[1] = G_e[1] + e^{-j\pi/4} G_o[1]$

$G[2] = G_e[2] + e^{-j\pi/2} G_o[2]$

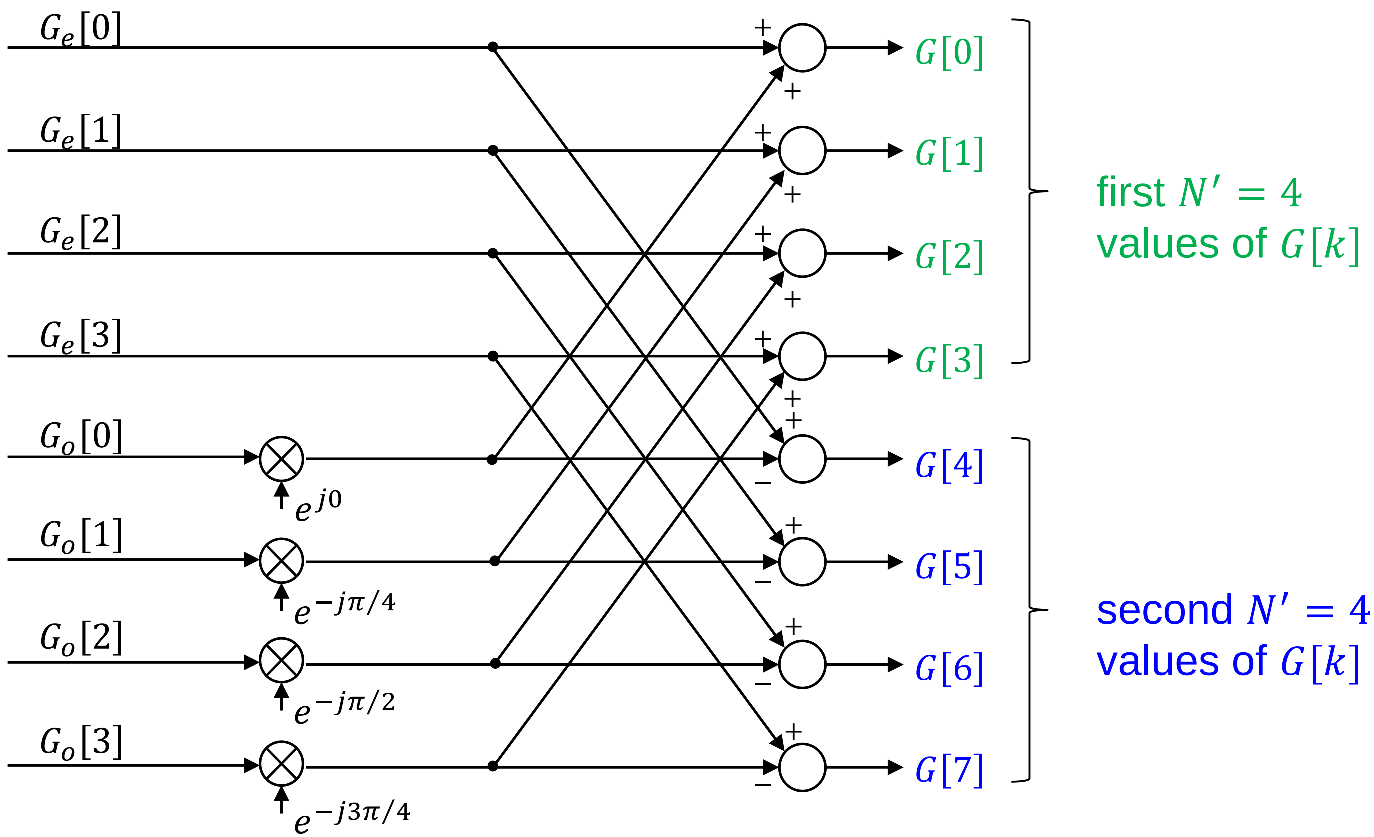
$G[3] = G_e[3] + e^{-j3\pi/4} G_o[3]$

$G[4] = G_e[0] - G_o[0]$

$G[5] = G_e[1] - e^{-j\pi/4} G_o[1]$

$G[6] = G_e[2] - e^{-j\pi/2} G_o[2]$

$G[7] = G_e[3] - e^{-j3\pi/4} G_o[3]$



The Fast Fourier Transform (FFT) - 4

- Hence in general $G[k] = G_e[k] + e^{-j\Omega_0 k} G_o[k]$ for $0 \le k \le N-1$ can be split as:

$\Rightarrow G[k] = G_e[k] + e^{-j\Omega_0 k} G_o[k]$ for $0 \le k \le N' - 1$

for the first N' values of $G[k]$

$\Rightarrow G[k+N'] = G_e[k] - e^{-j\Omega_0 k} G_o[k]$ for $0 \le k \le N' - 1$

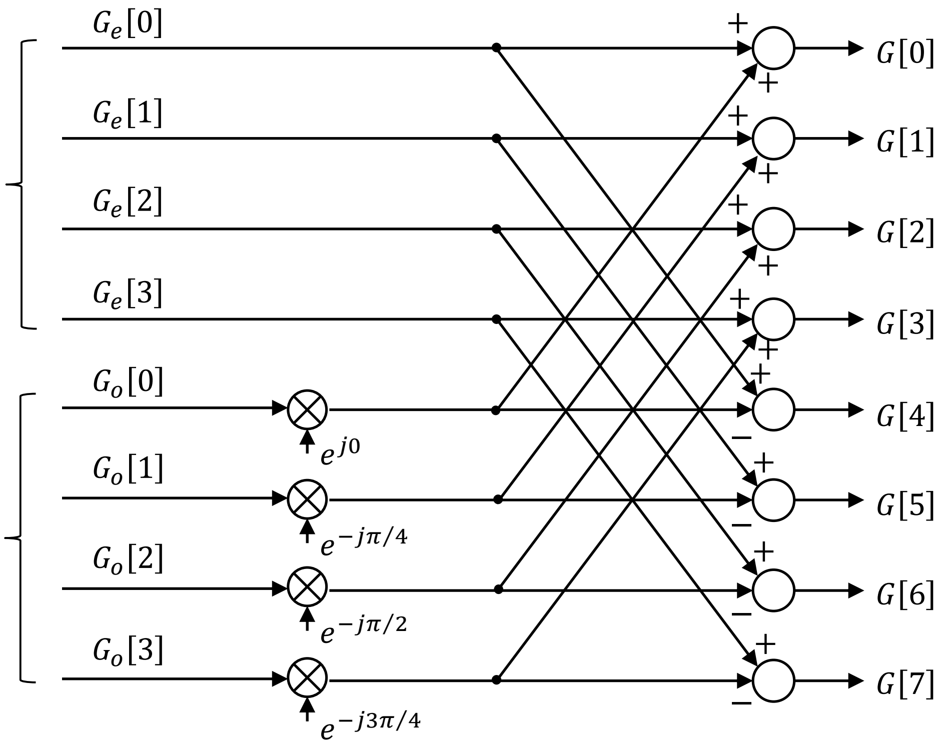
for the second N' values of $G[k]$

Note: we need to only multiply once by $e^{-j\Omega_0 k}$ to compute both equations

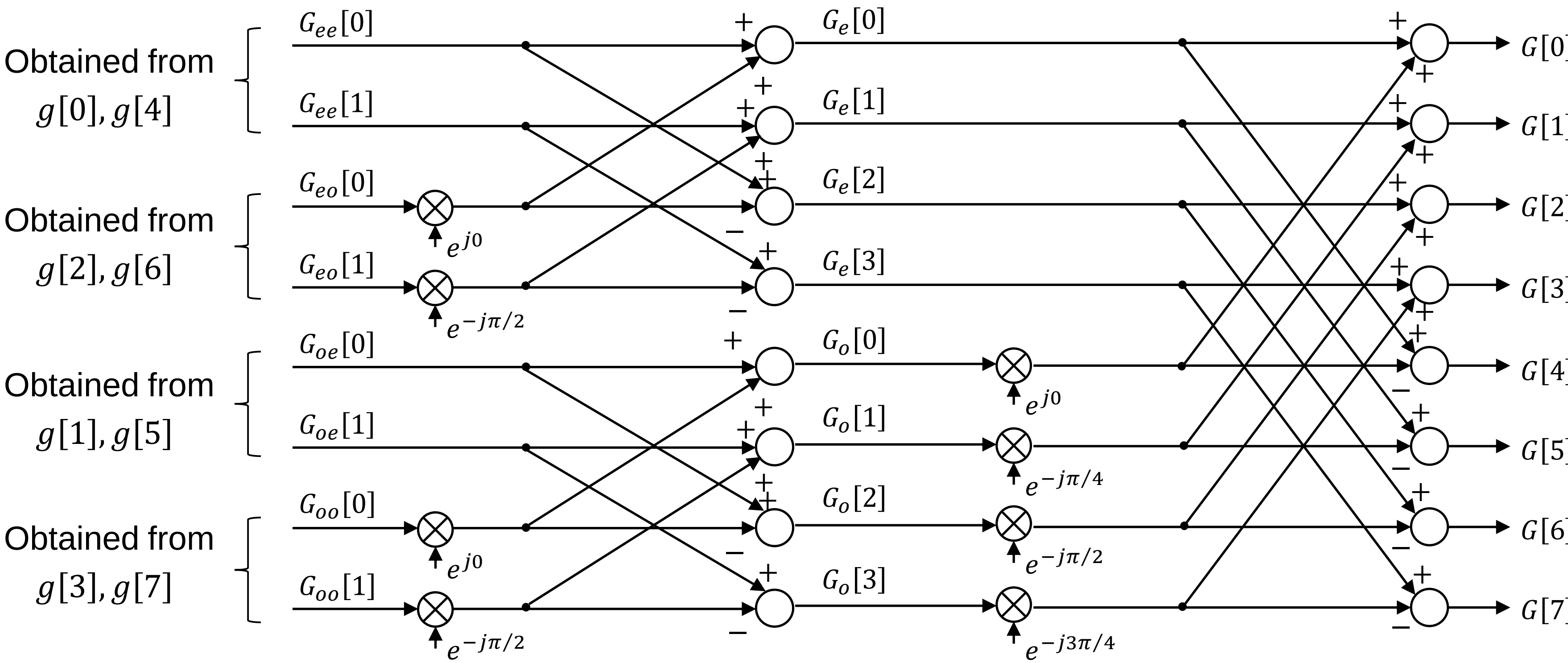
8-Point FFT $(N = 8,\ N' = 4)$

Obtained from the even samples
$g[0], g[2], g[4], g[6]$

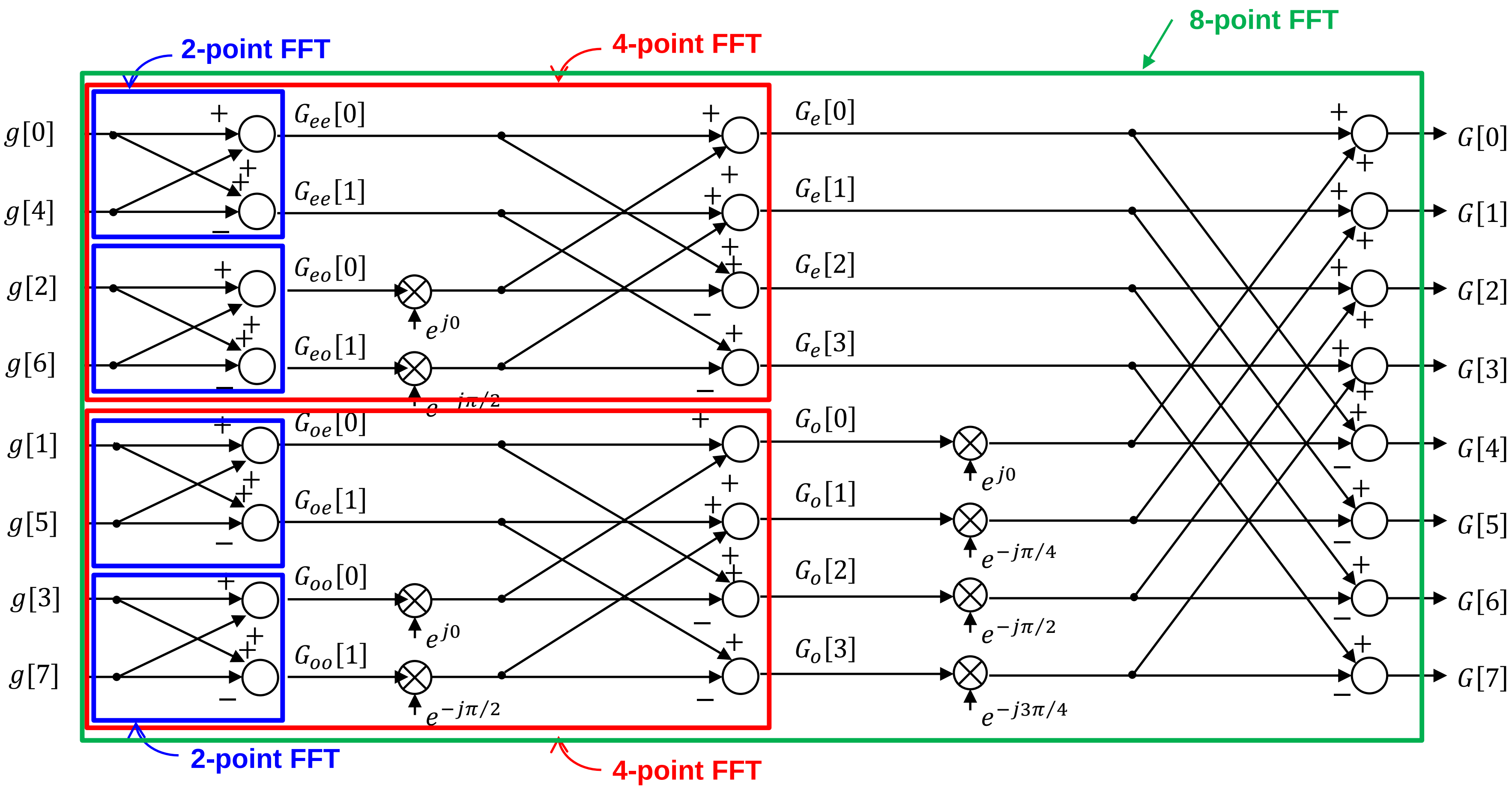
Obtained from the odd samples
$g[1], g[3], g[5], g[7]$



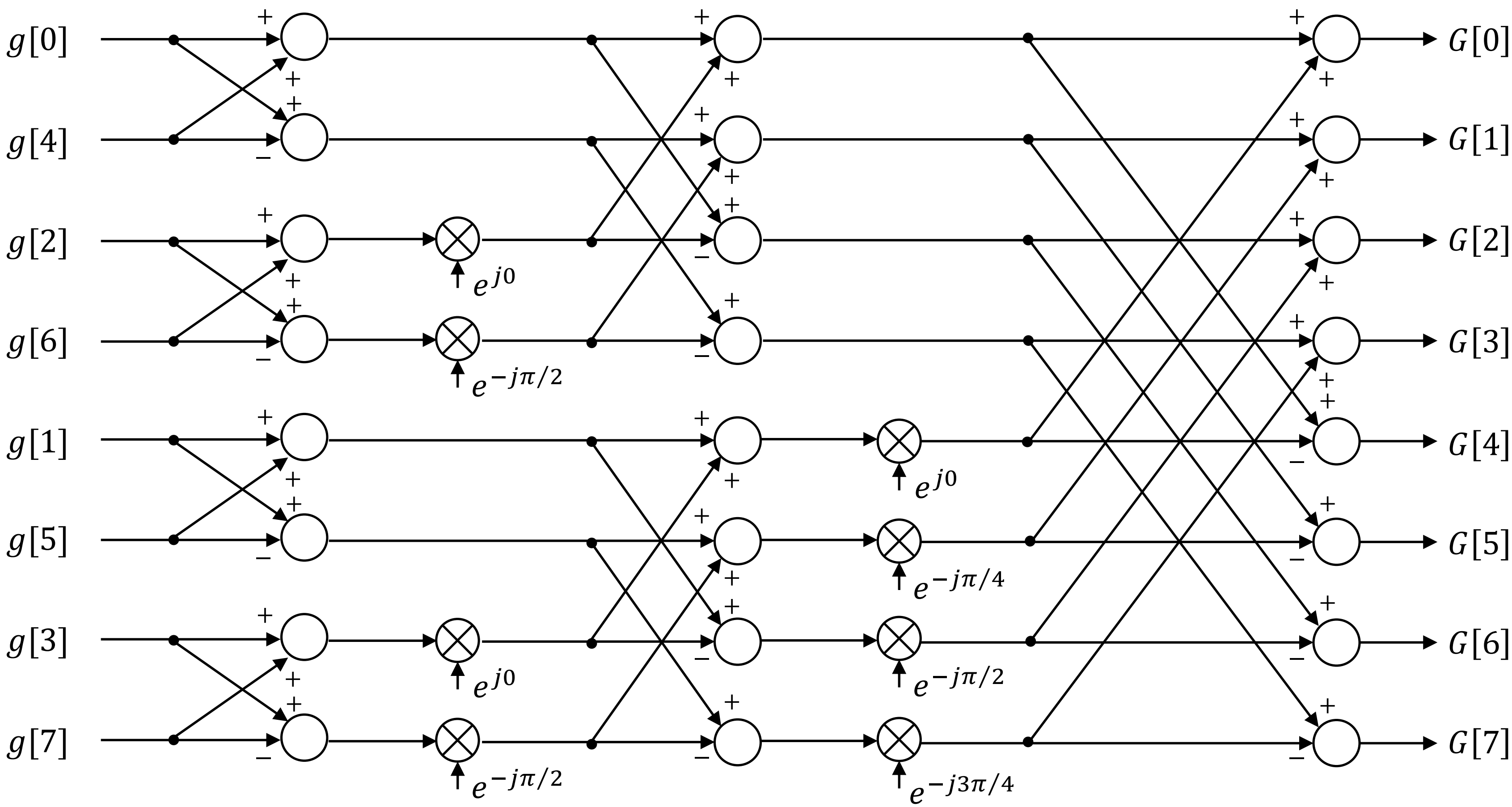
8-Point FFT $(N = 8,\ N' = 4)$



8-Point FFT $(N = 8,\ N' = 4)$



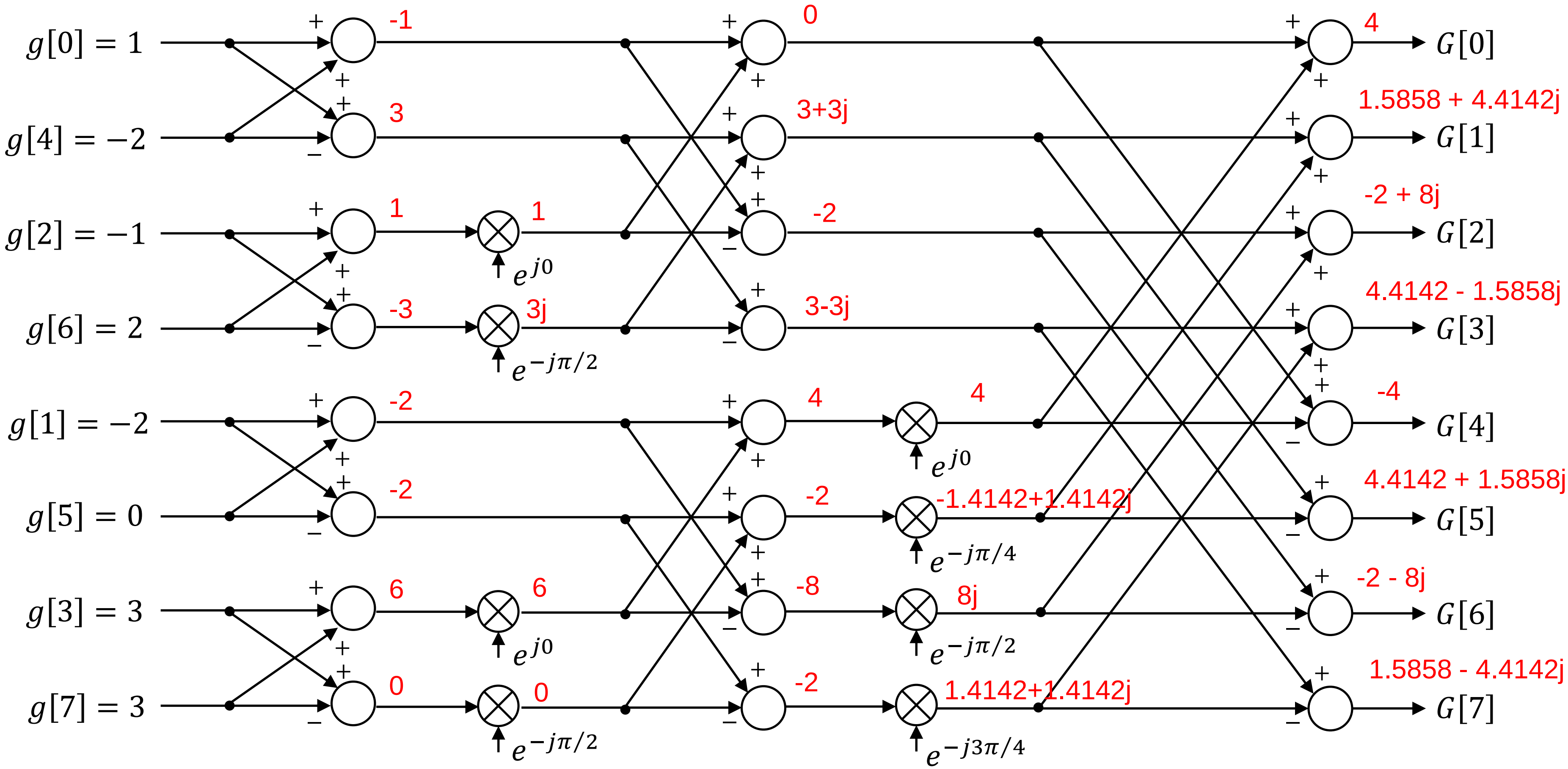
8-Point FFT



g[0]
g[4]
g[2]
g[6]
g[1]
g[5]
g[3]
g[7]
e^{j0}
e^{-jπ/2}
e^{j0}
e^{-jπ/2}
e^{j0}
e^{-jπ/4}
e^{-jπ/2}
e^{-j3π/4}
G[0]
G[1]
G[2]
G[3]
G[4]
G[5]
G[6]
G[7]

Example: 8-Point FFT

g = [1, -2, -1, 3, -2, 0 , 2, 3];



Matlab FFT function fft(.)

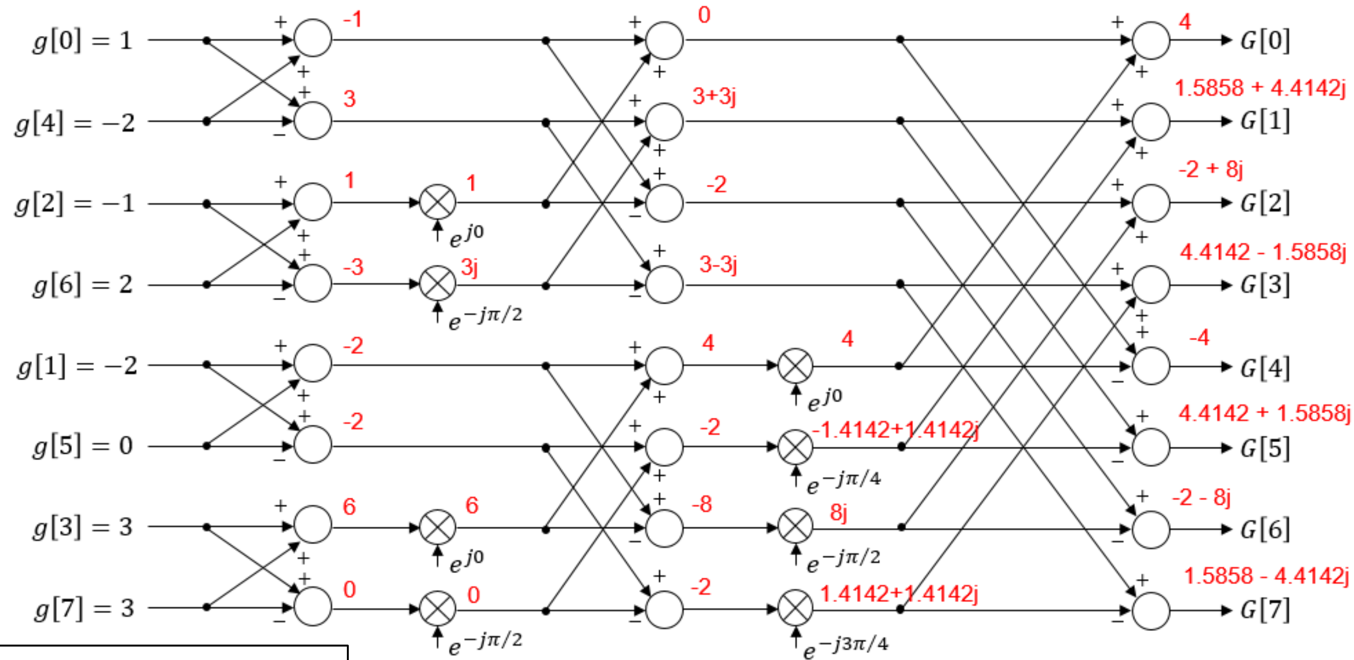
This MATLAB function returns the discrete Fourier transform (DFT) of vector x, computed with a fast Fourier transform (FFT) algorithm.

```
Y = fft(x)
Y = fft(X,n)
Y = fft(X,[],dim)
Y = fft(X,n,dim)
```

- If N is a power of 2 the FFT algorithm is applied

Matlab FFT function fft(.) -- Example

g = [1, -2, -1, 3, -2, 0 , 2, 3];




```
>> x=[1; -2; -1; 3; -2; 0 ; 2; 3];
>> fft(x)

ans =

   4.0000 + 0.0000i
   1.5858 + 4.4142i
  -2.0000 + 8.0000i
   4.4142 - 1.5858i
  -4.0000 + 0.0000i
   4.4142 + 1.5858i
  -2.0000 - 8.0000i
   1.5858 - 4.4142i
```