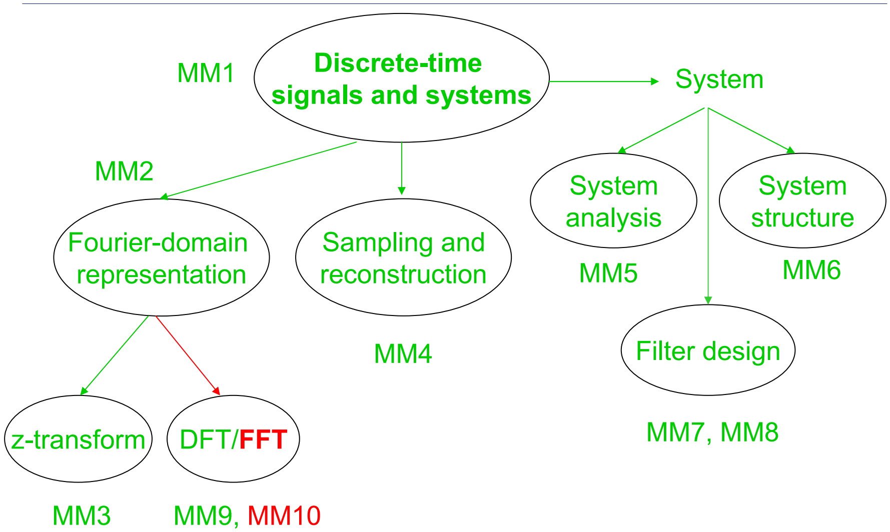
Digital Signal Processing, Fall 2005
- E-Study -

Lecture 10: Fast Fourier Transform

Zheng-Hua Tan

Department of Communication Technology
Aalborg University, Denmark
zt@kom.aau.dk

Course at a glance



Digital computation of the DFT

- The DFT of a finite-length sequence of length *N*

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

- The inverse DFT

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

- Due to the duality, focus on the DFT only.
- Use the number of arithmetic multiplications and additions as a measure of computational complexity.
- Fast Fourier transform (FFT) is a set of algorithms for the efficient and digital computation of the N-point DFT, rather than a new transform.

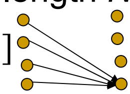
Part I: Direct computation of the DFT

- Direct computation of the DFT
- Decimation-in-time FFT algorithms
- Decimation-in-frequency FFT algorithms
- Fourier analysis of signals using the DFT

Direct computation of the DFT

- The DFT of a finite-length sequence of length *N*

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

$x[n]$  $X[k]$

- Direct computation: N^2 complex multiplications and N(N-1) complex additions
  - Compute and store (only over one period)

$$W_N^k = e^{-j(2\pi/N)k}$$
$$= \cos(2\pi k/N) + j\sin(2\pi k/N), \quad k = 0,1,\ldots,N-1$$

  - Compute the DFT using stored W_N^k and input $x[n]$

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

W_N^k and $x[n]$ may be complex

Direct computation of the DFT

- For each *k*

$$X[k] = \sum_{n=0}^{N-1} [(\mathrm{Re}\{x[n]\}\,\mathrm{Re}\{W_N^{kn}\} - (\mathrm{Im}\{x[n]\}\,\mathrm{Im}\{W_N^{kn}\})$$
$$+ j(\mathrm{Re}\{x[n]\}\,\mathrm{Im}\{W_N^{kn}\} + \mathrm{Im}\{x[n]\}\,\mathrm{Re}\{W_N^{kn}\}), \quad k = 0,1,\ldots,N-1$$

- Therefore, for each value of k, the direct computation of X[k] requires 4N real multiplications and (4N-2) real additions.
- The direct computation of the DFT requires $4N^2$ real multiplications and $N(4N-2)$ real additions.
- The efficiency can be improved by exploiting the symmetry and periodicity properties of W_N^{kn}

Symmetry and periodicity of complex exponential

- Complex conjugate symmetry

$$W_N^{k[N-n]} = W_N^{-kn} = (W_N^{kn})^* = \mathrm{Re}\{W_N^{kn}\} - j\,\mathrm{Im}\{W_N^{kn}\}$$

- Periodicity in n and k

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$$

- For example

$$\mathrm{Re}\{x[n]\}\,\mathrm{Re}\{W_N^{kn}\} + \mathrm{Re}\{x[N-n]\}\,\mathrm{Re}\{W_N^{k[N-n]}\}$$
$$= (\mathrm{Re}\{x[n]\} + \mathrm{Re}\{x[N-n]\})\,\mathrm{Re}\{W_N^{kn}\}$$

  - The number of multiplications is reduced by a factor of 2.

Part II: Decimation-in-time FFT algorithms

- Direct computation of the DFT
- Decimation-in-time FFT algorithms
- Decimation-in-frequency FFT algorithms
- Fourier analysis of signals using the DFT

FFT

- Cooley and Tukey (1965) published an algorithm for the computation of the DFT that is applicable when N is a composite number, i.e., the product of two or more integers. Later, it resulted in a number of highly efficient computational algorithms.
- The entire set of such algorithms are called the fast Fourier transform, FFT.
- FFT decomposes the computation of the DFT of a sequence of length N into successively smaller DFTs.

Decimation-in-time FFT algorithms

- Where
  - decomposition is done by decomposing the sequence into successively smaller subsequences,
  - and both the symmetry and periodicity of complex exponential $W_N^{kn} = e^{-j(2\pi/N)kn}$ are exploited.
- Consider $N = 2^v$ and separate x[n] into two (N/2)-point sequences

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

$$X[k] = \sum_{n \text{ even}} x[n]W_N^{kn} + \sum_{n \text{ odd}} x[n]W_N^{kn}$$

Decimation-in-time FFT algorithms

$$X[k] = \sum_{n \text{ even}} x[n] W_N^{kn} + \sum_{n \text{ odd}} x[n] W_N^{kn}$$

$$= \sum_{r=0}^{(N/2)-1} x[2r] W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x[2r+1] W_N^{(2r+1)k}$$

$$= \sum_{r=0}^{(N/2)-1} x[2r] (W_N^2)^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1] (W_N^2)^{rk}$$

$$= \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1] W_{N/2}^{rk}$$

$$= \underline{G[k] + W_N^k H[k], \quad k = 0,1,\ldots., N-1}$$

(only compute for $k = 0,1,\ldots, N/2-1$) due to the periodicity

Flow graph of the decimation-in-time

- Periodicity is applied, e.g. G[7]=G[3]

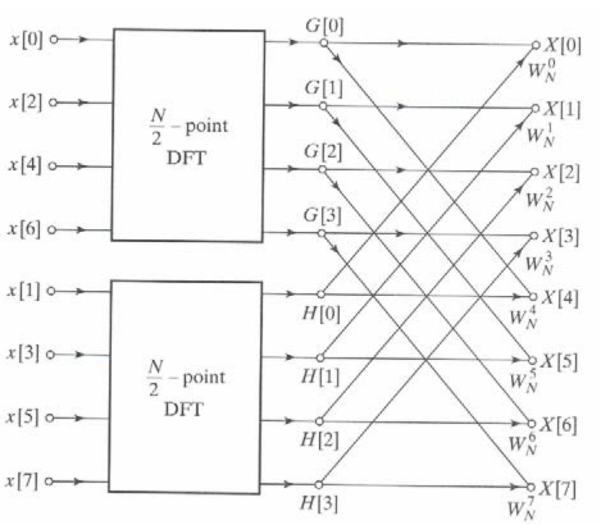


Figure 9.3 Flow graph of the decimation-in-time decomposition of an N-point DFT computation into two $(N/2)$-point DFT computations $(N = 8)$.

Decimation-in-time FFT

- Further break down

$$G[k]=\sum_{r=0}^{(N/2)-1} g[r]W_{N/2}^{rk}=\sum_{l=0}^{(N/4)-1} g[2l]W_{N/2}^{2lk}+\sum_{l=0}^{(N/4)-1} g[2l+1]W_{N/2}^{(2l+1)k}$$

$$=\sum_{l=0}^{(N/4)-1} g[2l]W_{N/4}^{lk}+W_{N/2}^{k}\sum_{l=0}^{(N/4)-1} g[2l+1]W_{N/4}^{lk}$$

$$H[k]=\sum_{l=0}^{(N/4)-1} h[2l]W_{N/4}^{lk}+W_{N/2}^{k}\sum_{l=0}^{(N/4)-1} h[2l+1]W_{N/4}^{lk}$$

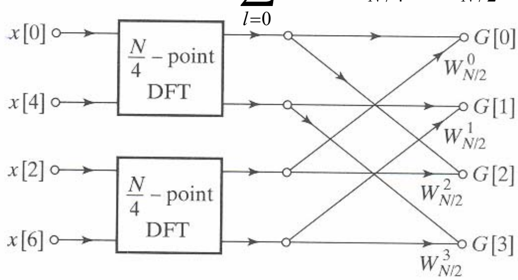


Figure 9.4 Flow graph of the decimation-in-time decomposition of an $(N/2)$-point DFT computation into two $(N/4)$-point DFT computations $(N=8)$.

Combination of Fig. 9.3 and 9.4

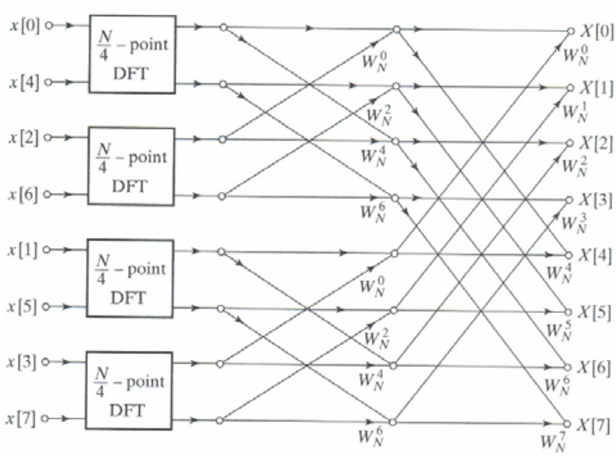


Figure 9.5 Result of substituting the structure of Figure 9.4 into Figure 9.3.

2-point DFT

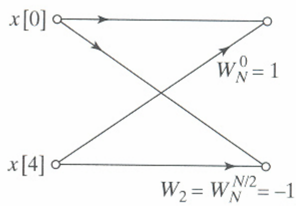
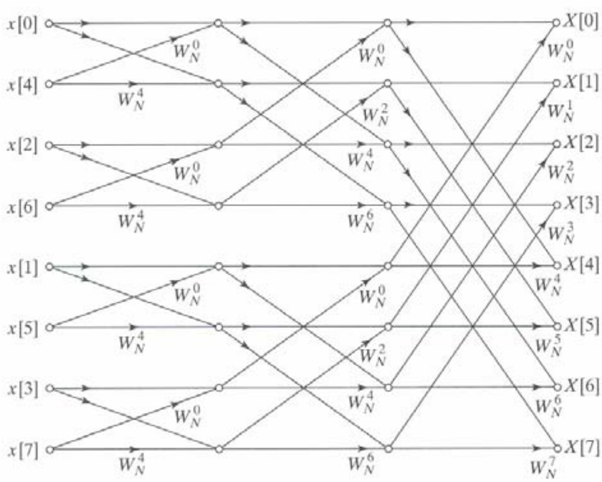


Figure 9.6 Flow graph of a 2-point DFT.

Flow graph



$\log_2 N$ stages and each stage has N complex multiplications and N complex additions .

In total, $N \log_2 N$ complex multiplications and additions

e.g.

$N = 2^{10} = 1024$

$N^2 = 1{,}048{,}576$

$N \log_2 N = 10{,}240$

A reduction of 2 orders!

Figure 9.7 Flow graph of complete decimation-in-time decomposition of an 8-point DFT computation.

Flow graph of butterfly computation

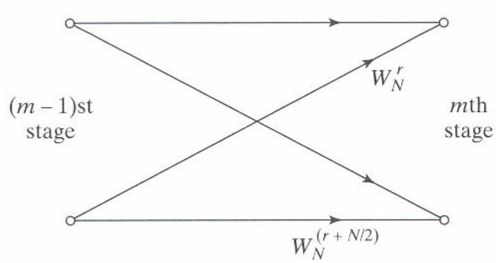


Figure 9.8 Flow graph of basic butterfly computation in Figure 9.7.

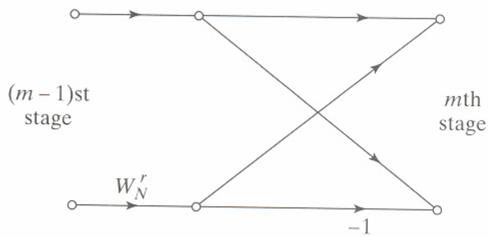


Figure 9.9 Flow graph of simplified butterfly computation requiring only one complex multiplication.

Flow graph

- The number of complex multiplications are reduced by a factor of 2 over the number in Fig. 9.7.

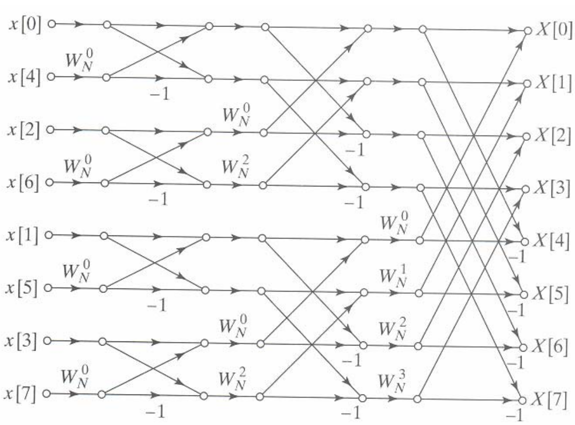


Figure 9.10 Flow graph of 8-point DFT using the butterfly computation of Figure 9.9.

Part III: Decimation-in-frequency FFT algorithms

- Direct computation of the DFT
- Decimation-in-time FFT algorithms
- Decimation-in-frequency FFT algorithms
- Fourier analysis of signals using the DFT

Decimation-in-frequency FFT algorithms

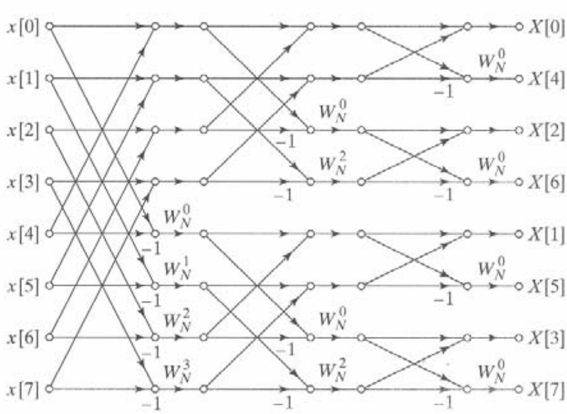


Figure 9.20 Flow graph of complete decimation-in-frequency decomposition of an 8-point DFT computation.

Part IV: Fourier analysis of signals using the DFT

- Direct computation of the DFT
- Decimation-in-time FFT algorithms
- Decimation-in-frequency FFT algorithms
- Fourier analysis of signals using the DFT

Fourier analysis of signals using the DFT

- Finite-duration requirement of DFT → windowing

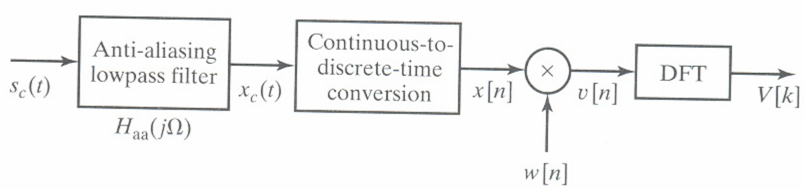
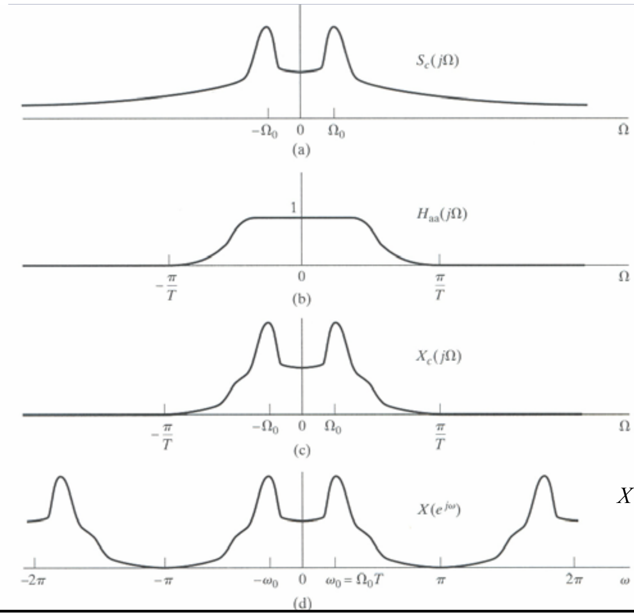


Figure 10.1 Processing steps in the discrete-time Fourier analysis of a continuous-time signal.

Fourier analysis of signals using the DFT



Tapers off but is not band-limited.

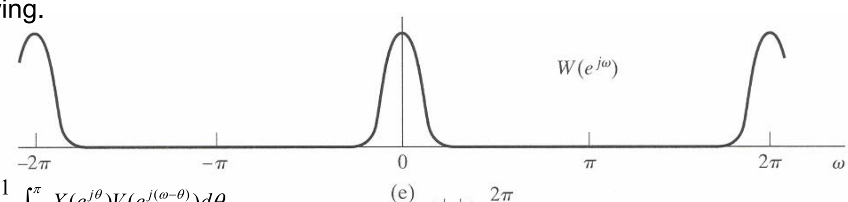
Not ideal.

Low-pass filtered and modified.

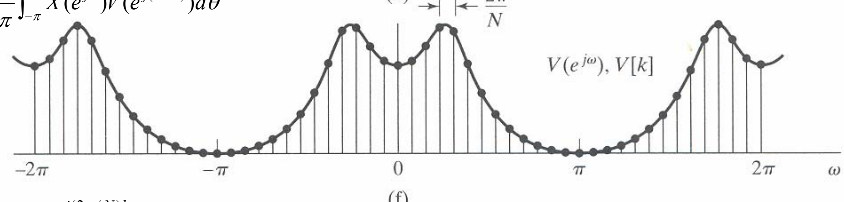
$$X(e^{j\omega})=\frac{1}{T}\sum_{r=-\infty}^{\infty}X_c\left(j\frac{\omega}{T}+j\frac{2\pi r}{T}\right)$$

Fourier analysis of signals using the DFT

Windowing.



$$V(e^{j\omega})=\frac{1}{2\pi}\int_{-\pi}^{\pi}X(e^{j\theta})V(e^{j(\omega-\theta)})d\theta$$



$$V(k)=\sum_{n=0}^{N-1}v[n]e^{-j(2\pi/N)kn},\quad k=0,1,\ldots,N-1$$

$$=V(e^{j\omega})|_{\omega=2\pi k/N}$$

Figure 10.2 Illustration of the Fourier transforms of the system of Figure 10.1. (a) Fourier transform of continuous-time input signal. (b) Frequency response of antialiasing filter. (c) Fourier transform of output of antialiasing filter. (d) Fourier transform of sampled signal. (e) Fourier transform of window sequence. (f) Fourier transform of windowed signal segment and frequency samples obtained using DFT samples.

CTiF

Effect of Windowing on Fourier analysis

A rectangular window of length 64.

$$s_c(t) = A_0 \cos(\Omega_0 t + \theta_0) + A_1 \cos(\Omega_1 t + \theta_1)$$
$$x[n] = A_0 \cos(\omega_0 n + \theta_0) + A_1 \cos(\omega_1 n + \theta_1)$$
$$v[n] = A_0 w[n]\cos(\omega_0 n + \theta_0) + A_1 w[n]\cos(\omega_1 n + \theta_1)$$
$$V(e^{j\omega}) = \frac{A_0}{2} e^{j\theta_0} W(e^{j(\omega-\omega_0)}) + \frac{A_0}{2} e^{-j\theta_0} W(e^{j(\omega+\omega_0)}) + \frac{A_1}{2} e^{j\theta_1} W(e^{j(\omega-\omega_1)}) + \frac{A_1}{2} e^{-j\theta_1} W(e^{j(\omega+\omega_1)})$$

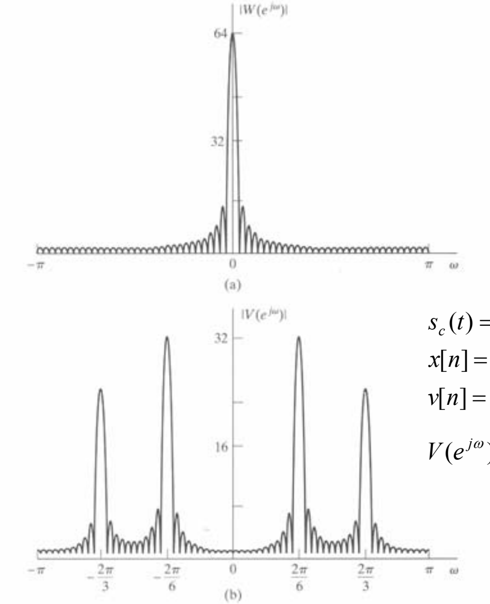
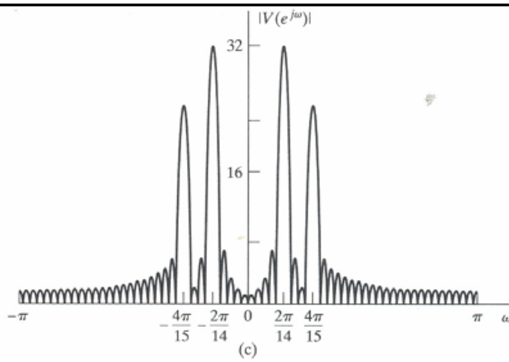
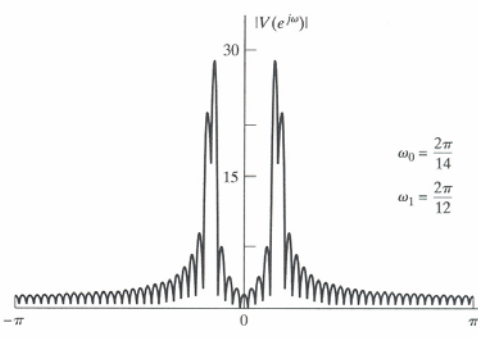


Figure 10.3 Illustration of Fourier analysis of windowed cosines with a rectangular window. (a) Fourier transform of window. (b)–(e) Fourier transform of windowed cosines as $\Omega_1 - \Omega_0$ becomes progressively smaller. (b) $\Omega_0 = (2\pi/6) \times 10^4$, $\Omega_1 = (2\pi/3) \times 10^4$.



The DTFT of a sinusoidal signal is a pair of impulses. Windowing broadens the impulses and reduces the distinction of signals that are close in frequency



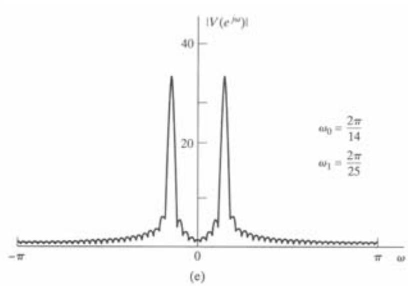
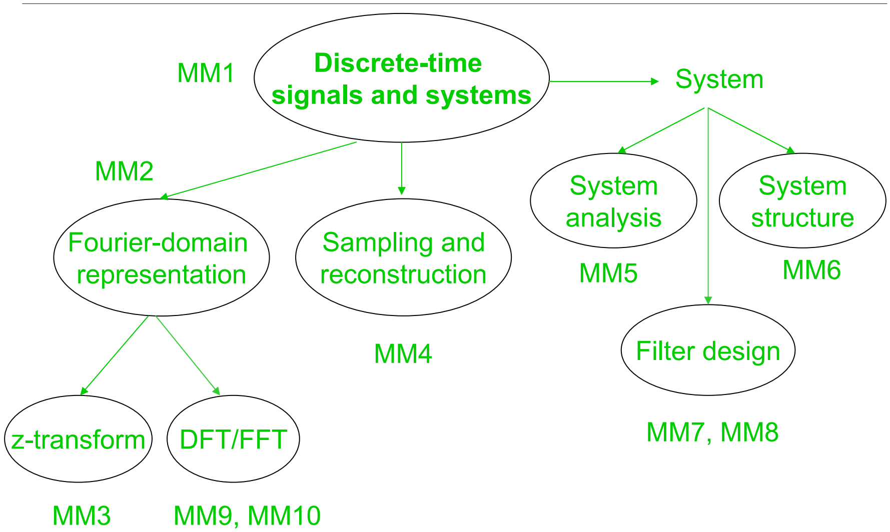


Figure 10.3 (*continued*) (c) $\Omega_0 = (2\pi/14) \times 10^4$, $\Omega_1 = (4\pi/15) \times 10^4$. (d) $\Omega_0 = (2\pi/14) \times 10^4$, $\Omega_1 = (2\pi/12) \times 10^4$. (e) $\Omega_0 = (2\pi/14) \times 10^4$, $\Omega_1 = (4\pi/25) \times 10^4$.

Summary

- Direct computation of the DFT
- Decimation-in-time FFT algorithms
- Decimation-in-frequency FFT algorithms
- Fourier analysis of signals using the DFT

Course at a glance



The end.

Thanks for your attention!