NTNU – Trondheim
Norwegian University of
Science and Technology

TTT4120 Digital Signal Processing
Fall 2017

Lecture: Discrete Fourier Transform for Filtering and Frequency Analysis

Prof. Stefan Werner
stefan.werner@ntnu.3no
Office B329

Institutt for elektronikk og telekommunikasjon

Lecture in course book*

- Proakis, Manolakis Digital Signal Processing, 4th Ed.
  - 7.3.1 Use of DFT in linear filtering
  - 7.3.2 Filtering of long sequences (overlap and add method)
  - 7.4 Frequency analysis using DFT

*Level of detail is defined by lectures and problem sets

Preliminary questions

- The discrete-time Fourier transform (DTFT) allows us to perform frequency analysis of signals and filtering of signals

$$X(\omega), Y(\omega), \text{ and } H(\omega)$$

- What practical problems arise when applying the DTFT for these tasks?

Contents and learning outcomes

- Linear filtering using discrete Fourier transform (DFT)
- Filtering of long sequences (overlap-add)
- Frequency analysis using DFT

Linear filtering using DFT

- Remember (Lecture 3):

$$\begin{matrix} x[n] \\ X(\omega) \end{matrix} \longrightarrow \boxed{h[n]} \longrightarrow \begin{matrix} y[n] = h[n] * x[n] \\ Y(\omega) \ = H(\omega)\, X(\omega) \end{matrix}$$

- Convolution can sometimes be computationally demanding
- If we know $X(\omega)$ and $H(\omega)$, we can obtain $y[n]$ from

$$y[n] = \mathcal{F}^{-1}\{Y(\omega)\} \ = \mathcal{F}^{-1}\{H(\omega)\, X(\omega)\}$$

- Conceptually simpler
- How to implement these calculations on a computer?

5

Linear filtering using DFT…

- DFT can be implemented efficiently on a computer

$$\begin{matrix} x[n] \\ \color{red}{X(\omega_k)} \end{matrix} \longrightarrow \boxed{h[n]} \longrightarrow \begin{matrix} y[n] = h[n] * x[n] \\ \color{red}{\updownarrow ?} \\ \color{red}{Y(\omega_k) \ = H(\omega_k)\, X(\omega_k)?} \end{matrix}$$

- Can compute $X(k) = \mathrm{DFT}_N\{x[n]\}$ and $H(k) = \mathrm{DFT}_N\{h[n]\}$
- Convenient if $y[n]$ could be obtained from

$$y[n] = \mathrm{IDFT}_N\{Y(k)\} \ = \mathrm{IDFT}_N\{H(k)\, X(k)\}$$

- Not true in general but we investigate when it can be done

6

Linear filtering using DFT…

- Product of two DFTs corresponds to circular convolution

$$x_1[n] \otimes_N x_2[n] \xleftrightarrow{\text{DFT}_N} X_1(k)X_2(k)$$

- Not useful to compute output $y[n]$ of linear filter $h[n]$
- Assume finite-duration input sequence $x[n]$ and impulse response $h[n]$, i.e.,

$$x[n] = 0, n < 0 \text{ and } n \geq L$$
$$h[n] = 0, n < 0 \text{ and } n \geq M$$

- Output $y[n]$ can be calculated

$$y[n] = \sum_{n=0}^{N-1} h(k)x[n-k] \overset{\mathcal{F}}{\leftrightarrow} Y(\omega) = H(\omega)X(\omega)$$

7

Linear filtering using DFT…

- Output has finite duration $M + L - 1$

$$y[n] = 0, n < 0 \text{ and } n \geq M + L - 1$$

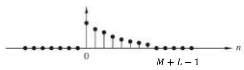
- We know from before that we can restore spectrum $Y(\omega)$ from its sampled spectrum $Y(\omega_k)$, $k = 0,1, \dots N-1$, if

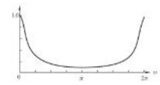
$$N \geq M + L - 1$$

∴ DFT of size $N \geq M + L - 1$ is required to uniquely represent $y[n]$ in frequency domain

8

Linear filtering using DFT…

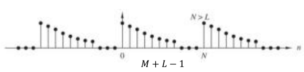


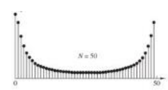


$y[n] \overset{DTFT}{\longleftrightarrow} Y(\omega)$

$\Uparrow$ $\Downarrow$ Sample at $\omega_k = \frac{2\pi k}{N}$

$y_p[n] = \sum_{l=-\infty}^{\infty} y[n-lN] \overset{DFT}{\longleftrightarrow} Y\left(\frac{2\pi k}{N}\right)$

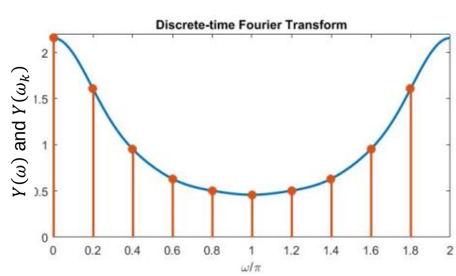




- Remember from last lecture, if $N \geq M + L - 1$,
  $y_p[n] = y[n]$ for $0 \leq n \leq N-1$

9

Linear filtering using DFT…



- In interval $0 \leq \omega \leq 2\pi$, take N equidistant samples,

$$Y(\omega_k) = Y(\omega)\big|_{\omega=\frac{2\pi k}{N}} = H(\omega)X(\omega)\big|_{\omega=\frac{2\pi k}{N}}, \; k = 0, \dots, N-1$$

$$\Rightarrow \boxed{Y(k) = H(k)X(k), \; k = 0, \dots, N-1}$$

N-point DFT of $h[n]$ N-point DFT of $x[n]$

10

Linear filtering using DFT…

- Since $x[n]$ and $h[n]$ have duration less than $N \Rightarrow$ need to *pad sequences with zeros to increase lengths to* $N \geq M + L - 1$

$$x[n] = \{x[0], x[1], \dots, x[L-1], \underbrace{0, \dots, 0}_{N-L}\}$$
$$h[n] = \{h[0], h[1], \dots, h[M-1], \underbrace{0, \dots, 0}_{N-M}\}$$

- Output sequence can now be computed as

$$y[n] = \mathrm{IDFT}_N\{Y(k)\} = \mathrm{IDFT}_N\{H(k)X(k)\}$$
$$= \mathrm{IDFT}_N\big\{\,\mathrm{DFT}_N\{h[n]\} \cdot \mathrm{DFT}_N\{h[n]\}\big\}$$

- Note that choosing $N < M + L - 1$ will lead to time-domain aliasing ($h[n] \otimes_N x[n] \neq h[n] * x[n]$)

11

Linear filtering using DFT…

- Example 1: Given $x[n] = \{\underline{1}, 2,2,1\}$, and $h[n] = \{\underline{1}, 2,3\}$.
  Which of the following calculations provide us with correct output sequence $y[n]$?

1. $y[n] = \mathrm{IDFT}_4\big\{\,\mathrm{DFT}_4\{x[n]\} \cdot \mathrm{DFT}_4\{h[n]\}\big\}$
2. $y[n] = \mathrm{IDFT}_3\big\{\,\mathrm{DFT}_3\{x[n]\} \cdot \mathrm{DFT}_3\{h[n]\}\big\}$
3. $y[n] = \mathrm{IDFT}_{16}\big\{\,\mathrm{DFT}_{16}\{x[n]\} \cdot \mathrm{DFT}_{16}\{h[n]\}\big\}$
4. $y[n] = \mathrm{IDFT}_6\big\{\,\mathrm{DFT}_6\{x[n]\} \cdot \mathrm{DFT}_6\{h[n]\}\big\}$

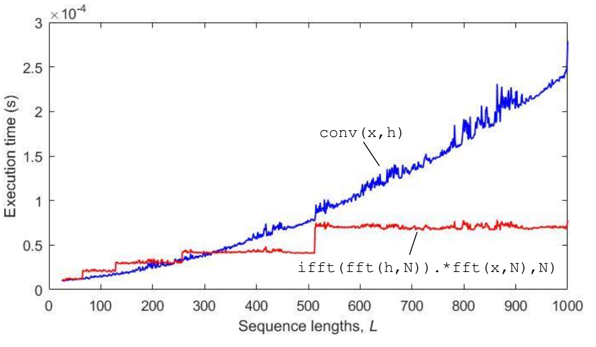
Matlab
```
N  = 4; % Try different N
x  = [1,2,2,1];
H  = [1,2,3];
y1 = ifft(fft(h,N).*fft(x,N),N)
y2 = conv(x,h)
```

12

Linear filtering using DFT…

- Example 2: When does frequency-domain filtering outperform time-domain filtering?

  Assume that both $x[n]$ and $h[n]$ have length L



Filtering of long sequences

$\{\ldots, x[0], x[1], x[2], \ldots, x[1000], x[1001], \ldots\}$ → $h[n]$ → $\{\ldots, y[0], y[1], y[2], \ldots, y[1000], y[1001], \ldots\}$

$X(k)$ $\qquad$ $Y(k) = X(k)H(k)$

- Assume that input sequence $x[n]$ is extremely long
- All N' input samples are required before we can perform DFT
- What are the implications on memory requirements and processing delay?

- Extreme case of real-time processing (no beginning or end)!

Filtering of long sequences…

$\{\dots, x[0], x[1], x[2], \dots, x[1000], x[1001], \dots\} \longrightarrow \boxed{h[n]} \longrightarrow \{\dots, y[0], y[1], y[2], \dots, y[1000], y[1001], \dots\}$

- All N' input samples are required before we can perform DFT
  $\Rightarrow$ Delay before output is produced increases with N'
- We need a method that can filter long sequences in time-domain that is memory- and delay-efficient
- Remember the *additivity property* of convolution

$$\begin{aligned} y[n] &= h[n] * (x_1[n] + x_2[n]) \\ &= h[n] * x_1[n] + h[n] * x_2[n] \\ &= y_1[n] + y_2[n] \end{aligned}$$

15

Filtering of long sequences…

$\{\dots, x[0], x[1], x[2], \dots, x[1000], x[1001], \dots\} \longrightarrow \boxed{h[n]} \longrightarrow \{\dots, y[0], y[1], y[2], \dots, y[1000], y[1001], \dots\}$

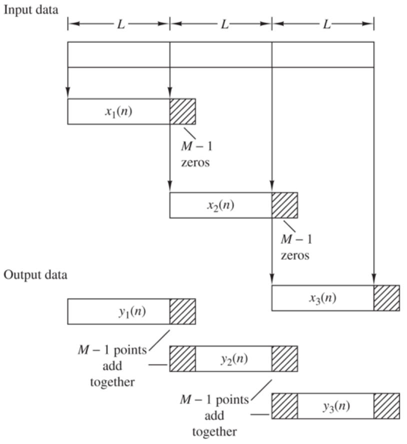
Strategy:

1. Divide input sequence $x[n]$ into *non-overlapping blocks* $x_m[n]$ *each of length* L
2. Filter each input block $x_m[n]$ to produce output block $y_m[n]$
3. Combine outputs: $y[n] = \sum_m y_m[n]$

- If length of $h[n]$ is M, the length of $y_m[n]$ is $L + M - 1$
  $\Rightarrow$ last $M - 1$ values of $y_{m-1}[n]$ added to beginning of $y_m[n]$

16

Filtering of long sequences…



- Filtering using N-point DFT requires zero-padding of sequences $x_m[n]$ and $h[n]$

17

Filtering of long sequences…

Steps of overlap-add:

1. Divide $x[n]$ into *non-overlapping* blocks $x_m[n]$ of length L
2. Pad $h[n]$ with zeros to length $N \geq M + L - 1$
3. Compute $H(k) = \mathrm{DFT}_N\ \{h[n]\}, k = 0, \ldots, N - 1$
4. For each block m:

   4.1 Pad $x_m[n]$ to with zeros to length $N \geq M + L - 1$

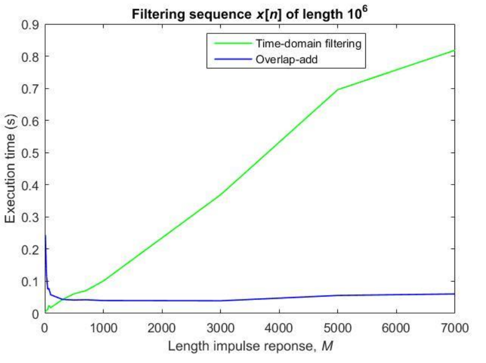
   4.2 Compute $X_m(k) = \mathrm{DFT}_N\ \{x_m[n]\}, k = 0, \ldots, N - 1$

   4.3 Multiply $Y_m(k) = H(k)X_m(k), k = 0, \ldots, N - 1$

   4.4 Compute $y_m[n] = \mathrm{IDFT}_N\ \{X_m(k)\}, n = 0, \ldots, N - 1$
5. Form $y[n]$ by overlapping and adding the last $M - 1$ values of $y_{m-1}[n]$ and the first $M - 1$ values of $y_m[n]$

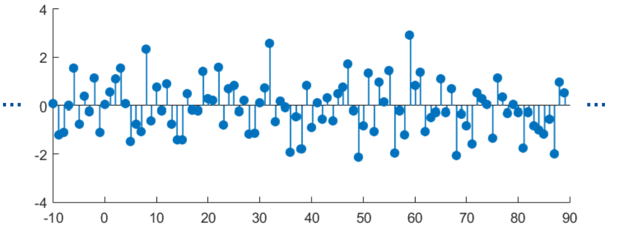
18

Filtering of long sequences…



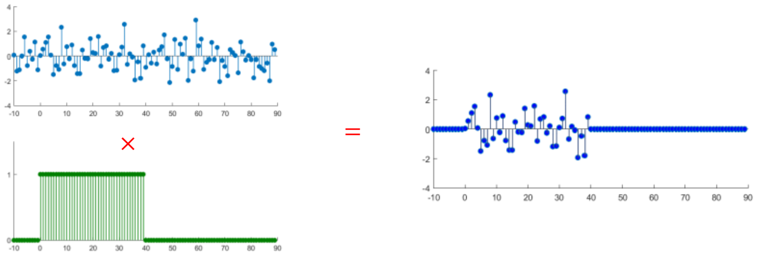

```
Matlab
M  = 2000; % Try different N
x = rand(1,1e6);
h = rand(1,M);
y1 = fftfilt(h,x);
y2 = filter(h,1,x);
```


Frequency analysis



- DTFT: $X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$
- In practice $x[n]$ needs to have finite duration
  $\Rightarrow$ Spectrum $X(\omega)$ approximated from a finite data record
- How does the approximation, $\hat{X}(\omega)$, depend on the number of available samples?

Frequency analysis…



- Limiting the number of samples is the same as multiplying original sequence $x[n]$ by a window $w[n]$

$$\hat{x}[n] = x[n]w[n]$$

where

$$w[n] = \begin{cases} 1 & \text{for } 0 \leq n \leq L-1 \\ 0 & \text{otherwise} \end{cases}$$

21

Frequency analysis…

- Multiplication in time-domain corresponds to

$$\hat{X}(\omega) = X(\omega) * W(\omega) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\theta)W(\omega-\theta)d\theta$$

- Using DFT we would get

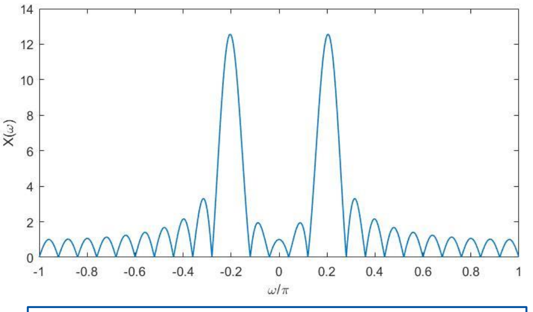
$$\hat{X}(k) = \sum_{n=0}^{N-1} \hat{x}[n]e^{-\frac{j2\pi k}{N}n} = \sum_{n=0}^{N-1} x[n]e^{-\frac{j2\pi k}{N}n}$$

$$= \hat{X}(\omega)\Big|_{\omega=\frac{2\pi k}{N}} = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\theta)W\left(\frac{2\pi k}{N}-\theta\right)d\theta$$

22

Frequency analysis…

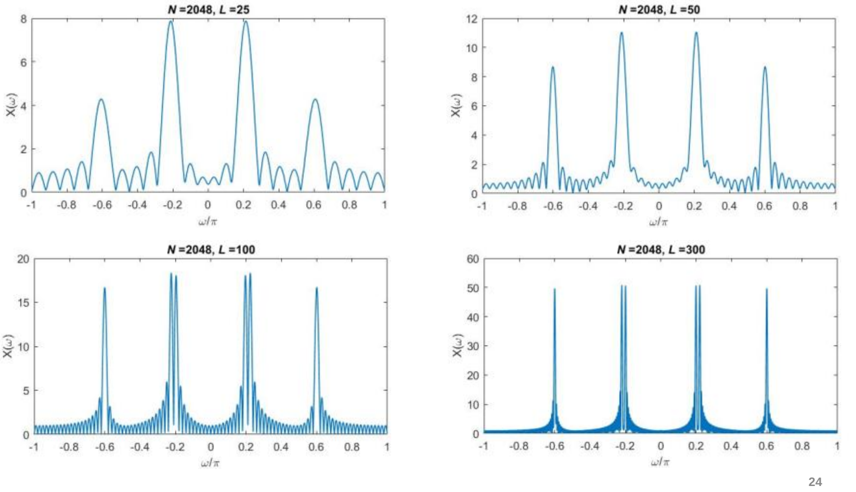
- Example: $x[n] = \cos 0.2\pi n$ for $N = 2048$ and $L = 25$



Matlab
```
N  = 2048; L  = 25;
n  = (0:N-1); k = (-1:2/N:1-2/N);
wn = [(L-n) > 0];
x  = cos(0.2*pi*n);
x_ = wn.*x;
plot(k,abs(fftshift(fft(x_hat,N))))
```


Frequency analysis…

- Example: $x[n] = \cos 0.2\pi n + \cos 0.22\pi n + \cos 0.6\pi n$

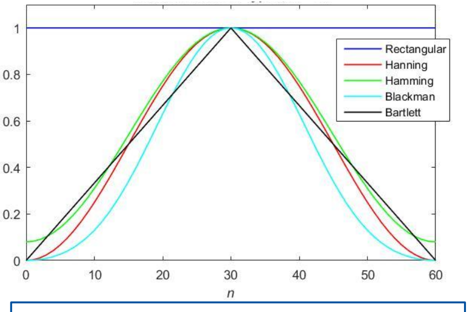


Frequency analysis …

- Windowing distorts the signal:
  - Spectrum peaks are smoothened out
  - Sidelobes are causing spectral leakage
- Increasing the window length, increases resolution
- Width of main lobe of rectangular window $4\pi/L$
- Use different windows to reduce spectral sidelobes
  - Width of main lobe is increasing when compared to rectangular window

Frequency analysis …

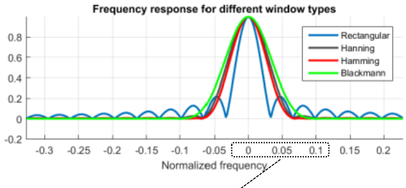
- Different window types, $L = 61$

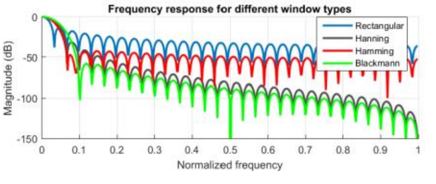


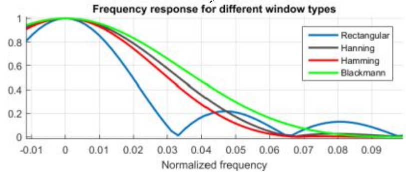
Matlab
```
% type 'help window' for options

L = 61; n = (0:L-1);
w1 = window(@hamming,L);
w2 = window(@bartlett,L);
plot(n,[w1,w2])
```

Frequency analysis…

- Frequency response for different window types





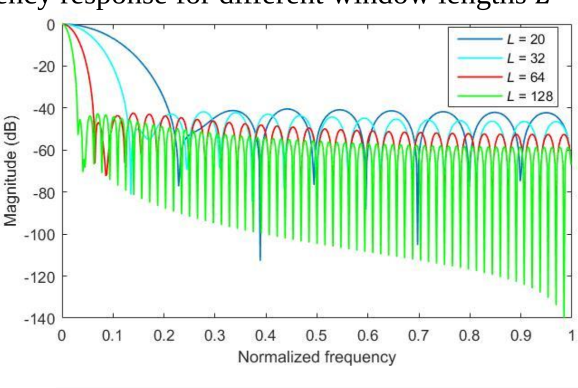


Matlab

```
L  = 61; wn = hamming(L);
[H,w] = freqz(wn);
plot(w/pi, abs(H)/max(abs(H)))
```


Frequency analysis…

- Frequency response for different window lengths L

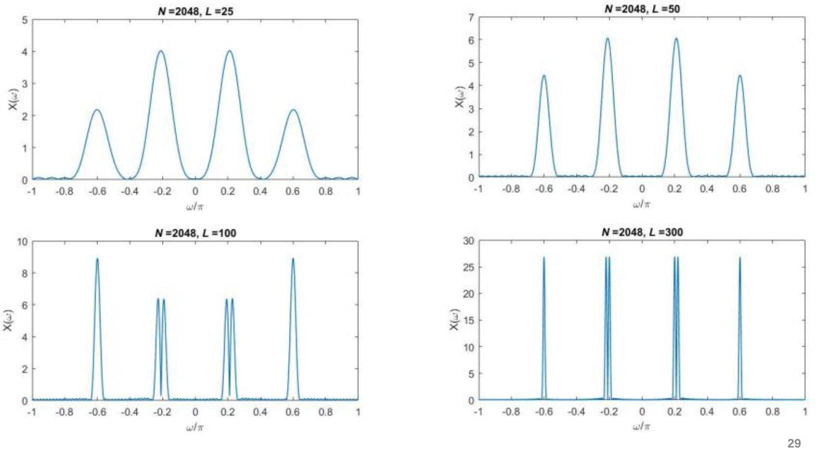


Matlab

```
L  = 20; wn = hamming(L);
[H,w] = freqz(wn);
plot(w/pi,20*log10(abs(H)/max(abs(H))))
```

Frequency analysis…

- Revisiting: $x[n] = \cos 0.2\pi n + \cos 0.22\pi n + \cos 0.6\pi n$ (Hamming window)



Frequency analysis …

- Increasing the window length, increases resolution
- Sidelobes are causing spectral leakage
- Width of main lobe versus sidelobe suppression
  - Use of different windows

Summary

- Today we discussed:
  - Filtering and frequency analysis using the DFT

- Next time:
  - Fast Fourier transform (FFT)