Discrete Fourier Transform (DFT)

Richard Heusdens

Frequency-Domain Sampling

Recall that the spectrum of the discrete-time signal x is given by

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$

where the signal x can be recovered from its spectrum by the inverse Fourier transform

$$x(n) = \frac{1}{2\pi}\int_0^{2\pi} X(\omega)e^{j\omega n}d\omega$$

Frequency-Domain Sampling

Frequency analysis of discrete-time signals is usually and most conveniently performed on a digital signal processor:

- we convert the discrete-time signal x to an equivalent frequency-domain representation
- such a representation is given by the Fourier transform $X(\omega)$ of x
- however, $X(\omega)$ is a continuous function of frequency and therefore not a convenient representation of x

We will consider the representation of x by samples of its spectrum $X(\omega)$, which will lead to the *discrete Fourier transform* (DFT)

Frequency-Domain Sampling

Recall that discrete-time aperiodic finite-energy signals have continuous 2π-periodic spectra

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$

Taking N equally spaced samples of $X(\omega)$ on the fundamental frequency range $[0, 2\pi)$ yields

$$X\left(\frac{2\pi}{N}k\right) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\frac{2\pi}{N}kn}, \qquad k = 0, \ldots, N-1$$

Frequency-Domain Sampling

What do these frequency samples tell us about x?

$$X\left(\frac{2\pi}{N}k\right) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\frac{2\pi}{N}kn}$$

$$= \sum_{m=-\infty}^{\infty} \sum_{n=mN}^{(m+1)N-1} x(n) e^{-j\frac{2\pi}{N}kn}$$

$$n \rightarrow n + mN$$

$$= \sum_{m=-\infty}^{\infty} \sum_{n=0}^{N-1} x(n+mN) e^{-j\frac{2\pi}{N}kn}$$

$$= \sum_{n=0}^{N-1} \left(\sum_{m=-\infty}^{\infty} x(n+mN) \right) e^{-j\frac{2\pi}{N}kn}$$

Frequency-Domain Sampling

The signal

$$x_p(n) = \sum_{m=-\infty}^{\infty} x(n + mN)$$

is an N-periodic repetition of x.

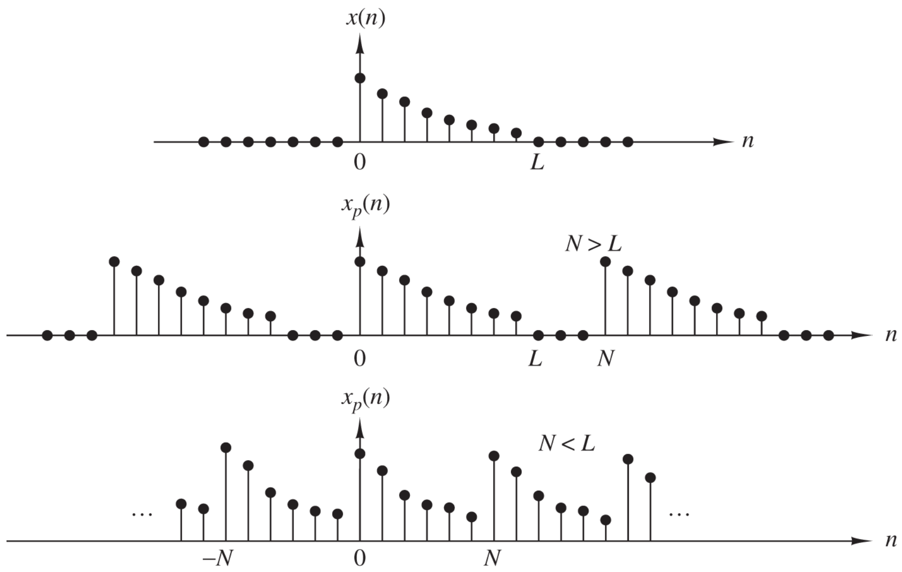


Figure 7.1.2 Aperiodic sequence $x(n)$ of length L and its periodic extension for $N \geq L$ (no aliasing) and $N < L$ (aliasing).

Frequency-Domain Sampling

As a consequence, it has a Fourier series expansion

$$x_p(n) = \sum_{k=0}^{N-1} c_k e^{j\frac{2\pi}{N}kn}$$

with Fourier coefficients

$$c_k = \frac{1}{N}\sum_{n=0}^{N-1} x_p(n)e^{-j\frac{2\pi}{N}kn} = \frac{1}{N}X\left(\frac{2\pi}{N}k\right)$$

As a consequence, we have

$$x_p(n) = \frac{1}{N}\sum_{k=0}^{N-1} X\left(\frac{2\pi}{N}k\right)e^{j\frac{2\pi}{N}kn}$$

Frequency-Domain Sampling

This relation shows how to reconstruct x_p from the samples of $X(\omega)$. However, it does not imply that we can recover $X(\omega)$ (and thus x) from its samples. To accomplish this, we need to consider the relation between x_p and x

Since x_p is the periodic extension of x, x can be recovered if it has finite support of L samples less than N $(N \geq L)$

On the other hand, if $N < L$, it is not possible to recover x from its periodic extension due to *time-domain aliasing*

Frequency-Domain Sampling

Assume $N \geq L$. As in the continuous-time case, we can express the spectrum $X(\omega)$ in terms of its samples $X(\frac{2\pi}{N}k)$ using an interpolation formula

$$X(\omega) = \sum_{n=0}^{N-1} \underbrace{\left(\frac{1}{N} \sum_{k=0}^{N-1} X\left(\frac{2\pi}{N}k\right) e^{j\frac{2\pi}{N}kn} \right)}_{x(n)} e^{-j\omega n}$$

$$= \sum_{k=0}^{N-1} X\left(\frac{2\pi}{N}k\right) \left(\frac{1}{N} \sum_{n=0}^{N-1} e^{-j(\omega - \frac{2\pi}{N}k)n} \right)$$

Frequency-Domain Sampling

Since

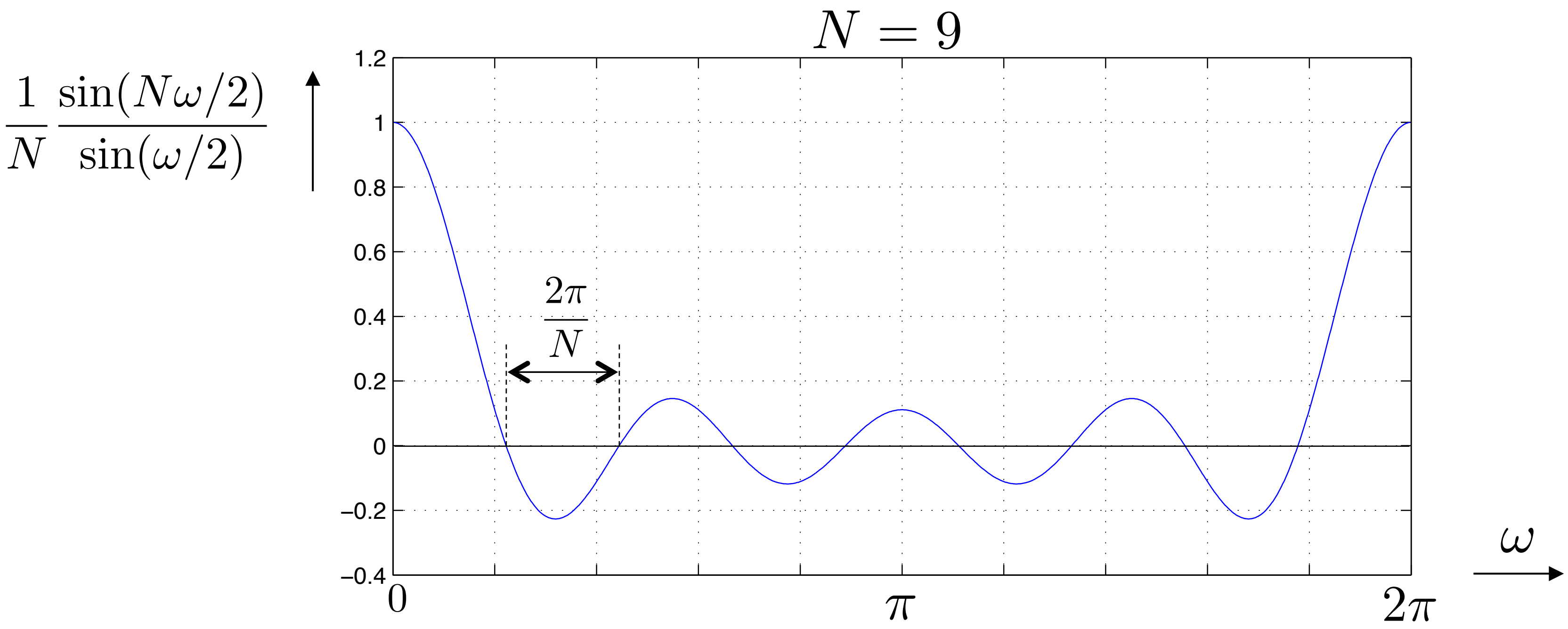
$$\sum_{n=0}^{N-1} e^{-j\omega n} = \frac{1-e^{-j\omega N}}{1-e^{-j\omega}} = e^{-j\omega\frac{N-1}{2}} \frac{\sin(\frac{\omega N}{2})}{\sin(\frac{\omega}{2})} = G(\omega)$$

we conclude that

$$X(\omega) = \frac{1}{N} \sum_{k=0}^{N-1} X\left(\frac{2\pi}{N}k\right) G\left(\omega - \frac{2\pi}{N}k\right)$$

The function $G(\omega)$ is not the familiar sinc-function, but a periodic counterpart of it

Frequency-Domain Sampling



Clearly, $\frac{1}{N}G(\omega)$ is an interpolation function since

$$\frac{1}{N}G\left(\frac{2\pi}{N}k\right) = \begin{cases} 1, & k = 0 \\ 0, & k = 1, \ldots, N-1 \end{cases}$$

Frequency-Domain Sampling

Similarly to what we did with sampling continuous-time signals, we can evaluate the time-domain description of interpolating the spectral samples. If

$$X(\omega) = \frac{1}{N}\sum_{k=0}^{N-1} X\left(\frac{2\pi}{N}k\right) G\left(\omega - \frac{2\pi}{N}k\right)$$

then

$$x(n) = g(n)x_p(n)$$

with

$$g(n) = \begin{cases} 1, & \text{for } n = 0, \dots, N-1 \\ 0, & \text{otherwise} \end{cases}$$

Hence, g is a rectangular window taking N samples out of x_p

Frequency-Domain Sampling

Example ($N < L$): $x(n) = a^n u(n),\ a < 1$. The Fourier transform of x is given by

$$X(\omega) = \frac{1}{1 - ae^{-j\omega}} \quad \Rightarrow \quad X\left(\frac{2\pi}{N}k\right) = \frac{1}{1 - ae^{-j\frac{2\pi}{N}k}}$$

The periodic extension of x is given by

$$x_p(n) = \sum_{m=-\infty}^{\infty} x(n + mN) = \sum_{m=0}^{\infty} a^{n+mN} = \frac{a^n}{1 - a^N}$$

The factor $1/(1 - a^N)$ represents the effect of aliasing which tends to zero as $N \to \infty$

Frequency-Domain Sampling

Ideal interpolation of the spectral samples corresponds to selecting the first N-samples of x_p

$$\hat{x}(n) = \begin{cases} x_p(n), & \text{for } n = 0, \ldots, N-1 \\ 0, & \text{otherwise} \end{cases}$$

Its Fourier transform is given by

$$\hat{X}(\omega) = \sum_{n=0}^{N-1} x_p(n) e^{-j\omega n} = \frac{1}{1-a^N} \frac{1-a^N e^{-j\omega N}}{1-ae^{-j\omega}}$$

Note that $\hat{X}(\omega) \neq X(\omega)$, except for sample values at $\omega_k = \frac{2\pi}{N} k$

Frequency-Domain Sampling

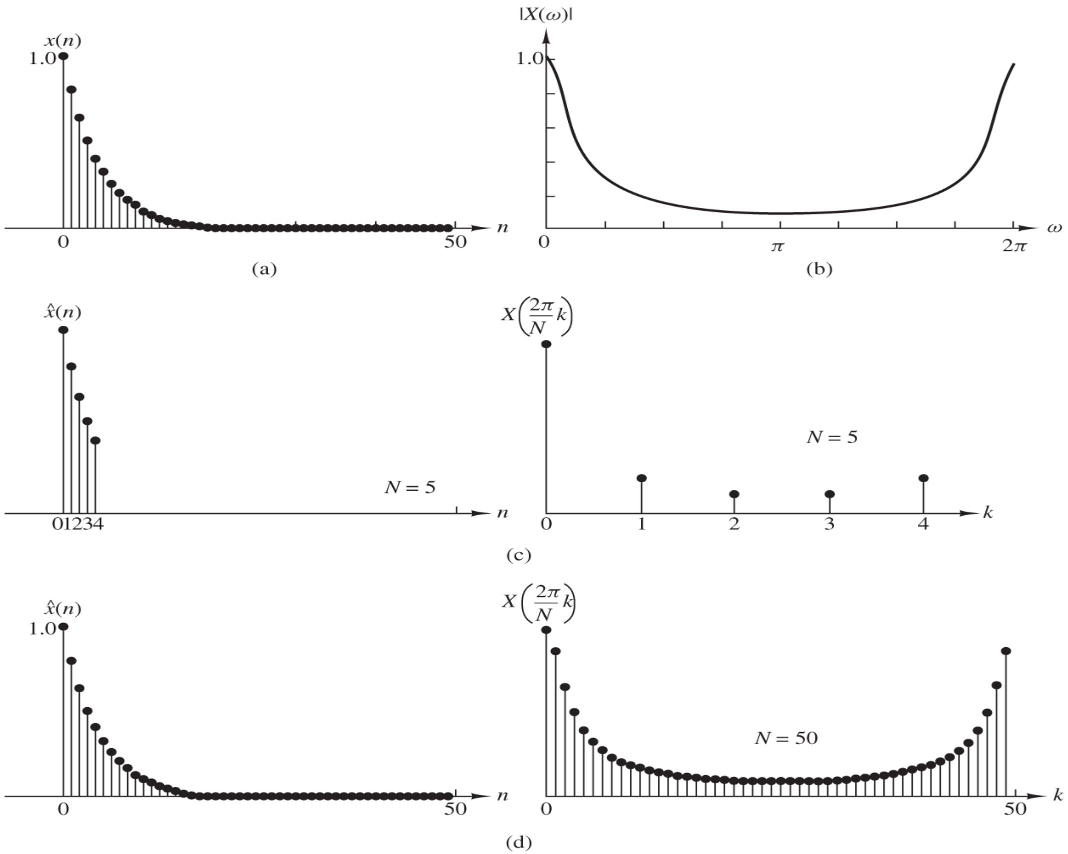


Figure 7.1.4 (a) Plot of sequence $x(n) = (0.8)^n u(n)$; (b) its Fourier transform (magnitude only); (c) effect of aliasing with $N = 5$; (d) reduced effect of aliasing with $N = 50$.

Discrete Fourier Transform

In summary, a finite support signal x of length N can be represented by N samples of its continuous-frequency spectrum $X(\omega)$. These samples uniquely determined x

Discrete Fourier transform (DFT):

DFT: $$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}, \quad k = 0, \ldots, N-1$$

IDFT: $$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn}, \quad n = 0, \ldots, N-1$$

Discrete Fourier Transform

Let $W_N = e^{j\frac{2\pi}{N}}$. We have

$$\begin{pmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \cdots & W_N^{-(N-1)} \\ 1 & W_N^{-2} & W_N^{-4} & \cdots & W_N^{-2(N-1)} \\ \vdots & & \vdots & \ddots & \vdots \\ 1 & W_N^{-(N-1)} & W_N^{-2(N-1)} & \cdots & W_N^{-(N-1)(N-1)}s \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \cdots & W_N^{1} \\ 1 & W_N^{-2} & W_N^{-4} & \cdots & W_N^{2} \\ \vdots & & \vdots & \ddots & \vdots \\ 1 & W_N^{1} & W_N^{2} & \cdots & W_N^{N-1} \end{pmatrix}}_{\text{matrix } F_N} \begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{pmatrix}$$

Discrete Fourier Transform

The DFT matrix F_N is unitairy:

$$F_N F_N^* = F_N^* F_N = NI$$

Proof: the (i, j)th-element of the product $F_N F_N^*$ is given by

$$(F_N F_N^*)_{i,j} = \sum_{n=0}^{N-1} W_N^{-in} W_N^{jn} = \sum_{n=0}^{N-1} W^{(j-i)n} = \begin{cases} N, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \vdots & & \vdots & \ddots & \vdots \\ 1 & W_N^{-i} & W_N^{-2i} & \cdots & W_N^{-(N-1)i} \\ \vdots & & \vdots & \ddots & \vdots \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{N-1} \end{pmatrix} \begin{pmatrix} 1 & \cdots & 1 & \cdots & 1 \\ 1 & \cdots & W_N^j & \cdots & W_N^{-1} \\ 1 & \cdots & W_N^{2j} & \cdots & W_N^{-2} \\ \vdots & & \vdots & \ddots & \vdots \\ 1 & \cdots & W_N^{(N-1)j} & \cdots & W_N^{-(N-1)} \end{pmatrix} = NI$$

Discrete Fourier Transform

Since $F_n F_N^* = NI$, we conclude that the inverse DFT is given by :

$$F_N^{-1} = \frac{1}{N} F_N^* = \frac{1}{N} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{-1} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{-2} \\ \vdots & & \vdots & \ddots & \vdots \\ 1 & W_N^{-1} & W_N^{-2} & \cdots & W_N^{-(N-1)} \end{pmatrix}$$

Hence,

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn}, \quad n = 0, \ldots, N-1$$

Frequency-Domain Sampling

Some facts:

- ☺ the DFT is a powerful computational tool for performing frequency analysis of discrete-time signals
- ☺ there exists a fast implementation of the DFT, the fast Fourier transform (FFT).
- ☹ we know from Fourier theory that pointwise multiplication in the frequency domain transforms into (linear) convolution in the time domain and vice versa. The DFT, however, transforms pointwise multiplication into *cyclic* convolution, rather than *linear* convolution
- ☺ linear convolution can be implemented using cyclic convolution

Fast Fourier Transform

A direct computation of an N-points DFT requires N^2 multiplications. However, we have

N-points DFT

$$
\begin{aligned}
X(k) &= \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn} \\
&= \sum_{n=0}^{N/2-1} x(2n)e^{-j\frac{2\pi}{N}2nk} + \sum_{n=0}^{N/2-1} x(2n+1)e^{-j\frac{2\pi}{N}(2n+1)k} \\
&= \sum_{n=0}^{N/2-1} x(2n)e^{-j\frac{2\pi}{N}2nk} + e^{-j\frac{2\pi}{N}k}\sum_{n=0}^{N/2-1} x(2n+1)e^{-j\frac{2\pi}{N}2nk} \\
&= \sum_{n=0}^{N/2-1} x(2n)e^{-j\frac{2\pi}{N/2}nk} + e^{-j\frac{2\pi}{N}k}\sum_{n=0}^{N/2-1} x(2n+1)e^{-j\frac{2\pi}{N/2}nk}
\end{aligned}
$$

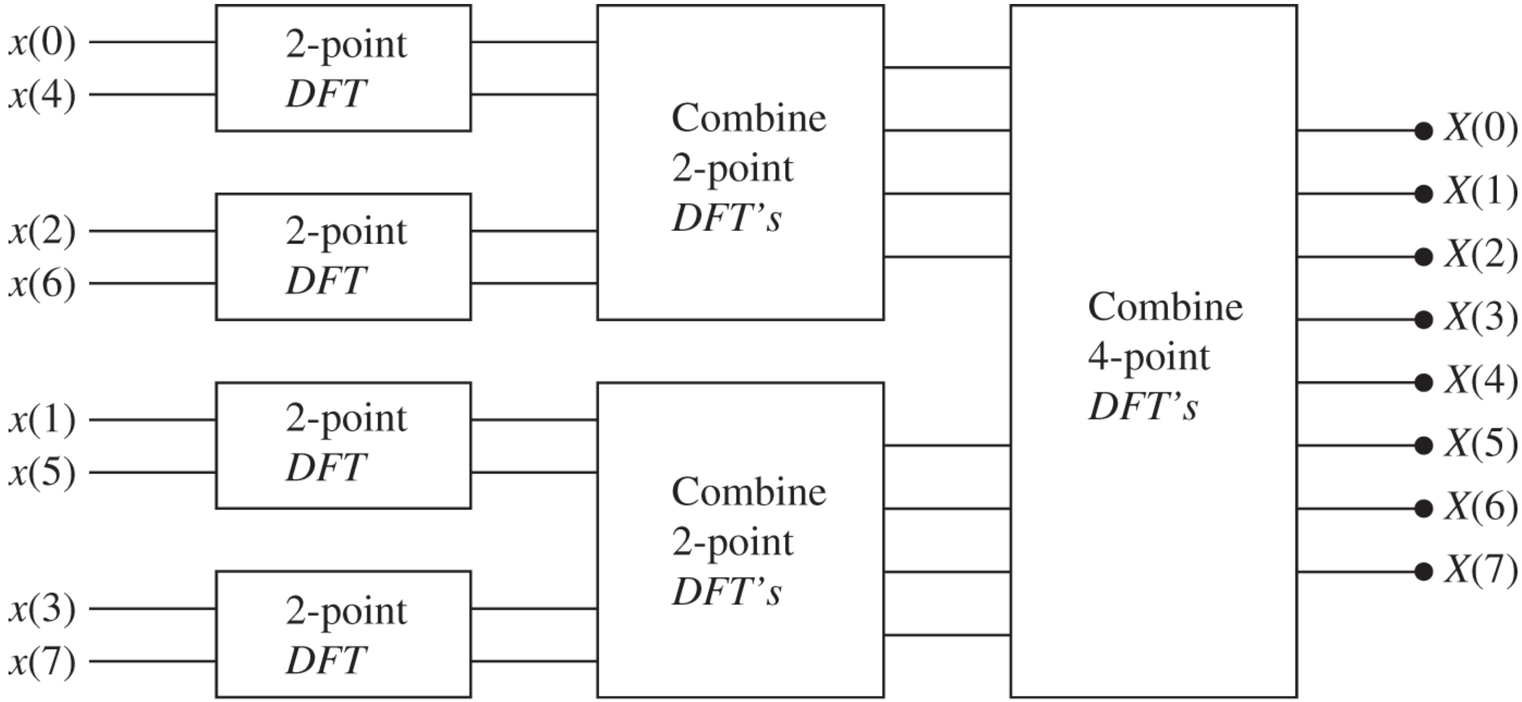
$N/2$-points DFT

$N/2$-points DFT

TU Delft

Fast Fourier Transform

Computational complexity $\mathcal{O}(N \log N)$ for the FFT versus $\mathcal{O}(N^2)$ for the DFT

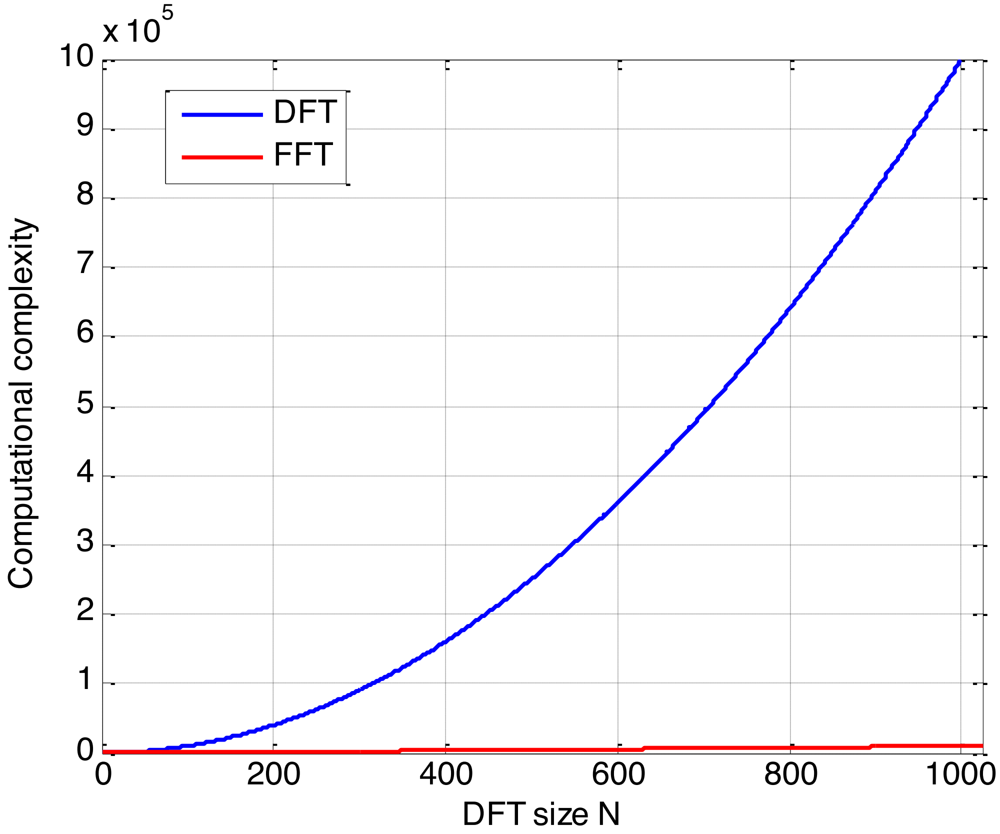


Fast Fourier Transform

Example: $N = 1024$

DFT: 1,000,000 mults/adds

FFT: 10,000 mults/adds



Radix-2 FFT Algorithm

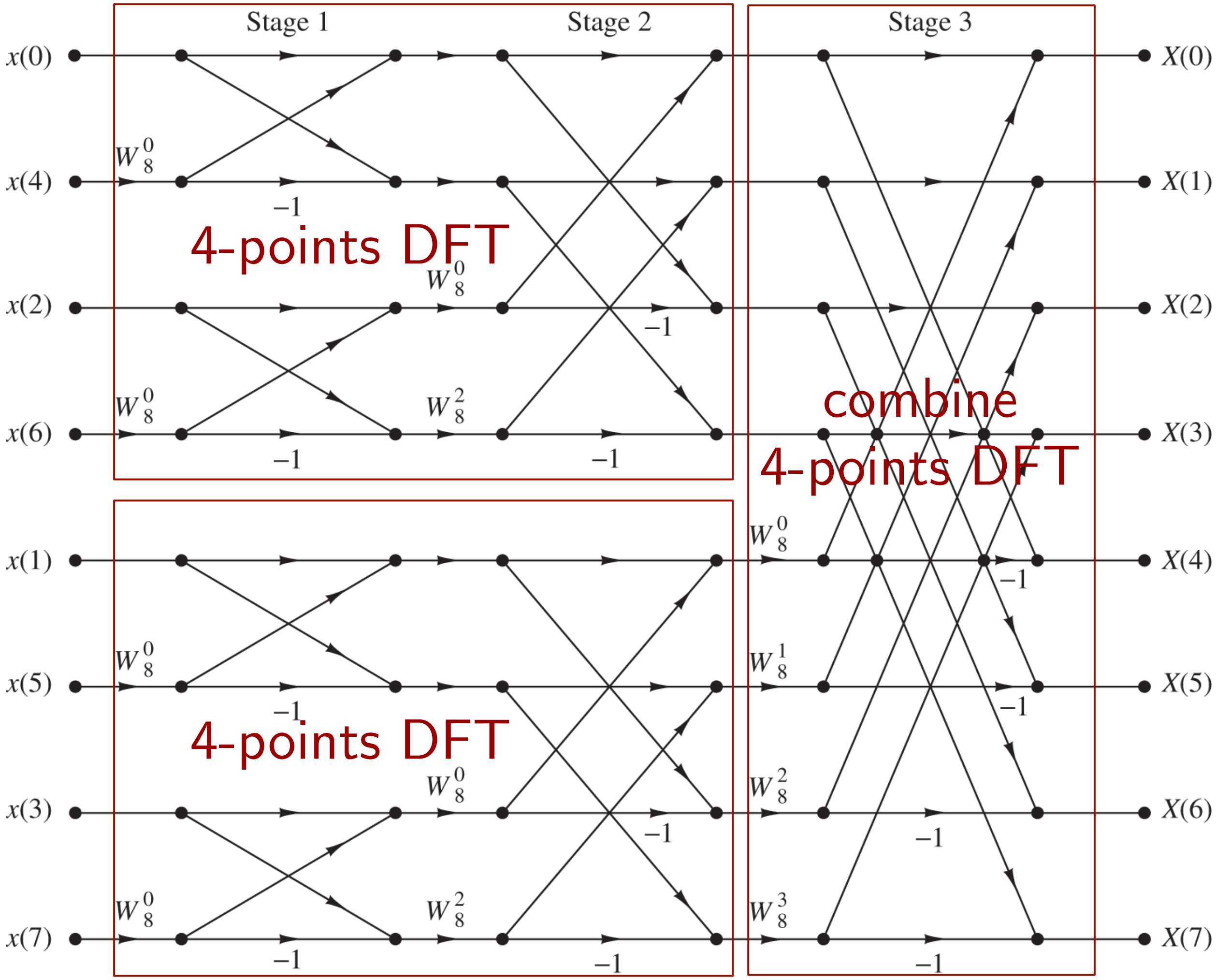
We have

$$X(k) = X_1(k) + W_N^k X_2(k), \quad k = 0, \ldots, N-1$$

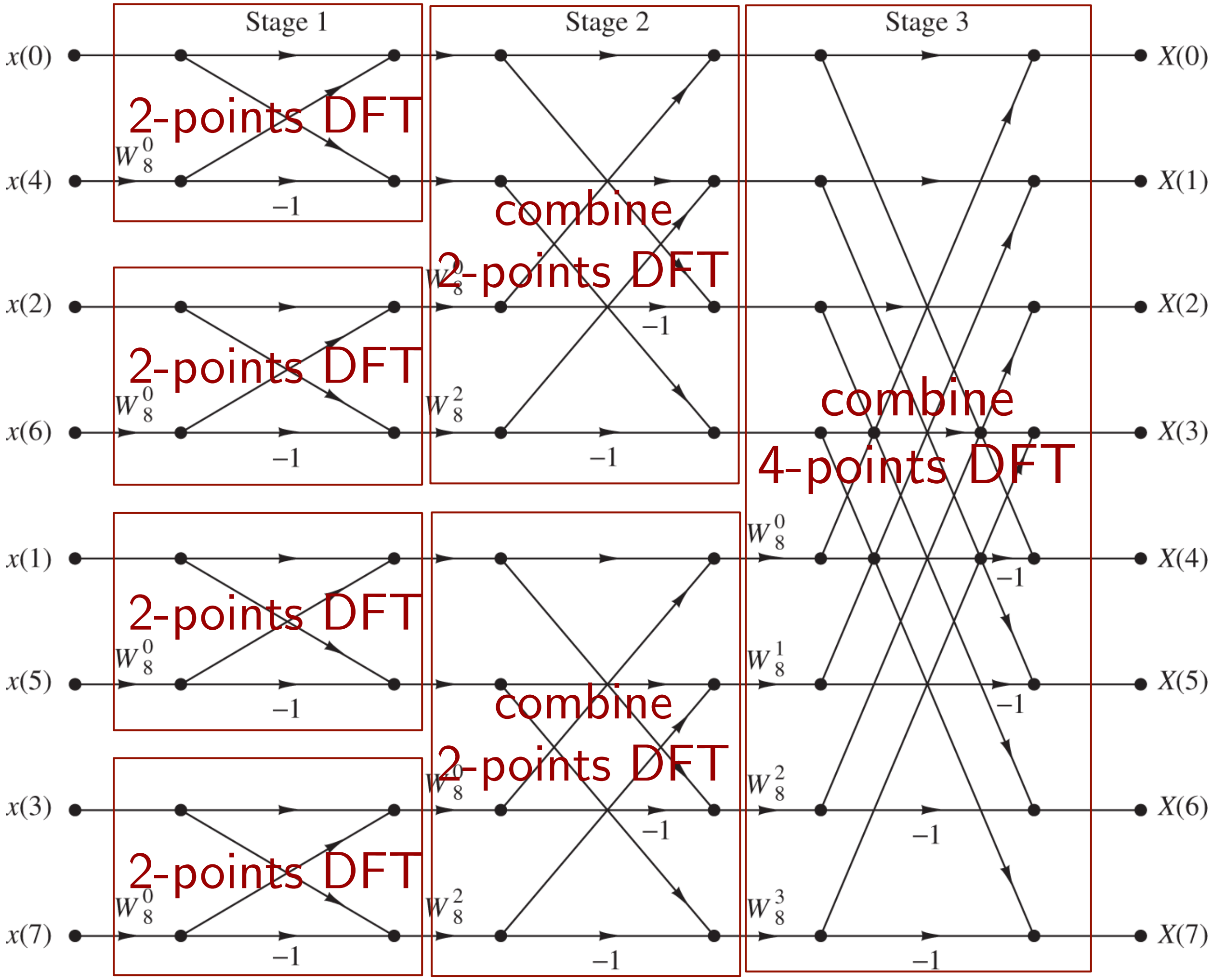
where X_1 and X_2 are the $N/2$-points DFTs of the sequences $x(2n)$ and $x(2n+1)$, $n = 0, \ldots, N/2$, and are, therefore, $N/2$ periodic. Moreover, we have that $W_N^{k+N/2} = -W_N^k$. With this, we have

$$X(k) = X_1(k) + W_N^k X_2(k), \quad k = 0, \ldots, \frac{N}{2}$$
$$X\left(k + \frac{N}{2}\right) = X_1(k) - W_N^k X_2(k), \quad k = 0, \ldots, \frac{N}{2}$$

Radix-2 FFT Algorithm



Radix-2 FFT Algorithm



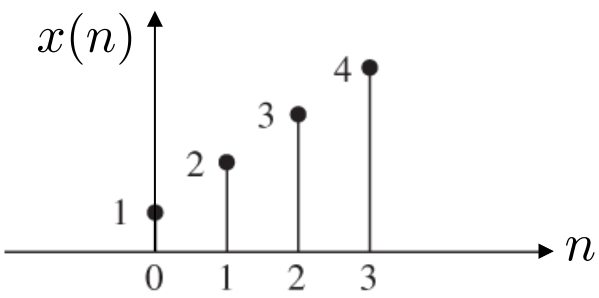
Discrete Fourier Transform

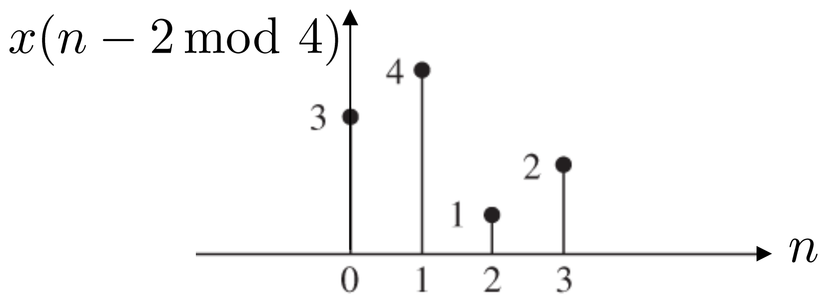
Proposition: $$X \cdot Y \overset{DFT}{\longleftrightarrow} x \circledast_N y$$

where

$$(x \circledast_N y)(n) = \sum_{m=0}^{N-1} x(m) y(n - m \mod N)$$

Hence, the product of two DFTs results in the DFT of *cyclicly* convolved sequences, *not* linearly convolved sequences!





TU Delft

Discrete Fourier Transform

Proof:

$$\frac{1}{N}\sum_{k=0}^{N-1}\left(\sum_{m=0}^{N-1} x(m)e^{-j\frac{2\pi}{N}km}\right)\left(\sum_{l=0}^{N-1} y(l)e^{-j\frac{2\pi}{N}kl}\right)e^{j\frac{2\pi}{N}kn}$$

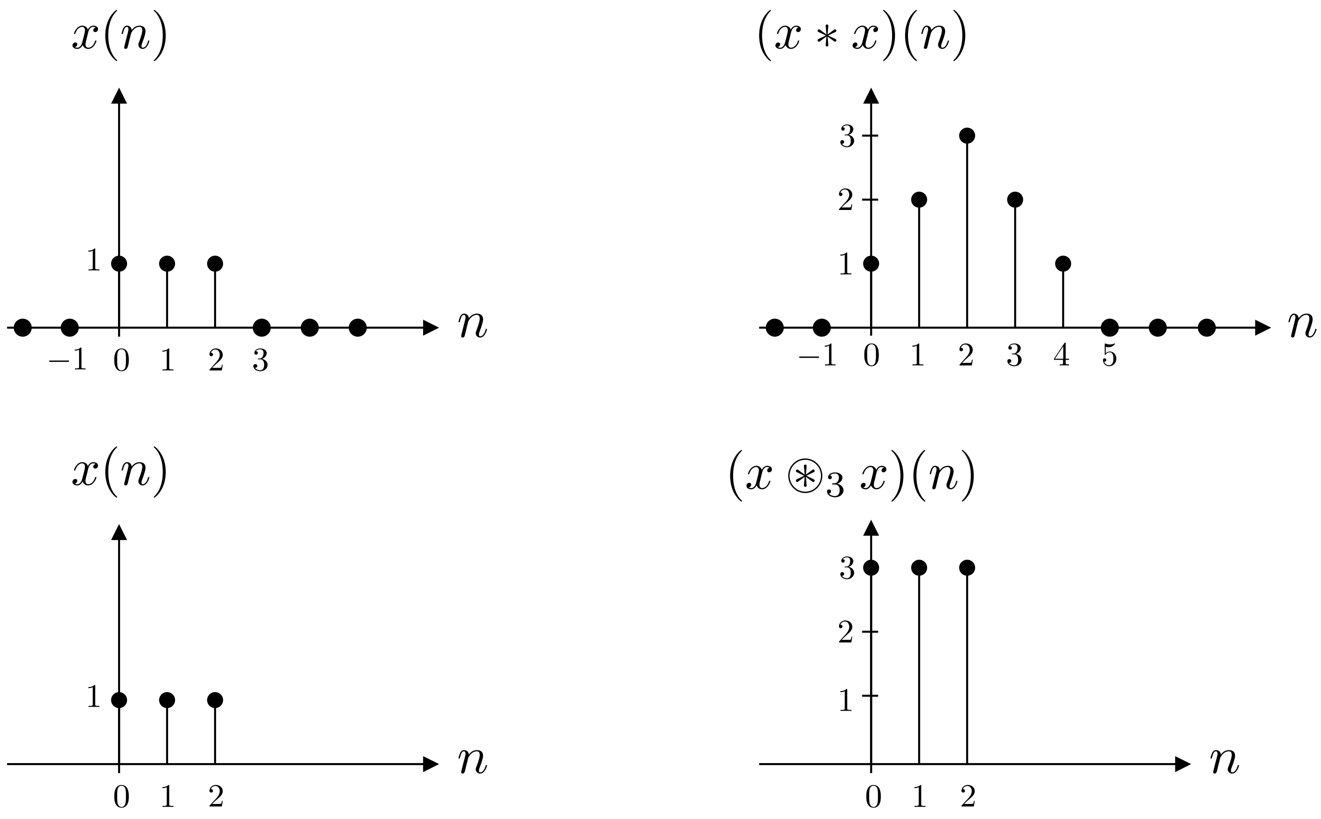
$$= \frac{1}{N}\sum_{m=0}^{N-1} x(m)\sum_{l=0}^{N-1} y(l)\underbrace{\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}k(n-l-m)}}_{=\begin{cases} N, & l=n-m \mod N \\ 0, & \text{otherwise}\end{cases}}$$

$$= \sum_{m=0}^{N-1} x(m)y(n-m \mod N)$$

$\square$

Discrete Fourier Transform

Example:



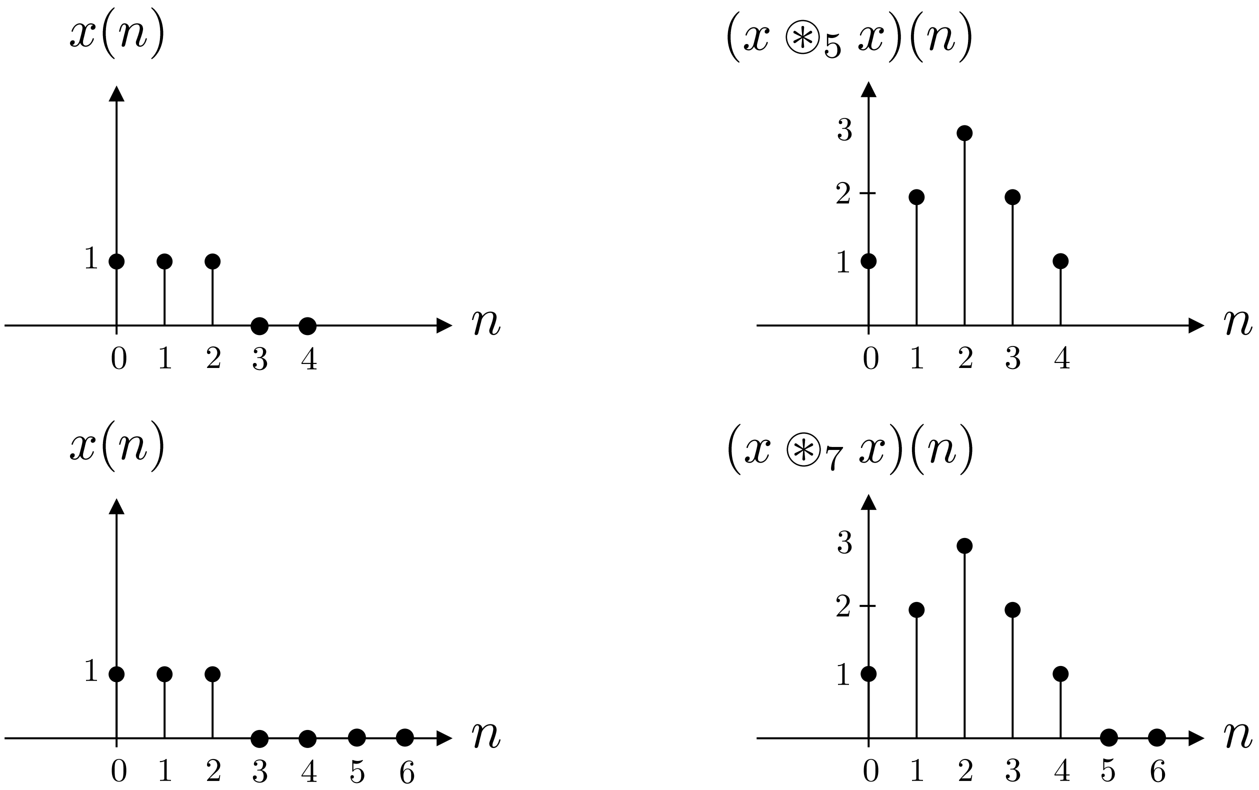
Discrete Fourier Transform

We can, however, implement a linear convolution using a cyclic convolution by zero-padding:

- let N_x and N_y denote the length of the sequences to be convolved
- length of the linearly convolved sequence is $N_x + N_y - 1$
- zero-pad both x and y to a length of *at least* $N_x + N_y - 1$
- the linear convolution is then given by the first $N_x + N_y - 1$ samples of $x \circledast y$

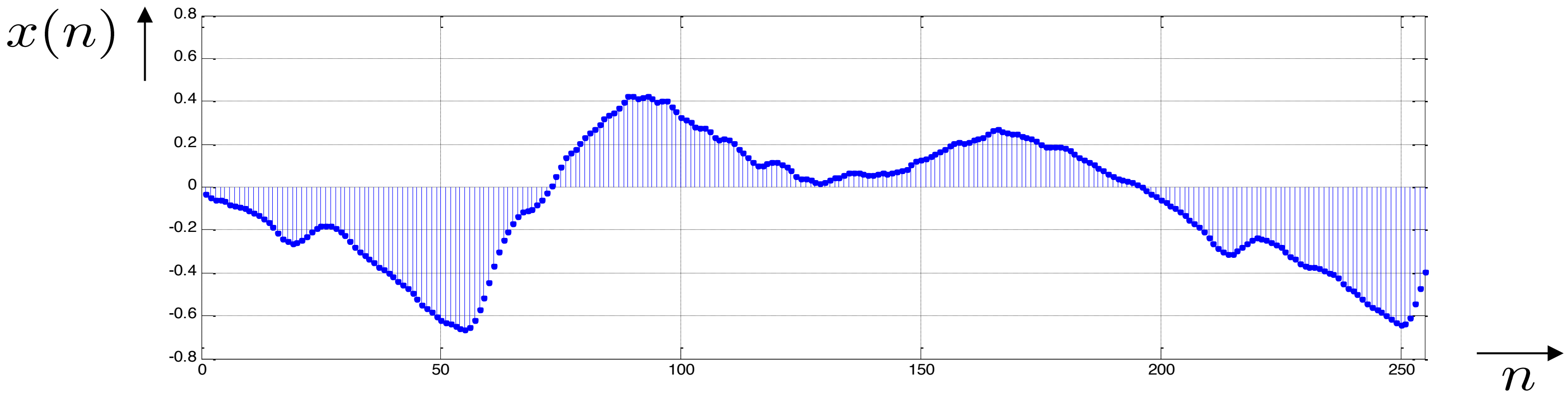
Discrete Fourier Transform

Example:



Overlap-Add Method

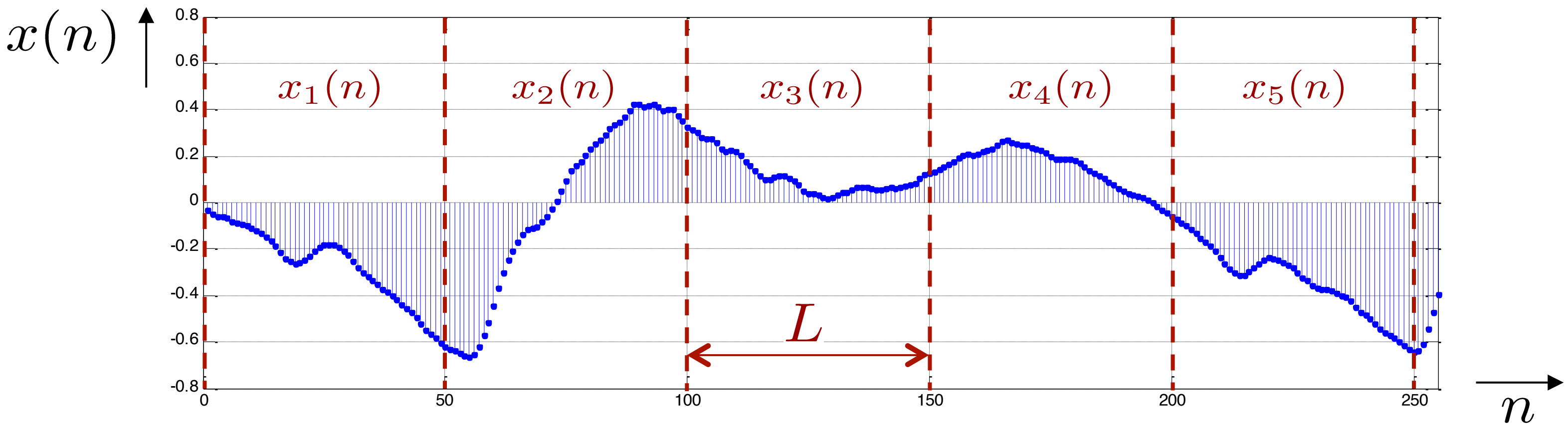
Application: FIR filtering of long data sequences



We assume that the FIR filter has a support of M samples

We apply the FIR filter on a block-by-block basis, where each block is of length L, and we assume that $L \gg M$ without loss of generality

Overlap-Add Method



We express x as

$$x(n) = \sum_{l=-\infty}^{\infty} x_l(n)$$

where

$$x_l(n) = \begin{cases} x(n - lL), & n = 0, \ldots, L-1 \\ 0, & \text{otherwise} \end{cases}$$

Overlap-Add Method

With this, the convolution $x * h$ can be expressed as

$$\begin{aligned} y(n) &= \sum_{k=0}^{M-1} h(k)x(n-k) \\ &= \sum_{k=0}^{M-1} h(k) \sum_{l=-\infty}^{\infty} x_l(n-k) \\ &= \sum_{l=-\infty}^{\infty} \underbrace{\sum_{k=0}^{M-1} h(k)x_l(n-k)}_{y_l(n)} \end{aligned}$$

Overlap-Add Method

In conclusion, the convolution $x * h$ can be expressed as a linear combination of convolutions $x_l * h$.

- In order to use the DFT (or FFT) to implement the convolutions, we have to apply a DFT of size $N = M + L - 1$. Hence, to each block of data x_l we append $M - 1$ zeros and compute the N-points DFT.

- Similarly, we compute the N-points DFT of h (padded with $L - 1$ zeros)

- We multiply the two N-points DFTs: $Y_l = X_l H$

- The inverse DFT yields data blocks of length N which have to be added (overlapped)

Overlap-Add Method

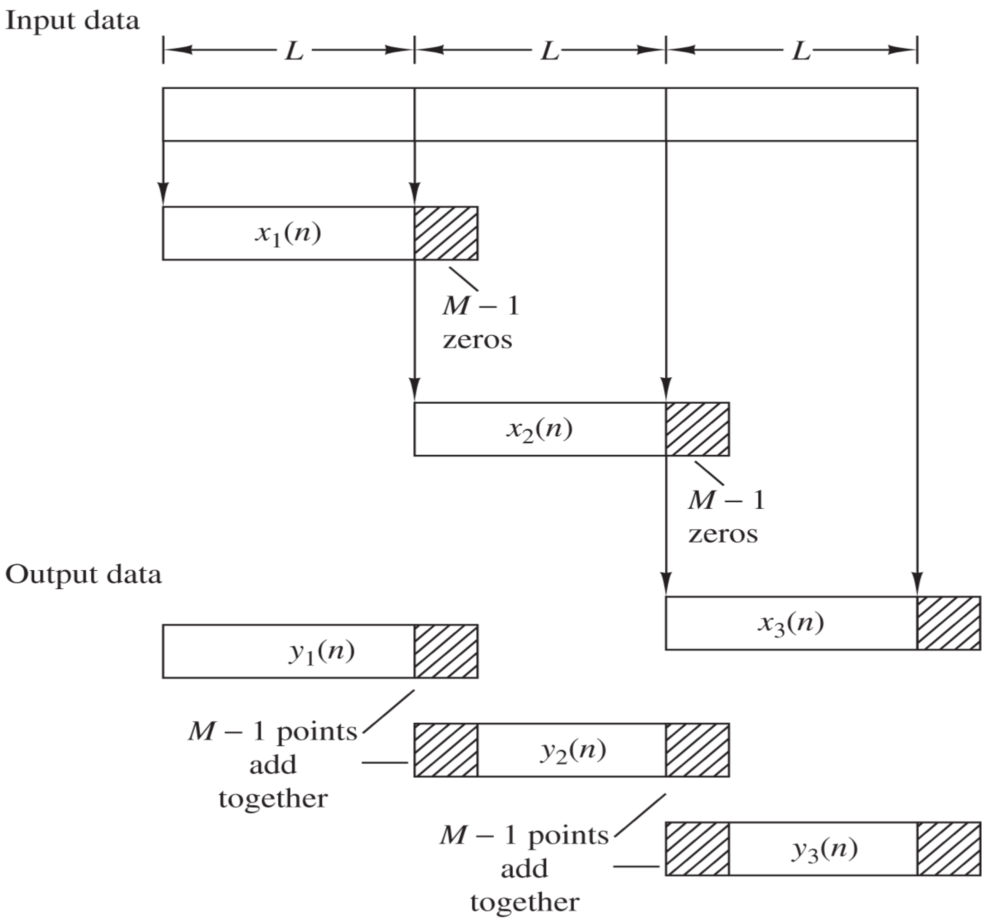


Figure 7.3.2 Linear FIR filtering by the overlap-add method.