Signal Processing for Interactive Systems

Lecture 2

February 19, 2024

Cumhur Erkut
cer@create.aau.dk

AALBORG UNIVERSITY
DENMARK

Agenda

Windowing

The Discrete Fourier Transform (DFT)

Agenda

Windowing

The Discrete Fourier Transform (DFT)

Windowing

Motivation

In about 20 minutes, you will know

- why we have to window the sampled data
- what consequences windowing have on the spectrum of the windown data
- that different windows trades off frequency resolution for side lobe attenuation

Windowing

In lecture 1, we saw that the DTFT of a discrete-time signal $x(n)$ is given by

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)\mathrm{e}^{-\mathrm{j}\omega n} \,. \tag{1}$$

Windowing

In lecture 1, we saw that the DTFT of a discrete-time signal $x(n)$ is given by

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} . \tag{1}$$

- Impossible to work with infinitely long signals in practice
- Instead, we only record N samples for $n = 0, 1, \ldots, N-1$, i.e.,
$$\boldsymbol{x} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-1) \end{bmatrix}^T \tag{2}$$
- But how do we compute the spectrum of such a finite set of samples?

Windowing

To allow us to use the theory from lecture 1, we invent a new infinite discrete-time signal $x_N(n)$ given by

$$x_N(n) = w(n)x(n) \tag{3}$$

where

- $w(n)$ is a rectangular window given by

$$w(n) = \begin{cases} 1 & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

- $x(n)$ is an infinitely long discrete-time signal (as considered in lecture 1)

Windowing

We then get that the DTFT of $x_N(n)$ is

$$X_N(\omega) = \sum_{n=-\infty}^{\infty} x_N(n)\mathrm{e}^{-\mathrm{j}\omega n} = \sum_{n=0}^{N-1} x(n)\mathrm{e}^{-\mathrm{j}\omega n} = \boldsymbol{f}^H(\omega)\boldsymbol{x} \qquad (5)$$

where $(\cdot)^H$ denotes the hermitian (complex conjugation and transposition) and

$$\boldsymbol{x} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-1) \end{bmatrix}^T \qquad (6)$$

$$\boldsymbol{f}(\omega) = \begin{bmatrix} 1 & \mathrm{e}^{\mathrm{j}\omega} & \cdots & \mathrm{e}^{\mathrm{j}\omega(N-1)} \end{bmatrix}^T \qquad (7)$$

Thus, computing the DTFT of $x_N(n)$ gives us the spectrum of the finite length signal $\boldsymbol{x}$!

Windowing

Windowed DTFT

From the modulation property, we now get

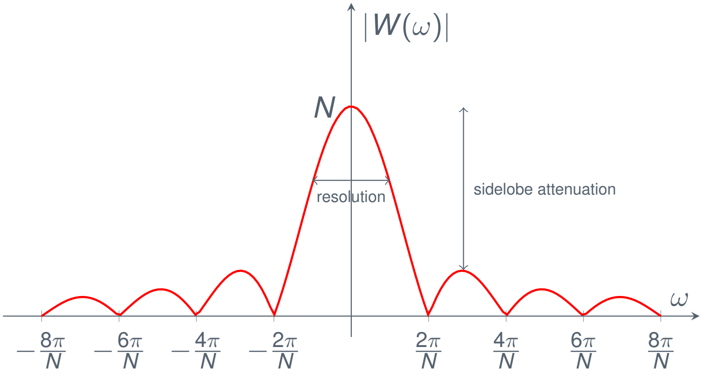
$$X_N(\omega) = (W \circledast X)(\omega) = \frac{1}{2\pi}\int_{-\pi}^{\pi} W(\nu)X(\omega-\nu)d\nu \qquad (8)$$

where $W(\nu)$ is the DTFT of the window. For a rectangular window, we saw in lecture 1 that

$$W(\omega) = \begin{cases} N & \omega = 0 \\ \frac{\sin(\omega N/2)}{\sin(\omega/2)} \mathrm{e}^{-\mathrm{j}\omega\frac{N-1}{2}} & \text{otherwise} \end{cases} \qquad (9)$$

Windowing

The amplitude spectrum of a rectangular window.



Windowing

Example: windowed phasor

Assume that we observed N samples from

$$x(n) = e^{j\omega_0 n} \,. \tag{10}$$

Since the DTFT of $x(n)$ is

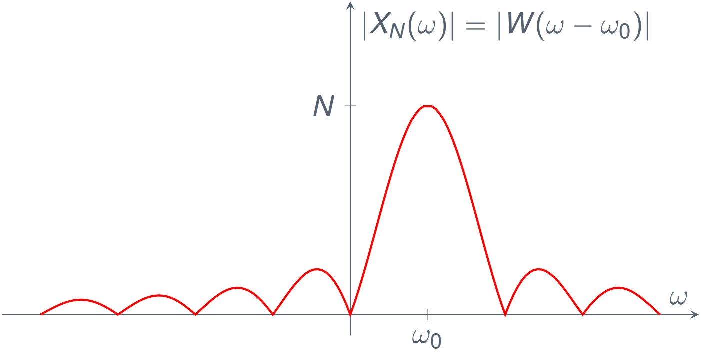
$$X(\omega) = 2\pi\delta(\omega - \omega_0) \,, \tag{11}$$

the DTFT of $x_N(n) = w(n)x(n)$ is

$$\begin{aligned} X_N(\omega) &= \frac{1}{2\pi}\int_{-\pi}^{\pi} W(\nu)X(\omega-\nu)d\nu = \int_{-\pi}^{\pi} W(\nu)\delta(\omega-\omega_0-\nu)d\nu \\ &= W(\omega-\omega_0) \,. \end{aligned} \tag{12}$$

Windowing

The amplitude spectrum of windows phasor.



Windowing

Other windows

- Many other windows than the rectangular window exists
- A popular alternative is the Hamming window given by

$$w(n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

- Windows trades off frequency resolution for sidelobe attenuation
- The rectangular window has the best frequency resolution, but the lowest sidelobe attenuation
- Increasing N increases the frequency resolution and the sidelobe attenuation

Windowing

Summary

- Windows are a necessary since we have to work with finite discrete-time signals
- Windowing limits the frequency resolution and introduces sidelobes, i.e., other frequency components
- Many windows exist and they trades off frequency resolution for sidelobe attenuation
- Use long windows for stationary signals

Windowing

Five minutes active break

- In MATLAB, compute the amplitude spectra of the rectangular and hamming window of length $N = 100$ (use the function `fft(w,nDft)` where you set the variable nDft to 2048).
- Which window has the best frequency resolution and and which function has the best sidelobe attenuation?

Agenda

Windowing

The Discrete Fourier Transform (DFT)

The Discrete Fourier Transform

Motivation

In about 20 minutes, you will know

- what the differences are between the DTFT and the DFT
- how the DFT can be conveniently modelled using linear algebra
- when and when not zero-padding is necessary for performing linear convolution in the frequency domain

The Discrete Fourier Transform

- DTFT of infinite sequence $x(n)$

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)\mathrm{e}^{-\mathrm{j}\omega n} \tag{14}$$

The Discrete Fourier Transform

- DTFT of infinite sequence $x(n)$

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \tag{14}$$

- DTFT of a rectangularly windowed infinite sequence $(x_N(n) = w(n)x(n))$

$$X_N(\omega) = \sum_{n=-\infty}^{\infty} x_N(n)e^{-j\omega n} = \sum_{n=0}^{N-1} x(n)e^{-j\omega n} = \boldsymbol{f}^H(\omega)\boldsymbol{x} \tag{15}$$

where

$$\boldsymbol{x} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-1) \end{bmatrix}^T \tag{16}$$

$$\boldsymbol{f}(\omega) = \begin{bmatrix} 1 & e^{j\omega} & \cdots & e^{j\omega(N-1)} \end{bmatrix}^T \tag{17}$$

The Discrete Fourier Transform

K-point DFT (analysis)

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\omega_k n}, \quad \omega_k = 2\pi k/K \tag{18}$$

- In many cases, $K = N$
- The DFT is a sampled DTFT of $x_N(n)$

The Discrete Fourier Transform

K-point DFT (analysis)

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\omega_k n}, \quad \omega_k = 2\pi k/K \tag{18}$$

- In many cases, $K = N$
- The DFT is a sampled DTFT of $x_N(n)$

K-point Inverse DFT (synthesis)

$$x(n) = \frac{1}{K} \sum_{k=0}^{K-1} X(k) e^{j2\pi kn/K} \tag{19}$$

The Discrete Fourier Transform

Matrix form of the DFT

The K-point DFT can be written as

$$\boldsymbol{X} = \boldsymbol{F}\boldsymbol{x} \tag{20}$$

where $\boldsymbol{x} \in \mathbb{C}^{N\times 1}$ and $\boldsymbol{F} \in \mathbb{C}^{K\times N}$ are the data vector and the DFT matrix, respectively, and given by

$$\boldsymbol{x} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-1) \end{bmatrix}^T \tag{21}$$

$$[\boldsymbol{F}]_{k+1,n+1} = \mathrm{e}^{-\mathrm{j}2\pi nk/K} \tag{22}$$

for $k = 0, 1, \ldots, K-1$ and $n = 0, 1, \ldots, N-1$.

The Discrete Fourier Transform

Inverse DFT

Since (for $K \geq N$)

$$\boldsymbol{F}^H \boldsymbol{F} = K \boldsymbol{I}_N \ , \tag{23}$$

we have that

$$K^{-1} \boldsymbol{F}^H \boldsymbol{X} = K^{-1} \boldsymbol{F}^H \boldsymbol{F} \boldsymbol{x} = K^{-1} K \boldsymbol{I}_N \boldsymbol{x} = \boldsymbol{x} \tag{24}$$

The Discrete Fourier Transform

Inverse DFT

Since (for $K \geq N$)

$$\boldsymbol{F}^H \boldsymbol{F} = K \boldsymbol{I}_N \,, \tag{23}$$

we have that

$$K^{-1} \boldsymbol{F}^H \boldsymbol{X} = K^{-1} \boldsymbol{F}^H \boldsymbol{F} \boldsymbol{x} = K^{-1} K \boldsymbol{I}_N \boldsymbol{x} = \boldsymbol{x} \tag{24}$$

Thus,

DFT $\boldsymbol{X} = \boldsymbol{F}\boldsymbol{x}$

iDFT $\boldsymbol{x} = K^{-1} \boldsymbol{F}^H \boldsymbol{X}$

The Discrete Fourier Transform

Linear convolution using the DFT

We wish to perform linear convolution between the two sequences $\{h(n)\}_{n=0}^{N_1-1}$ and $\{x(n)\}_{n=0}^{N_2-1}$.

$$y(n) = (h * x)(n) \tag{25}$$

The Discrete Fourier Transform

Linear convolution using the DFT

We wish to perform linear convolution between the two sequences $\{h(n)\}_{n=0}^{N_1-1}$ and $\{x(n)\}_{n=0}^{N_2-1}$.

$$y(n) = (h * x)(n) \tag{25}$$

1. Set the DFT-length to $K \geq N_1 + N_2 - 1$

The Discrete Fourier Transform

Linear convolution using the DFT

We wish to perform linear convolution between the two sequences $\{h(n)\}_{n=0}^{N_1-1}$ and $\{x(n)\}_{n=0}^{N_2-1}$.

$$y(n) = (h * x)(n) \tag{25}$$

1. Set the DFT-length to $K \geq N_1 + N_2 - 1$
2. Compute the K-point DFTs of $h(n)$ and $x(n)$

The Discrete Fourier Transform

Linear convolution using the DFT

We wish to perform linear convolution between the two sequences $\{h(n)\}_{n=0}^{N_1-1}$ and $\{x(n)\}_{n=0}^{N_2-1}$.

$$y(n) = (h * x)(n) \tag{25}$$

1. Set the DFT-length to $K \geq N_1 + N_2 - 1$
2. Compute the K-point DFTs of $h(n)$ and $x(n)$
3. Compute $Y(k) = H(k)X(k)$

The Discrete Fourier Transform

Linear convolution using the DFT

We wish to perform linear convolution between the two sequences $\{h(n)\}_{n=0}^{N_1-1}$ and $\{x(n)\}_{n=0}^{N_2-1}$.

$$y(n) = (h * x)(n) \tag{25}$$

1. Set the DFT-length to $K \geq N_1 + N_2 - 1$
2. Compute the K-point DFTs of $h(n)$ and $x(n)$
3. Compute $Y(k) = H(k)X(k)$
4. Compute the inverse DFT of $Y(k)$

The Discrete Fourier Transform

Linear convolution using the DFT

We wish to perform linear convolution between the two sequences $\{h(n)\}_{n=0}^{N_1-1}$ and $\{x(n)\}_{n=0}^{N_2-1}$.

$$y(n) = (h * x)(n) \tag{25}$$

1. Set the DFT-length to $K \geq N_1 + N_2 - 1$
2. Compute the K-point DFTs of $h(n)$ and $x(n)$
3. Compute $Y(k) = H(k)X(k)$
4. Compute the inverse DFT of $Y(k)$

If 1. is not satisfied, circular convolution is performed, unless either $h(n)$ or $x(n)$ are periodic in K.

The Discrete Fourier Transform

Linear convolution using linear algebra

Let $\boldsymbol{h}_{\text{zp}} \in \mathbb{C}^{K\times 1}$ and $\boldsymbol{x}_{\text{zp}} \in \mathbb{C}^{K\times 1}$ be zero-padded versions of $\boldsymbol{h} \in \mathbb{C}^{N_1\times 1}$ and $\boldsymbol{x} \in \mathbb{C}^{N_2\times 1}$.

The Discrete Fourier Transform

Linear convolution using linear algebra

Let $\boldsymbol{h}_{\text{zp}} \in \mathbb{C}^{K\times 1}$ and $\boldsymbol{x}_{\text{zp}} \in \mathbb{C}^{K\times 1}$ be zero-padded versions of $\boldsymbol{h} \in \mathbb{C}^{N_1\times 1}$ and $\boldsymbol{x} \in \mathbb{C}^{N_2\times 1}$. Then,

$$\boldsymbol{y} = K^{-1}\boldsymbol{F}^H \operatorname{diag}(\boldsymbol{F}\boldsymbol{h}_{\text{zp}})\boldsymbol{F}\boldsymbol{x}_{\text{zp}} = \boldsymbol{H}\boldsymbol{x}_{\text{zp}} \tag{26}$$

where $\boldsymbol{F}$ is a $K \times K$ DFT matrix and $\boldsymbol{H}$ is a convolution matrix

$$\boldsymbol{H} = \begin{bmatrix} h(0) & h(K-1) & & h(2) & h(1) \\ h(1) & \ddots & \ddots & & h(2) \\ & \ddots & \ddots & \ddots & \\ h(K-2) & & \ddots & \ddots & h(K-1) \\ h(K-1) & h(K-2) & & h(1) & h(0) \end{bmatrix} \tag{27}$$

The Discrete Fourier Transform

Example (Linear convolution)

- Let $\boldsymbol{h} = \begin{bmatrix} -1 & -2 & -3 \end{bmatrix}^T$ and $\boldsymbol{x} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$.

The Discrete Fourier Transform

Example (Linear convolution)

- Let $\boldsymbol{h} = \begin{bmatrix} -1 & -2 & -3 \end{bmatrix}^T$ and $\boldsymbol{x} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$.
- Then, $\boldsymbol{h}_{\text{zp}} = \begin{bmatrix} -1 & -2 & -3 & 0 \end{bmatrix}^T$ and $\boldsymbol{x}_{\text{zp}} = \begin{bmatrix} 1 & 2 & 0 & 0 \end{bmatrix}^T$.

The Discrete Fourier Transform

Example (Linear convolution)

- Let $\boldsymbol{h} = \begin{bmatrix} -1 & -2 & -3 \end{bmatrix}^T$ and $\boldsymbol{x} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$.
- Then, $\boldsymbol{h}_{\text{zp}} = \begin{bmatrix} -1 & -2 & -3 & 0 \end{bmatrix}^T$ and $\boldsymbol{x}_{\text{zp}} = \begin{bmatrix} 1 & 2 & 0 & 0 \end{bmatrix}^T$.
- Consequently,

$$\boldsymbol{y} = \begin{bmatrix} -1 & 0 & -3 & -2 \\ -2 & -1 & 0 & -3 \\ -3 & -2 & -1 & 0 \\ 0 & -3 & -2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ -2 & -1 \\ -3 & -2 \\ 0 & -3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -1 \\ -4 \\ -7 \\ -6 \end{bmatrix}$$

The Discrete Fourier Transform

The DFT vs the FFT

- Calculating

$$\boldsymbol{X} = \boldsymbol{F}\boldsymbol{x} \tag{28}$$

directly costs $\mathcal{O}(KN)$

The Discrete Fourier Transform

The DFT vs the FFT

- Calculating
$$\boldsymbol{X} = \boldsymbol{F}\boldsymbol{x} \tag{28}$$
directly costs $\mathcal{O}(KN)$
- Calculating $\boldsymbol{X}$ using an FFT algorithm costs $\mathcal{O}(K \log_2 K)$

The Discrete Fourier Transform

The DFT vs the FFT

- Calculating

$$\boldsymbol{X} = \boldsymbol{F}\boldsymbol{x} \tag{28}$$

  directly costs $\mathcal{O}(KN)$
- Calculating $\boldsymbol{X}$ using an FFT algorithm costs $\mathcal{O}(K \log_2 K)$
- Most FFT algorithms are working most efficiently when $\log_2(K)$ is an integer
- Most FFT algorithms are slow (relatively speaking) if K is prime or has large prime factors

The Discrete Fourier Transform

Summary

- The DFT is a sampled version of the DTFT of a window discrete-time signal
- The DFT operations can be modelled as a simple matrix-vector multiplication
- To perform linear convolution in the frequency domain, zero-padding is necessary, unless we are working with periodic signals

Questions?

AALBORG UNIVERSITY
DENMARK