Discrete Fourier Transformation (DFT)

Prof. Seungchul Lee
Industrial AI Lab.

Eigen-Analysis
(System or Linear Transformation)

Eigenvector and Eigenvalues

- Given matrix A

$$Av = \lambda v$$

- Eigenvectors v are input signals that emerge at the system output unchanged (except for a scaling by the eigenvalue λ_k) and so are somehow "fundamental" to the system

- Using this, we can find the following equation

$$AV = V\Lambda$$

$$AV = [v_0 \mid v_1 \mid \cdots \mid v_{N-1}] \begin{bmatrix} \lambda_0 & & & \\ & \lambda_1 & & \\ & & \ddots & \\ & & & \lambda_{N-1} \end{bmatrix}$$

- We can change to

$$A = V\Lambda V^{-1} \implies \text{Eigen-decomposition}$$

$$V^{-1}AV = \Lambda$$

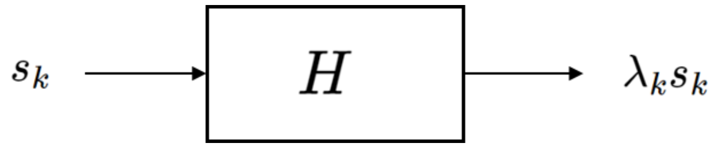
Eigen-analysis of LTI Systems (Finite-Length Signals)

- For length-N signals, H is an $N \times N$ circulent matrix with entries

$$[H]_{n,m} = h[(n-m)_N]$$

  where h is the impulse response

- Goal: calculate the eigenvectors and eigenvalues of H



  – Fact: the eigenvectors of a circulent matrix (LTI system) are the complex harmonic sinusoids

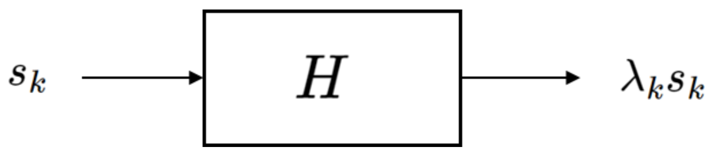
$$s_k[n] = \frac{1}{\sqrt{N}} e^{j\frac{2\pi}{N}kn}$$

  – The eigenvalue $\lambda_k \in \mathbb{C}$ corresponding to the sinusoid eigenvectors s_k is called the frequency response at frequency k since it measures how the system “responds” to s_k

$$\lambda_k = \sum_{n=0}^{N-1} h[n] e^{-j\frac{2\pi}{N}kn} = \langle h, s_k \rangle = H_u[k]$$

Eigenvector of LTI Systems (Finite-Length Signals)

- Prove that
  - harmonic sinusoids are the eigenvectors of LTI systems simply by computing the circular convolution with input s_k and applying the periodicity of the harmonic sinusoids



$$s_k[n] \ast h[n] = \sum_{m=0}^{N-1} s_k[(n-m)_N]\, h[m] = \sum_{m=0}^{N-1} \frac{e^{j\frac{2\pi}{N}k(n-m)_N}}{\sqrt{N}}\, h[m]$$

$$= \sum_{m=0}^{N-1} \frac{e^{j\frac{2\pi}{N}k(n-m)}}{\sqrt{N}}\, h[m] = \sum_{m=0}^{N-1} \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}} e^{-j\frac{2\pi}{N}km}\, h[m]$$

$$= \left(\sum_{m=0}^{N-1} e^{-j\frac{2\pi}{N}km}\, h[m] \right) \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}} = \lambda_k\, s_k[n]$$

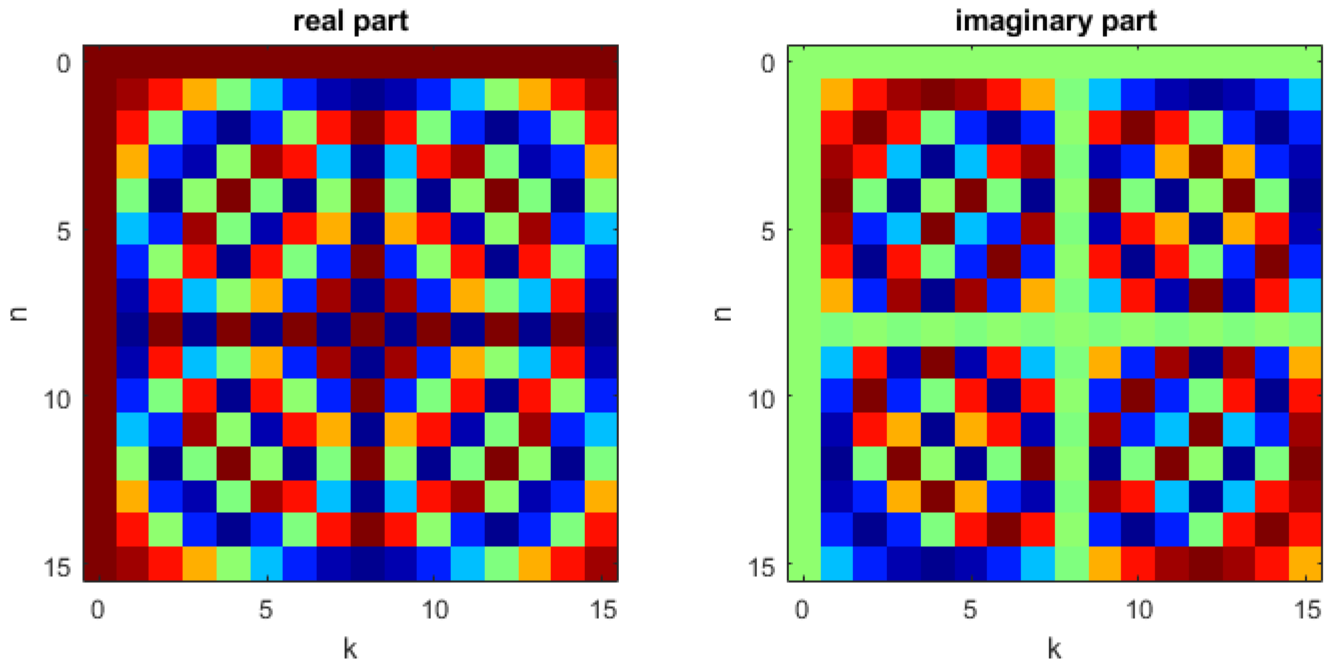
- λ_k means the number of s_k in $h[n]$ $\Rightarrow$ similarity

Eigenvector Matrix of Harmonic Sinusoids

- Stack N normalized harmonic sinusoid $\{s_k\}_{k=0}^{N-1}$ as columns into an $N \times N$ complex orthonormal basis matrix

$$S = [s_0 \mid s_1 \mid \cdots \mid s_{N-1}]$$

$$s_k[n] = \frac{1}{\sqrt{N}} e^{j\frac{2\pi}{N}kn}$$



Signal Decomposition by Harmonic Sinusoids

Basis

- A basis $\{b_k\}$ for a vector space V is a collection of vectors from V that linearly independent and span V

- Basis matrix: stack the basis vectors b_k as columns

$$B = [b_0 \mid b_1 \mid b_2 \mid \cdots \mid b_{N-1}]$$

- Using this matrix B, we can now write a linear combination of basis elements as the matrix/vector product

$$x = \alpha_0 b_0 + \alpha_1 b_1 + \cdots + \alpha_{N-1} b_{N-1} = \sum_{k=0}^{N-1} \alpha_k b_k$$

$$= [b_0 \mid b_1 \mid b_2 \mid \cdots \mid b_{N-1}] \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{bmatrix} = B\alpha$$

Orthonormal Basis

- An orthogonal basis $\{b_k\}_{k=0}^{N-1}$ for a vector space V
  - a basis whose elements are mutually orthogonal

$$\langle b_k, b_l \rangle = 0 \quad k \neq l$$

- An orthonormal basis $\{b_k\}_{k=0}^{N-1}$ for a vector space V
  - a basis whose elements are mutually orthogonal and normalized in the 2-norm

$$\langle b_k, b_l \rangle = 0 \quad k \neq l, \quad \text{and}$$

$$||b_k||_2 = 1 \quad \forall k$$

Orthonormal Basis

- B is a unitary matrix

$$B^H B = I \implies B^{-1} = B^H, \qquad \text{where } B^H \text{ is Hermitian (complex conjugate) transpose}$$

$$B^H B = \begin{bmatrix} b_0^H \\ b_1^H \\ \vdots \\ b_{N-1}^H \end{bmatrix} [b_0 \mid b_1 \mid b_2 \mid \cdots \mid b_{N-1}] = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

Signal Represented by Orthonormal Basis

- Signal representation by orthonormal basis $\{b_k\}_{k=0}^{N-1}$ and orthonormal basis matrix B

$$x = B\alpha = \sum_{k=0}^{N-1} \alpha_k b_k, \qquad \text{(synthesis)}$$

$$\alpha = B^H x \qquad \text{or} \qquad \alpha_k = \langle x, b_k \rangle, \qquad \text{(analysis)}$$

- Synthesis: build up the signal x as a linear combination of the basis elements b_k weighted by the weights α_k

- Analysis: compute the weights α_k such that the synthesis produces x; the weights α_k measures the similarity between x and the basis element b_k

Harmonic Sinusoids are an Orthonormal Basis

- Stack N normalized harmonic sinusoid $\{s_k\}_{k=0}^{N-1}$ as columns into an $N \times N$ complex orthonormal basis matrix

$$S = [s_0 \mid s_1 \mid \cdots \mid s_{N-1}] \quad \text{where} \quad s_k[n] = \frac{1}{\sqrt{N}} e^{j\frac{2\pi}{N}kn}$$

$$S^H S = I \implies \langle s_k, s_l \rangle = 0 \quad k \neq l, \quad \text{and} \quad \|s_k\|_2 = 1$$

POSTECH

Discrete Fourier Transform (DFT)

DFT and Inverse DFT

- Jean Baptiste Joseph Fourier had the radical idea of proposing that all signals could be represented as a linear combination of sinusoids

- Analysis (Forward DFT)
  - The weight $X[k]$ measures the similarity between x and the harmonic sinusoid s_k
  - It finds the "frequency contents" of x at frequency k

$$X = S^H x$$

$$X[k] = \langle x, s_k \rangle = \sum_{n=0}^{N-1} x[n] \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

DFT and Inverse DFT

- Jean Baptiste Joseph Fourier had the radical idea of proposing that all signals could be represented as a linear combination of sinusoids

- Synthesis (Inverse DFT)
  - It is returning to time domain
  - It builds up the signal x as a linear combination of s_k weighted by the $X[k]$

$$x = SX$$

$$x[n] = \sum_{k=0}^{N-1} X[k] \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

Unnormalized DFT

- Normalized forward and inverse DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] \, \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

$$x[n] = \sum_{k=0}^{N-1} X[k] \, \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

- Unnormalized forward and inverse DFT

$$X_u[k] = \sum_{n=0}^{N-1} x[n] \, e^{-j\frac{2\pi}{N}kn}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \, e^{j\frac{2\pi}{N}kn}$$

Harmonic Sinusoids are an Orthonormal Basis

- Stack N normalized harmonic sinusoid $\{s_k\}_{k=0}^{N-1}$ as columns into an $N \times N$ complex orthonormal basis matrix

$$S = [s_0 \mid s_1 \mid \cdots \mid s_{N-1}] \quad \text{where} \quad s_k[n] = \frac{1}{\sqrt{N}} e^{j\frac{2\pi}{N}kn}$$

$$S^H S = I \implies \langle s_k, s_l \rangle = 0 \quad k \neq l, \quad \text{and} \quad \|s_k\|_2 = 1$$

Eigen-decomposition and Diagonalization

- H is circulent LTI System matrix
- S is harmonic sinusoid eigenvectors matrix (corresponds to DFT/IDFT)
- Λ is eigenvalue diagonal matrix (frequency response)

$$H = S \Lambda S^H$$

- The eigenvalues are the frequency response (unnormalized DFT of the impulse response)

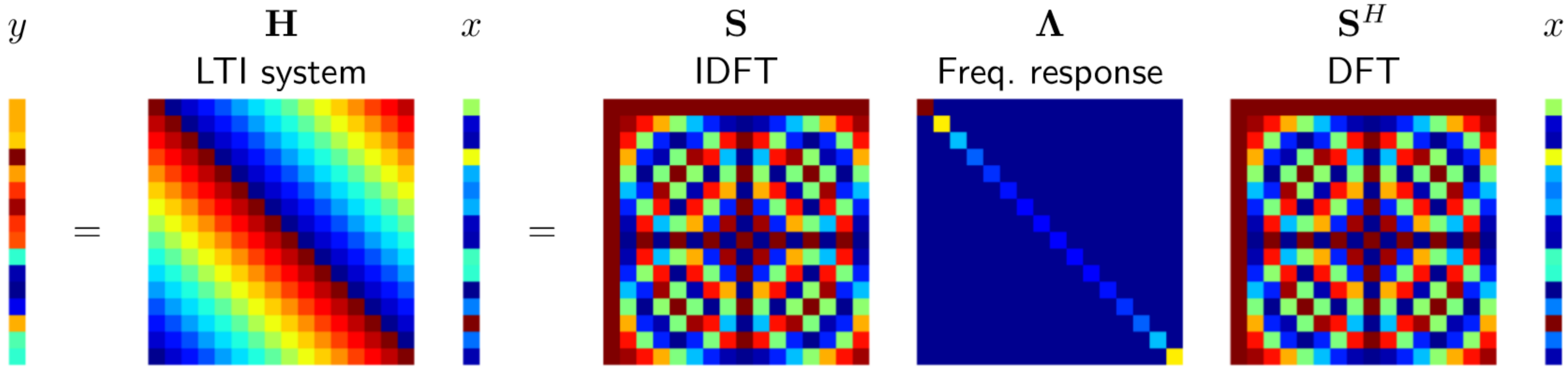
$$\lambda_k = \sum_{n=0}^{N-1} h[n] e^{-j\frac{2\pi}{N}kn} = \langle h, s_k \rangle = H_u[k]$$

- Place the N eigenvalues $\{\lambda_k\}_{k=0}^{N-1}$ on the diagonal of an $N \times N$ matrix

$$\Lambda = \begin{bmatrix} \lambda_0 & & & \\ & \lambda_1 & & \\ & & \ddots & \\ & & & \lambda_{N-1} \end{bmatrix} = \begin{bmatrix} H_u[0] & & & \\ & H_u[1] & & \\ & & \ddots & \\ & & & H_u[N-1] \end{bmatrix}$$

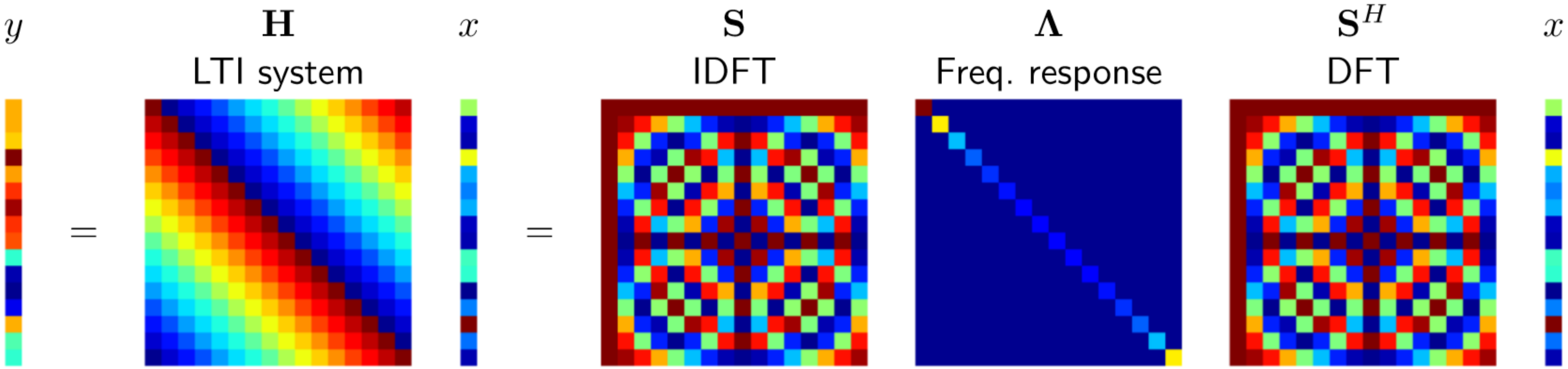
Eigen-decomposition and Diagonalization

- H is circulent LTI System matrix
- S is harmonic sinusoid eigenvectors matrix (corresponds to DFT/IDFT)
- Λ is eigenvalue diagonal matrix (frequency response)



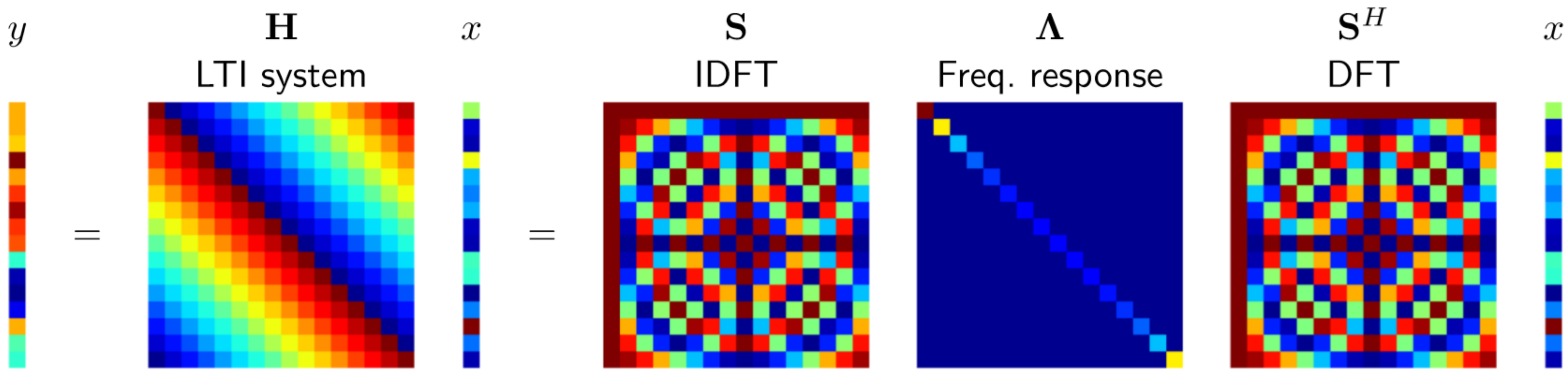
Eigen-decomposition and Diagonalization

$$\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$



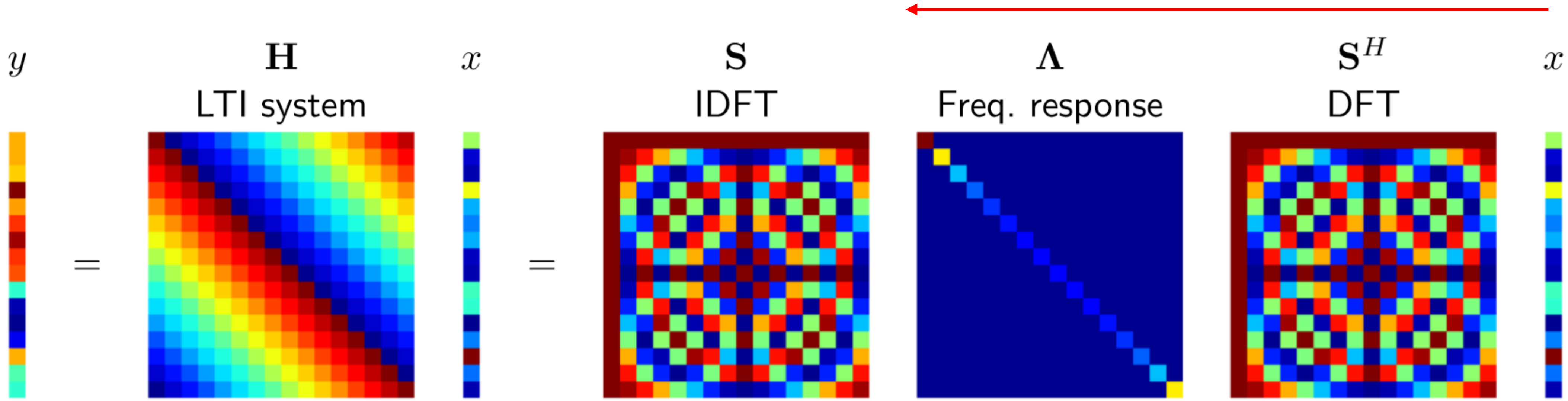
Eigen-decomposition and Diagonalization

$$\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix}$$



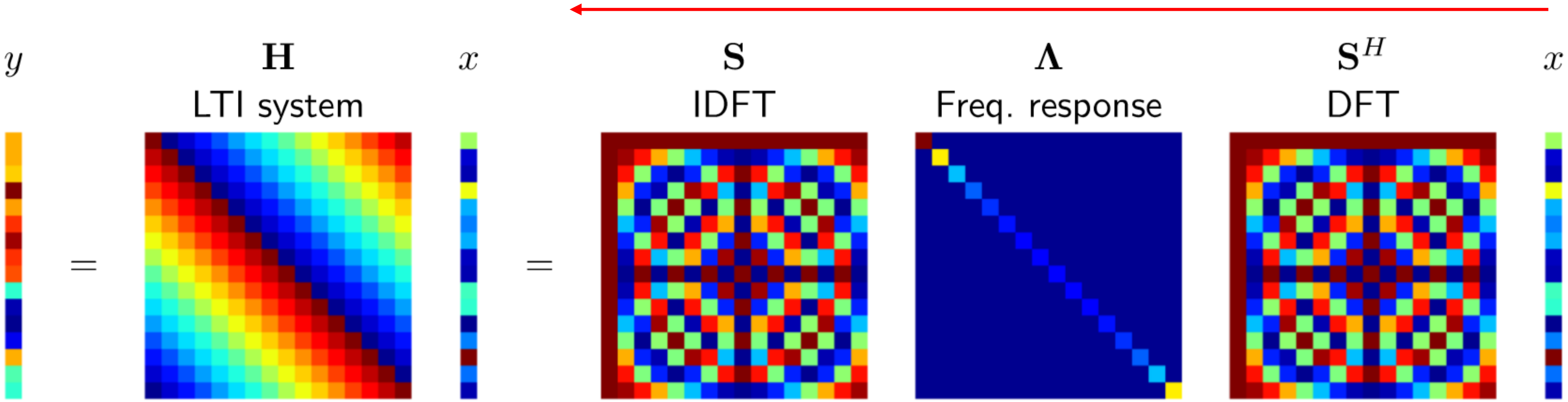
Eigen-decomposition and Diagonalization

$$\begin{bmatrix} \lambda_0 X[0] \\ \lambda_1 X[1] \\ \vdots \\ \lambda_{N-1} X[N-1] \end{bmatrix}$$



Eigen-decomposition and Diagonalization

$$\begin{bmatrix} y[0] \\ y[1] \\ \vdots \\ y[N-1] \end{bmatrix}$$

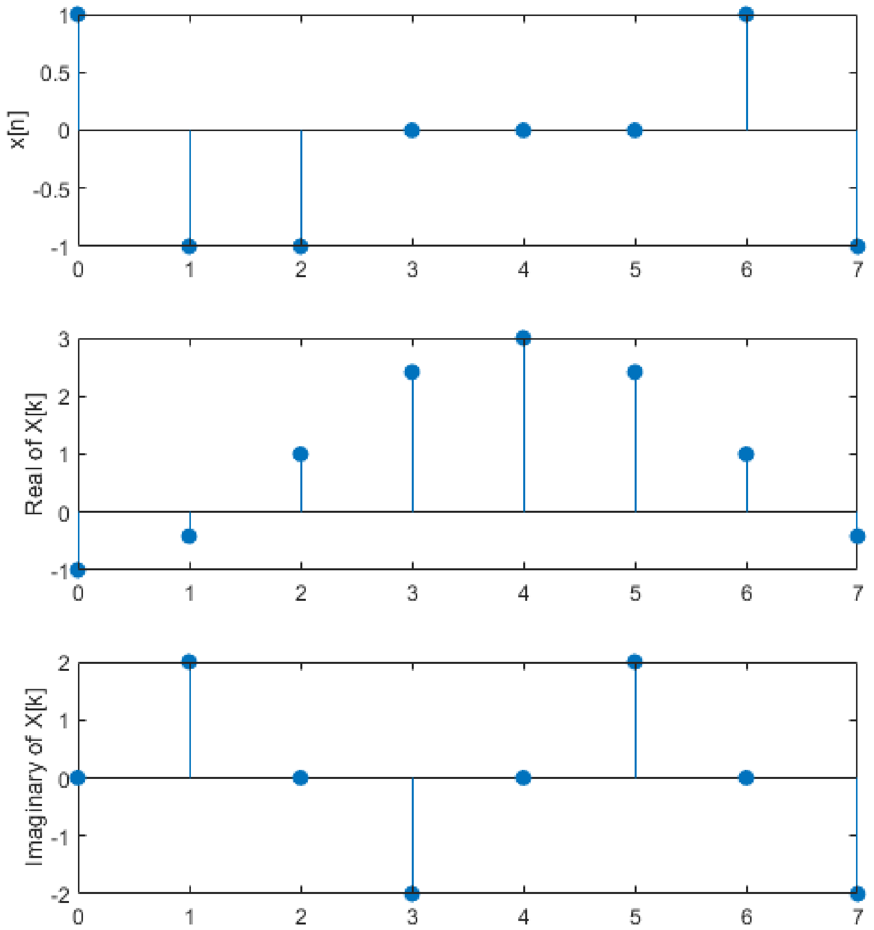


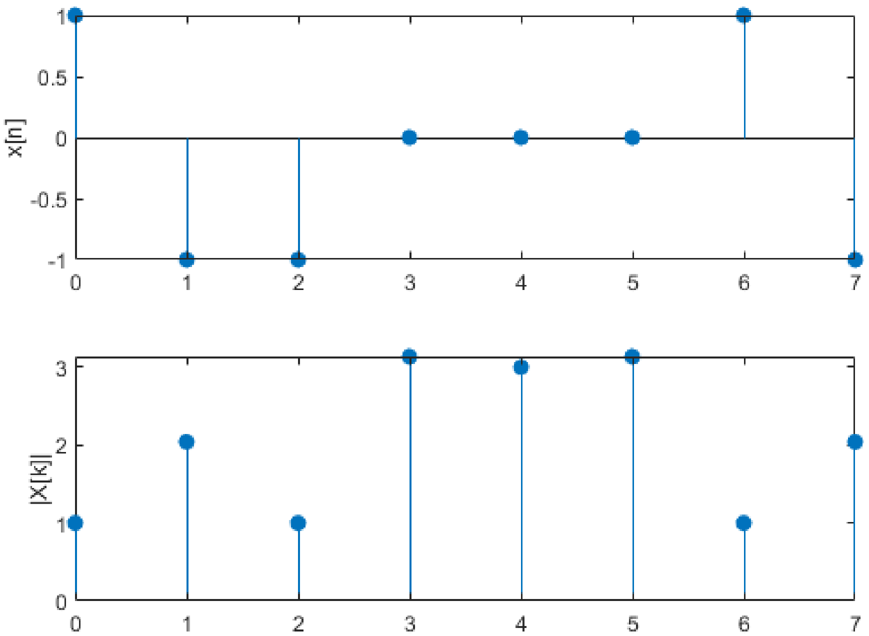
DFT in MATLAB

```
%% DFT without using the built-in function

x = [1 -1 -1 0 0 0 1 -1];
N = length(x);
X = zeros(1,N);

for k = 0:N-1
    for n = 0:N-1
        X(k+1) = X(k+1) + x(n+1)*exp(-1j*2*pi/N*n*k);
    end
end
```

$$X_u[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$

DFT in MATLAB





DFT Function

$$X_u[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$

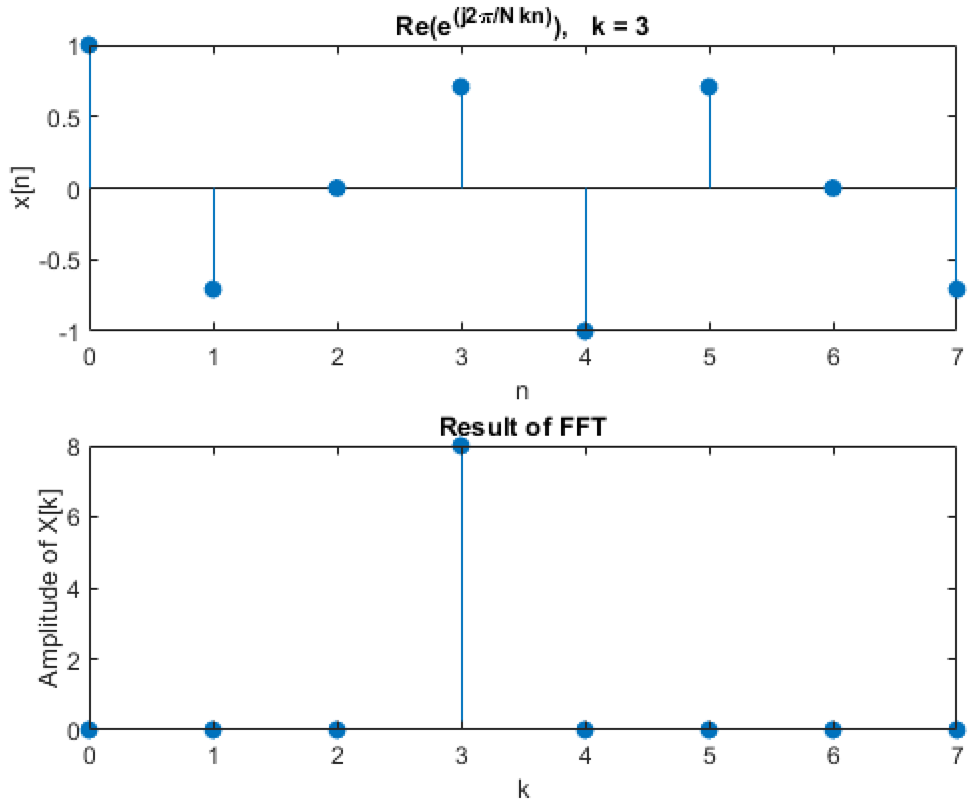
```
function [Xk] = dft(xn,N)

% Computes Discrete Fourier Transform
% [Xk] = dft(xn,N)
% Xk = DFT coeff. array over 0 <= k <= N-1
% xn = N-point finite-duration sequence
%  N = Length of DFT
%

n = [0:1:N-1];                      % row vector for n
k = [0:1:N-1];                      % row vecor for k
WN = exp(-1j*2*pi/N);               % Wn factor
nk = n'*k;                          % creates a N by N matrix of nk values
WNnk = WN.^nk;                      % DFT matrix
Xk = xn*WNnk;                       % row vector for DFT coefficients
```

Example: DFT

$$x[n] = e^{j\frac{2\pi}{8}3n}$$

```
k = 3;                          % index for freqeuncy

N = 8;
n = 0:N-1;                      % sampling period

x = exp(1j*2*pi/N*k*n);         % harmonic complex exponential
```

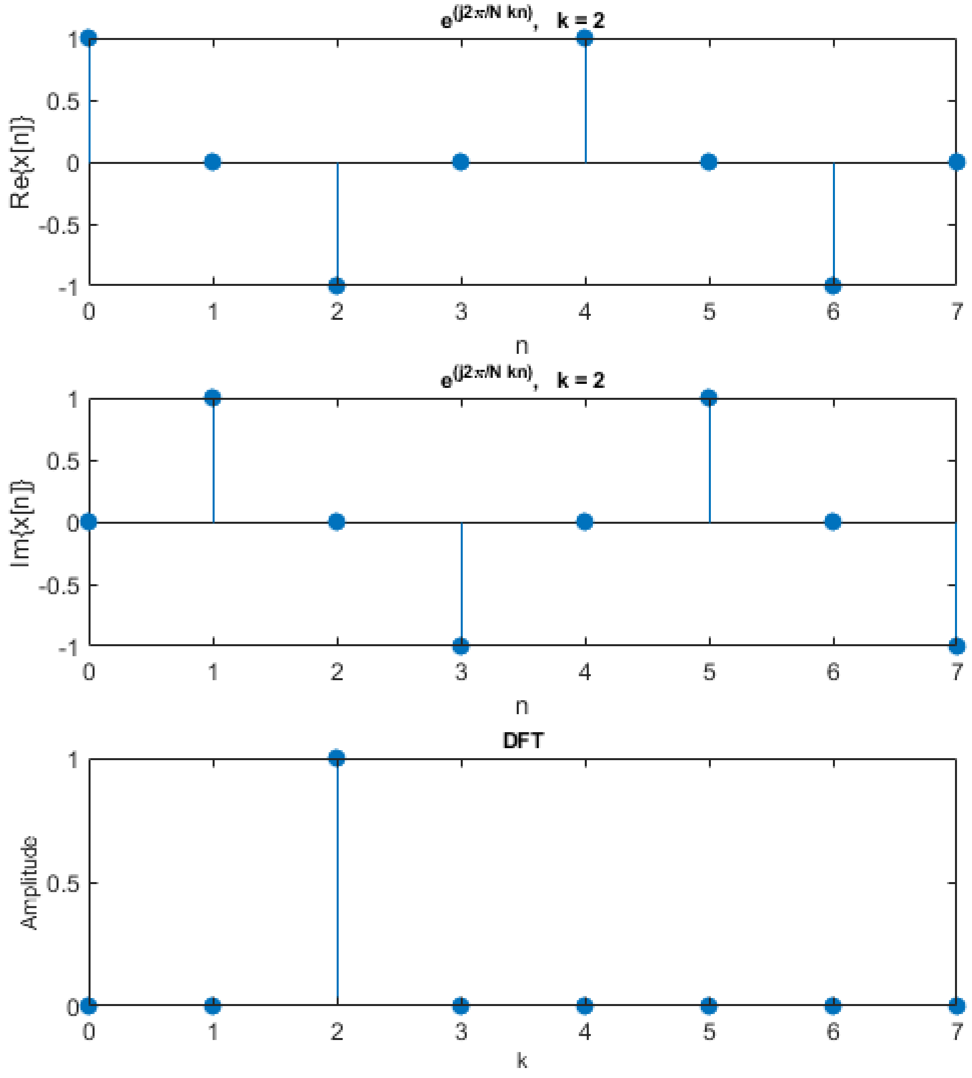
```
X = dft(x,N);
%X = fft(x,N);
```



Example: DFT

$$x[n] = e^{j\frac{2\pi}{8}2n}$$

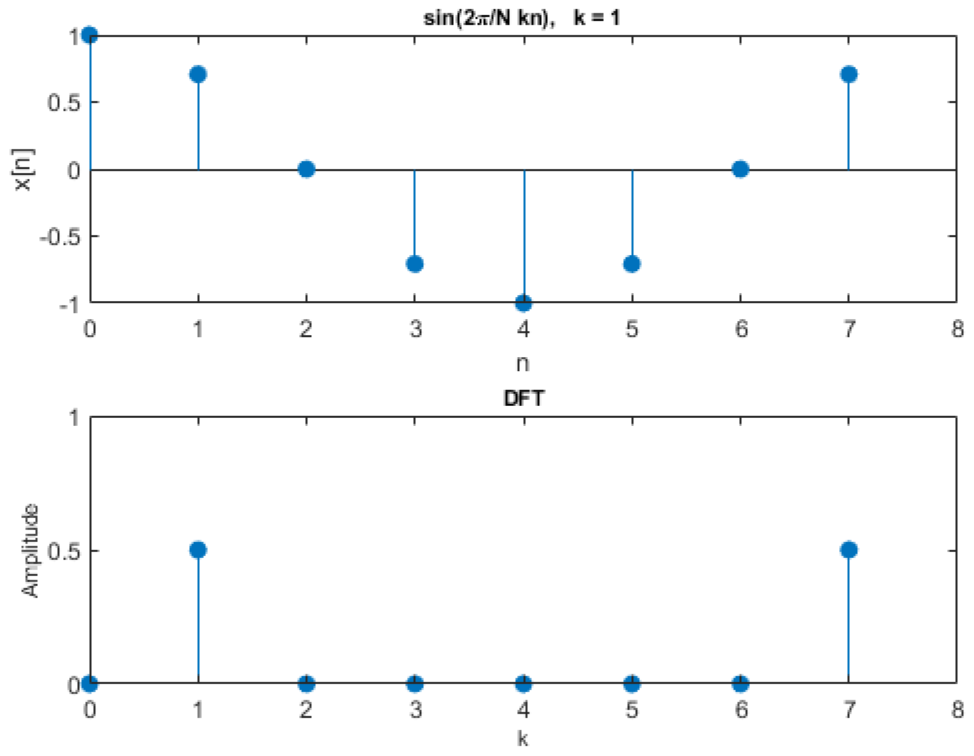
```
% Normalized DFT

k = 2;                        % index for freqeuncy

N = 8;
n = 0:N-1;                    % sampling period

x = exp(1j*2*pi/N*k*n);       % harmonic complex exponential
X = dft(x,N)/N;
```



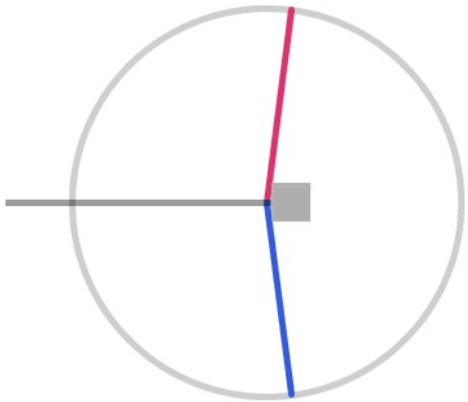
Example: DFT

$$x[n] = \cos\left(\frac{2\pi}{8}1n\right)$$

```
k = 1;                          % index for freqeuncy

N = 8;
n = 0:N-1;                      % sampling period

x = cos(2*pi/N*k*n);            % harmonic complex exponential
X = dft(x,N)/N;
```



$$\cos\omega t = \frac{e^{i\omega t} + e^{-i\omega t}}{2}$$



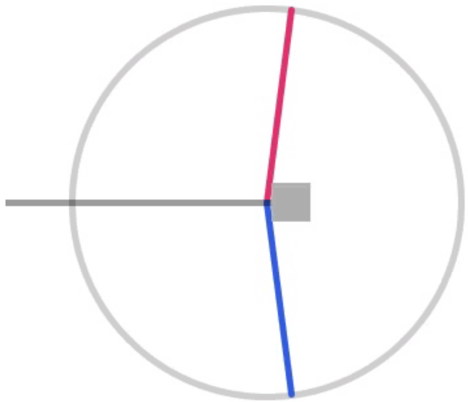
Example: DFT

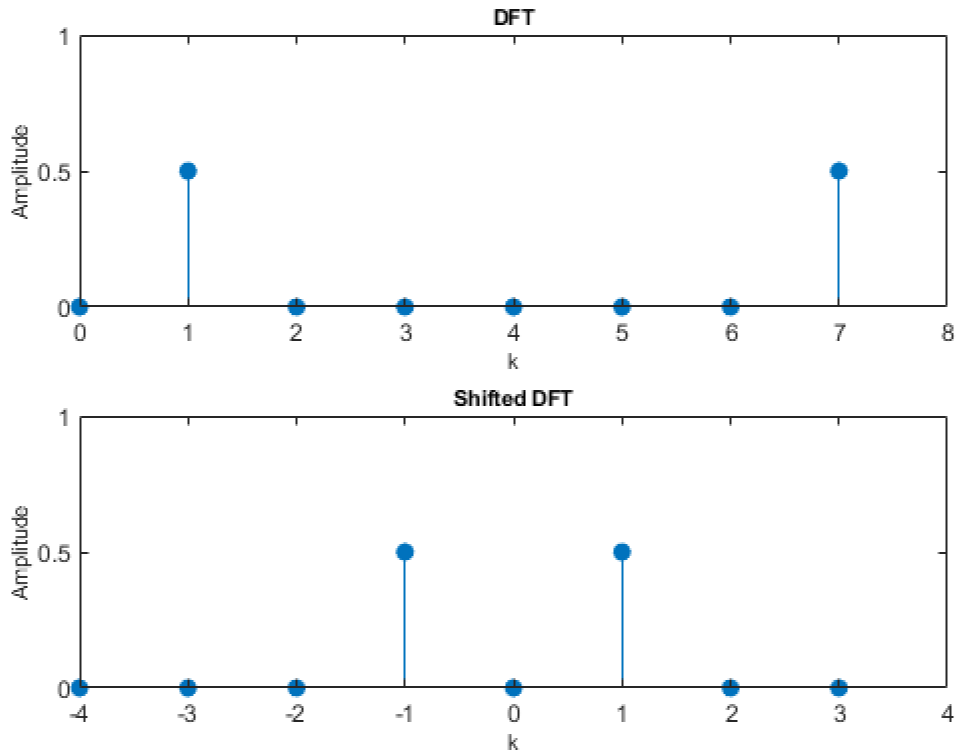
Typical interval 1: $0 \leq k \leq N-1$ corresponds to frequencies ω_k in the interval $0 \leq \omega \leq 2\pi$

Typical interval 2: $-\frac{N}{2} \leq k \leq \frac{N}{2} - 1$ corresponds to frequencies ω_k in the interval $-\pi \leq \omega \leq \pi$

```
n = 0:N-1;
k = 0:N-1;

kr = [0:N/2-1 -N/2:-1];
ks = fftshift(kr);

X = fft(x,N)/N;
Xs = fftshift(X);
```

$$\cos \omega t = \frac{e^{i\omega t} + e^{-i\omega t}}{2}$$





Fast Fourier Transform (FFT)

- FFT algorithms are so commonly employed to compute DFT that the term 'FFT' is often used to mean 'DFT'
  - The FFT has been called the "most important computational algorithm of our generation"
  - It uses the dynamic programming algorithm (or divide and conquer) to efficiently compute DFT.

- DFT refers to a mathematical transformation or function, whereas 'FFT' refers to a specific family of algorithms for computing DFTs.
  - use fft command to compute dft
  - fft (computationally efficient)

- We will use the embedded fft function without going too much into detail.

DFT Properties

$$X[k] = \sum_{n=0}^{N-1} x[n] \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

$$x[n] = \sum_{k=0}^{N-1} X[k] \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

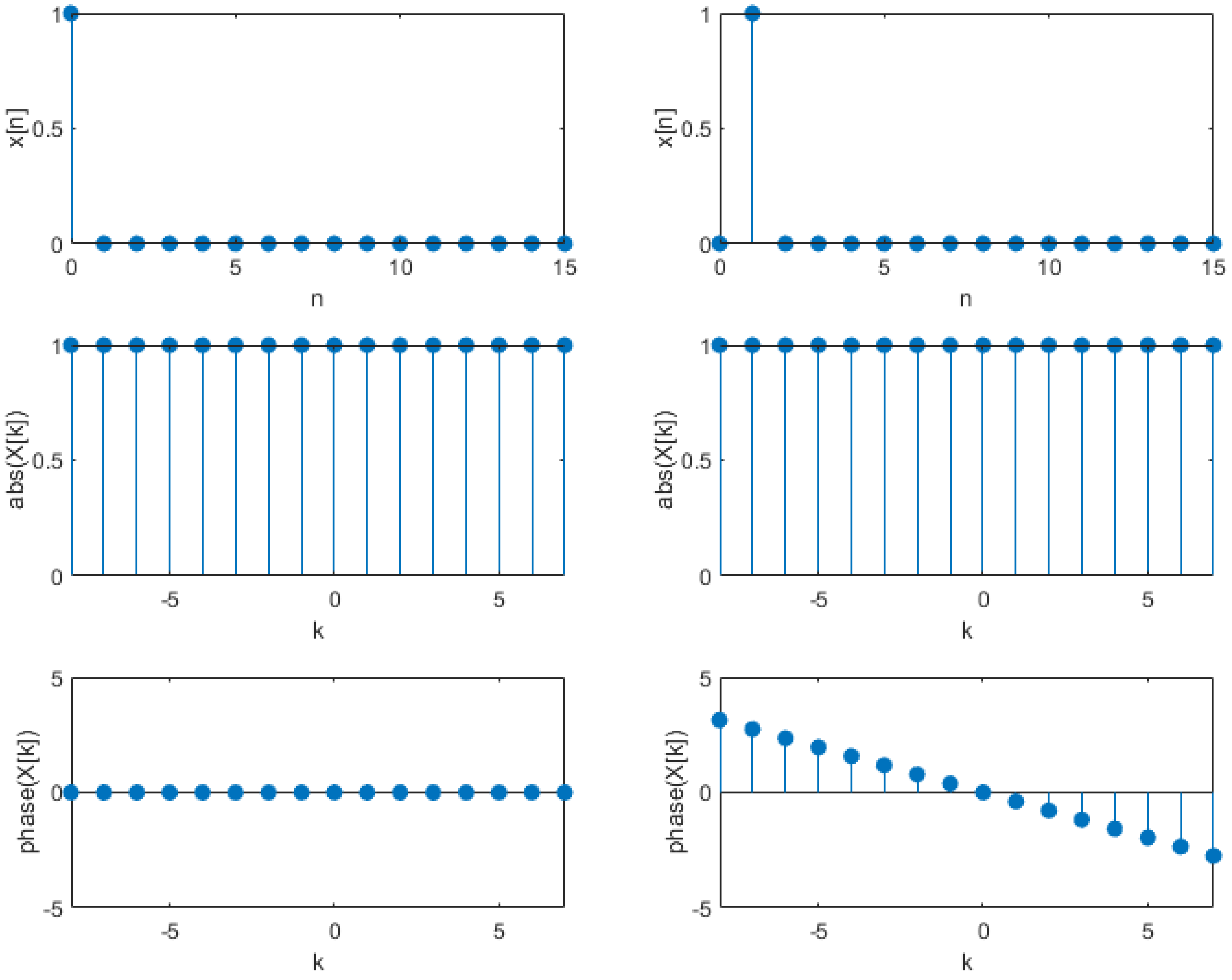
- DFT pair $$x[n] \longleftrightarrow X[k]$$

- DFT Frequencies
  - $X[k]$ measures the similarity between the time signal $x[n]$ and the harmonic sinusoid $s_k[n]$
  - $X[k]$ measures the "frequency content" of $x[n]$ at frequency $\omega_k = \frac{2\pi}{N}k$

DFT Properties

- DFT and Circular Shift
  - No amplitude changed
  - Phase changed

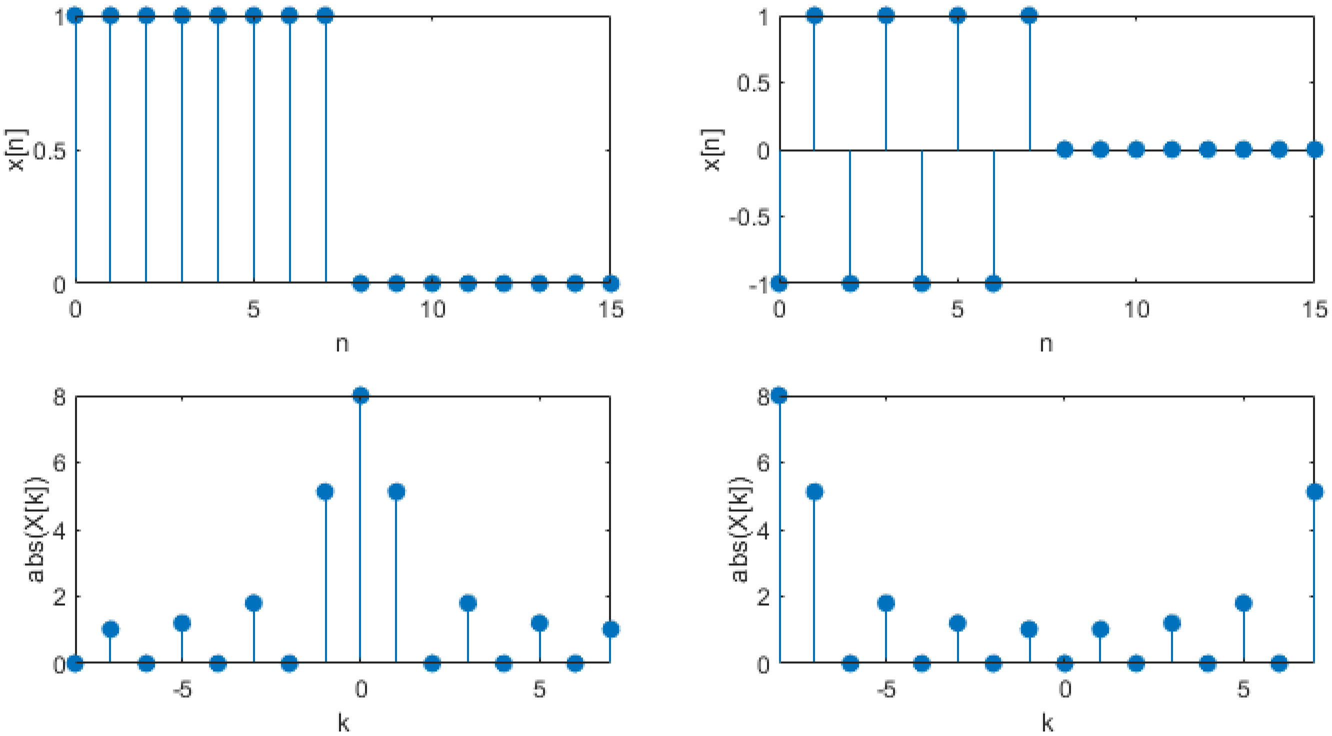
$$x[(n-m)_N] \longleftrightarrow e^{-j\frac{2\pi}{N}km}X[k]$$



DFT Properties

- DFT and Modulation

$$e^{j\frac{2\pi}{N}r\,n}x[n] \longleftrightarrow X[(k-r)_N]$$



DFT Properties

- DFT and Circular Convolution
  - Circular convolution in the time domain = multiplication in the frequency domain

$$Y[k] = H[k]X[k]$$

$$h[n] \otimes x[n] \longleftrightarrow H[k]X[k]$$

$$y[n] = \mathrm{IDFT}(Y[k])$$

- Proof

$$Y_u[k] = \sum_{n=0}^{N-1} y[n] e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} \left(\sum_{m=0}^{N-1} h[(n-m)_N] x[m] \right) e^{-j\frac{2\pi}{N}kn}$$

$$= \sum_{m=0}^{N-1} x[m] \left(\sum_{n=0}^{N-1} h[(n-m)_N] e^{-j\frac{2\pi}{N}kn} \right)$$

$$= \sum_{m=0}^{N-1} x[m] \left(\sum_{r=0}^{N-1} h[r] e^{-j\frac{2\pi}{N}k(r+m)} \right)$$

$$= \left(\sum_{m=0}^{N-1} x[m] e^{-j\frac{2\pi}{N}km} \right) \left(\sum_{r=0}^{N-1} h[r] e^{-j\frac{2\pi}{N}kr} \right)$$

$$= X_u[k] H_u[k]$$

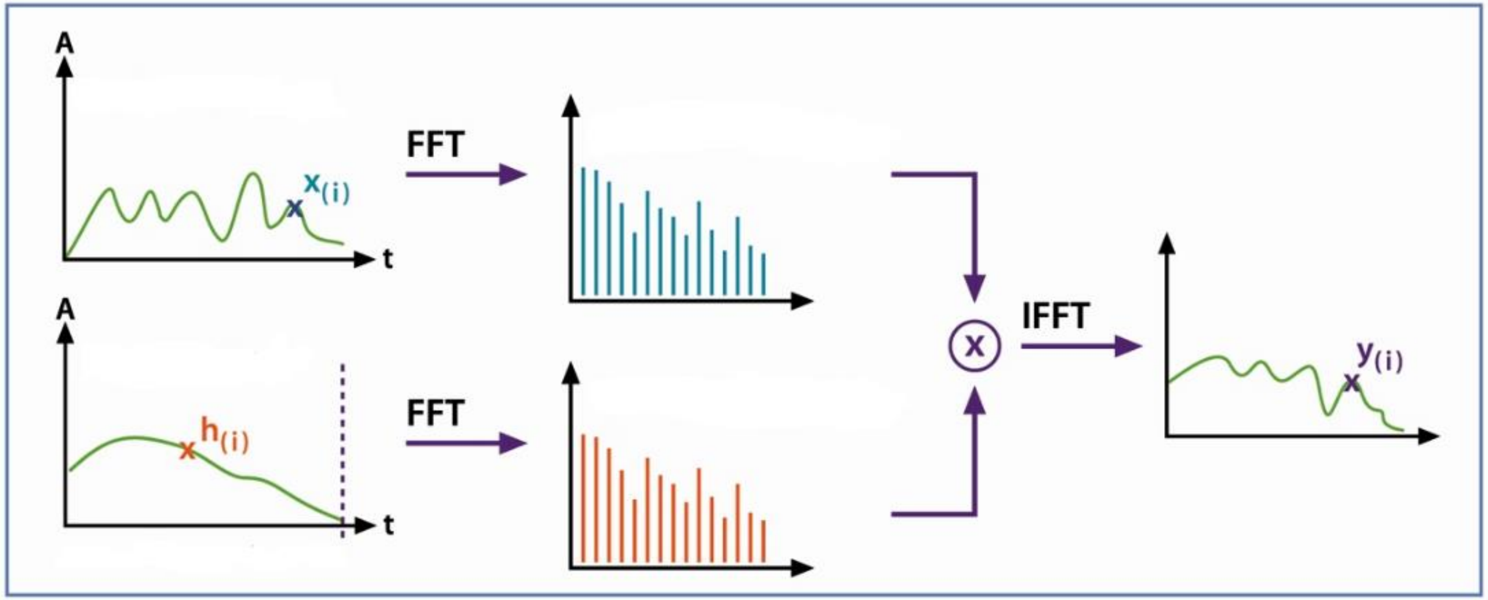
Filtering in Frequency Domain

- Circular convolution in the time domain = multiplication in the frequency domain

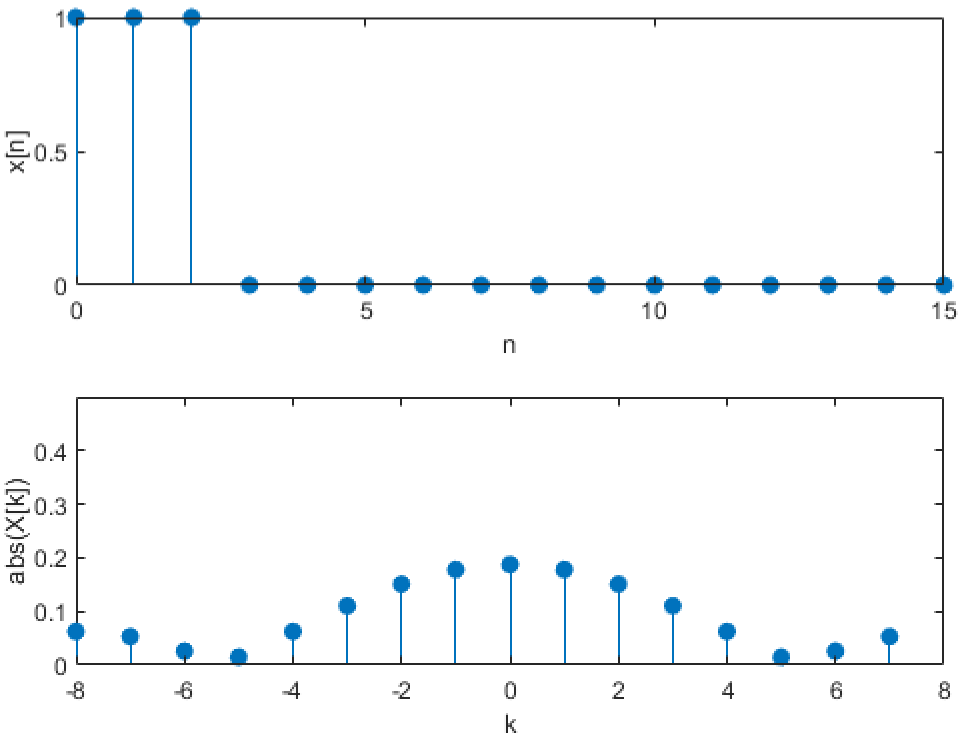
$$Y[k] = H[k]X[k]$$

$$h[n] \otimes x[n] \longleftrightarrow H[k]X[k]$$

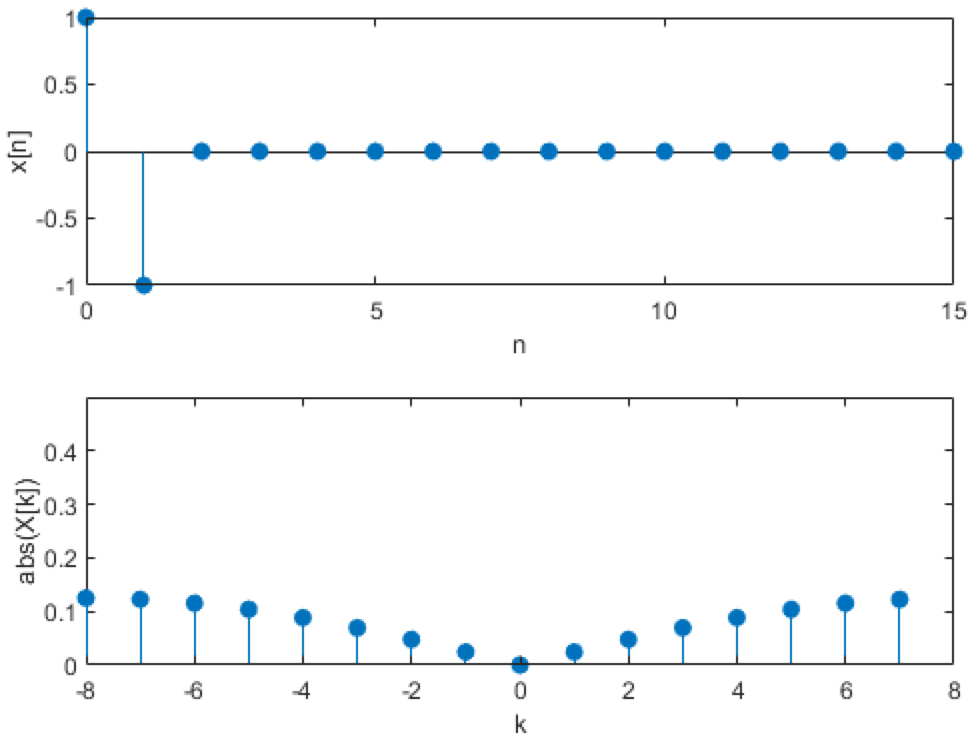
$$y[n] = \mathrm{IDFT}(Y[k])$$



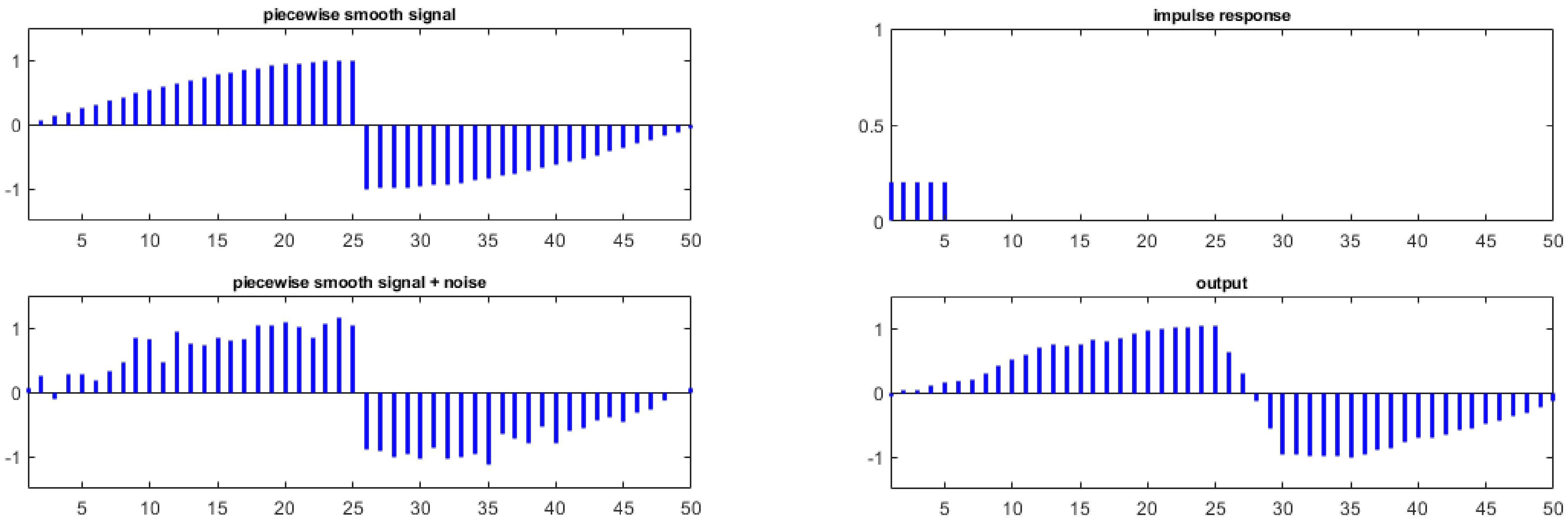
Example: Low-Pass Filter



Example: High-Pass Filter



Filtering in Time Domain




```
y = cconv(x,h,N);
```

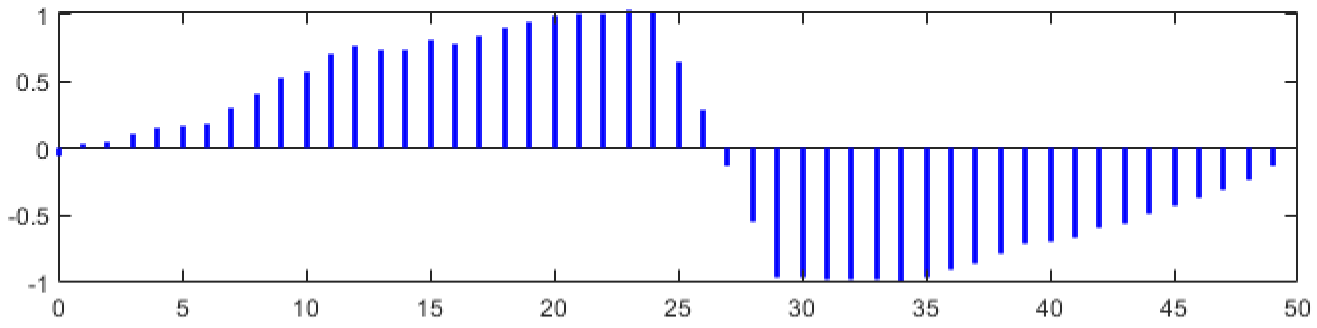
Filtering in Frequency Domain

$$Y[k] = H[k]X[k]$$

$$h[n] \otimes x[n] \longleftrightarrow H[k]X[k]$$

$$y[n] = \mathrm{IDFT}(Y[k])$$

```
y = cconv(x,h,N);
```




```
H = fft(h,N);
X = fft(x,N);

Y = H.*X;
yi = ifft(Y);
```