Advanced Digital Signal Processing

Part 4: DFT and FFT

Gerhard Schmidt

Christian-Albrechts-Universität zu Kiel
Faculty of Engineering
Institute of Electrical and Information Engineering
Digital Signal Processing and System Theory

Contents

Definitions

Basic definitions (assumed to be known from lectures about signals and systems):

The ***Discrete Fourier Transform*** (***DFT***):

$$v(n) \circ\!\!-\!\!\bullet\; V_M(\mu) \;=\; \sum_{n=0}^{M-1} v(n)\, e^{-j\frac{2\pi}{M}\mu n} \;=\; \sum_{n=0}^{M-1} v(n)\, W_M^{\mu n}.$$

The ***inverse Discrete Fourier Transform*** (***IDFT***):

$$V_M(\mu) \bullet\!\!-\!\!\circ\; v(n) \;=\; \frac{1}{M}\sum_{\mu=0}^{M-1} V_M(\mu)\, e^{j\frac{2\pi}{M}\mu n} \;=\; \frac{1}{M}\sum_{\mu=0}^{M-1} V_M(\mu)\, W_M^{-\mu n},$$

with the so-called ***twiddle factors***

$$W_M \;=\; e^{-j\frac{2\pi}{M}}$$

and M being the number of DFT points.

Linear and Circular Convolution – Part 1

Basic definitions of both types of convolutions

A ***linear convolution*** of two sequences $v_1(n)$ and $v_2(n)$ with $n \in \mathbb{Z}$ is defines as

$$\begin{aligned} y_{\text{lin}}(n) &= v_1(n) * v_2(n) &&= v_2(n) * v_1(n) \\ &= \sum_{\kappa=-\infty}^{\infty} v_1(\kappa)\, v_2(n-\kappa) &&= \sum_{\kappa=-\infty}^{\infty} v_2(\kappa)\, v_1(n-\kappa). \end{aligned}$$

A ***circular convolution*** of two periodic sequences $v_1(n)$ and $v_2(n)$ with $n \in \{0,\, 1 \ldots,\, M_{1,2} - 1\}$ and with the same period $M_1 = M_2 = M$ if defined as

$$\begin{aligned} y_{\text{cir}}(n) &= v_1(n) \circledast v_2(n) &&= v_2(n) \circledast v_1(n) \\ &= \sum_{\kappa=\kappa_0}^{\kappa_0+M-1} v_1(\kappa)\, v_2(n-\kappa) &&= \sum_{\kappa=\kappa_0}^{\kappa_0+M-1} v_2(\kappa)\, v_1(n-\kappa). \end{aligned}$$

The parameter κ_0 needs only to fulfill $\kappa_0 \in \mathbb{Z}$.

Linear and Circular Convolution – Part 2

The DFT and it's relation to circular convolution – Part 1:

The DFT is defined as the transform of the periodic signal with length M. Thus, we have

$$y(n) \circ\!\!-\!\!\bullet\ Y_M(\mu) \bullet\!\!-\!\!\circ\ y(n) \ = \ y(n+\lambda M).$$

Applying the DFT to a circular convolution leads to

$$\boxed{y(n) \ = \ v_1(n) \circledast v_2(n) \circ\!\!-\!\!\bullet\ Y_M(\mu) \ = \ V_{1,M}(\mu) \cdot V_{2,M}(\mu),}$$

with

$$v_1(n) \quad \circ\!\!-\!\!\bullet \quad V_{1,M}(\mu),$$

$$v_2(n) \quad \circ\!\!-\!\!\bullet \quad V_{2,M}(\mu).$$

This means that a circular convolution can be performed very efficiently (see next slides) in the DFT domain!

Linear and Circular Convolution – Part 3

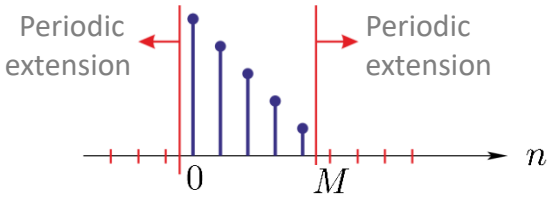
The DFT and it's relation to circular convolution – Part 2:

Proof of the DFT relation with the circular convolution on the blackboard …

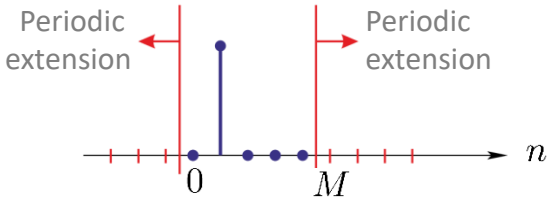
Linear and Circular Convolution – Part 4

Example:

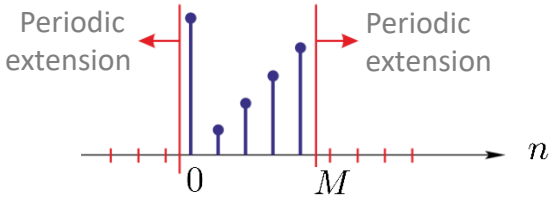
$v_1(n)$



$v_2(n) = \gamma_0(n-1)$

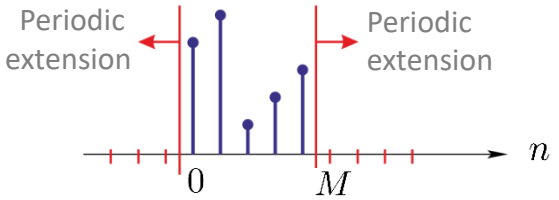


$v_1(0-n)$



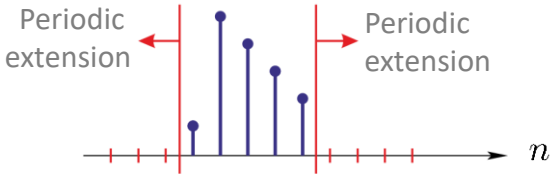
Due to the "single impulse behavior" of $v_2(n)$ the value at $n = 1$ is extracted and used at $y(0)$!

$v_1(1-n)$



Due to the "single impulse behavior" of $v_2(n)$ the value at $n = 1$ is extracted and used at $y(1)$!

$y(n) = v_1(n) \circledast v_2(n)$



Linear Filtering in the DFT Domain – Part 1

DFT and linear convolution for finite-length sequences – Part 1

Basic ideas

- The ***filtering operation*** can also be carried out in the ***frequency domain*** using the DFT.
  This is very attractive, since fast algorithms (fast Fourier transforms) exist.
- The DFT only realizes a ***circular convolution***. However, the desired operation for linear filtering is linear convolution.
  How can this be achieved by means of the DFT?

Given a finite-length sequence $v_1(n)$ with length M_1 and $v_2(n)$ with length M_2:

- The ***linear convolution*** is defined as:

$$y(n) = \sum_{\kappa=0}^{M_1-1} v_1(\kappa)\, v_2(n-\kappa),$$

  with a length $M_1 + M_2 - 1$ of the convolution result $y(n)$.
- The ***frequency domain equivalent*** is

$$Y(e^{j\Omega}) = V_1(e^{j\Omega})\, V_2(e^{j\Omega}).$$

Linear Filtering in the DFT Domain – Part 2

DFT and linear convolution for finite-length sequences – Part 2

Given a finite-length sequence $v_1(n)$ with length M_1 and $v_2(n)$ with length M_2 (continued):

- In order to represent the sequence $y(n)$ uniquely in the frequency domain by samples of its spectrum $Y(e^{j\Omega})$, the number of samples must be equal or exceed $M_1 + M_2 - 1$. Thus, a DFT of size $M \geq M_1 + M_2 - 1$ is required.
- Then, the DFT of the linear convolution

$$y(n) = v_1(n) * v_2(n)$$

is

$$Y_M(\mu) = V_{1,M}(\mu) \cdot V_{2,M}(\mu).$$

Explanation on the blackboard (if required) …

This ***result can be summarized as follows***:

- The circular convolution of two sequences $v_1(n)$ with length M_1 and $v_2(n)$ with length M_2 leads to the same result as the linear convolution $v_1(n) * v_2(n)$ when the lengths of $v_1(n)$ and $v_2(n)$ are increased to $M_1 + M_2 - 1$ by ***zero padding***.

Linear Filtering in the DFT Domain – Part 3

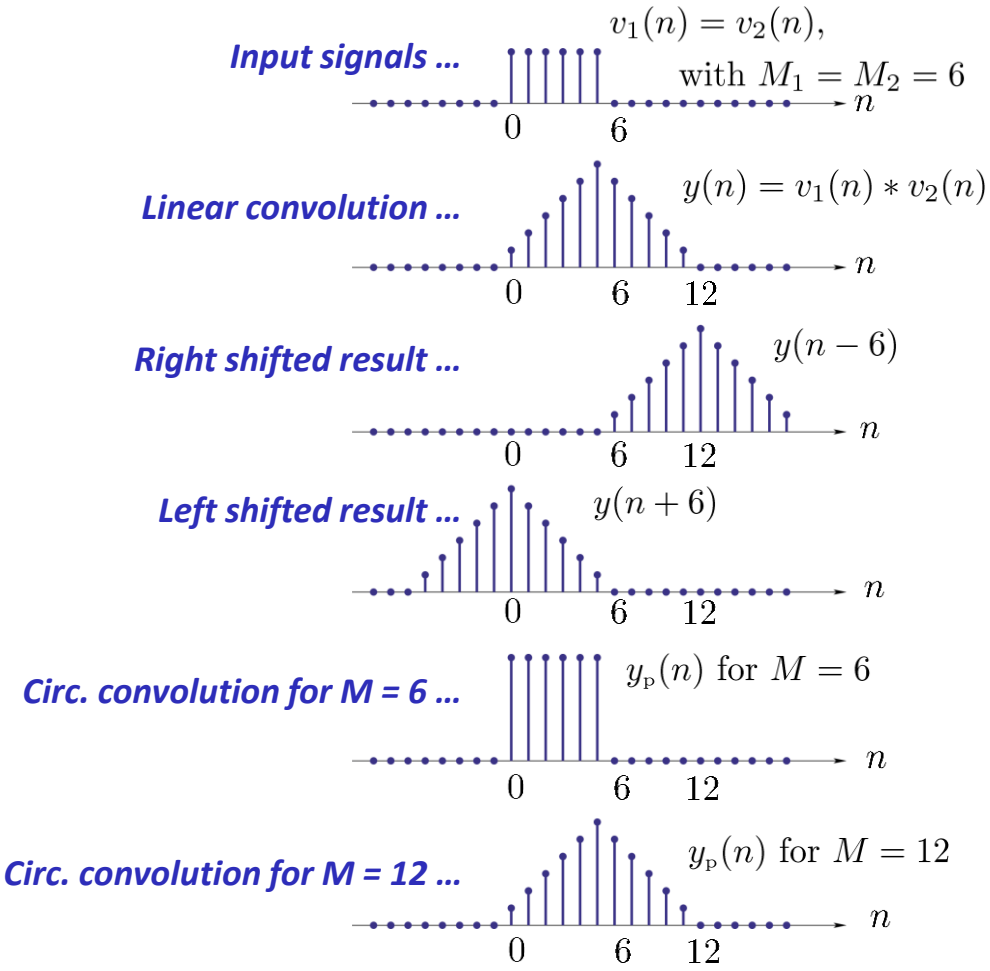
DFT and linear convolution for finite-length sequences – Part 3

Alternative interpretation:

- The circular convolution can be interpreted as a ***linear convolution with aliasing***.
- The inverse DFT leads to the following sequence in the time-domain:

$$y_{\mathrm{p}}(n) = \begin{cases} \sum_{\lambda=-\infty}^{\infty} y(n-\lambda M), \\ \qquad \text{if } 0 \leq n < M, \\ 0, \\ \qquad \text{else.} \end{cases}$$

- For clarification, see example on the right.



Linear Filtering in the DFT Domain – Part 4

DFT and linear convolution for infinite or long sequences – Part 1

Basic objective:

- Filtering a long input signal $v(n)$ with a finite impulse response $h(n)$ of length M_2:

$$y(n) = \sum_{\kappa=0}^{M_2-1} h(\kappa)\, v(n-\kappa).$$

First possible realization: the ***overlap-add method***

- Segment the input signal into separate (non-overlapping) blocks:

$$v_\nu(n) = \begin{cases} v(n-\nu M_1), & \text{if } 0 \le n < M_1, \\ 0, & \text{else.} \end{cases}$$

- Apply zero-padding for the signal blocks $v_\nu(n) \rightarrow \tilde{v}_\nu(n)$ and for the impulse response $h(n) \rightarrow \tilde{h}(n)$ to obtain a block length $M \ge M_1 + M_2 - 1$. The non-segmented input signal can be reconstructed according to

$$v(n) = \sum_{\nu=-\infty}^{\infty} \tilde{v}_\nu(n-\nu M_1).$$

DFT and linear convolution for infinite or long sequences – Part 2

First possible realization: the ***overlap-add method*** (continued)

- Compute M-point DFTs of $\tilde{v}_\nu(n)$ and $\tilde{h}(n)$ (need to be done only once) and multiply the results:

$$Y_{\nu,M}(\mu) = \widetilde{V}_{\nu,M}(\mu)\,\widetilde{H}_M(\mu), \quad \text{with } \mu \in \{0, \ldots, M-1\}.$$

- The M–point inverse DFT $y_\nu(n) = \mathrm{IDFT}\{Y_{\nu,M}(\mu)\}$ yields data blocks that are ***free from aliasing due to the zero padding*** applied before.
- Since each input data block $v_\nu(n)$ is terminated with $M - M_1$ zeros, the last $M - M_1$ signal samples from each output block $y_\nu(n)$ must be ***overlapped*** with (added to) the first $M - M_1$ signal samples of the succeeding block (linearity of convolution):

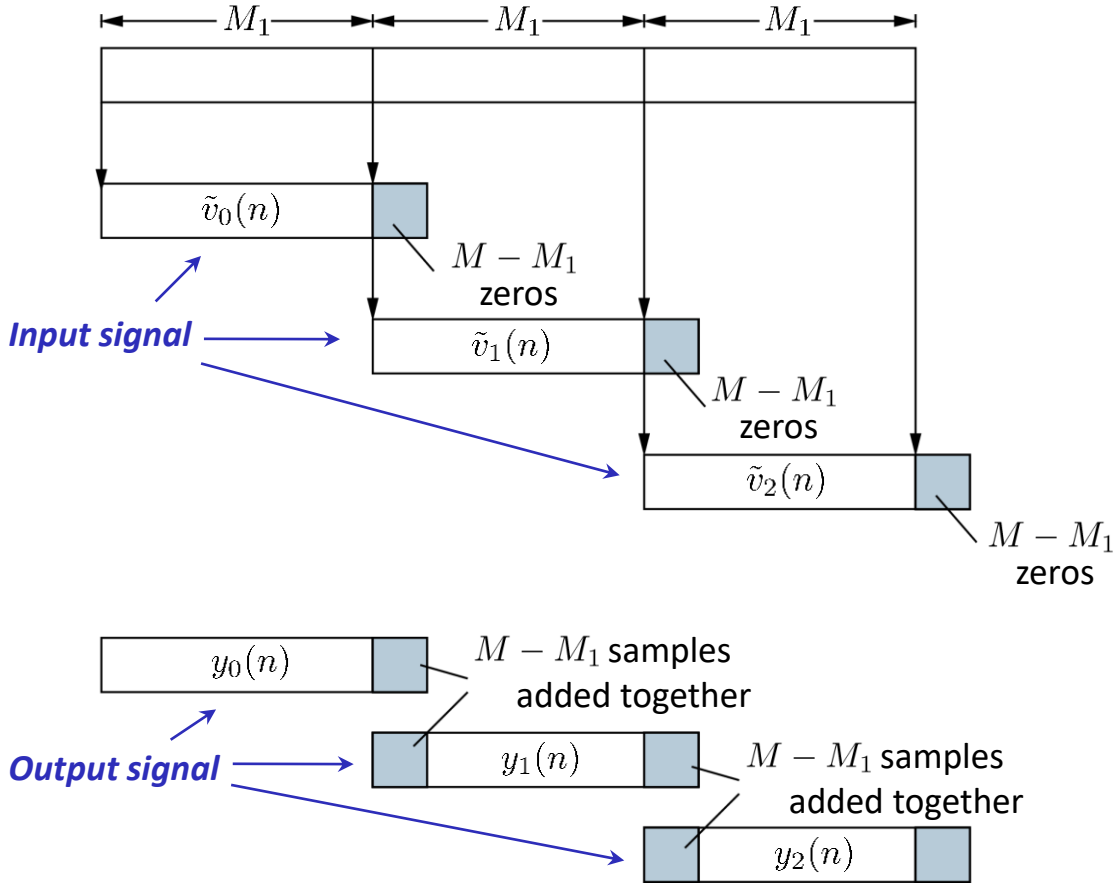
$$y(n) = \sum_{\nu=-\infty}^{\infty} \tilde{y}_\nu(n - \nu M_1).$$

As we will see later on, this can result in an immense reduction in computational complexity (compared to the direct time-domain realization) since efficient computations of the DFT and IDFT exist.

Linear Filtering in the DFT Domain – Part 6

DFT and linear convolution for infinite or long sequences – Part 3

First possible realization: the ***overlap-add method*** (continued)



Linear Filtering in the DFT Domain – Part 7

DFT and linear convolution for infinite or long sequences – Part 4

Second possible realization: the ***overlap-save method***

- Segment the input signal into blocks of length M with an overlap of length $M - M_1$:

$$v_\nu(n) = \begin{cases} v(n - \nu M_1), & \text{if } 0 \leq n < M, \\ 0, & \text{else.} \end{cases}$$

- Apply zero-padding for the impulse response $h(n) \rightarrow \tilde{h}(n)$ to obtain a block length $M \geq M_1 + M_2 - 1$.
- Compute M–point DFTs of $v_\nu(n)$ and $\tilde{h}(n)$ (need to be done only once) and multiply the results:

$$Y_{\nu,M}(\mu) = V_{\nu,M}(\mu)\, \widetilde{H}_M(\mu), \quad \text{with } \mu \in \{0, \ldots, M-1\}.$$

Linear Filtering in the DFT Domain – Part 8

DFT and linear convolution for infinite or long sequences – Part 5

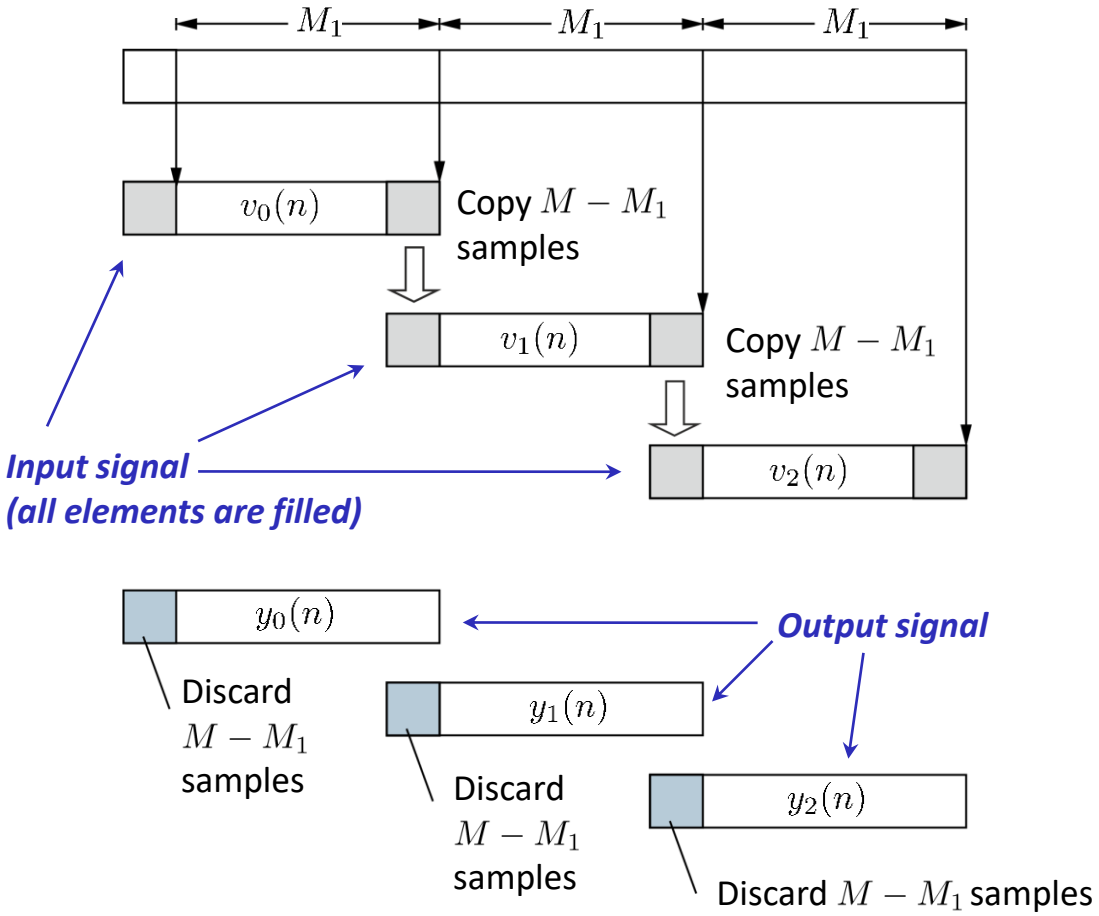
Second possible realization: the ***overlap-save method*** (continued)

- The M–point inverse DFT $y_\nu(n) = \mathrm{IDFT}\{Y_{\nu,M}(\mu)\}$ yields data blocks of length M with aliasing in the first $M - M_1$ samples. These samples must be ***discarded***. The last $M_1 = M - M_2 - 1$ samples of $y_\nu(n)$ are exactly the same as the result of a linear convolution.
- In order to avoid the loss of samples due to aliasing the last $M - M_1$ samples are saved and ***appended*** at the beginning of the next block. The processing is started by setting the first $M - M_1$ samples of the first block to zero.

Linear Filtering in the DFT Domain – Part 9

DFT and linear convolution for infinite or long sequences – Part 6

Second possible realization: the ***overlap-save method*** (continued)



Linear Filtering in the DFT Domain – Part 10

DFT and linear convolution for infinite or long sequences – Part 7

Partner work – Please think about the following questions and try to find answers
(first group discussions, afterwards broad discussion in the whole group).

- What are the differences between the overlap-add and the overlap-save method?

  ……

  ……

- Are there advantages and disadvantages?

  ……

  ……

- Can you think of applications where you would prefer either overlap-save or overlap-add? Please explain your choice!

  ……

  ……

Linear Filtering in the DFT Domain – Part 11

Frequency analysis of stationary signals – Leakage effect – Part 1

Preprocessing for spectral analysis of an analog signal $v(t)$ in practice:

- ***Anti-aliasing lowpass filtering*** and sampling with $\omega_s > 2\omega_b$, ω_b denoting the cut-off frequency of the signal.
- For practical purposes (delay, complexity):
  Limitation of the signal duration to the time interval $T_0 = LT$
  (L: number of samples under consideration, T: sampling interval).

Consequence of the duration limitation:

- The limitation to a signal duration of T_0 can be modeled as multiplication of the sampled input signal $v(n)$ with a rectangular ***window*** $w(n)$:

$$\hat{v}(n) = v(n) \cdot w(n) \quad \text{with} \quad w(n) = \begin{cases} 1, & \text{if } 0 \leq n \leq L-1, \\ 0, & \text{else.} \end{cases}$$

Linear Filtering in the DFT Domain – Part 12

Frequency analysis of stationary signals – Leakage effect – Part 2

The ***consequence of the duration limitation*** is shown using an example:

Suppose that the input sequence consists of a single sinusoid

$$v(n) \;=\; \cos\left(\Omega_0 n\right).$$

The Fourier transform is

$$V(e^{j\Omega}) \;=\; \pi\Big[\delta_0(\Omega-\Omega_0)+\delta_0(\Omega+\Omega_0)\Big].$$

The Fourier transform of the window function $w(n)$ can be obtained as

$$W(e^{j\Omega}) \;=\; \sum_{n=0}^{L-1} e^{-j\Omega n} \;=\; \frac{1-e^{-j\Omega L}}{1-e^{-j\Omega}} \;=\; e^{-j\Omega(L-1)/2}\cdot\frac{\sin(\Omega L/2)}{\sin(\Omega/2)}.$$

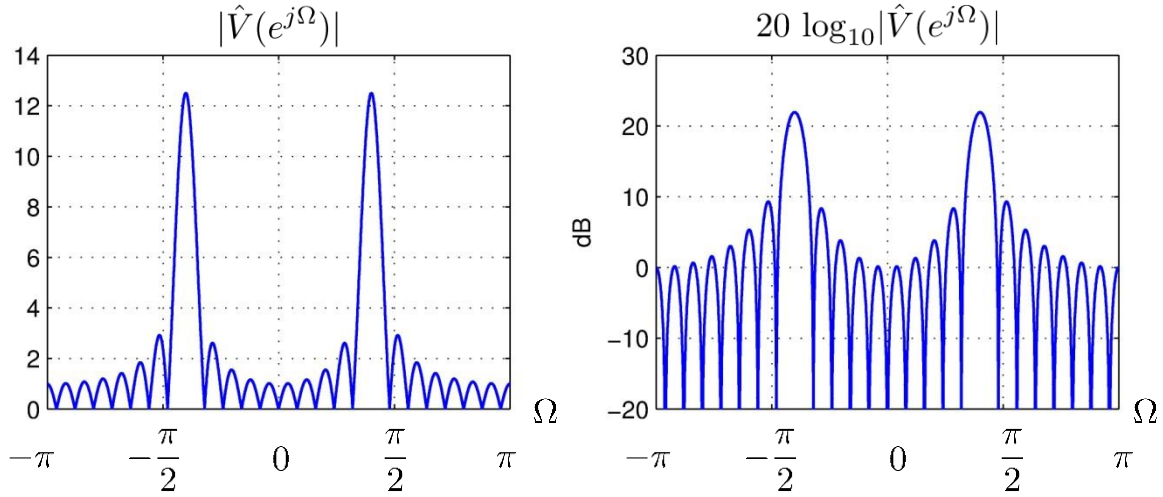
The Fourier transform of the windowed sequence $\hat{v}(n)$ is

$$\begin{aligned}\hat{V}(e^{j\Omega}) \;&=\; \frac{1}{2\pi}\Big[V(e^{j\Omega}) \circledast W(e^{j\Omega})\Big]\\ &=\; \frac{1}{2}\Big[W\big(e^{j(\Omega-\Omega_0)}\big)+W\big(e^{j(\Omega+\Omega_0)}\big)\Big].\end{aligned}$$

Linear Filtering in the DFT Domain – Part 13

Frequency analysis of stationary signals – Leakage effect – Part 3

Magnitude frequency response $|\hat{V}(e^{j\Omega})|$ for $L = 25$ and $\Omega_0 = 0,2 \cdot 2\pi$:



The windowed spectrum $\hat{V}(e^{j\Omega})$ is not localized to the frequency of the cosine $v(n)$ any more.
It is spread out over the whole frequency range.

$\Rightarrow$ This is called "***spectral leaking***".

Linear Filtering in the DFT Domain – Part 14

Frequency analysis of stationary signals – Leakage effect – Part 4

Properties of the rectangle windowing:

- First zero crossing of $W(e^{j\Omega})$ at $\Omega_z = \pm 2\pi/L$
- $\Rightarrow$ The larger the number of sampling points L (and thus also the width of the rectangular window) the smaller becomes Ω_z (and thus the main lobe of the frequency response).
- $\Rightarrow$ Decreasing the frequency resolution (making the window width smaller) leads to an increase of the time resolution and vice versa.
  Duality of time and frequency domain.

Practical scope of the DFT:

Use of the DFT in order to obtain a sampled representation of the spectrum according to

$$\widehat{V}(e^{j\Omega_\mu}), \quad \Omega_\mu = \frac{2\pi}{M}\mu, \quad \mu = 0, ..., M-1.$$

Linear Filtering in the DFT Domain – Part 15

Frequency analysis of stationary signals – Leakage effect – Part 5

Special case: If $M = L \quad \text{and} \quad \Omega = \frac{2\pi}{M}\nu, \quad \nu = 0, ..., M-1$

then the Fourier transform is exactly zero at the sampled frequencies Ω_μ except for $\mu = \nu$.

Example: $M = 50, \; n = 0, \ldots, M-1$, rectangular window $w(n)$

$$v_0(n) = \cos\left(1.0\frac{2\pi}{M}n\right), \quad v_1(n) = \cos\left(1.2\frac{2\pi}{M}n\right).$$

Results:

- DFT of $\hat{v}_0(n) = v_0(n)\, w(n)$:

$$\widehat{V}_0(e^{j\Omega}) = \frac{1}{2\pi}\left[V_0(e^{j\Omega}) \circledast W(e^{j\Omega})\right] = 0 \quad \text{for } \Omega_\mu = \mu 2\pi/M$$

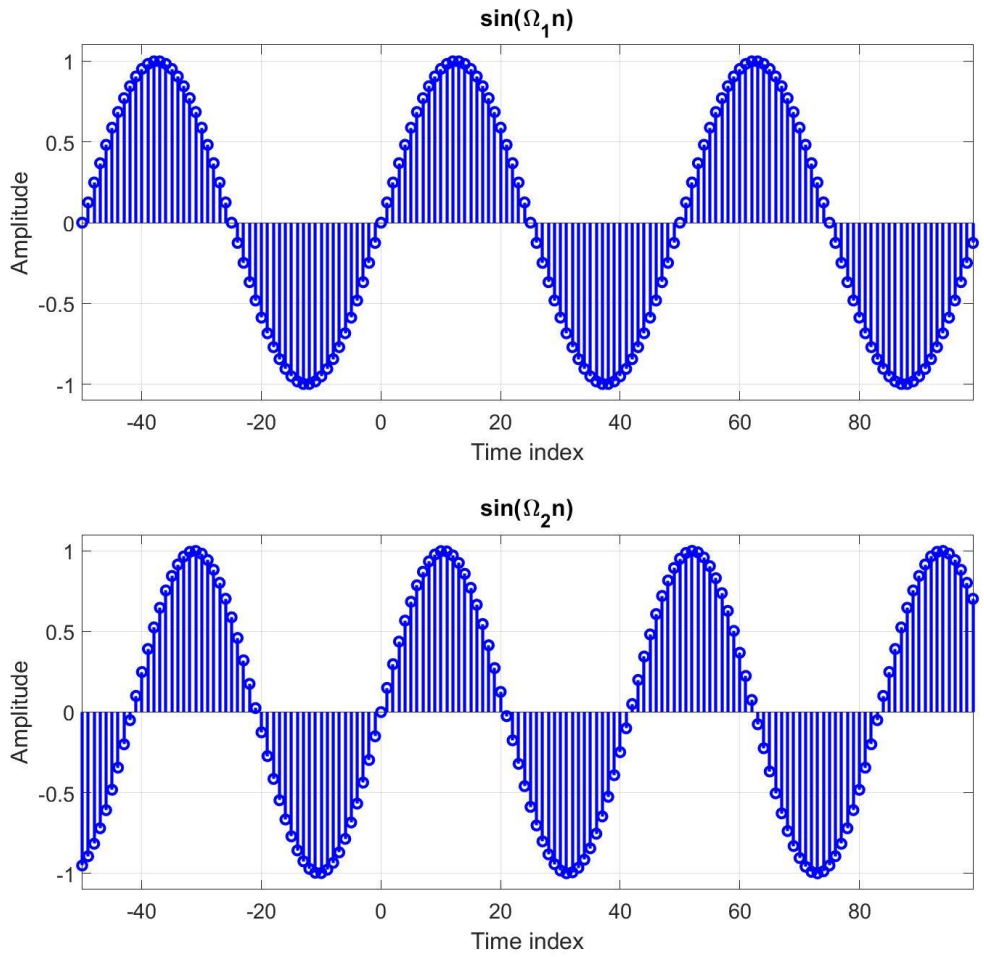
  except for $\mu = 1$ since Ω_0 is exactly an integer multiple of $2\pi/M$.
  $\Rightarrow$ The periodic repetition of $v_0(n)$ leads to a pure cosine.

- DFT of $v_1(n)$: $\Omega_\nu \neq \nu 2\pi/M \quad \Rightarrow \widehat{V}(e^{j\Omega_\mu}) \neq 0$ for $\Omega_\mu = \mu 2\pi/M$
  $\Rightarrow$ The periodic repetition of $v_1(n)$ is not a cosine sequence anymore.

Linear Filtering in the DFT Domain – Part 16

Frequency analysis of stationary signals – Leakage effect – Part 6

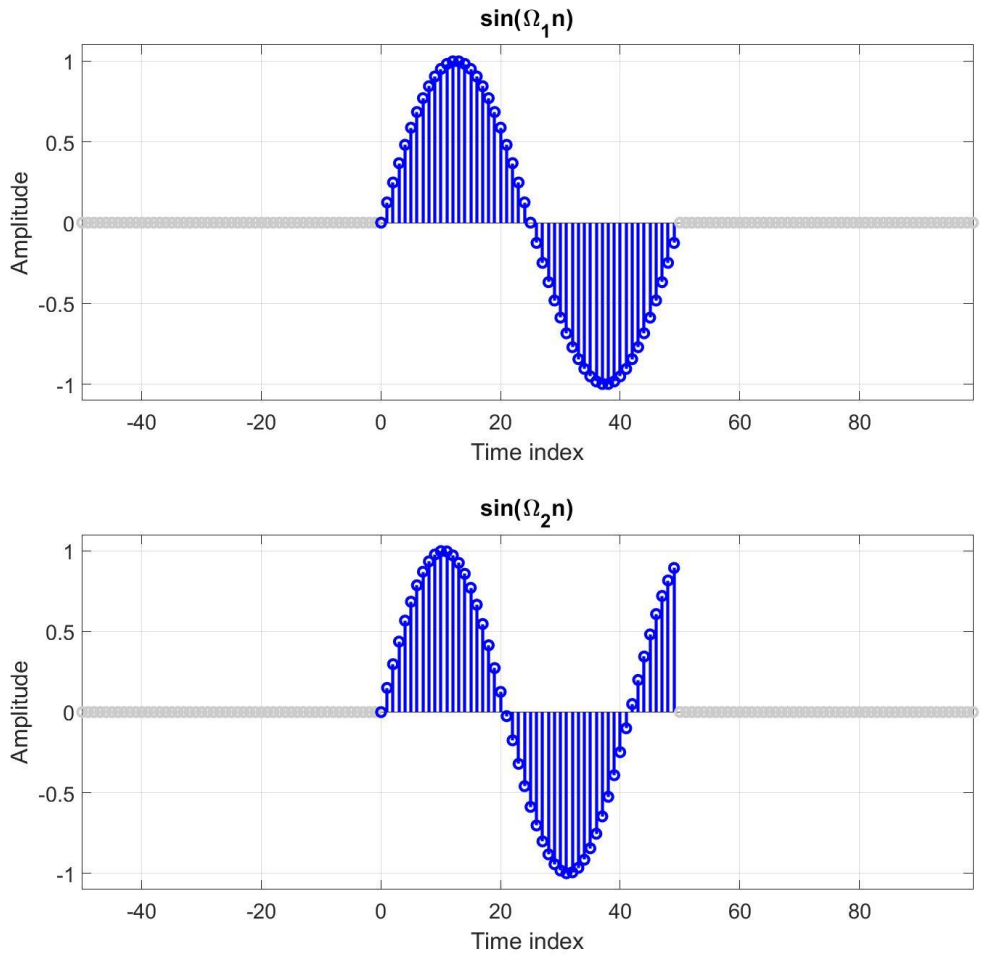
Example (continued):



Linear Filtering in the DFT Domain – Part 16

Frequency analysis of stationary signals – Leakage effect – Part 7

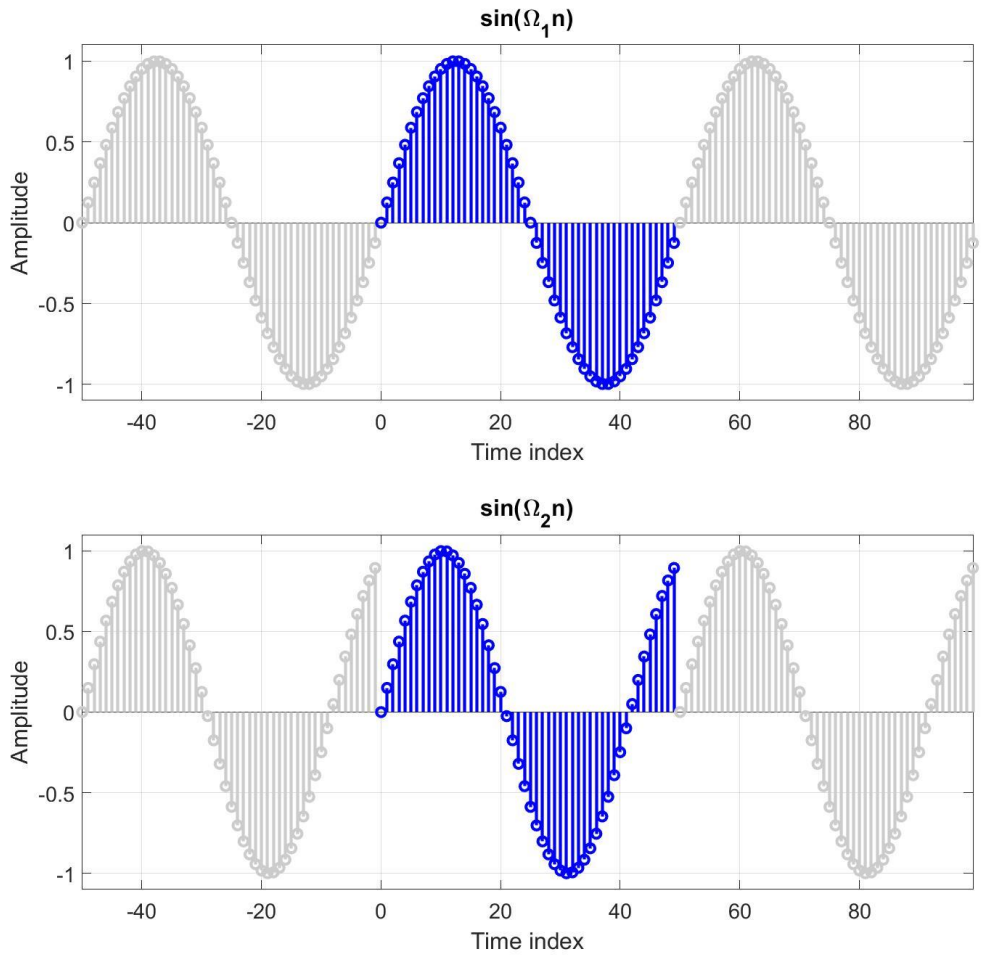
Example (continued):



Linear Filtering in the DFT Domain – Part 16

Frequency analysis of stationary signals – Leakage effect – Part 8

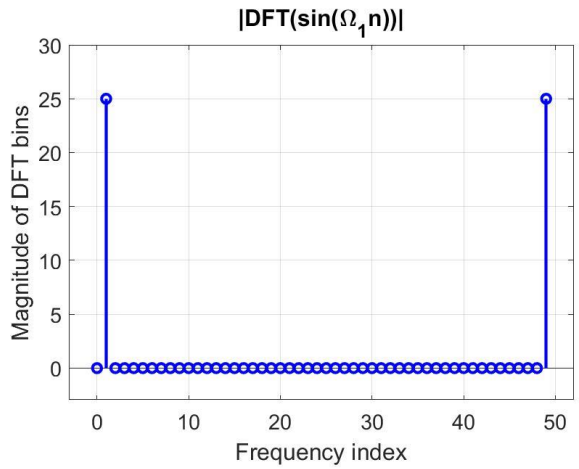
Example (continued):

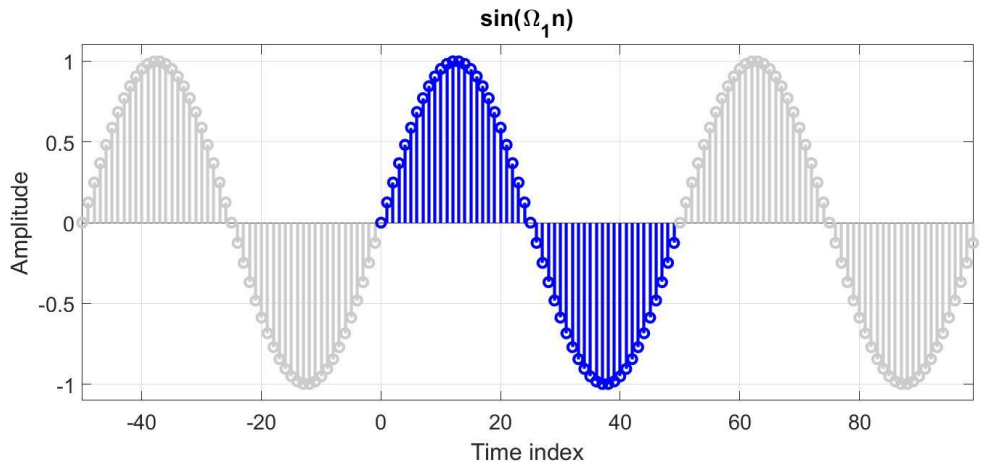


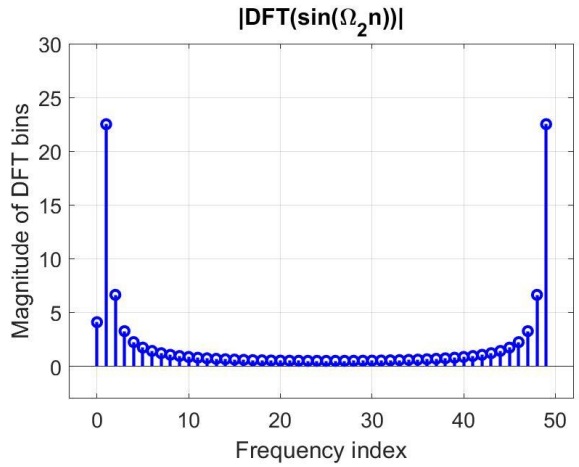
Linear Filtering in the DFT Domain – Part 16

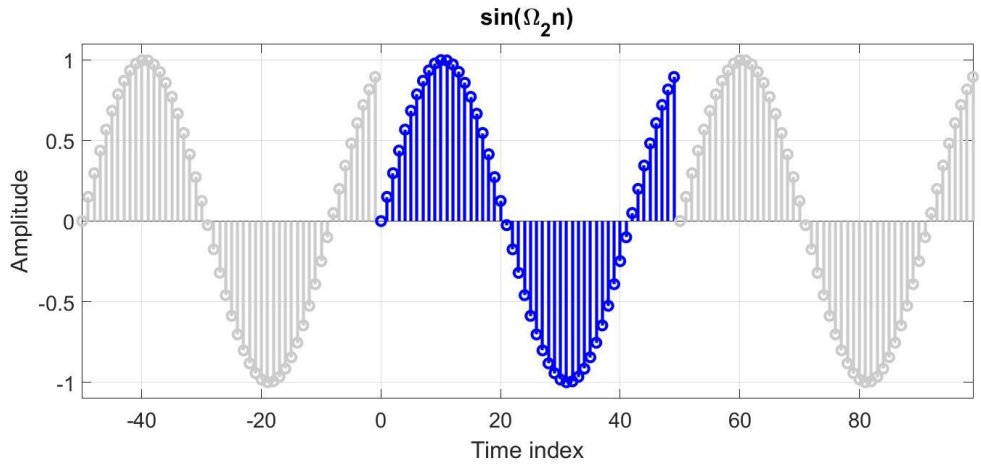
Frequency analysis of stationary signals – Leakage effect – Part 9

Example (continued):









Linear Filtering in the DFT Domain – Part 17

Frequency analysis of stationary signals – Leakage effect – Part 10

Partner work – Please think about the following questions and try to find answers
(first group discussions, afterwards broad discussion in the whole group).

- Why is the spectrum of a signal that you analyze using a DFT “widened” and “smeared” in general?

  ……

  ……

- What can you do in order to minimize the effect?

  ……

  ……

- Why is a longer sequence length not always the better choice?

  ……

  ……

Linear Filtering in the DFT Domain – Part 18

Frequency analysis of stationary signals – Windowing – Part 1

Windowing not only distorts the spectral estimate due to leakage effects, it also influences the spectral resolution.

First example:

Consider a sequence of two frequency components

$$\begin{aligned} v(n) &= \cos(\Omega_1 n) + \cos(\Omega_2 n), \\ \hat{v}(n) &= v(n) \cdot w(n) \\ \widehat{V}(e^{j\Omega}) &= \frac{1}{2}\left[W(e^{j(\Omega-\Omega_1)}) + W(e^{j(\Omega-\Omega_2)}) + W(e^{j(\Omega+\Omega_1)}) + W(e^{j(\Omega+\Omega_2)})\right] \end{aligned}$$

where $W(e^{j\Omega})$ is the Fourier transform of the rectangular window $w(n)$.

Linear Filtering in the DFT Domain – Part 19

Frequency analysis of stationary signals – Windowing – Part 2

Consideration of three cases for the relation between Ω_1, Ω_2 and L:

- $2\pi/L < |\Omega_1 - \Omega_2|$: The two maxima (the main lobes) for both window spectra $W(e^{j(\Omega-\Omega_1)})$ and $W(e^{j(\Omega-\Omega_2)})$ can be separated.
- $2\pi/L = |\Omega_1 - \Omega_2|$: Correct values for the spectral samples, but the main lobes cannot be separated anymore.
- $2\pi/L > |\Omega_1 - \Omega_2|$: The main lobes of $W(e^{j(\Omega-\Omega_1)})$ and $W(e^{j(\Omega-\Omega_2)})$ overlap.

The ability to resolve spectral lines of different frequencies is limited by the main lobe width, which also depends on the length of the window impulse response L.

Linear Filtering in the DFT Domain – Part 20

Frequency analysis of stationary signals – Windowing – Part 3

Second example:

Depicted (on the next slide) are the frequency responses $|\widehat{V}(e^{j\Omega})|$ for

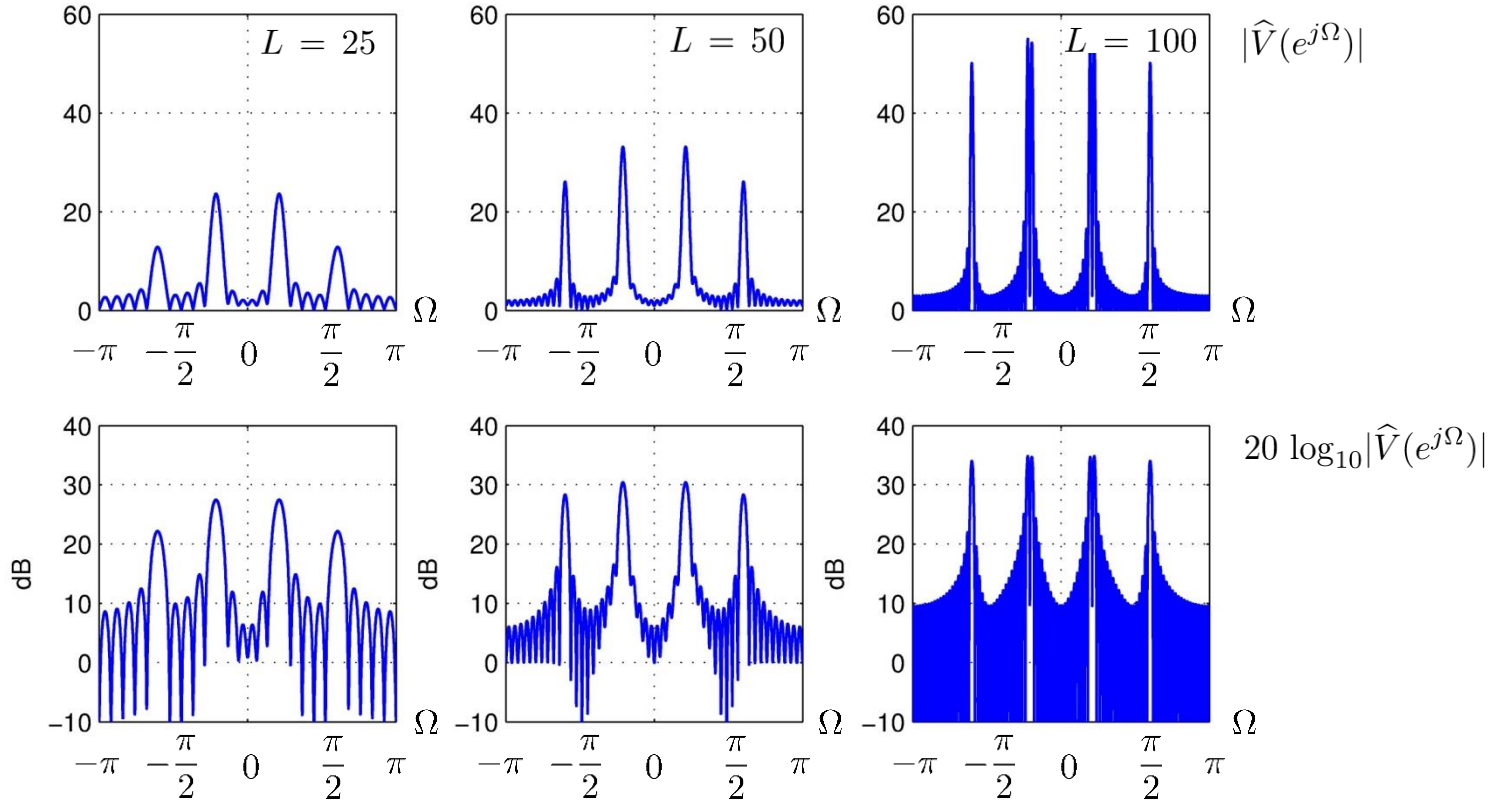
$$\hat{v}(n) \;=\; \Big[\cos(\Omega_0 n) + \cos(\Omega_1 n) + \cos(\Omega_2 n)\Big]\, w(n)$$

with $\Omega_0 = 0.2\pi,\ \Omega_1 = 0.22\pi$ and $\Omega_2 = 0.6\pi$ for different window lengths L .

Linear Filtering in the DFT Domain – Part 21

Frequency analysis of stationary signals – Windowing – Part 4

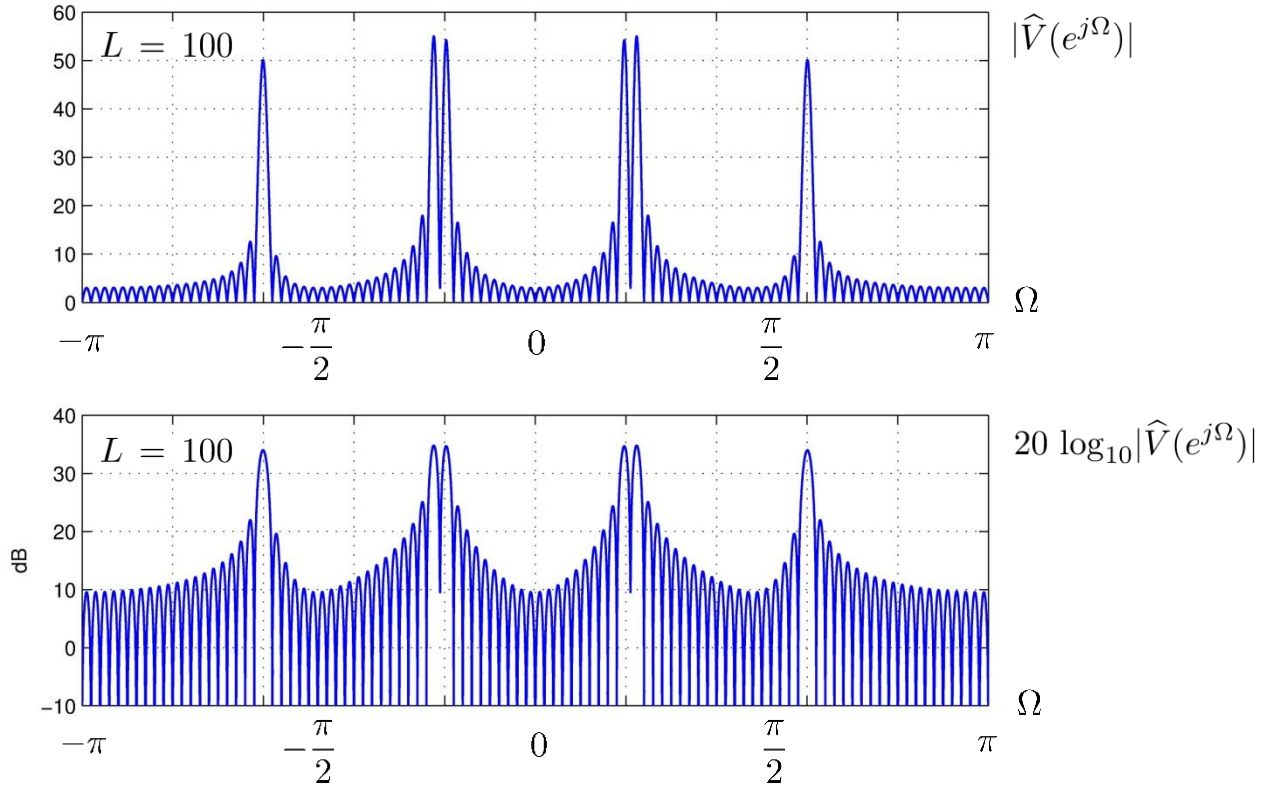
Second example (continued):



Linear Filtering in the DFT Domain – Part 22

Frequency analysis of stationary signals – Windowing – Part 5

Second example (continued):



Linear Filtering in the DFT Domain – Part 23

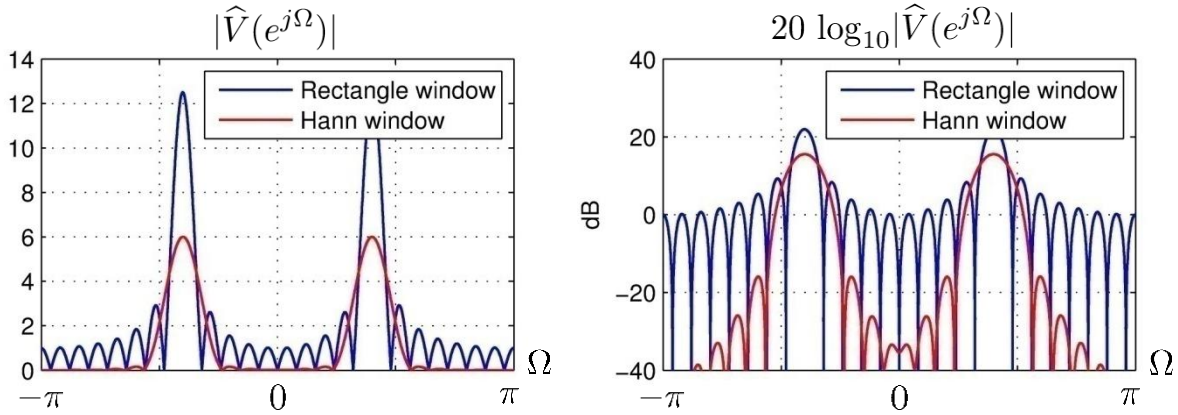
Frequency analysis of stationary signals – Windowing – Part 6

Approach to reduce leakage: Other window functions with lower side lobes (however, this comes with an increase of the width of the main lobe).

One possible (often used) window*:* the ***Hann window***, defined as

$$w_{\mathrm{Han}}(n) = \begin{cases} \frac{1}{2}\left[1 - \cos\left(\frac{2\pi}{L-1}n\right)\right], & \text{if } 0 \le n \le L-1, \\ 0, & \text{else.} \end{cases}$$

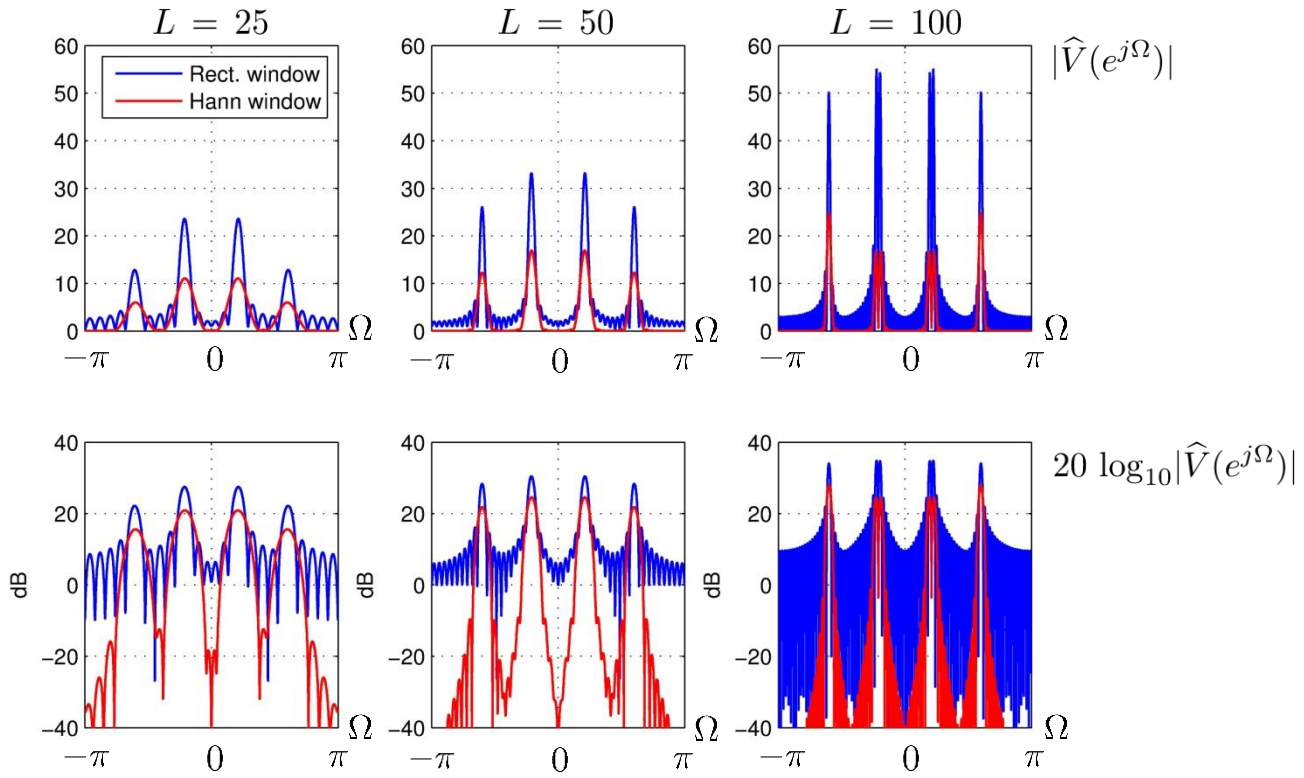
Magnitude frequency response $|\widehat{V}(e^{j\Omega})|$ of the cosine-function $v(n) = \cos(\Omega_0 n)$ after windowing with the ***Hann window***:



Linear Filtering in the DFT Domain – Part 24

Frequency analysis of stationary signals – Windowing – Part 7

Spectrum of the signal $v(n) = \cos(\Omega_0 n) + \cos(\Omega_1 n) + \cos(\Omega_2 n)$ after windowing with the ***Hann window***:

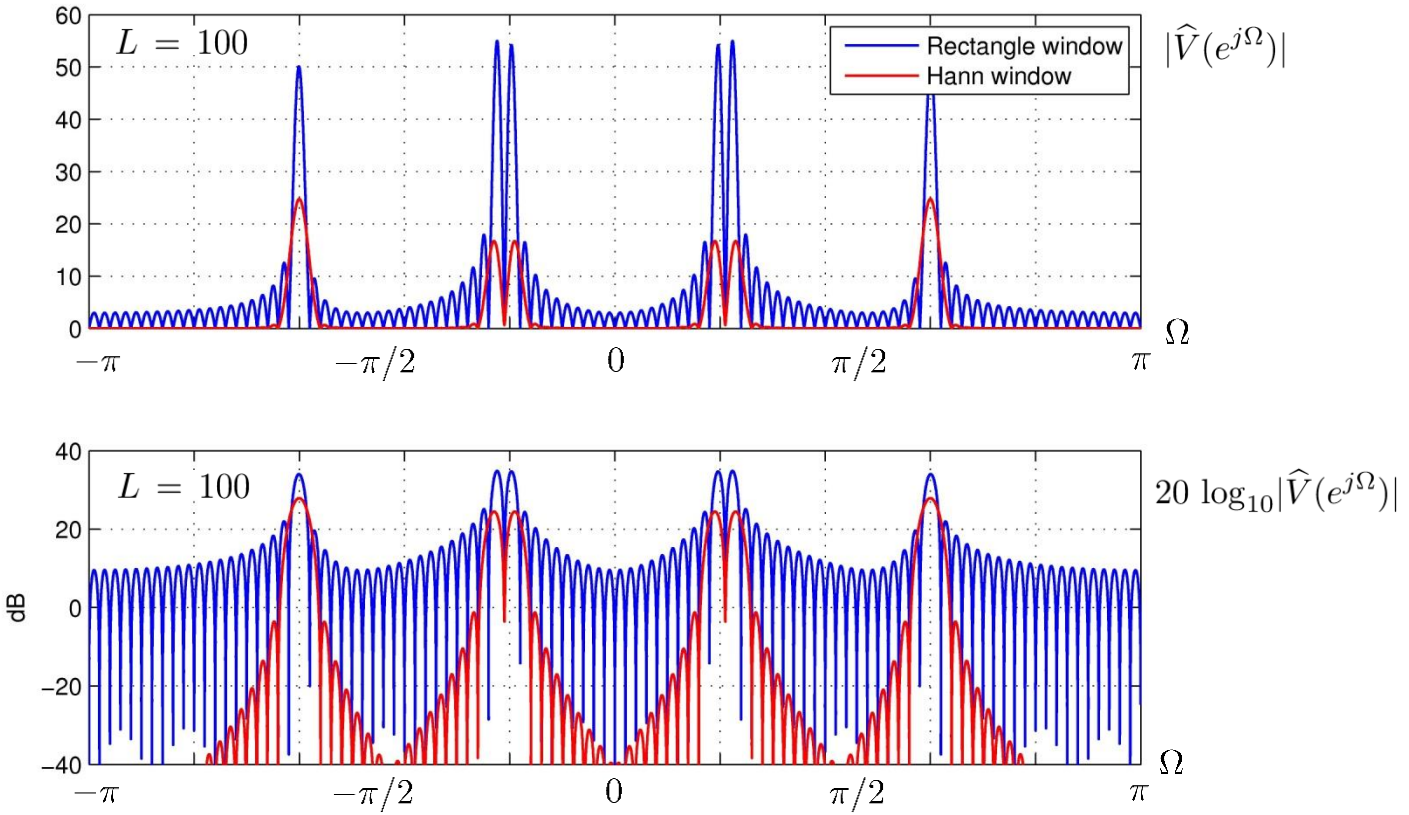


The reduction of the sidelobes and the reduced resolution compared to the rectangular window can be clearly observed.

Linear Filtering in the DFT Domain – Part 25

Frequency analysis of stationary signals – Windowing – Part 8

Spectrum of the signal $v(n) = \cos(\Omega_0 n) + \cos(\Omega_1 n) + \cos(\Omega_2 n)$ after windowing with the ***Hann window*** (continued):



Linear Filtering in the DFT Domain – Part 26

Frequency analysis of stationary signals – Comparison of window sequences – Part 1

In the following we will ***compare different window sequences***. Before we start, some ***global remarks***:

- All windows will have the ***same length***: $L = 50$ (which means that 51 coefficients are used).
- The windows will be ***centered around 0***, meaning that they start at index -25 and end at index 25.
- From time to time the so-called ***Dirichlet kernel*** will be used. This an abbreviation for the following function:

$$\begin{aligned} D_k(\Omega) &= \frac{1}{2\pi} \sum_{m=-k}^{k} e^{j\Omega m} \\ &= \frac{1}{2\pi} \frac{\sin\left(\frac{k+1}{2}\Omega\right)}{\sin\left(\frac{\Omega}{2}\right)}. \end{aligned}$$

- In the following all ***time sequences*** are ***normalized*** to have ***maximum amplitude of one***.
- The ***magnitude frequency responses*** are ***normalized*** to have ***unity gain at frequency zero***.

Johann Peter Gustav Lejeune Dirichlet, German mathematician, 1805 – 1859 (Source: Wikipedia)

Linear Filtering in the DFT Domain – Part 27

Frequency analysis of stationary signals – Comparison of window sequences – Part 2

Some ***global remarks*** (continued):

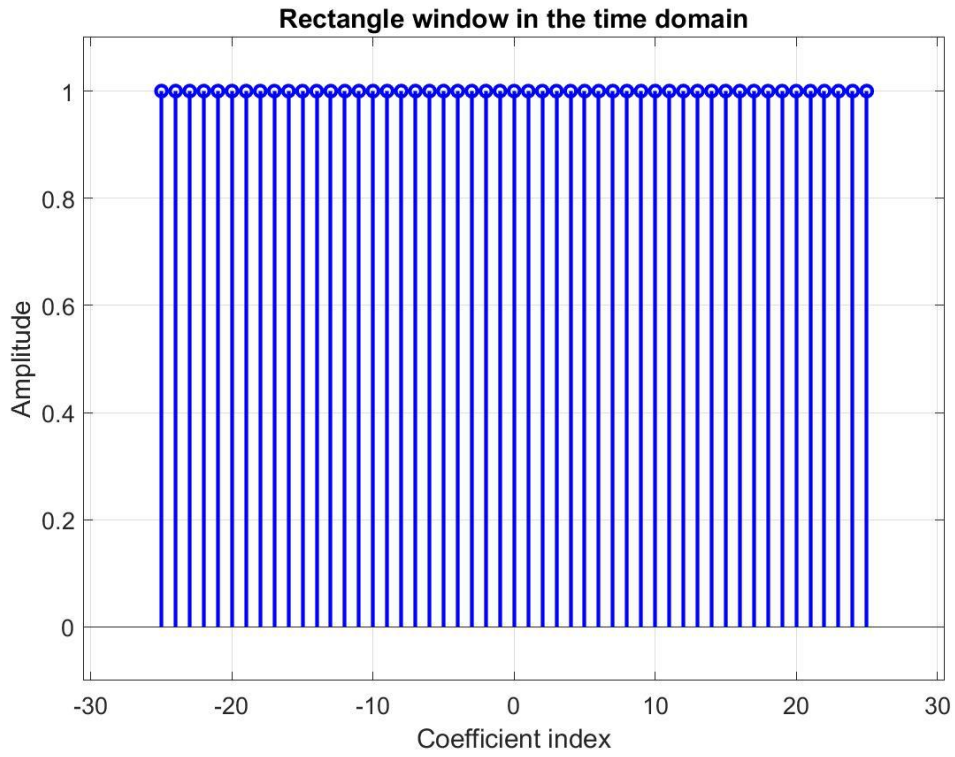
- Creation of window sequences:
  - One can start with basic sequences (sin/cos, Gauss function, polynomials, …)
  - One can combine basic sequences (by means of addition, multiplication, convolution, …)
- Desired shapes of sequences (their corresponding transforms):
  - Small main lobe / large attenuation of side lobes (most of the time)
  - Sometimes also a certain width of the main lobe (SONAR, RADAR, …)
  - Addition of shifted (und multiplied) window sequences (overlap-add filterbanks, …)
  - Small aliasing components (if combined with subsampling in filterbanks, …)

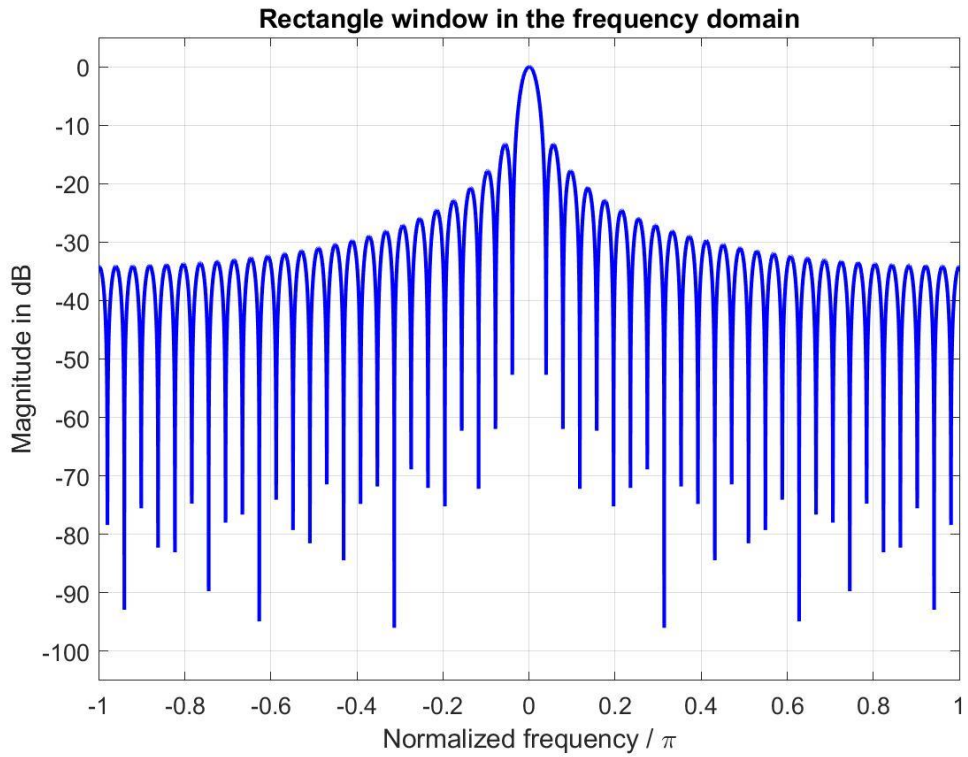
The more window sequences you know, the better …

Linear Filtering in the DFT Domain – Part 28

Frequency analysis of stationary signals – Comparison of window sequences – Part 3

The simplest window is a rectangle sequence.

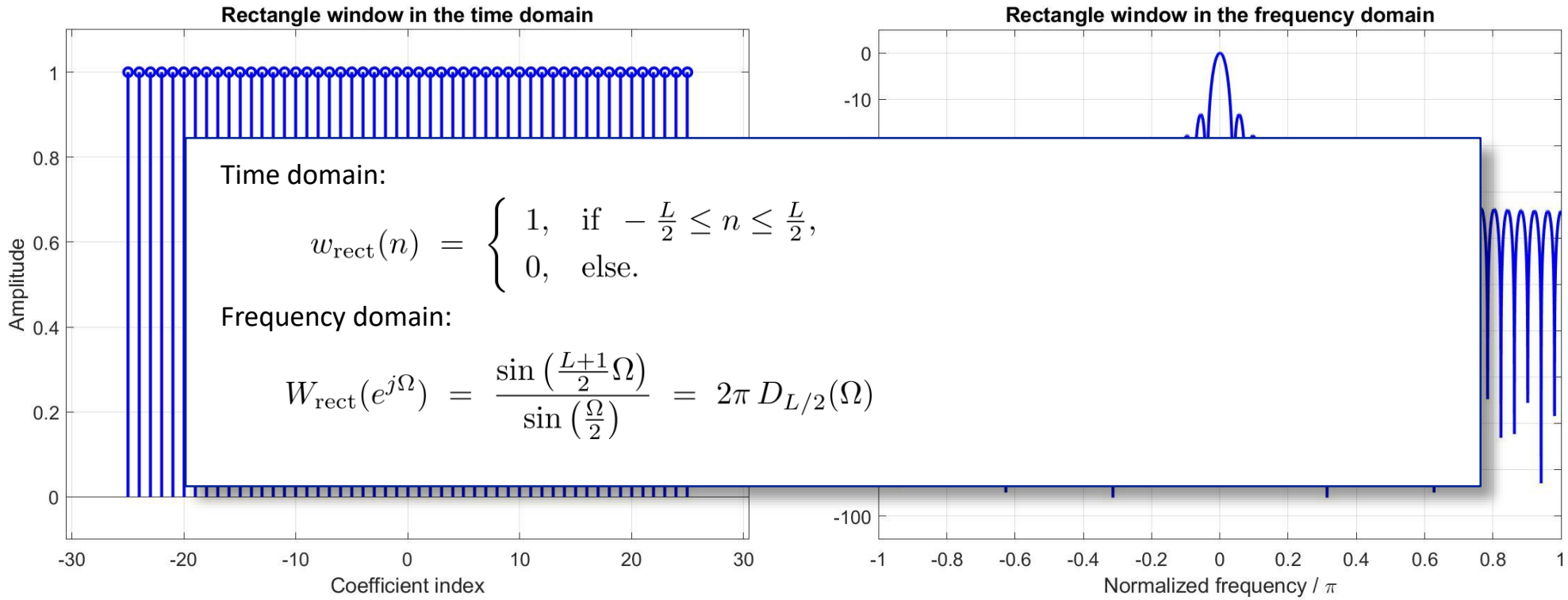




Linear Filtering in the DFT Domain – Part 29

Frequency analysis of stationary signals – Comparison of window sequences – Part 3

The simplest window is a rectangle sequence.



Time domain:

$$w_{\mathrm{rect}}(n) = \begin{cases} 1, & \text{if } -\frac{L}{2} \le n \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

Frequency domain:

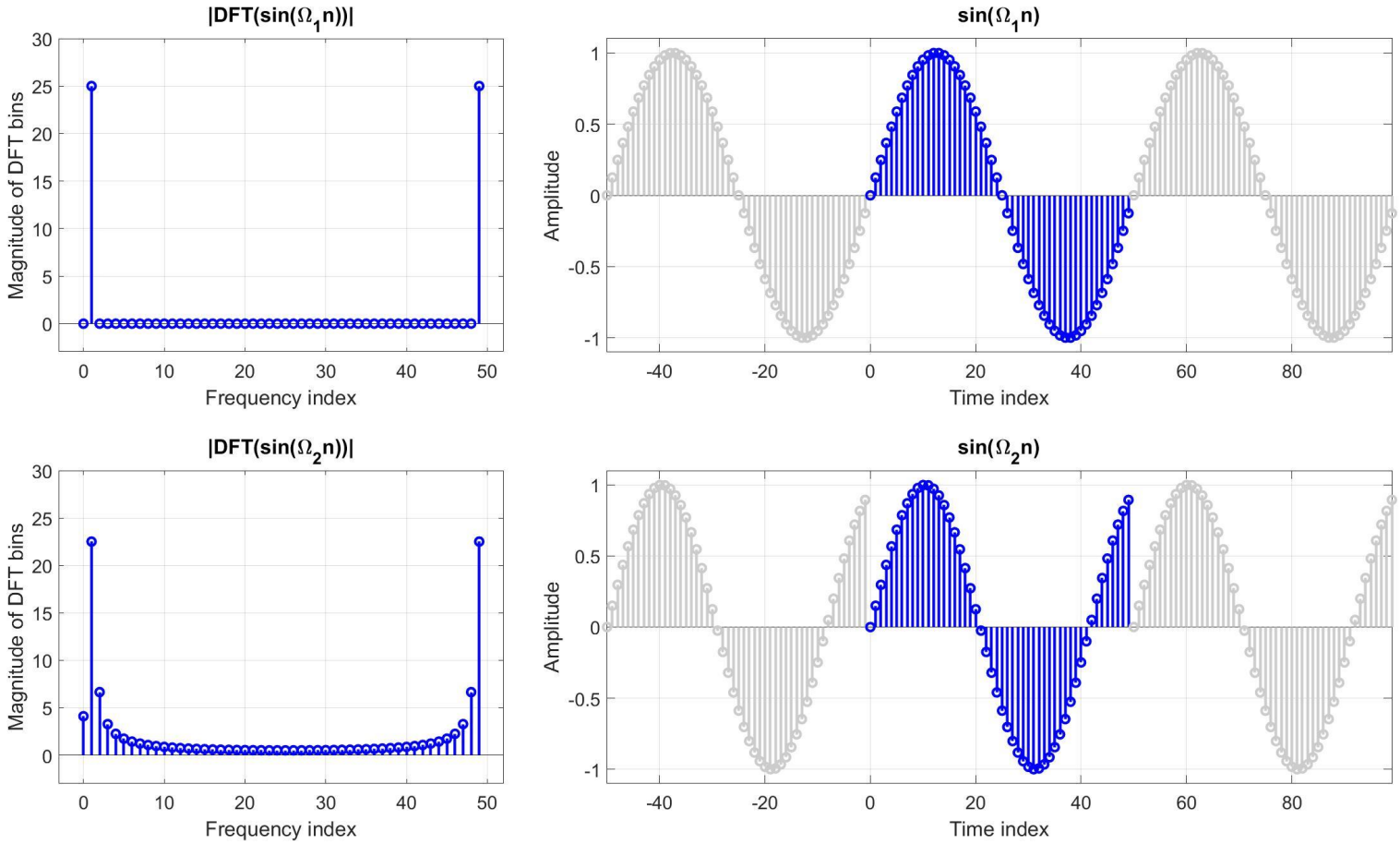
$$W_{\mathrm{rect}}(e^{j\Omega}) = \frac{\sin\left(\frac{L+1}{2}\Omega\right)}{\sin\left(\frac{\Omega}{2}\right)} = 2\pi\, D_{L/2}(\Omega)$$

Linear Filtering in the DFT Domain – Part 30

Frequency analysis of stationary signals – Comparison of window sequences – Part 4

Some ***problems*** you already know …

… Create windows that go down to zero at the edges. A simple idea is to use a window that has a ***triangular*** shape (instead of a rectangular one).

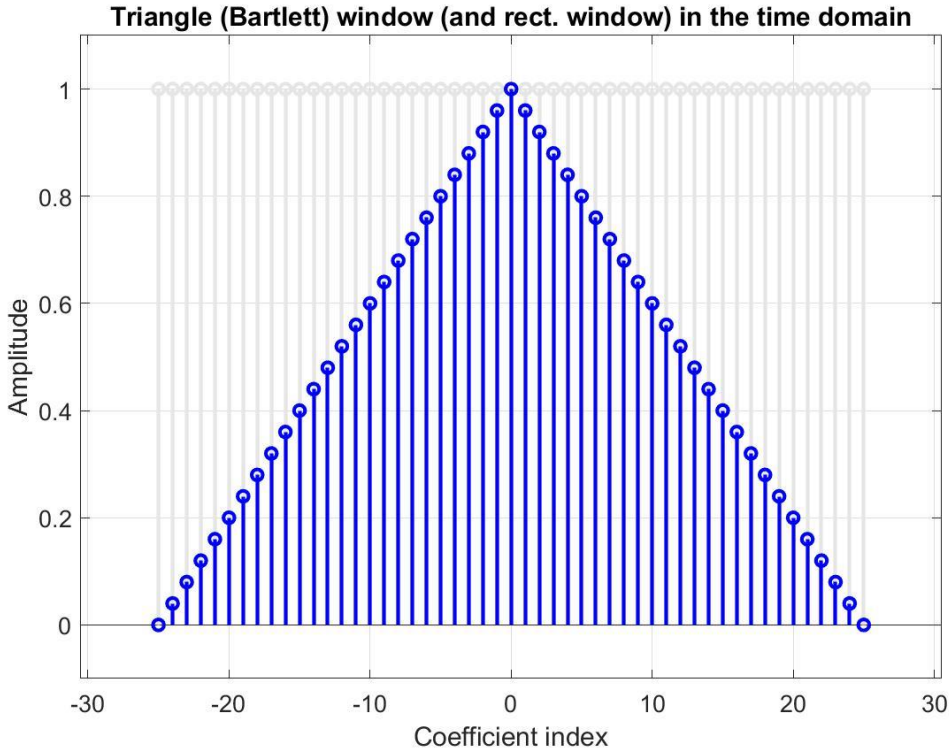


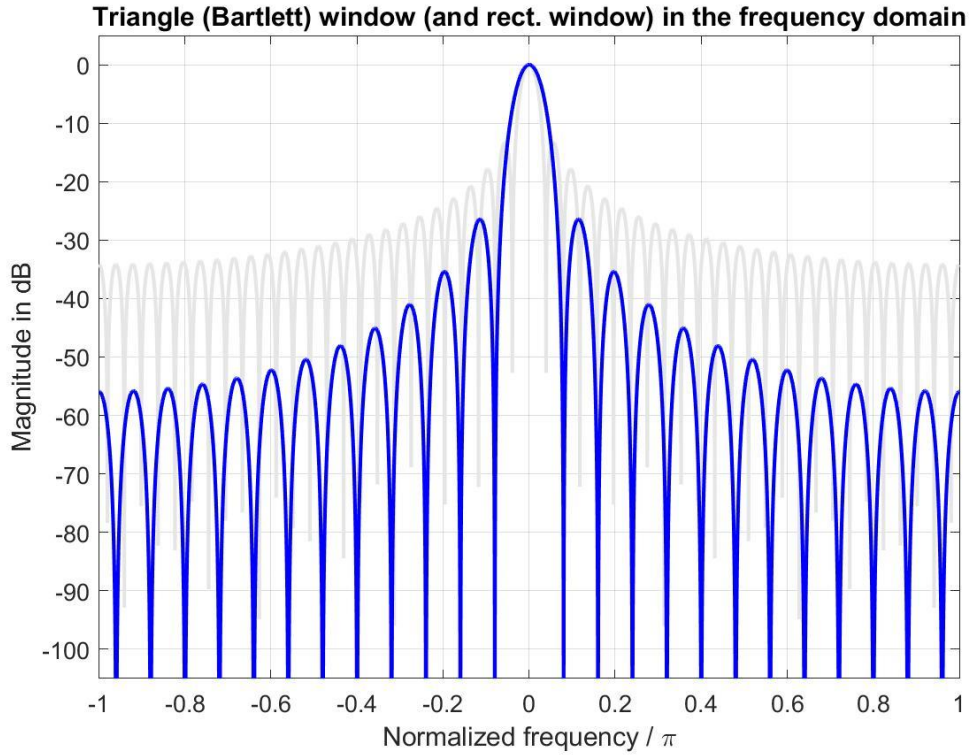
Linear Filtering in the DFT Domain – Part 31

Frequency analysis of stationary signals – Comparison of window sequences – Part 5

Properties to be mentioned:

- Better side-lobe attenuation
- Can be created as a convolution of two rectangle windows
- Allows for overlap-add processing with 50 % overlap



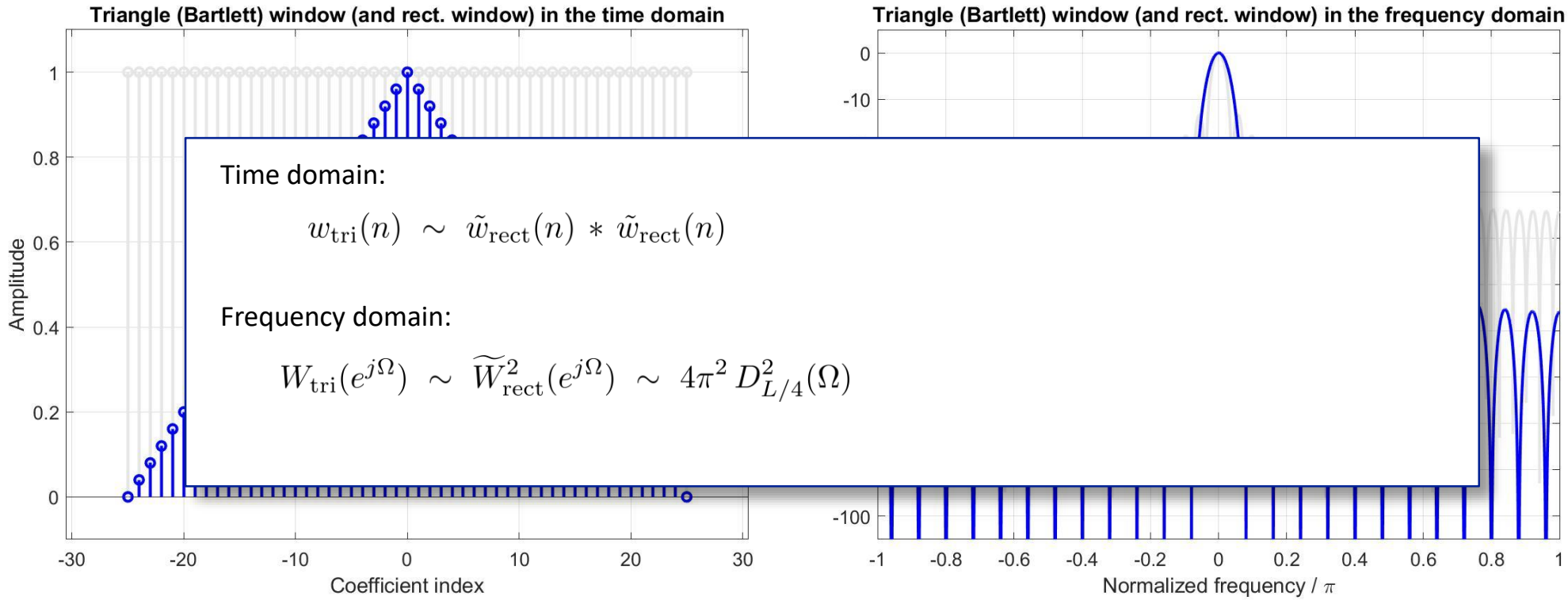


Linear Filtering in the DFT Domain – Part 32

Frequency analysis of stationary signals – Comparison of window sequences – Part 6

Properties to be mentioned:

- Better side-lobe attenuation
- Can be created as a convolution of two rectangle windows
- Allows for overlap-add processing with 50 % overlap



Time domain:

$$w_{\mathrm{tri}}(n) \;\sim\; \tilde{w}_{\mathrm{rect}}(n) * \tilde{w}_{\mathrm{rect}}(n)$$

Frequency domain:

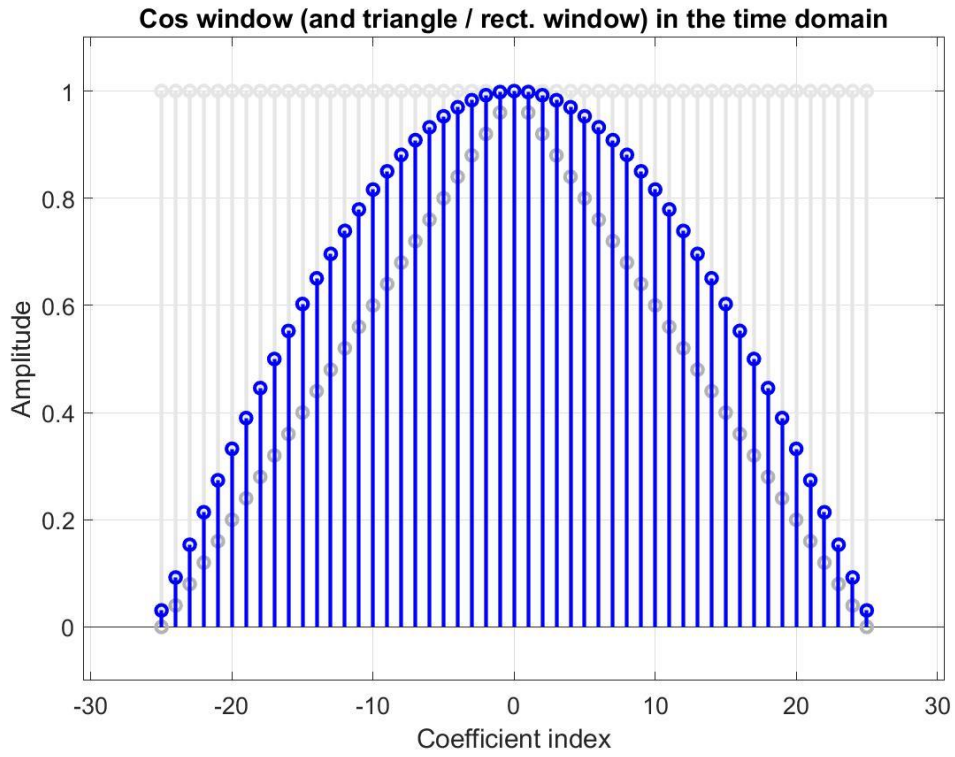
$$W_{\mathrm{tri}}(e^{j\Omega}) \;\sim\; \widetilde{W}^2_{\mathrm{rect}}(e^{j\Omega}) \;\sim\; 4\pi^2\, D^2_{L/4}(\Omega)$$

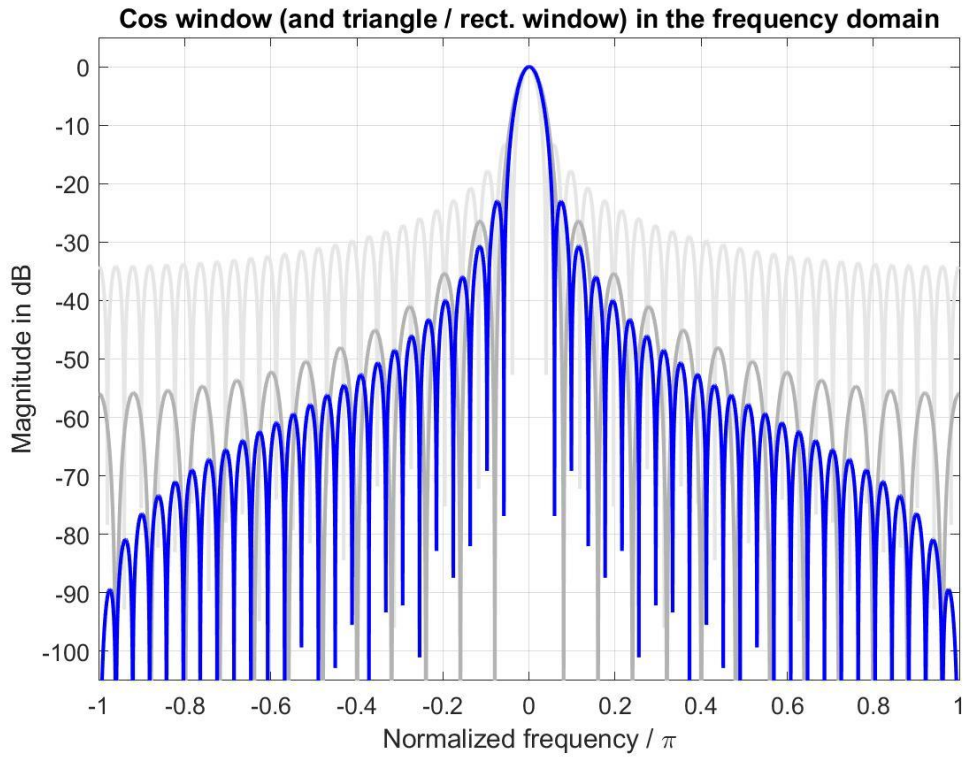
Linear Filtering in the DFT Domain – Part 33

Frequency analysis of stationary signals – Comparison of window sequences – Part 8

Properties to be mentioned:

- Using now sin/cos functions
- First, only part of a cos sequence



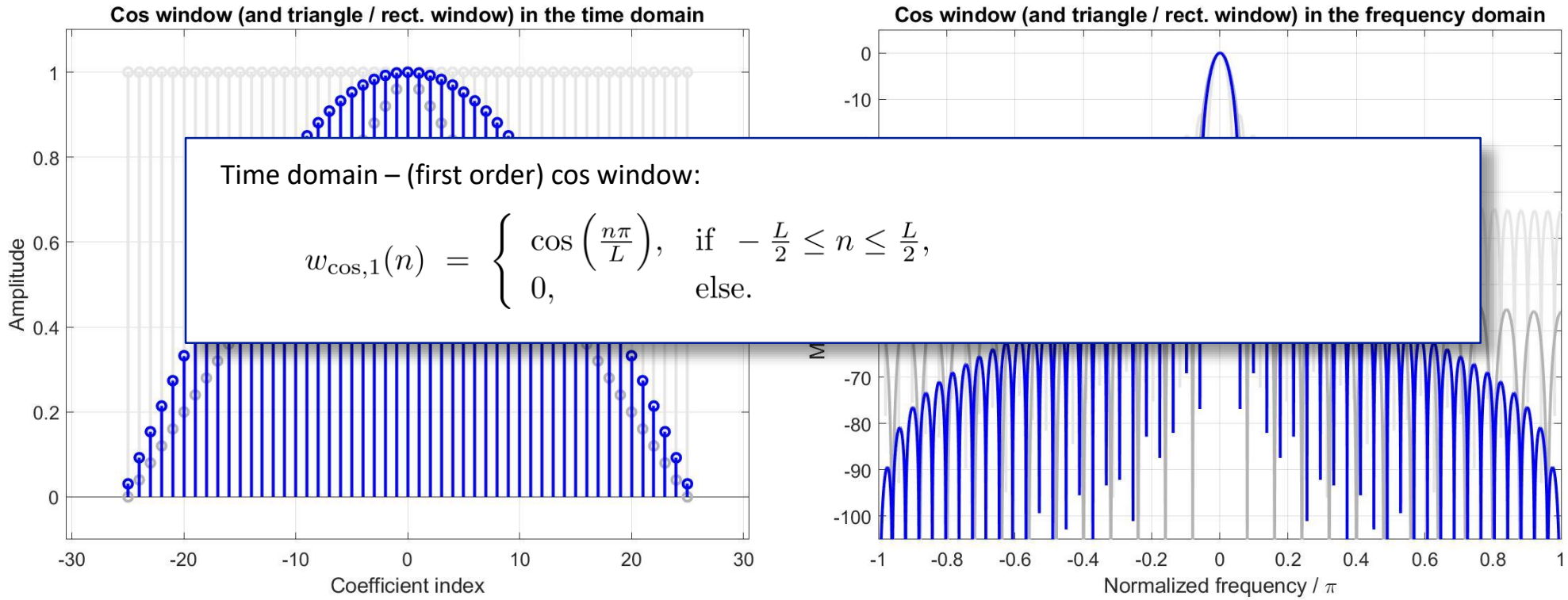


Linear Filtering in the DFT Domain – Part 34

Frequency analysis of stationary signals – Comparison of window sequences – Part 9

Properties to be mentioned:

- Using now sin/cos functions
- First, only part of a cos sequence



Time domain – (first order) cos window:

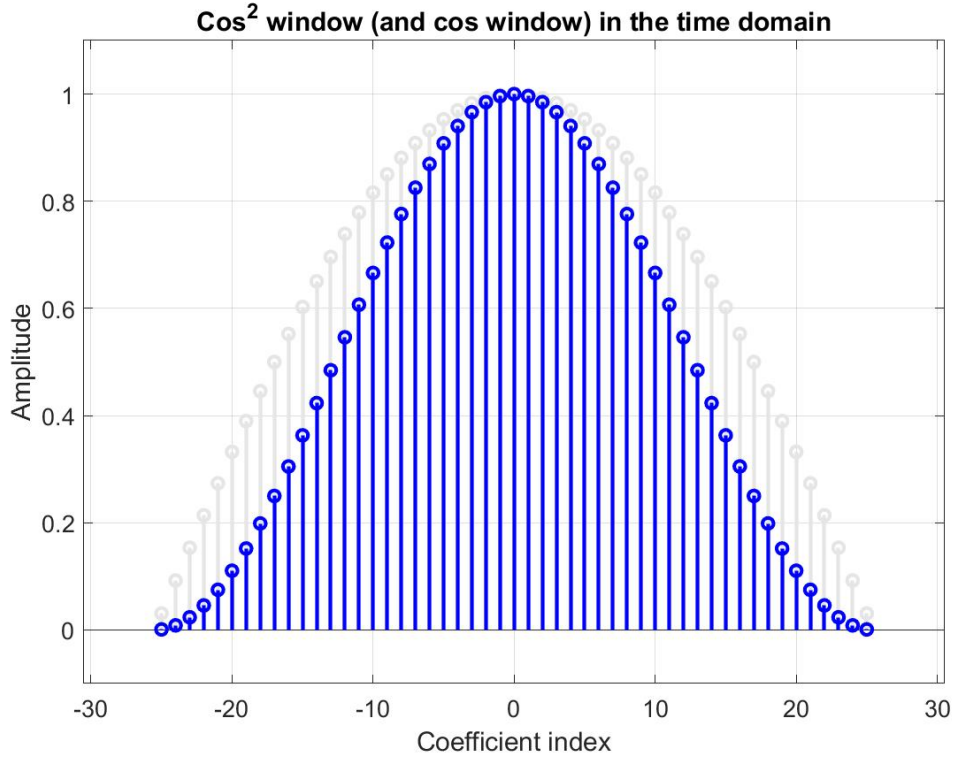
$$w_{\cos,1}(n) = \begin{cases} \cos\left(\frac{n\pi}{L}\right), & \text{if } -\frac{L}{2} \leq n \leq \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

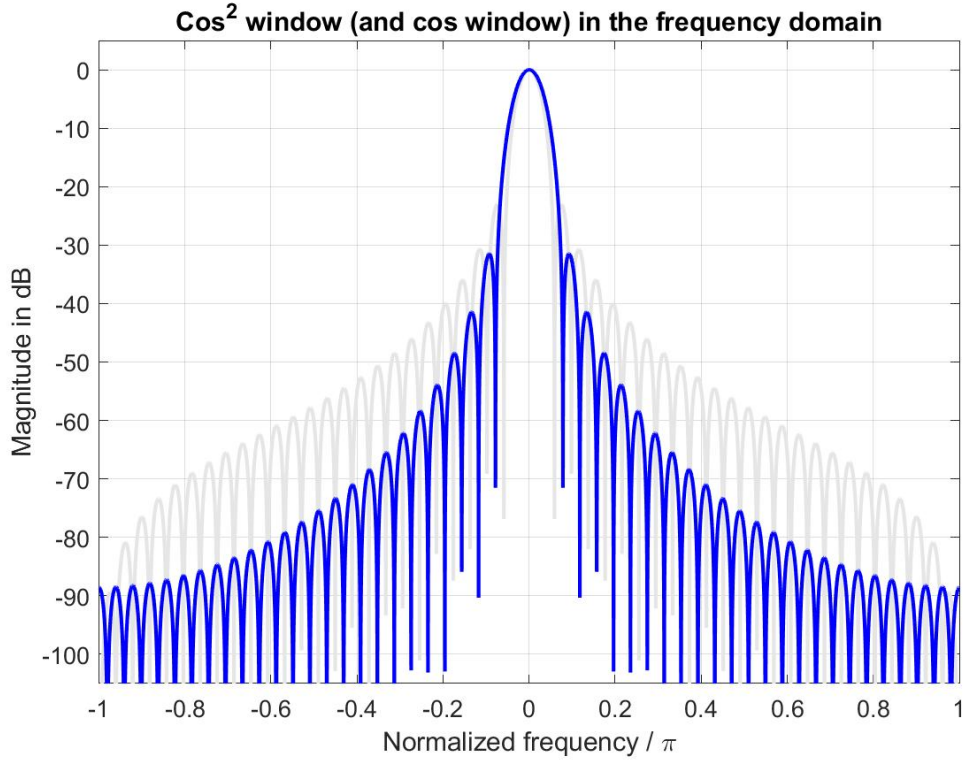
Linear Filtering in the DFT Domain – Part 35

Frequency analysis of stationary signals – Comparison of window sequences – Part 10

Properties to be mentioned:

- Using now sin/cos functions
- Only one half of a cos wave is used (can be changed)
- Higher orders (powers) are possible
- Cos properties can be exploited (summing to a constant)



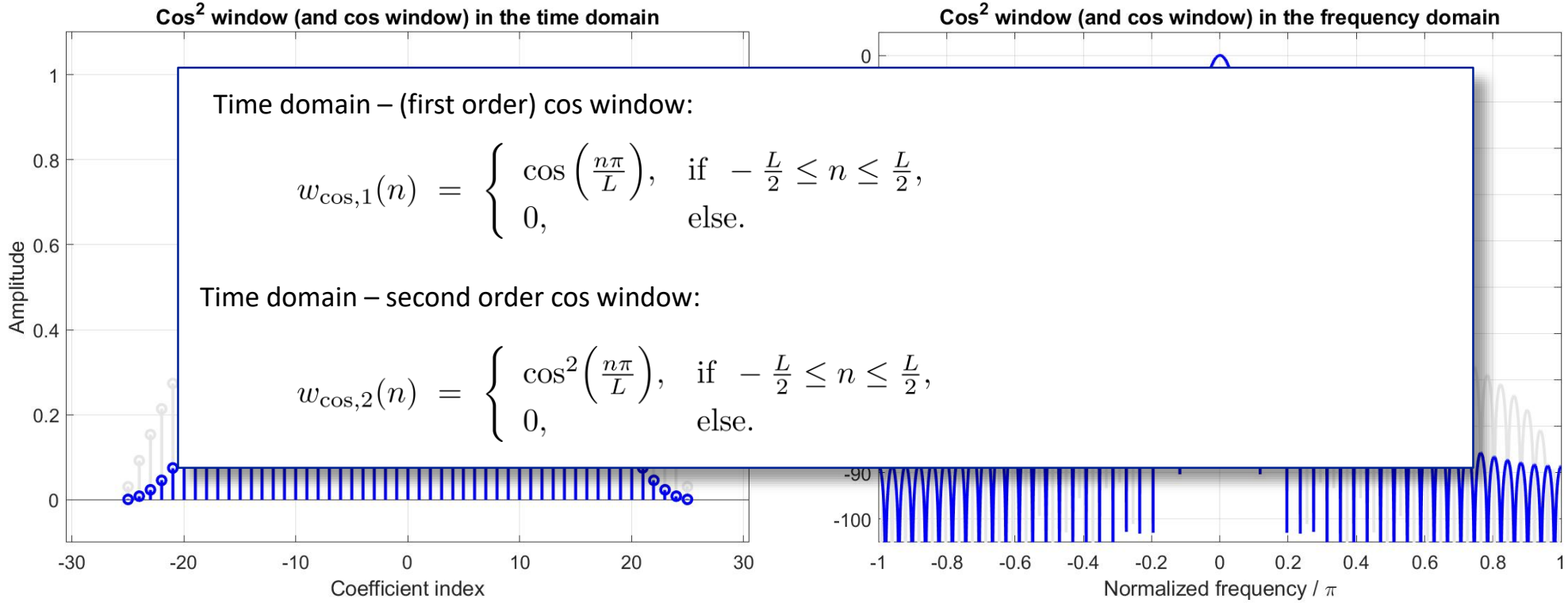


Linear Filtering in the DFT Domain – Part 36

Frequency analysis of stationary signals – Comparison of window sequences – Part 11

Properties to be mentioned:

- Using now sin/cos functions
- Only one half of a cos wave is used (can be changed)
- Higher orders (powers) are possible
- Cos properties can be exploited (summing to a constant)



Time domain – (first order) cos window:

$$w_{\cos,1}(n) = \begin{cases} \cos\left(\frac{n\pi}{L}\right), & \text{if } -\frac{L}{2} \le n \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

Time domain – second order cos window:

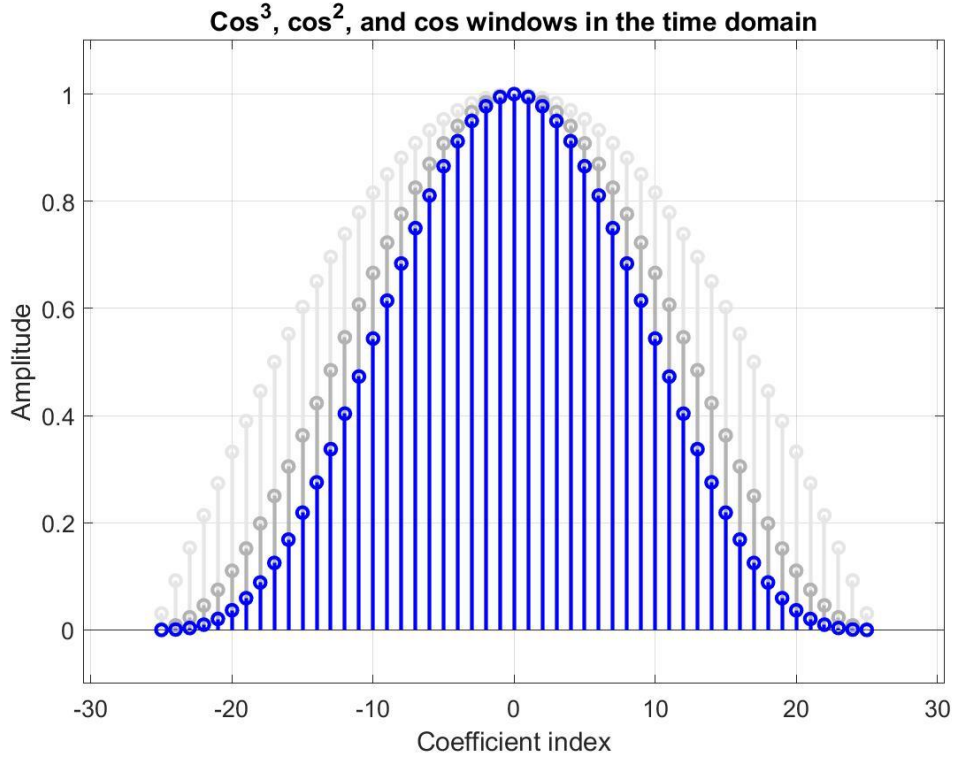
$$w_{\cos,2}(n) = \begin{cases} \cos^2\left(\frac{n\pi}{L}\right), & \text{if } -\frac{L}{2} \le n \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

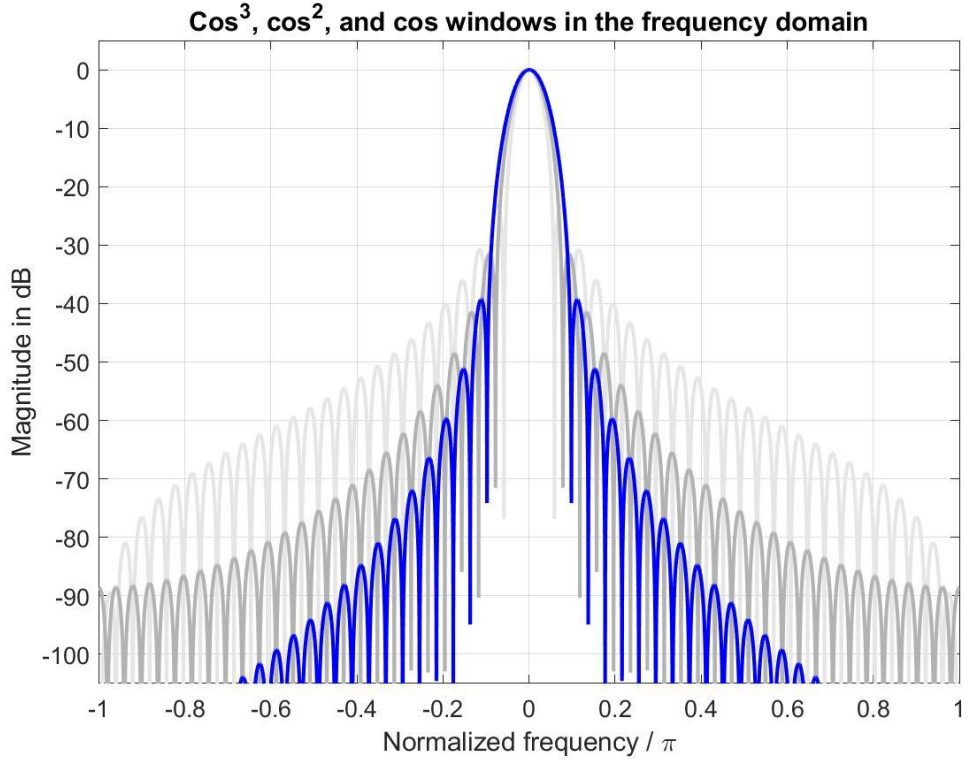
Linear Filtering in the DFT Domain – Part 37

Frequency analysis of stationary signals – Comparison of window sequences – Part 12

Properties to be mentioned:

- Using now sin/cos functions
- Only one half of a cos wave is used (can be changed)
- Higher orders (powers) are possible
- Cos properties can be exploited (summing to a constant)



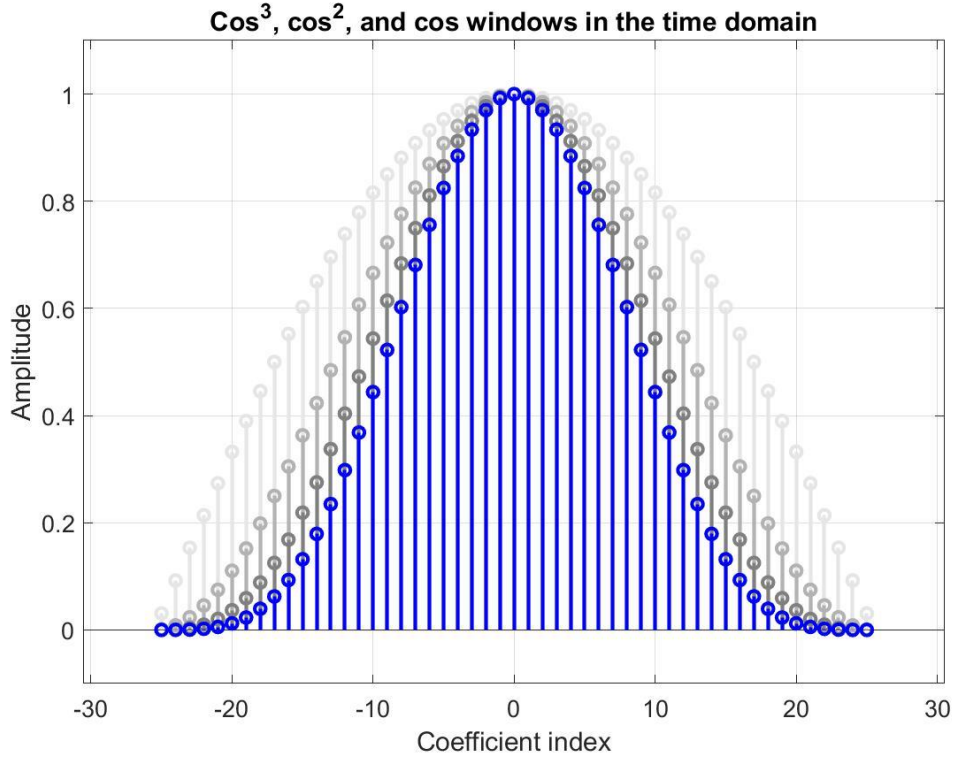


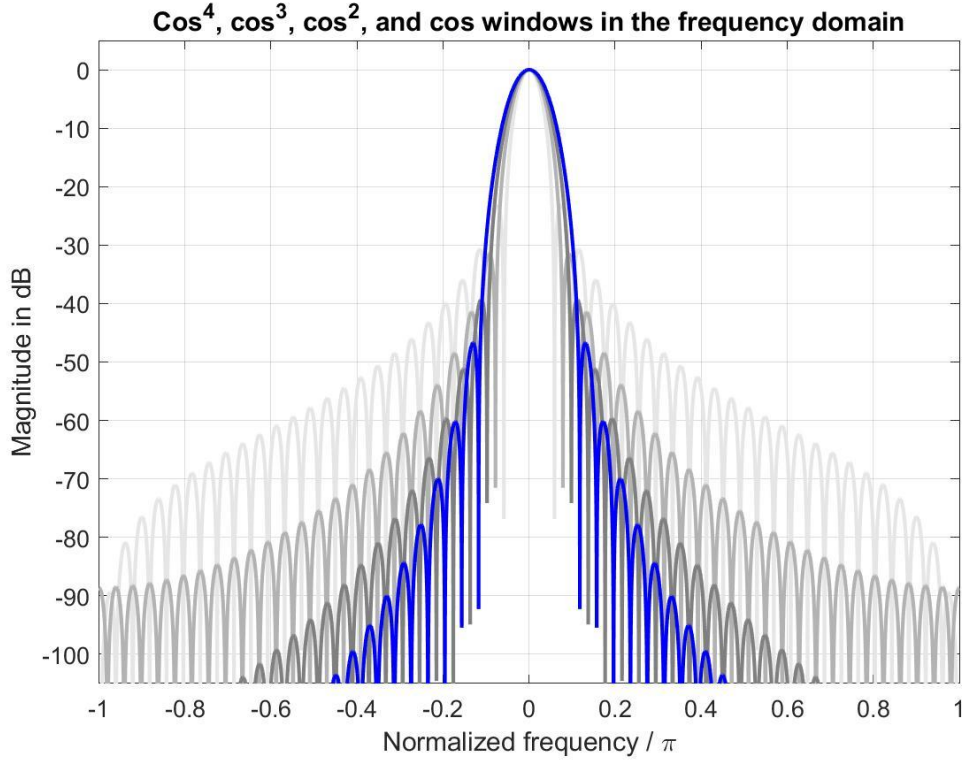
Linear Filtering in the DFT Domain – Part 38

Frequency analysis of stationary signals – Comparison of window sequences – Part 13

Properties to be mentioned:

- Using now sin/cos functions
- Only one half of a cos wave is used (can be changed)
- Higher orders (powers) are possible
- Cos properties can be exploited (summing to a constant)



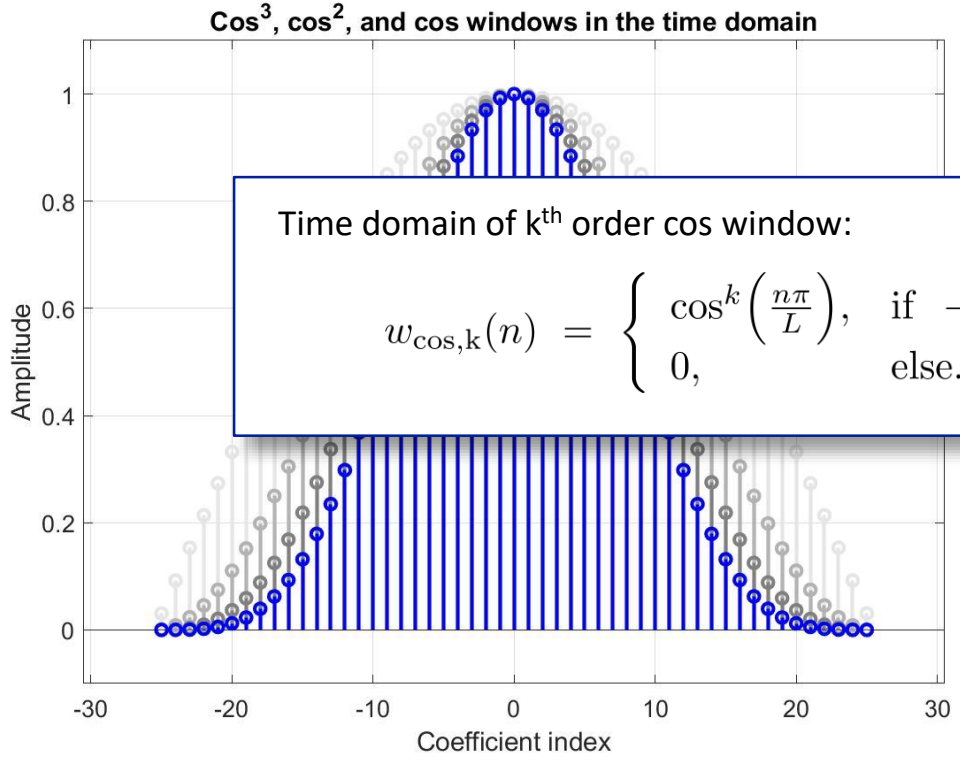


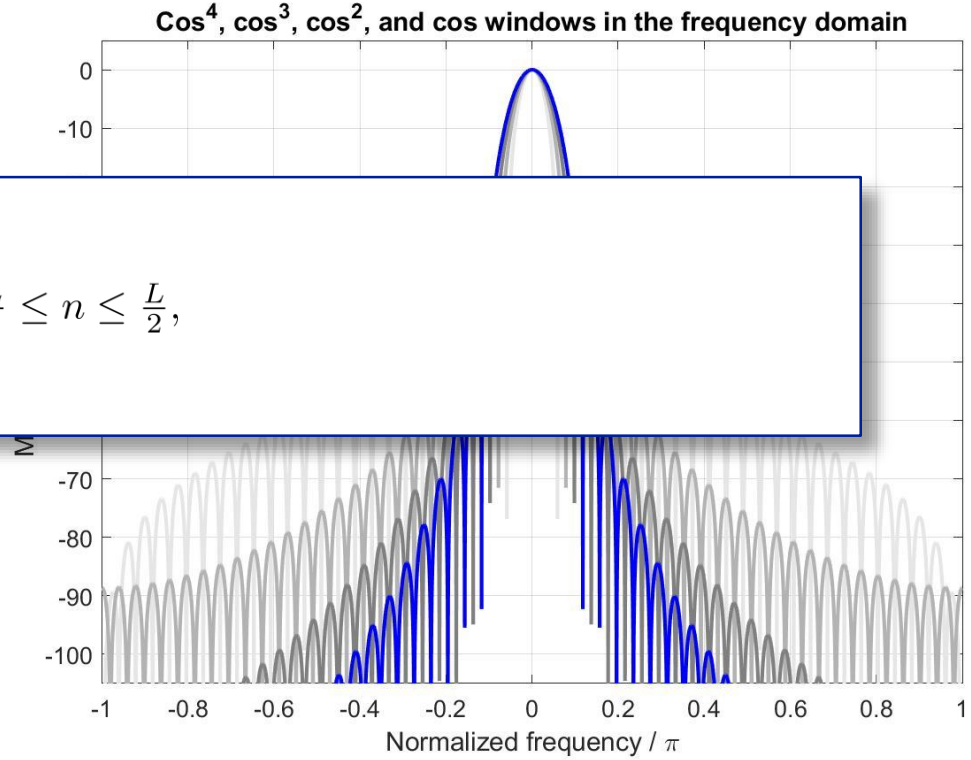
Linear Filtering in the DFT Domain – Part 39

Frequency analysis of stationary signals – Comparison of window sequences – Part 14

Properties to be mentioned:

- Using now sin/cos functions
- Only one half of a cos wave is used (can be changed)
- Higher orders (powers) are possible
- Cos properties can be exploited (summing to a constant)





Time domain of kth order cos window:

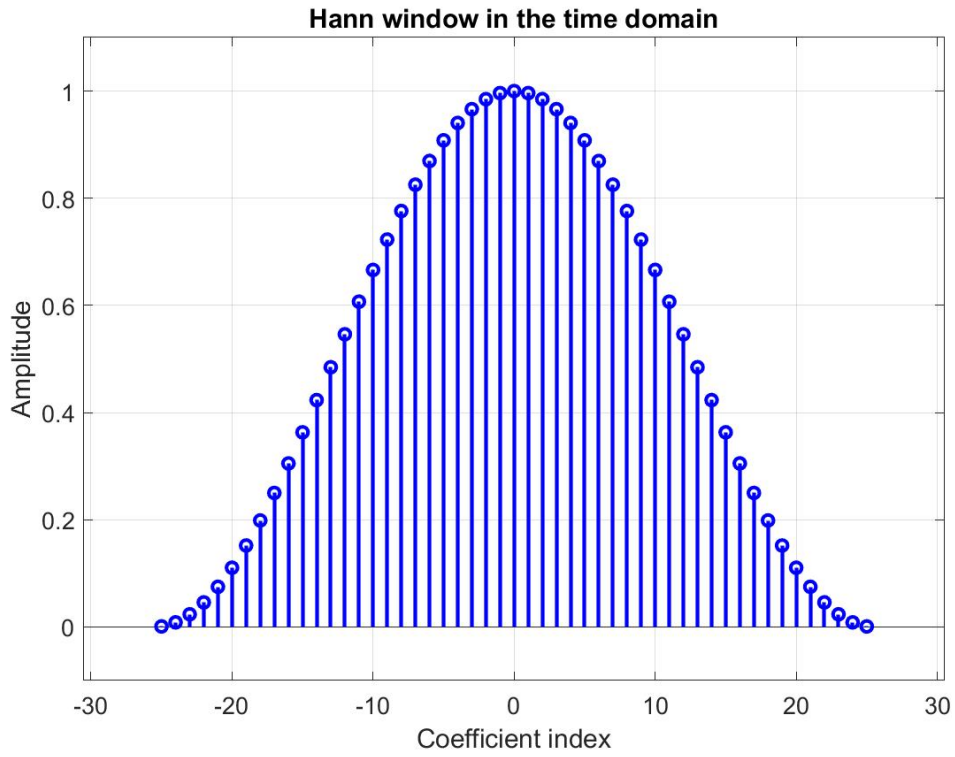
$$w_{\cos,\mathrm{k}}(n) = \begin{cases} \cos^k\left(\frac{n\pi}{L}\right), & \text{if } -\frac{L}{2} \le n \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

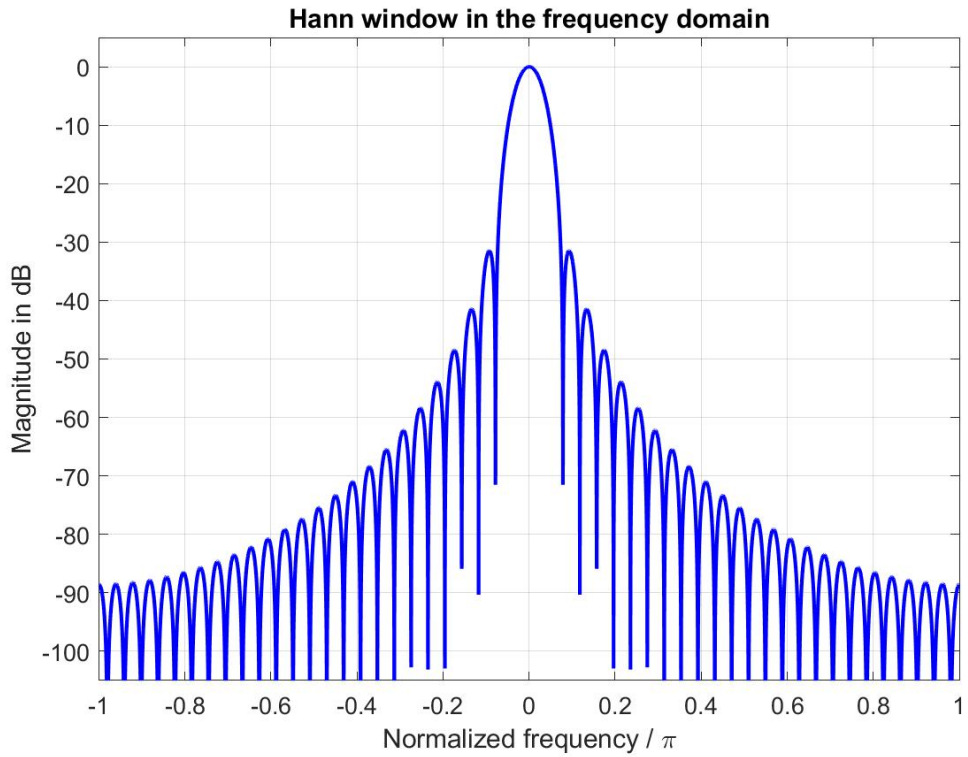
Linear Filtering in the DFT Domain – Part 40

Frequency analysis of stationary signals – Comparison of window sequences – Part 15

Properties to be mentioned:

- Combining rectangle and cos sequences
- Hann window



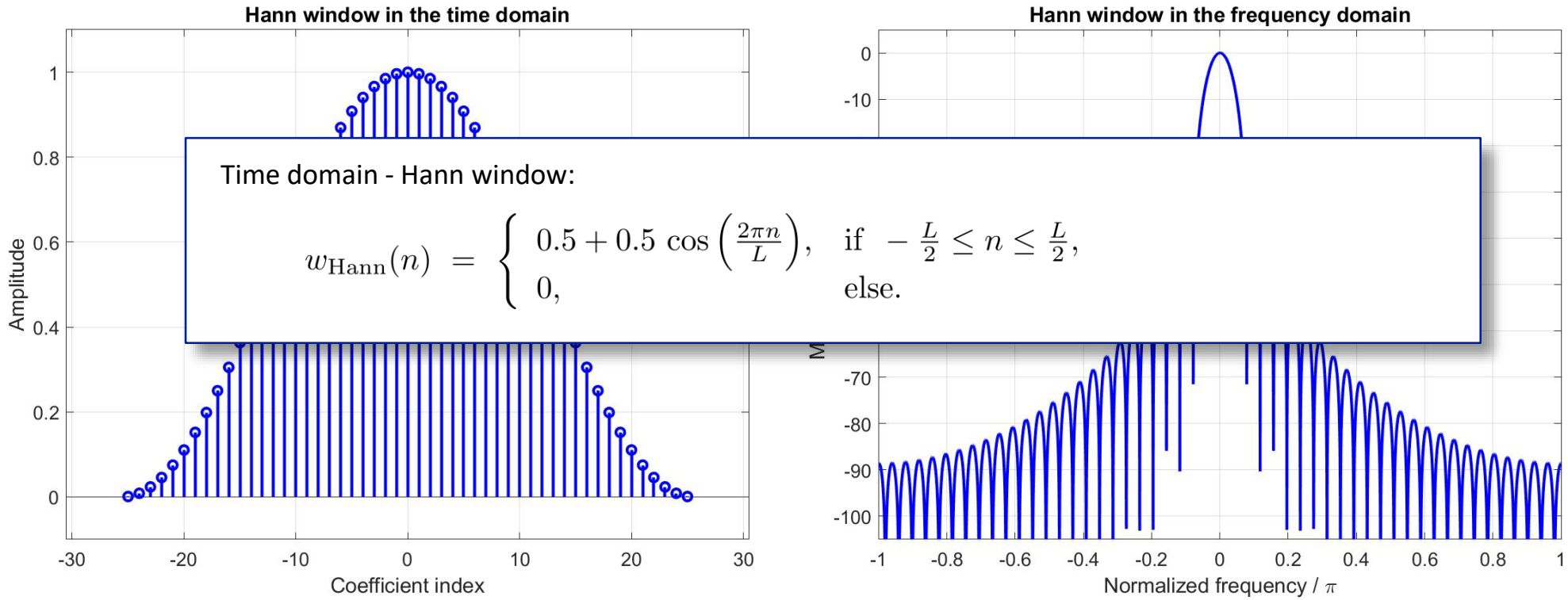


Linear Filtering in the DFT Domain – Part 41

Frequency analysis of stationary signals – Comparison of window sequences – Part 16

Properties to be mentioned:

- Combining rectangle and cos sequences
- Hann window



Time domain - Hann window:

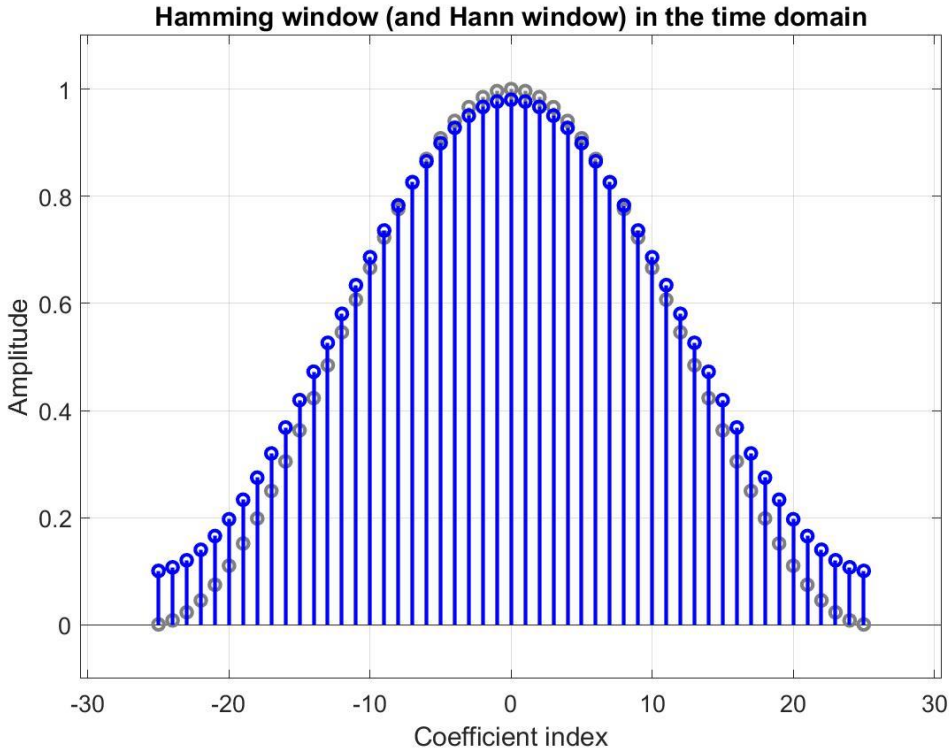
$$w_{\text{Hann}}(n) = \begin{cases} 0.5 + 0.5\,\cos\left(\frac{2\pi n}{L}\right), & \text{if } -\frac{L}{2} \le n \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

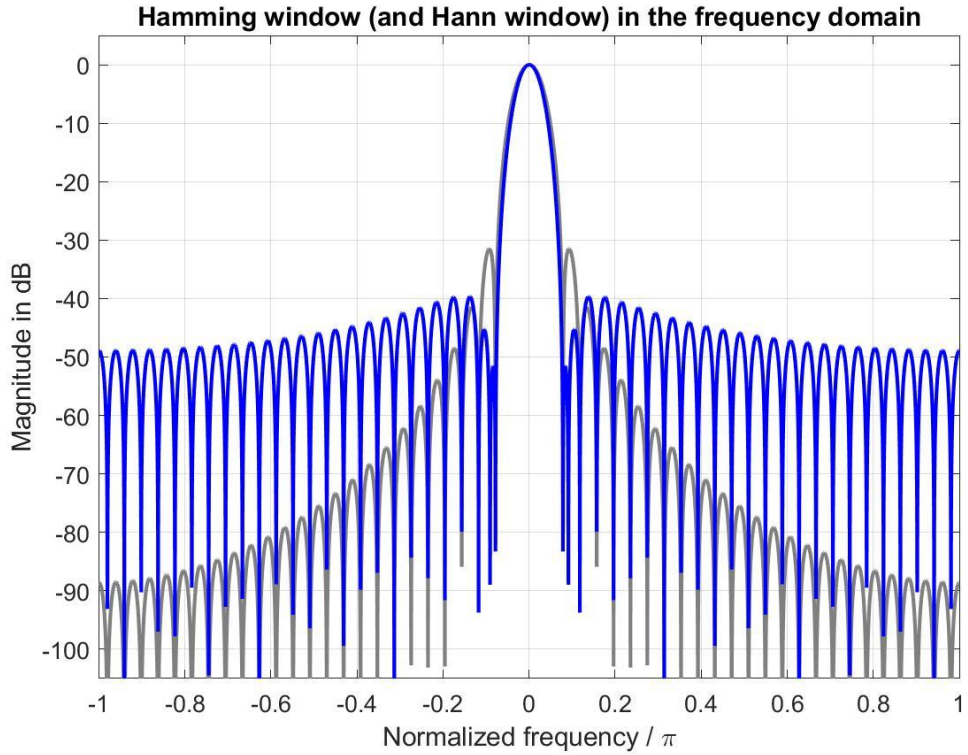
Linear Filtering in the DFT Domain – Part 42

Frequency analysis of stationary signals – Comparison of window sequences – Part 17

Properties to be mentioned:

- Combining rectangle and cos sequences
- Different weighting in order to minimize the largest side lobe
- Hamming window



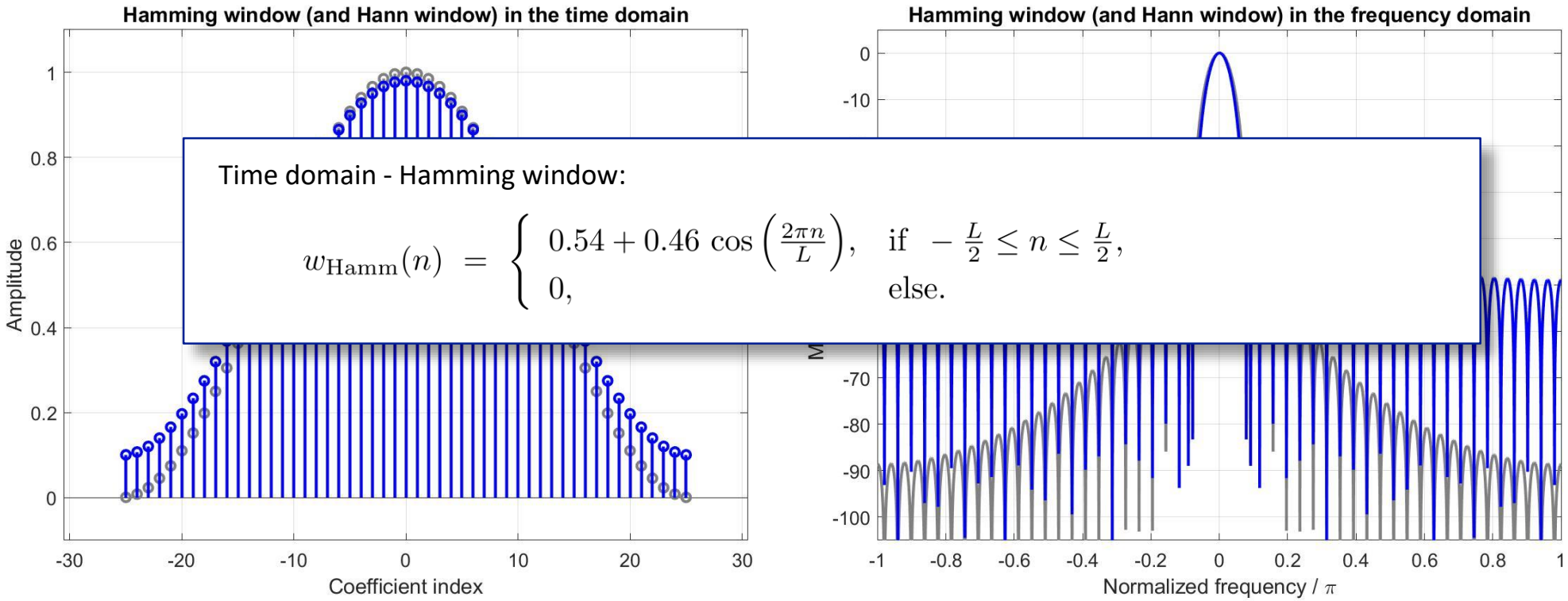


Linear Filtering in the DFT Domain – Part 43

Frequency analysis of stationary signals – Comparison of window sequences – Part 18

Properties to be mentioned:

- Combining rectangle and cos sequences
- Different weighting in order to minimize the largest side lobe
- Hamming window



Time domain - Hamming window:

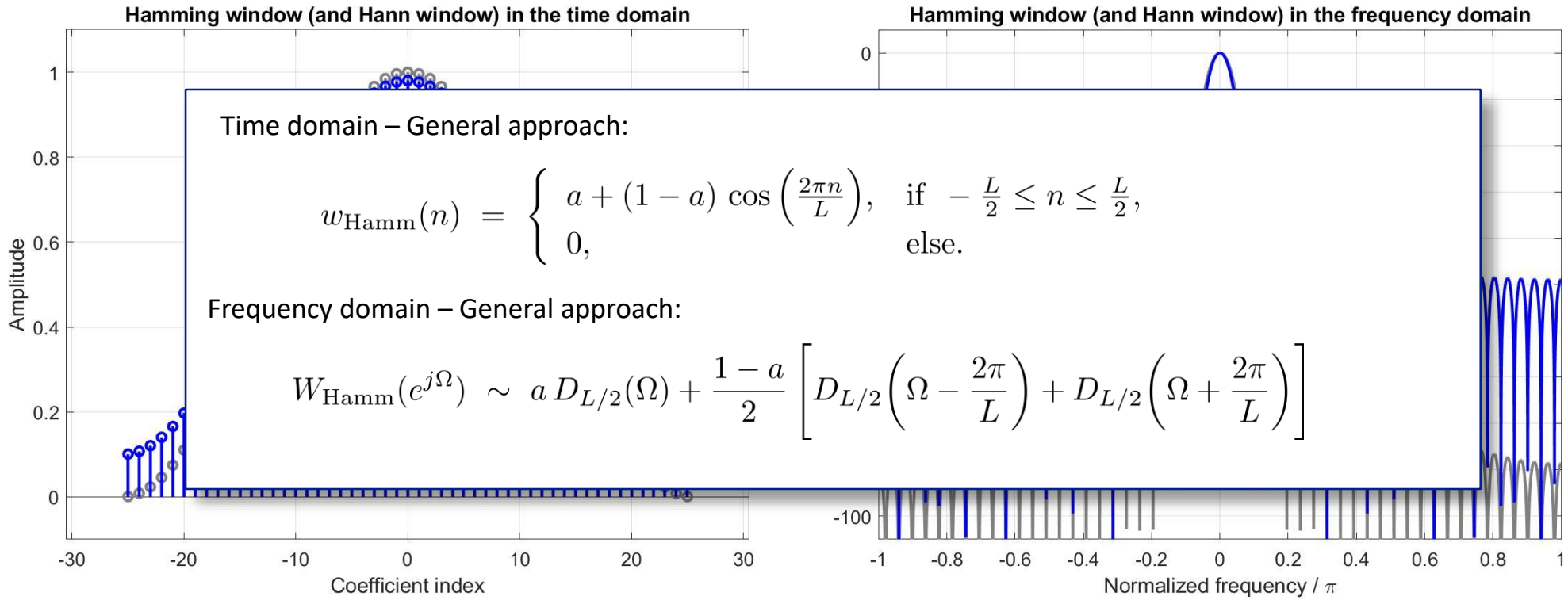
$$w_{\mathrm{Hamm}}(n) = \begin{cases} 0.54 + 0.46\,\cos\left(\frac{2\pi n}{L}\right), & \text{if } -\frac{L}{2} \le n \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

Linear Filtering in the DFT Domain – Part 44

Frequency analysis of stationary signals – Comparison of window sequences – Part 19

Properties to be mentioned:

- Combining rectangle and cos sequences
- Different weighting in order to minimize the largest side lobe
- Hamming window



Time domain – General approach:

$$w_{\mathrm{Hamm}}(n) = \begin{cases} a + (1-a)\cos\left(\frac{2\pi n}{L}\right), & \text{if } -\frac{L}{2} \le n \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

Frequency domain – General approach:

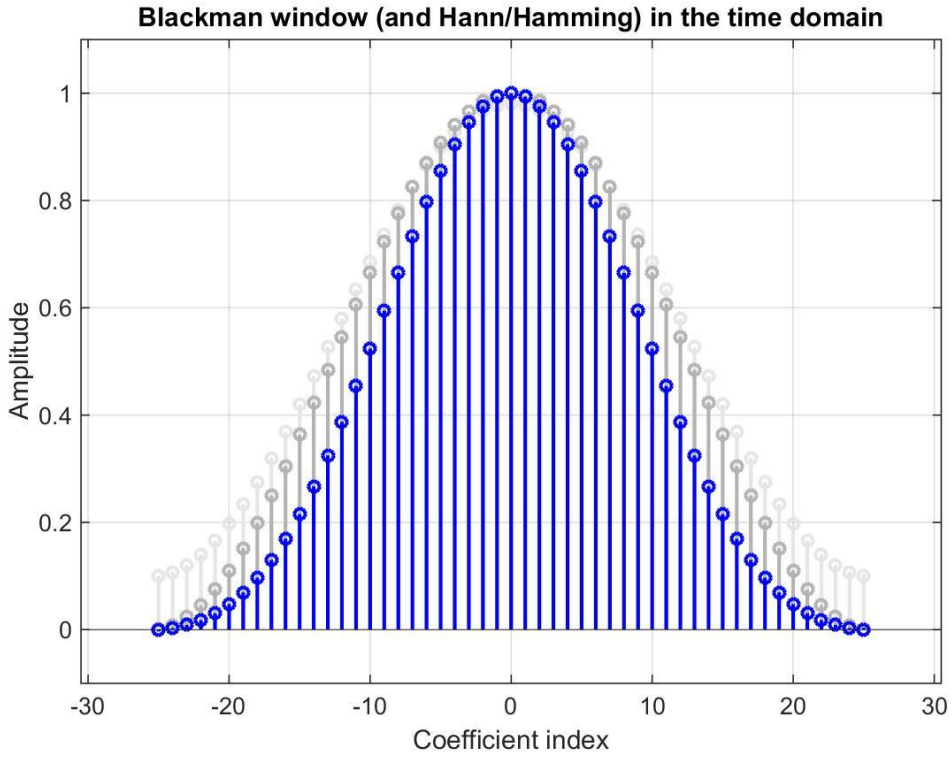
$$W_{\mathrm{Hamm}}(e^{j\Omega}) \sim a\,D_{L/2}(\Omega) + \frac{1-a}{2}\left[D_{L/2}\left(\Omega - \frac{2\pi}{L}\right) + D_{L/2}\left(\Omega + \frac{2\pi}{L}\right)\right]$$

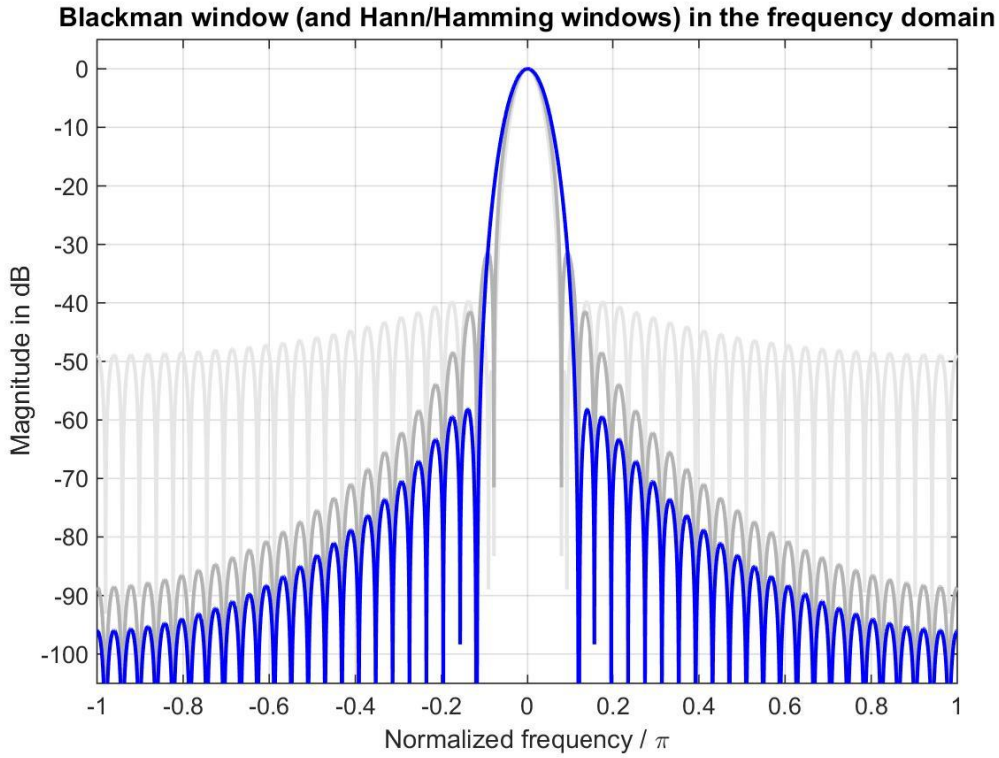
Linear Filtering in the DFT Domain – Part 45

Frequency analysis of stationary signals – Comparison of window sequences – Part 20

Properties to be mentioned:

- Combining rectangle and cos sequences
- Different weighting in order to minimize the largest side lobe
- Three cos sequences (Blackman window)



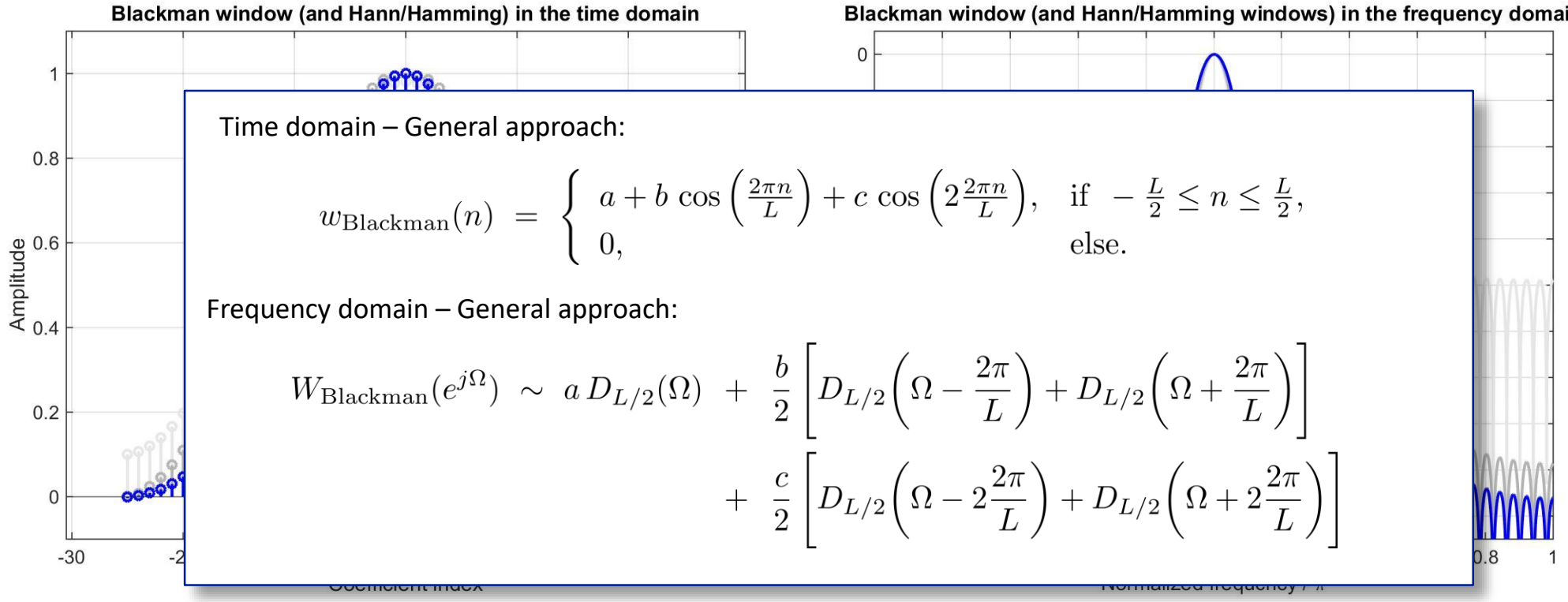


Linear Filtering in the DFT Domain – Part 46

Frequency analysis of stationary signals – Comparison of window sequences – Part 21

Properties to be mentioned:

- Combining rectangle and cos sequences
- Different weighting in order to minimize the largest side lobe
- Three cos sequences (Blackman window)



Time domain – General approach:

$$w_{\text{Blackman}}(n) = \begin{cases} a + b\cos\left(\frac{2\pi n}{L}\right) + c\cos\left(2\frac{2\pi n}{L}\right), & \text{if } -\frac{L}{2} \le n \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

Frequency domain – General approach:

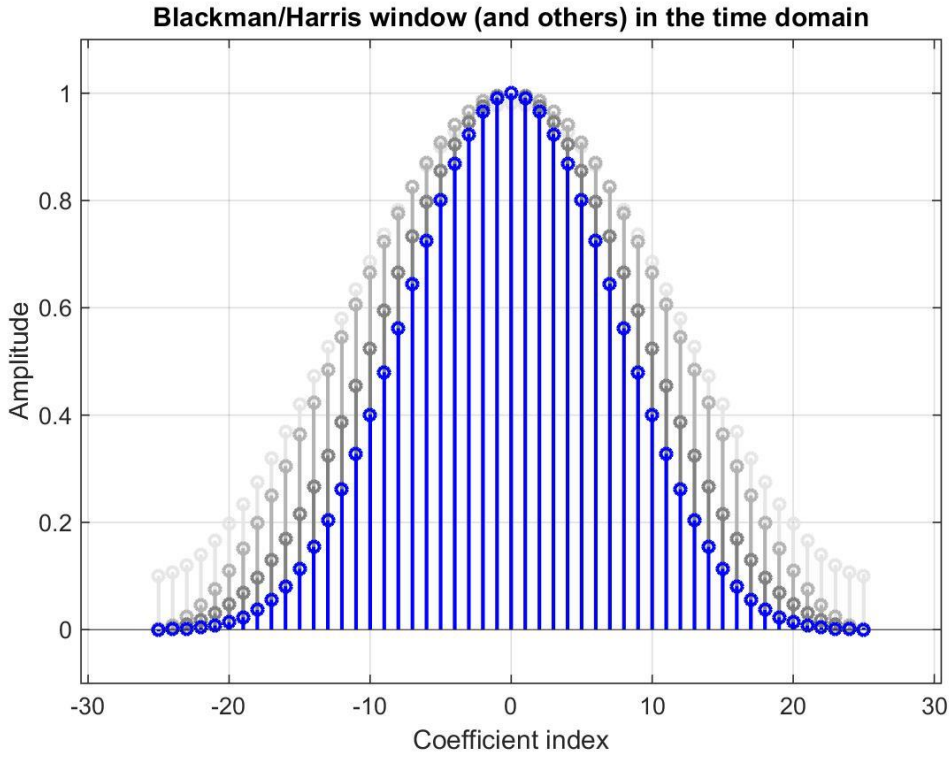
$$W_{\text{Blackman}}(e^{j\Omega}) \sim a\,D_{L/2}(\Omega) + \frac{b}{2}\left[D_{L/2}\left(\Omega - \frac{2\pi}{L}\right) + D_{L/2}\left(\Omega + \frac{2\pi}{L}\right)\right] + \frac{c}{2}\left[D_{L/2}\left(\Omega - 2\frac{2\pi}{L}\right) + D_{L/2}\left(\Omega + 2\frac{2\pi}{L}\right)\right]$$

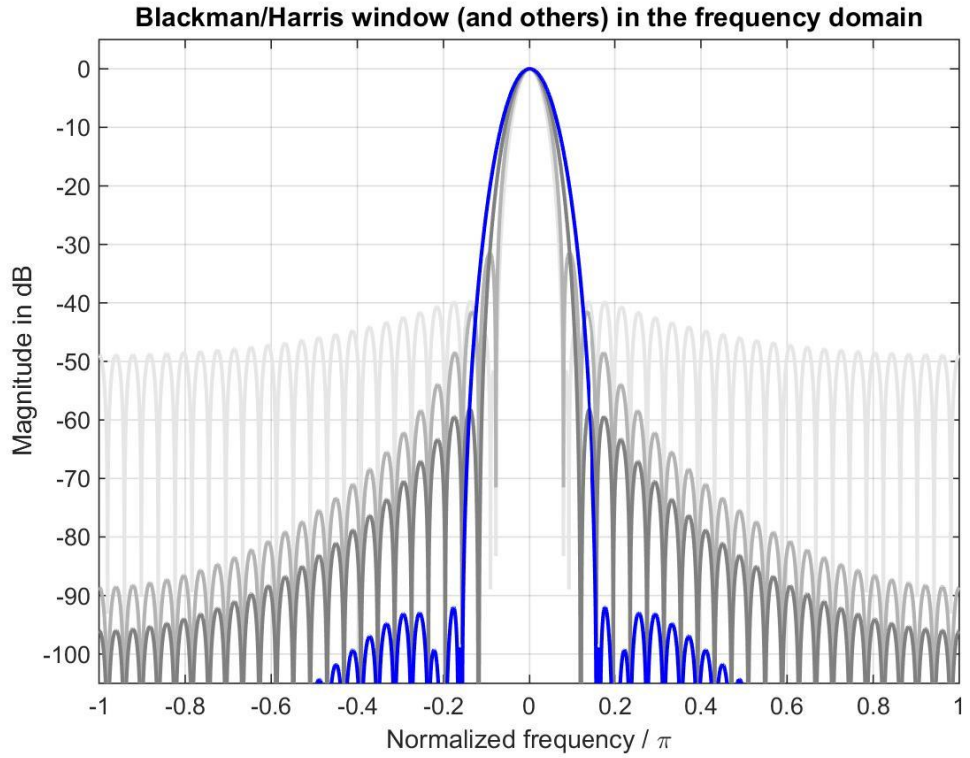
Linear Filtering in the DFT Domain – Part 47

Frequency analysis of stationary signals – Comparison of window sequences – Part 22

Properties to be mentioned:

- Combining rectangle and cos sequences
- Different weighting in order to minimize the largest side lobe
- Four cos sequences (Blackman/Harris window)





Linear Filtering in the DFT Domain – Part 48

Frequency analysis of stationary signals – Comparison of window sequences – Part 23

Properties to be mentioned:

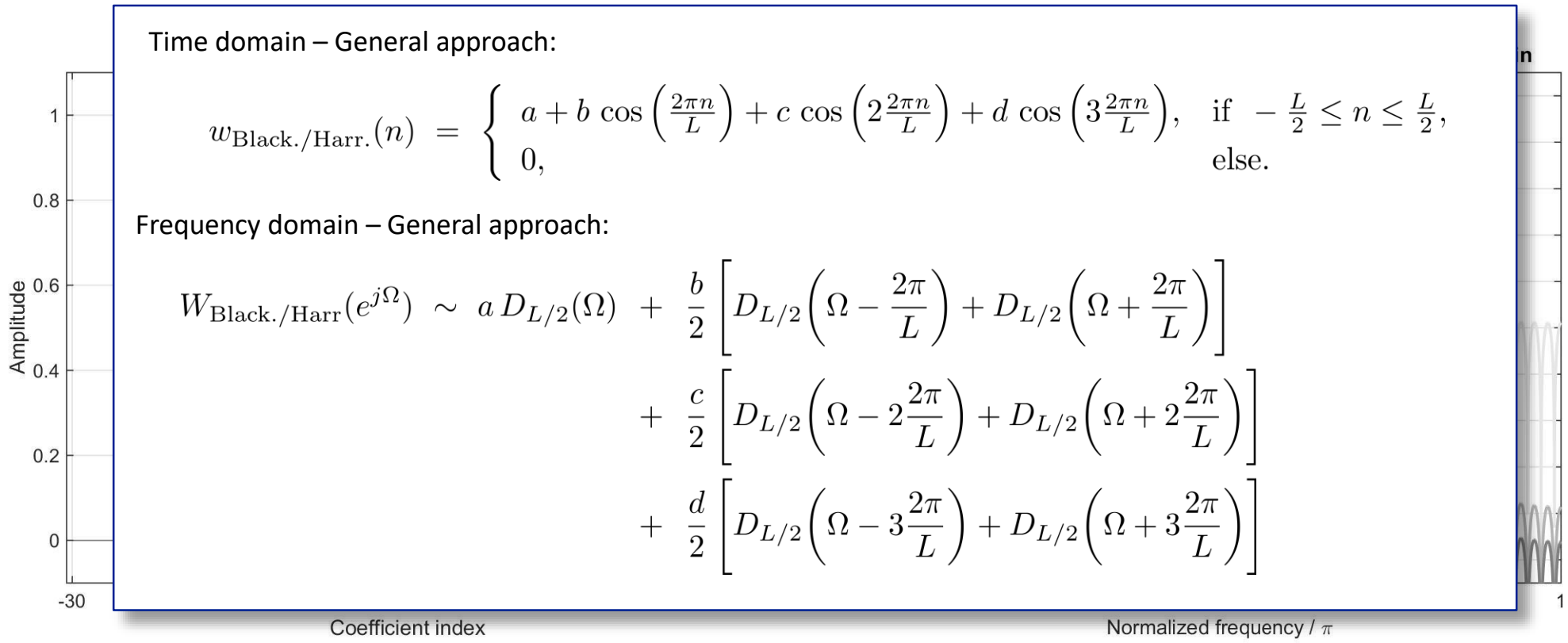
- Combining rectangle and cos sequences
- Different weighting in order to minimize the largest side lobe
- Four cos sequences (Blackman/Harris window)

Time domain – General approach:

$$w_{\text{Black./Harr.}}(n) = \begin{cases} a + b\cos\left(\frac{2\pi n}{L}\right) + c\cos\left(2\frac{2\pi n}{L}\right) + d\cos\left(3\frac{2\pi n}{L}\right), & \text{if } -\frac{L}{2} \le n \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

Frequency domain – General approach:

$$\begin{aligned} W_{\text{Black./Harr}}(e^{j\Omega}) \sim\ & a\,D_{L/2}(\Omega) + \frac{b}{2}\left[D_{L/2}\left(\Omega - \frac{2\pi}{L}\right) + D_{L/2}\left(\Omega + \frac{2\pi}{L}\right)\right] \\ & + \frac{c}{2}\left[D_{L/2}\left(\Omega - 2\frac{2\pi}{L}\right) + D_{L/2}\left(\Omega + 2\frac{2\pi}{L}\right)\right] \\ & + \frac{d}{2}\left[D_{L/2}\left(\Omega - 3\frac{2\pi}{L}\right) + D_{L/2}\left(\Omega + 3\frac{2\pi}{L}\right)\right] \end{aligned}$$

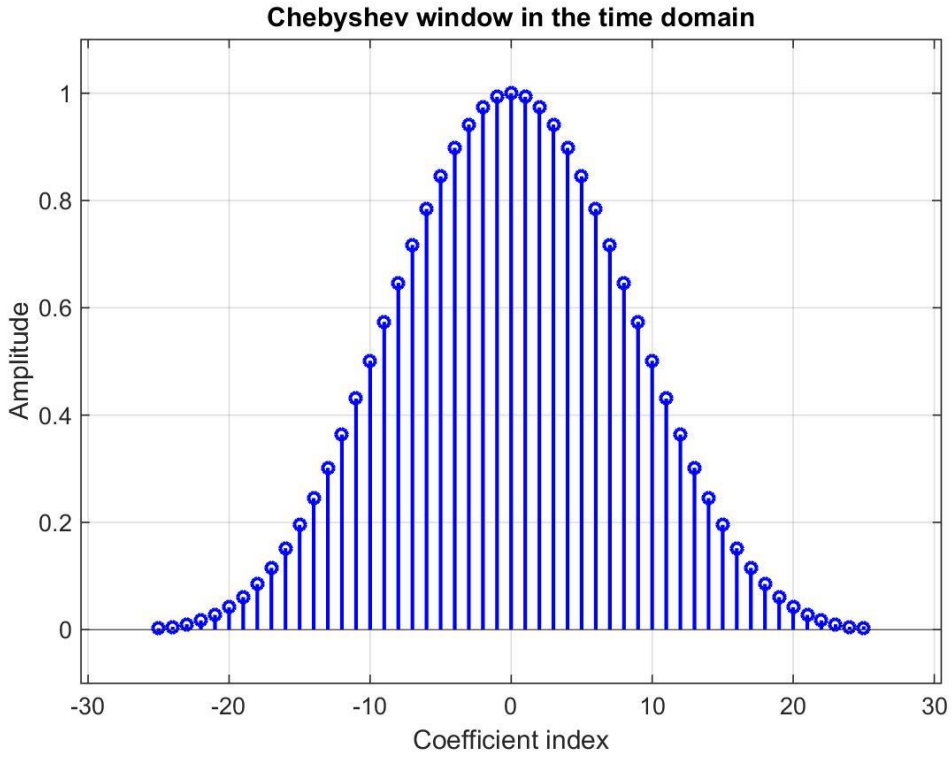


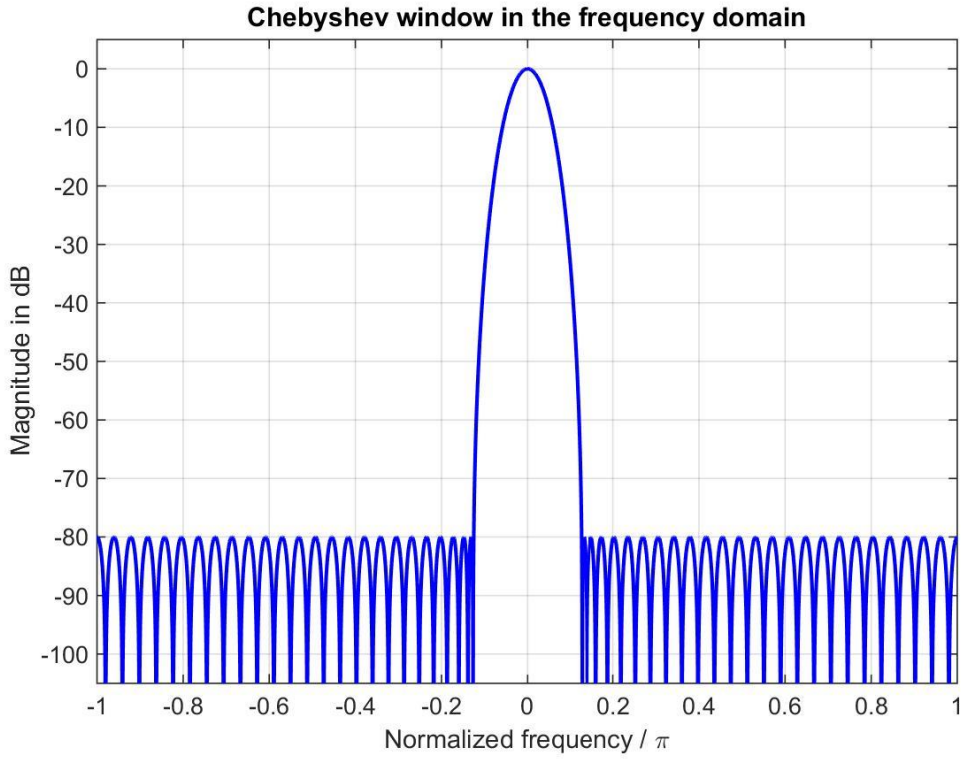
Linear Filtering in the DFT Domain – Part 49

Frequency analysis of stationary signals – Comparison of window sequences – Part 24

Properties to be mentioned:

- Dolph-Chebychev window
- Optimized according to the minimax criterion
- Based on (very simple) poly-nomials in the frequency domain
- Side lobes adjusted to -80 dB





Linear Filtering in the DFT Domain – Part 50

Frequency analysis of stationary signals – Comparison of window sequences – Part 25

Properties to be mentioned:

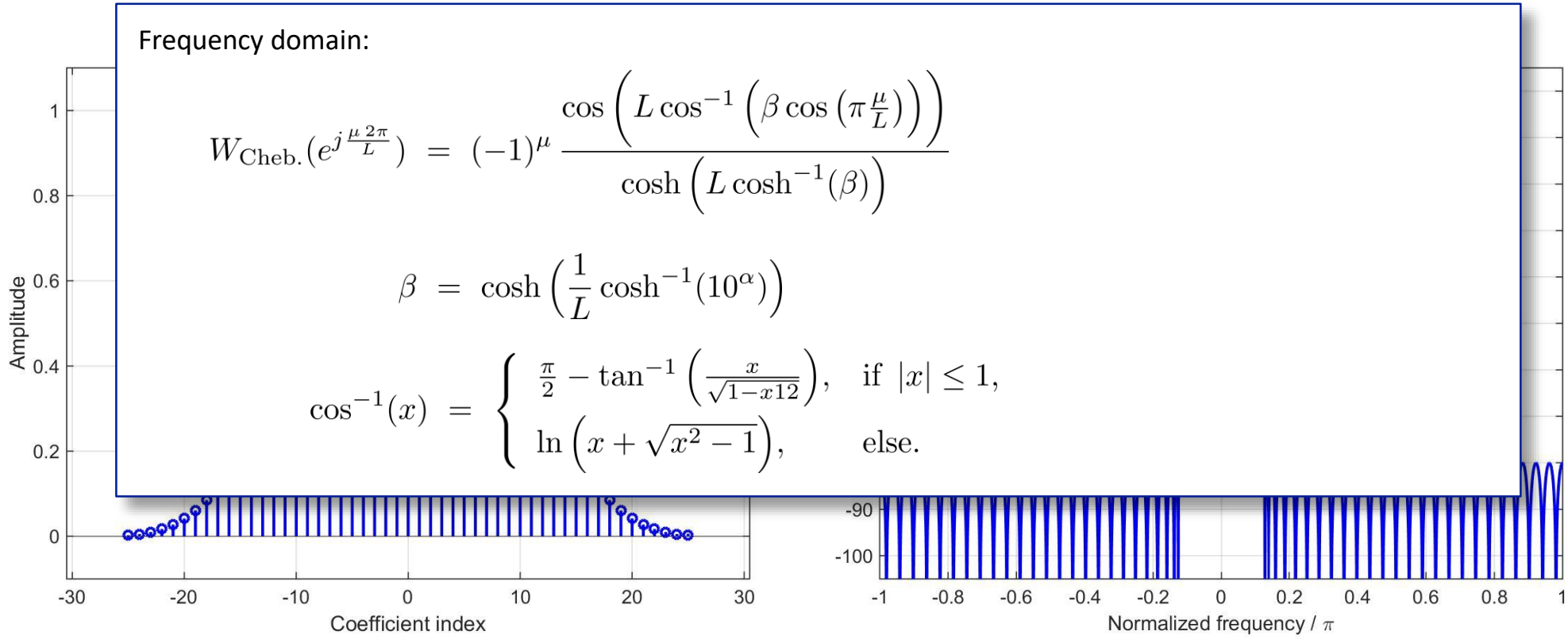
- Dolph-Chebychev window
- Optimized according to the minimax criterion
- Based on (very simple) poly-nomials in the frequency domain
- Side lobes adjusted to -80 dB

Frequency domain:

$$W_{\text{Cheb.}}(e^{j\frac{\mu 2\pi}{L}}) = (-1)^{\mu}\frac{\cos\left(L\cos^{-1}\left(\beta\cos\left(\pi\frac{\mu}{L}\right)\right)\right)}{\cosh\left(L\cosh^{-1}(\beta)\right)}$$

$$\beta = \cosh\left(\frac{1}{L}\cosh^{-1}(10^{\alpha})\right)$$

$$\cos^{-1}(x) = \begin{cases} \frac{\pi}{2} - \tan^{-1}\left(\frac{x}{\sqrt{1-x12}}\right), & \text{if } |x| \le 1, \\ \ln\left(x+\sqrt{x^2-1}\right), & \text{else.} \end{cases}$$

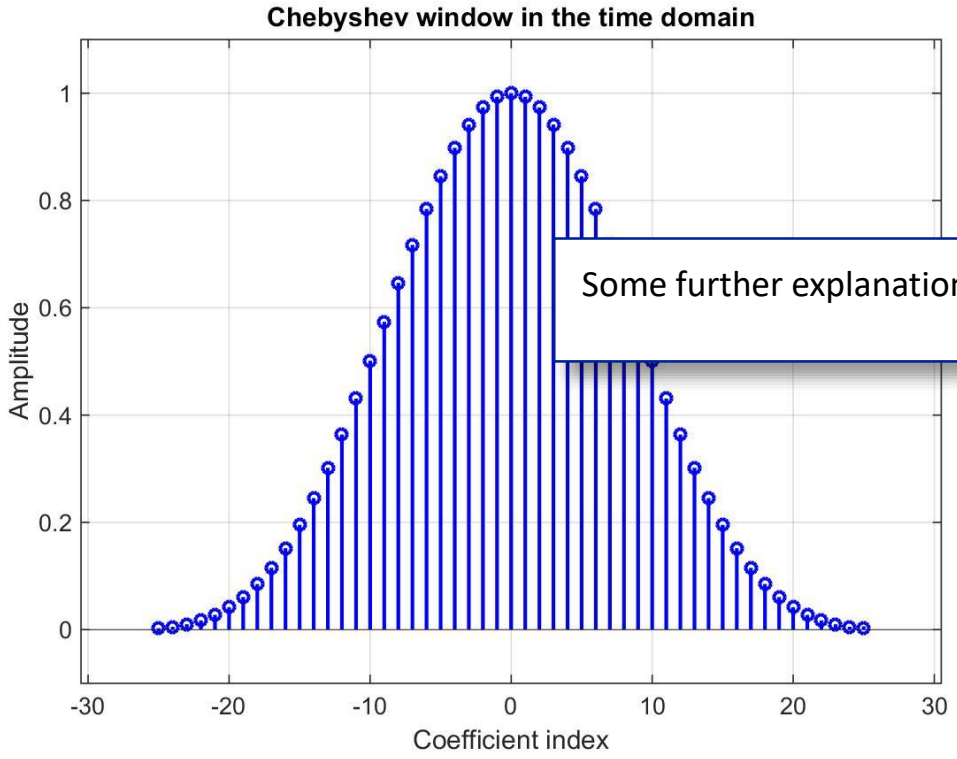


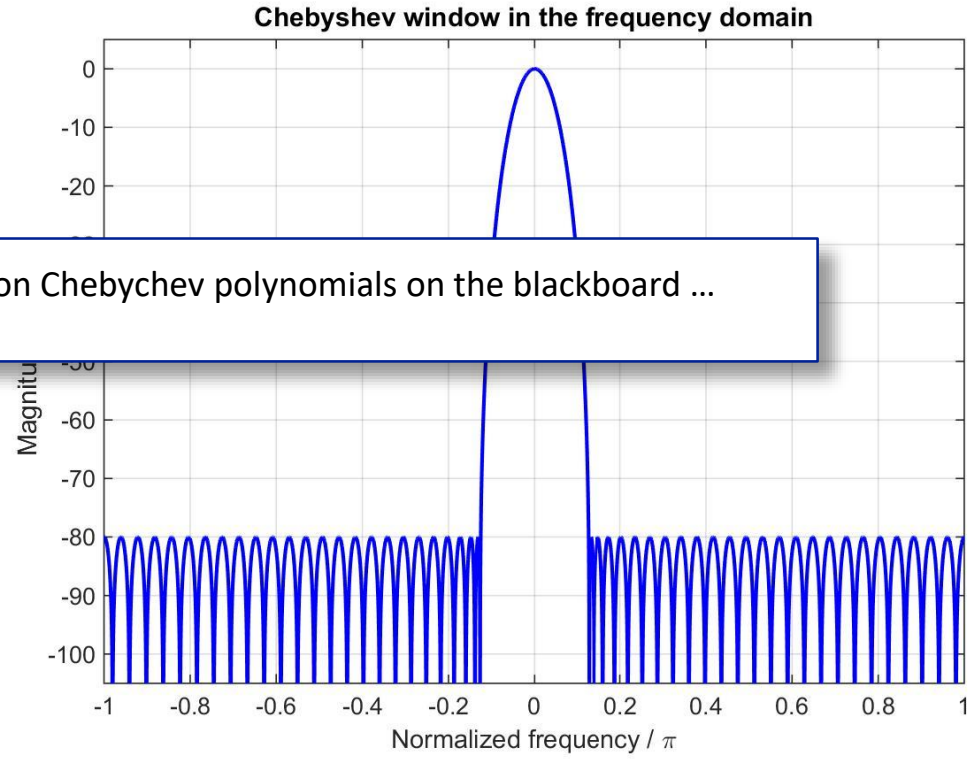
Linear Filtering in the DFT Domain – Part 51

Frequency analysis of stationary signals – Comparison of window sequences – Part 26

Properties to be mentioned:

- Dolph-Chebychev window
- Optimized according to the minimax criterion
- Based on (very simple) poly-nomials in the frequency domain
- Side lobes adjusted to -80 dB





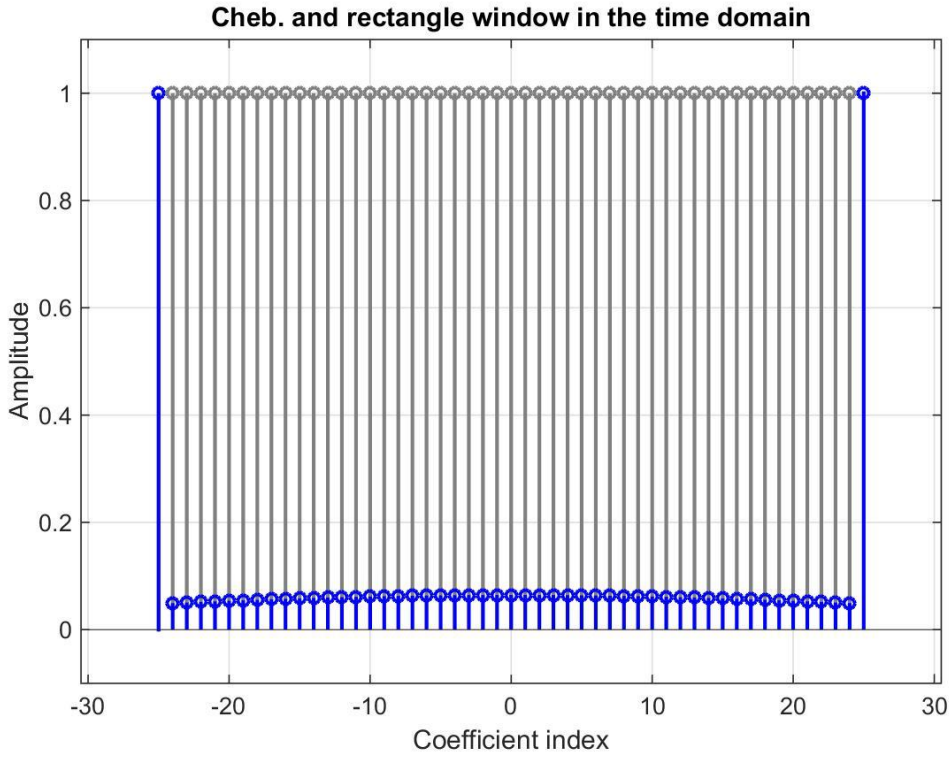
Some further explanations on Chebychev polynomials on the blackboard …

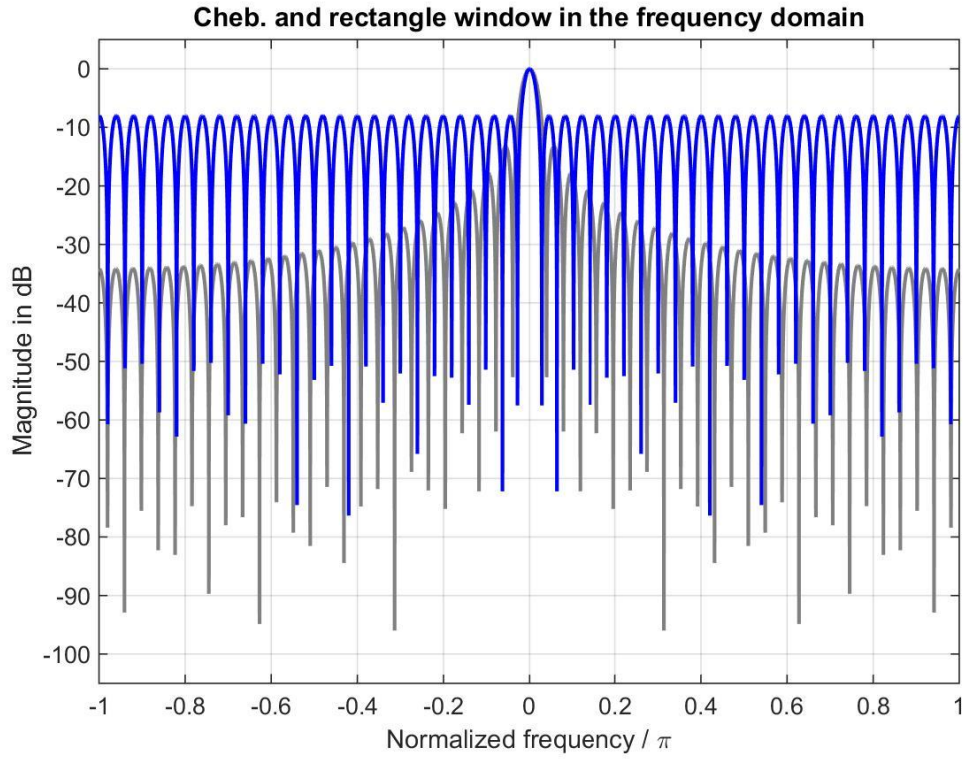
Linear Filtering in the DFT Domain – Part 52

Frequency analysis of stationary signals – Comparison of window sequences – Part 27

Properties to be mentioned:

- Dolph-Chebychev window
- Optimized according to the minimax criterion
- Based on (very simple) poly-nomials in the frequency domain
- Side lobes adjusted to -10 dB



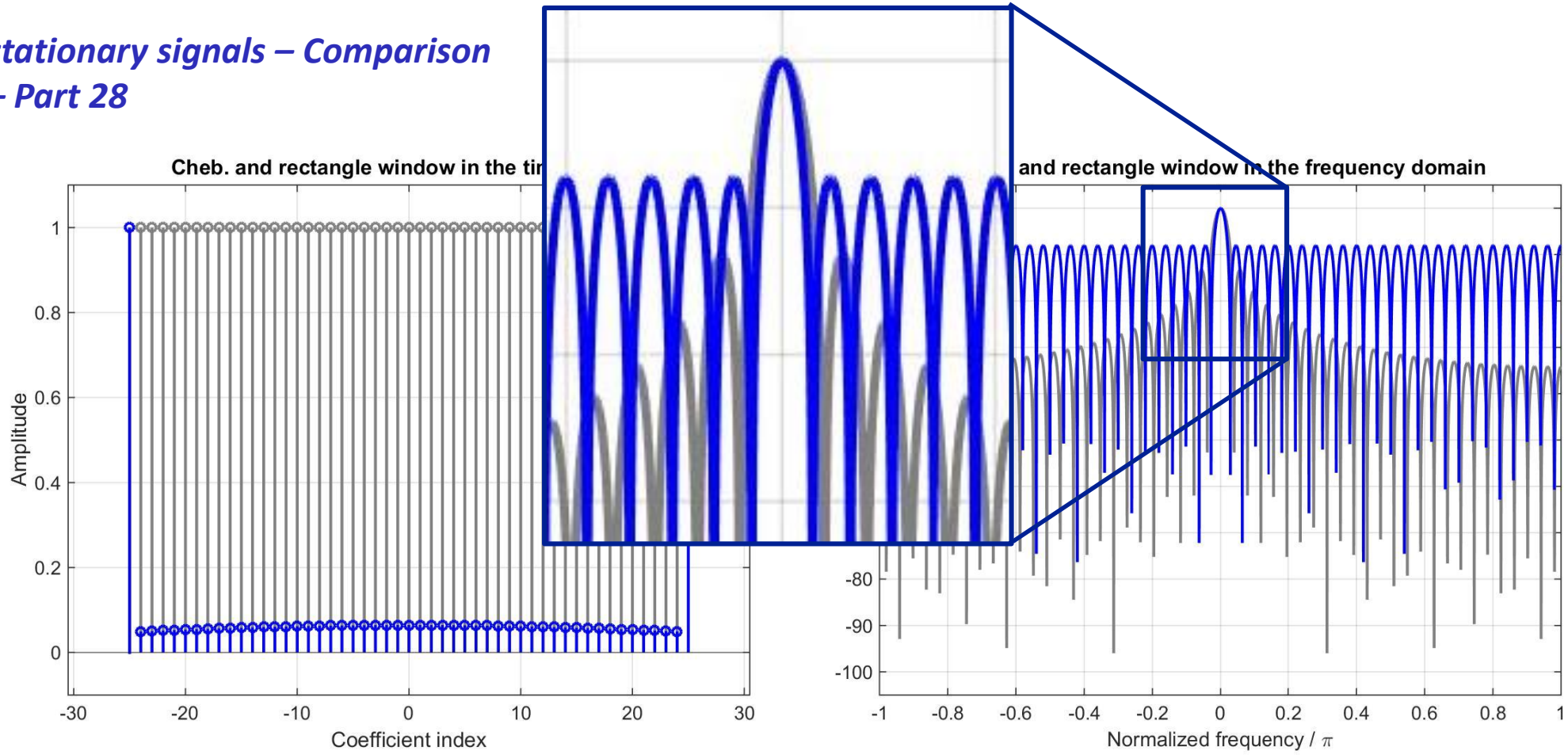


Linear Filtering in the DFT Domain – Part 53

Frequency analysis of stationary signals – Comparison of window sequences – Part 28

Properties to be mentioned:

- Dolph-Chebychev window
- Optimized according to the minimax criterion
- Based on (very simple) poly-nomials in the frequency domain
- Side lobes adjusted to -10 dB

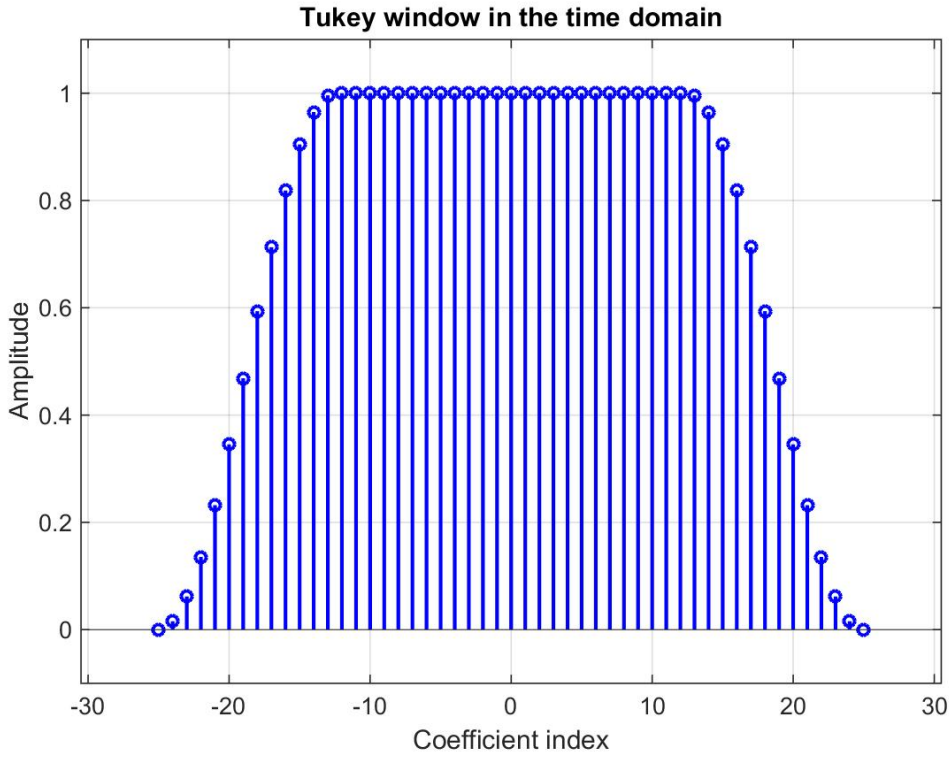


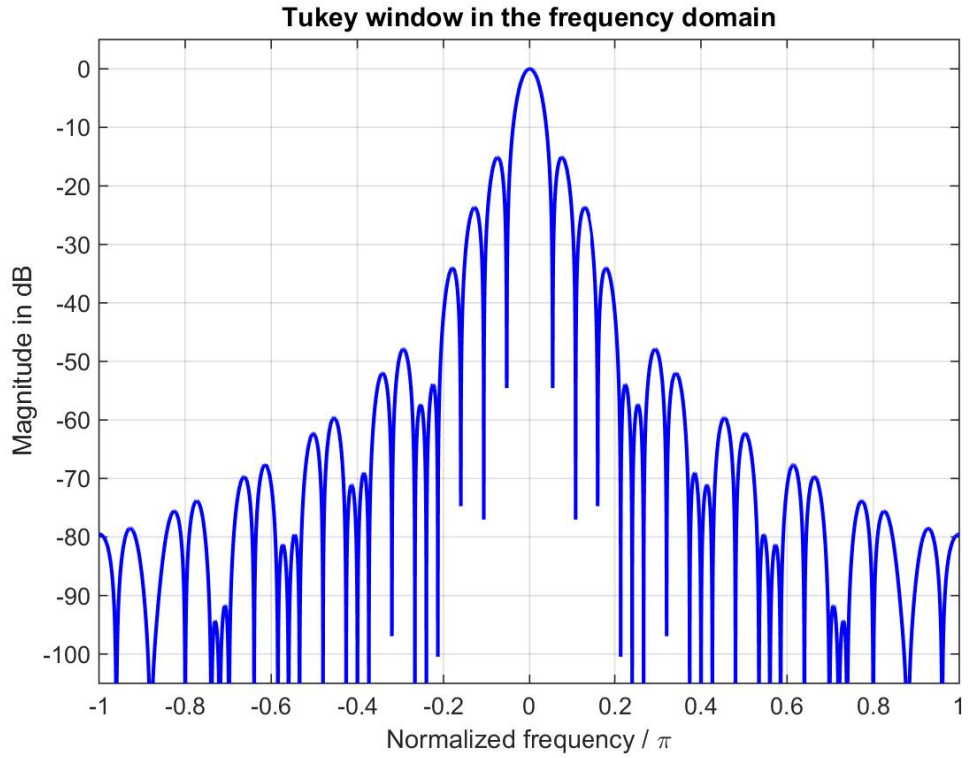
Linear Filtering in the DFT Domain – Part 54

Frequency analysis of stationary signals – Comparison of window sequences – Part 29

Properties to be mentioned:

- Combining cos and rectangle sequences
- Well suited for extraction certain time-domain parts



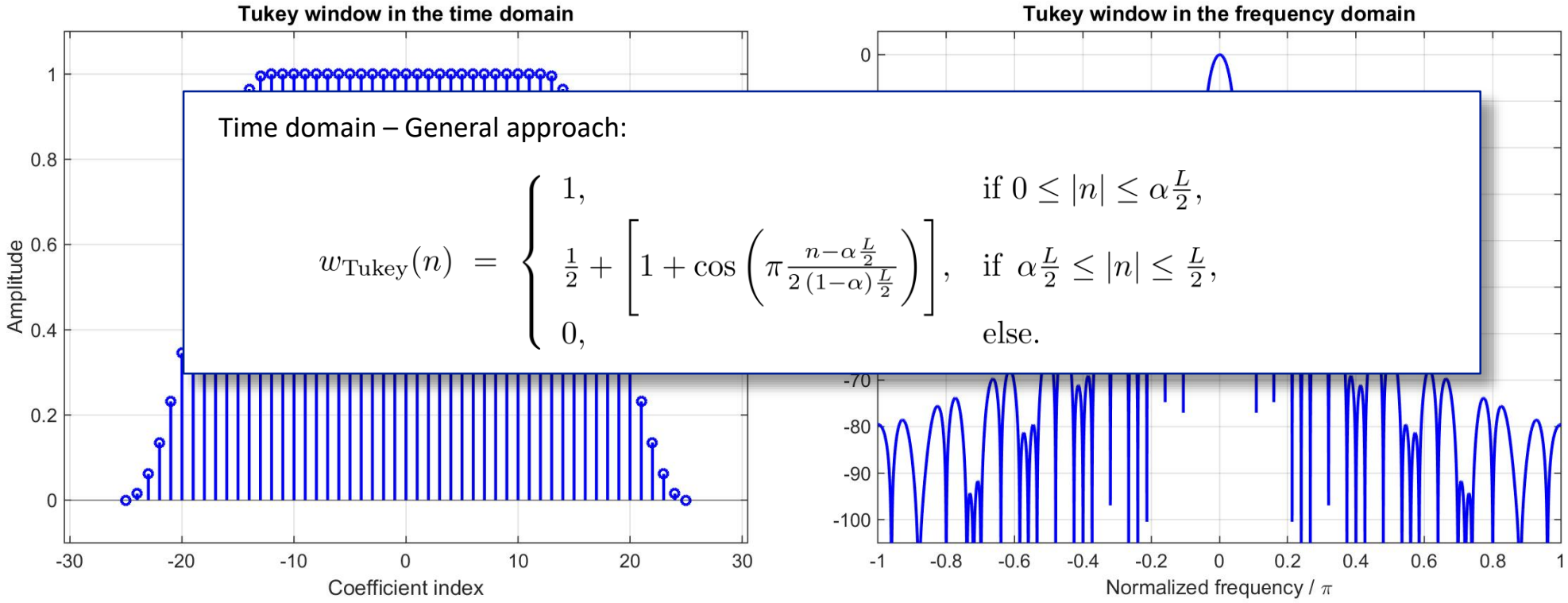


Linear Filtering in the DFT Domain – Part 55

Frequency analysis of stationary signals – Comparison of window sequences – Part 30

Properties to be mentioned:

- Combining cos and rectangle sequences
- Well suited for extraction certain time-domain parts



Time domain – General approach:

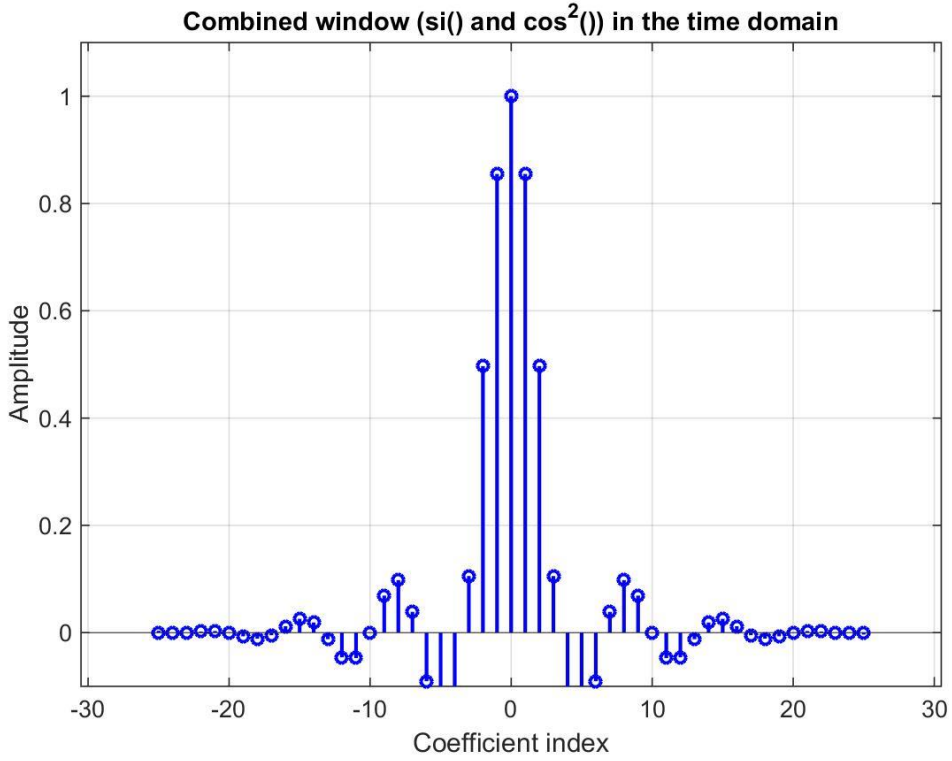
$$w_{\text{Tukey}}(n) = \begin{cases} 1, & \text{if } 0 \le |n| \le \alpha\frac{L}{2}, \\ \frac{1}{2} + \left[1 + \cos\left(\pi \frac{n - \alpha\frac{L}{2}}{2\,(1-\alpha)\frac{L}{2}}\right)\right], & \text{if } \alpha\frac{L}{2} \le |n| \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

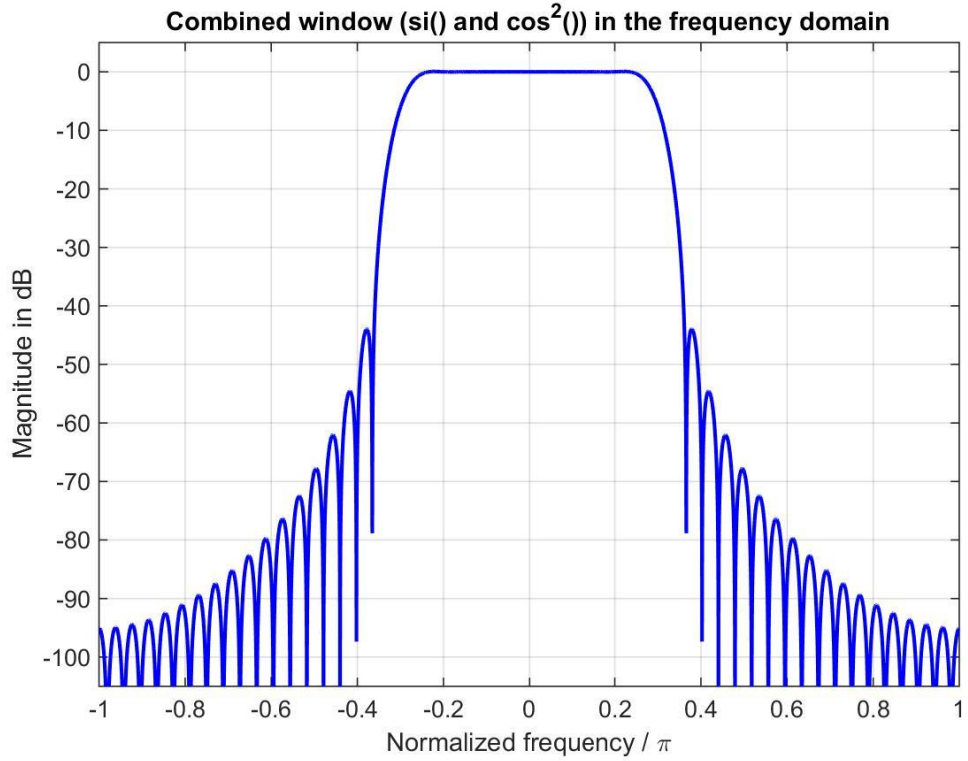
Linear Filtering in the DFT Domain – Part 56

Frequency analysis of stationary signals – Comparison of window sequences – Part 31

Properties to be mentioned:

- Combining finite and infinite windows
- Example: cos² and sin(x)/x
- E.g. used for specific beam-forming applications



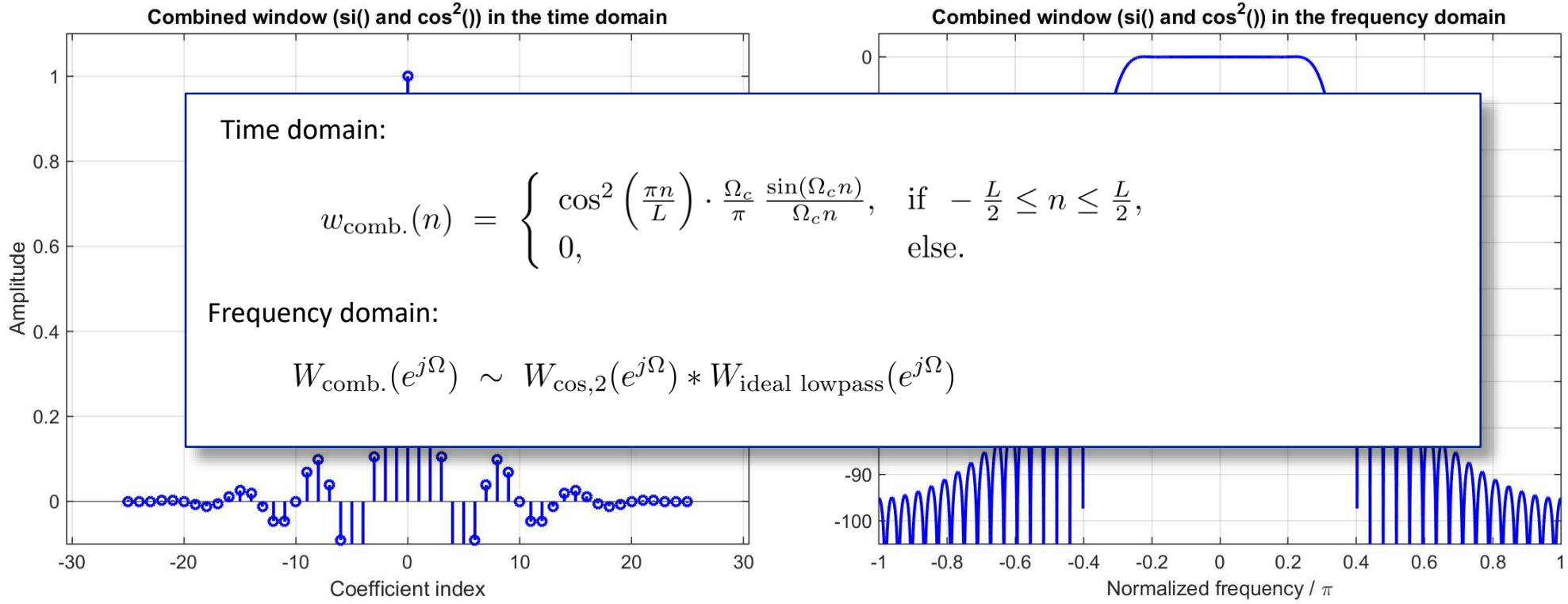


Linear Filtering in the DFT Domain – Part 57

Frequency analysis of stationary signals – Comparison of window sequences – Part 32

Properties to be mentioned:

- Combining finite and infinite windows
- Example: cos² and sin(x)/x
- E.g. used for specific beam-forming applications



Time domain:

$$w_{\text{comb.}}(n) = \begin{cases} \cos^2\left(\frac{\pi n}{L}\right) \cdot \frac{\Omega_c}{\pi} \frac{\sin(\Omega_c n)}{\Omega_c n}, & \text{if } -\frac{L}{2} \le n \le \frac{L}{2}, \\ 0, & \text{else.} \end{cases}$$

Frequency domain:

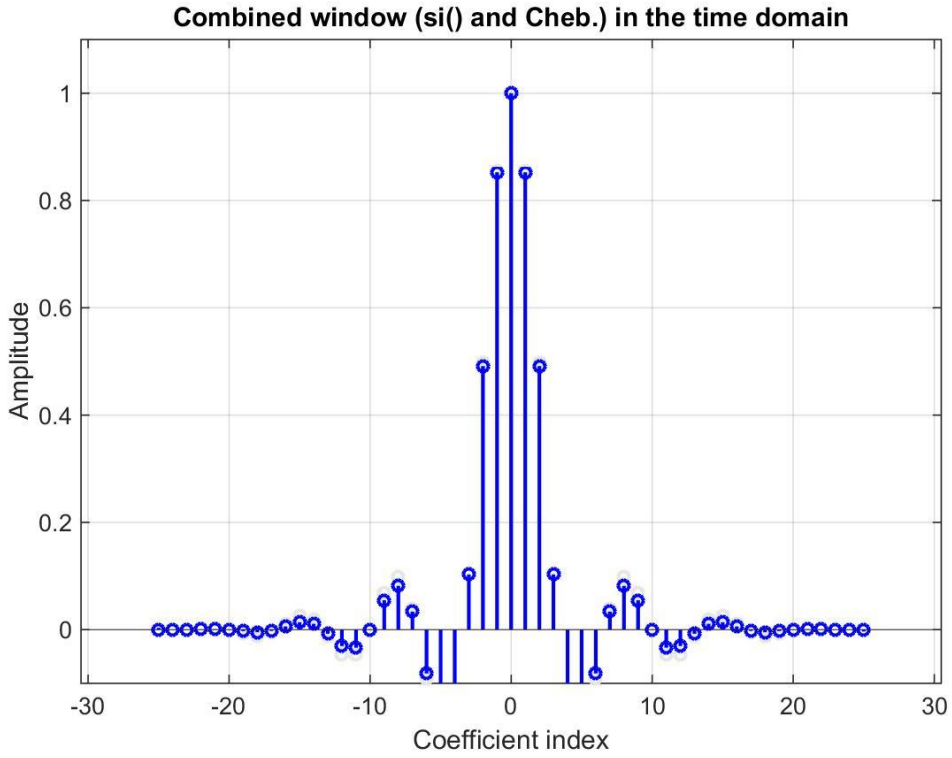
$$W_{\text{comb.}}(e^{j\Omega}) \sim W_{\cos,2}(e^{j\Omega}) * W_{\text{ideal lowpass}}(e^{j\Omega})$$

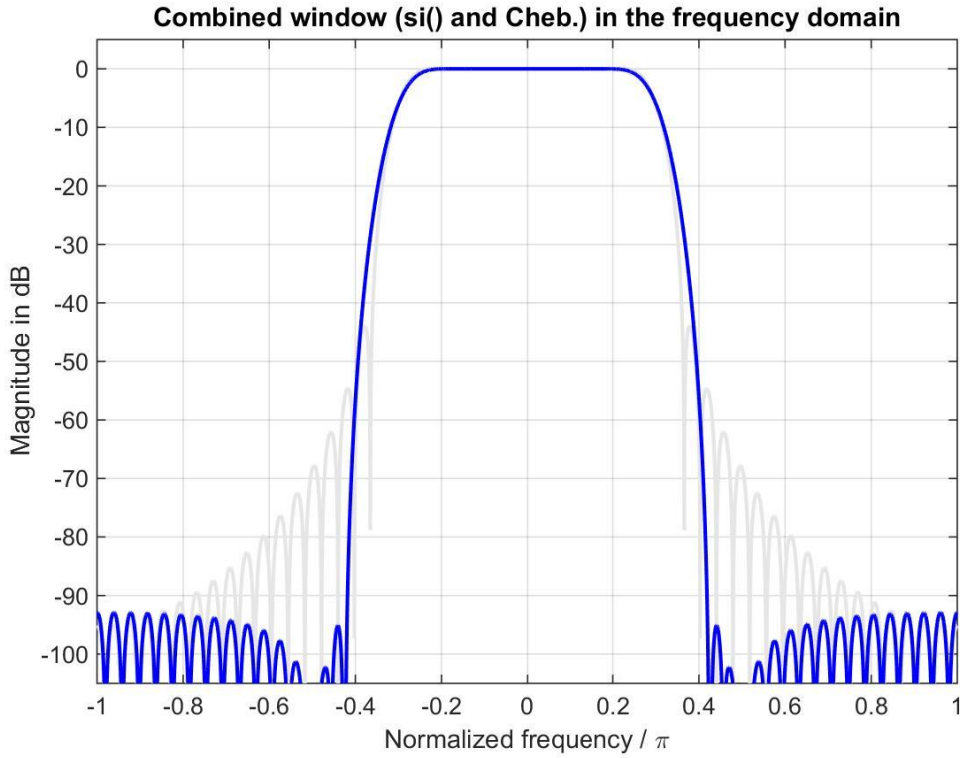
Linear Filtering in the DFT Domain – Part 58

Frequency analysis of stationary signals – Comparison of window sequences – Part 33

Properties to be mentioned:

- Combining finite and infinite windows
- Example: sin(x)/x and Chebychev window
- E.g. used for specific beam-forming applications





Linear Filtering in the DFT Domain – Part 59

Frequency analysis of stationary signals – Comparison of window sequences – Part 34

Could be continued virtually endlessly, but for now it should be enough …

Linear Filtering in the DFT Domain – Part 59

Frequency analysis of stationary signals – Windowing

Partner work – Please think about the following questions and try to find answers
(first group discussions, afterwards broad discussion in the whole group).

- Why do we apply a window function before performing a Fourier transform?

  ……

  ……

- How do you select a window function? What prior information might be useful to know before you chose a window function?

  ……

  ……

- Which window would you chose if you need a narrow main lobe? Is your choice optimal in some sense?

  ……

  ……

Fast Computation of the DFT: The FFT – Part 1

Basics – Part 1

When computing the ***complexity of a DFT*** for the complex signals $v(n) \in \mathbb{C}$ and complex spectra $V_M(\mu) \in \mathbb{C}$, we obtain

$$V_M(\mu) = \underbrace{\sum_{n=0}^{M-1} \underbrace{v(n) \cdot e^{-j\mu\frac{2\pi}{M}n}}_{1 \text{ compl. mult.}}}_{M \text{ compl. mult.}} \quad \text{with} \quad \underbrace{\mu \in \{0, 1, ..., M-1\}}_{M \text{ result values}}.$$

In total we need about M^2 complex multiplications and additions

Remarks (part 1):

- 1 complex multiplication $\rightarrow$ 4 real multiplications + 2 real additions,
  1 complex addition $\rightarrow$ 2 real additions.
- When looking closer we see that not all operations require the above mentioned complexity:
  - M values have to be added, $\Rightarrow (M-1)$ additions,
  - for the factors e^{j0}, $e^{j\pi\lambda}$, $e^{\pm j\lambda\pi/2}$ multiplications are not required,
  - for $\mu = 0$ multiplications are not necessary.

Fast Computation of the DFT: The FFT – Part 2

Basics – Part 2

Remarks (part 2):

- ***Multiplications*** and ***additions*** are not the only operations that should be considered for analyzing the computational complexity.
- ***Memory access operations***, ***checking conditions***, etc. are as important as additions and multiplications.
- Cost functions for ***complexity measures*** should be ***adapted*** to the ***individually used hardware***.

Basic idea for reducing the computational complexity:

The basic idea for reducing the complexity of a DFT is to decompose the „***big problem***“ into several „***smaller problems***“. This usually leads to a reduction in complexity. However, this „trick“ can not always be applied.

Fast Computation of the DFT: The FFT – Insertion Part 1

The Goertzel Algorithm

Basics:

- The ***name*** stems for an American physicist (1919 – 2002).
- The ***basic idea*** of this algorithm is to ***reduce the number of multiplications*** as much as possible if only a ***single DFT pin*** should be computed.
- The principle works also if an ***arbitrary frequency supporting point*** (not on the DFT grid) should be computed.

Number of real multiplication for a single bin:

- The signal $v(n)$ is assumed to be real.

$$V_M(\mu_0) = \underbrace{\sum_{n=0}^{M-1} \underbrace{v(n) \cdot e^{-j\mu_0 \frac{2\pi}{M} n}}_{2 \text{ real mult.}}}_{2M \text{ real mult.}}.$$

Fast Computation of the DFT: The FFT – Insertion Part 2

The Goertzel Algorithm

Derivation:

$$V_M(\mu_0) = \sum_{n=0}^{M-1} v(n) \cdot e^{-j\mu_0 \frac{2\pi}{M} n}$$

... inserting the abbreviation $W_M = e^{-j\frac{2\pi}{M}}$ *(twiddle factor) ...*

$$= \sum_{n=0}^{M-1} v(n) \cdot W_M^{\mu_0 n}$$

... exploiting that $W_M^{-\mu_0 M} = e^{j\frac{2\pi}{M}\mu_0 M} = e^{j2\pi\mu_0} = 1$ *...*

$$= \sum_{n=0}^{M-1} v(n) \cdot W_M^{-\mu_0\,(M-n)}.$$

Interpretation as a convolution (but only at one specific sample):

$$V_M(\mu_0) = \sum_{n=0}^{M-1} v(n) \cdot W_M^{-\mu_0\,(M-n)} = \left(v(n) * W_M^{-n\mu_0}\right)\Big|_{n=M}.$$

Fast Computation of the DFT: The FFT – Insertion Part 3

The Goertzel Algorithm

Interpretation as a convolution (but only at one specific sample):

$$V_M(\mu_0) = \sum_{n=0}^{M-1} v(n) \cdot W_M^{-\mu_0\,(M-n)} = \left(v(n) * W_M^{-(n+1)\mu_0}\right)\Big|_{n=M-1}.$$

FIR filter with the coefficients $h_i = W_M^{-(i+1)\mu_0}$ ***with*** $i \in \{0, 1, ..., M-1\}$ **.**

Example for $n = 4$:

$v(4)$	$v(3)$	$v(2)$	$v(1)$	$v(0)$	0	0	0
$W_M^{-1\mu_0}$	$W_M^{-2\mu_0}$	$W_M^{-3\mu_0}$	...	...	...	...	$W_M^{-M\mu_0}$

Example for $n = M-1$:

$v(M-1)$	$v(M-2)$	$v(M-3)$	...	...	...	...	$v(0)$
$W_M^{-1\mu_0}$	$W_M^{-2\mu_0}$	$W_M^{-3\mu_0}$	...	...	...	...	$W_M^{-M\mu_0}$

Fast Computation of the DFT: The FFT – Insertion Part 4

The Goertzel Algorithm

Extending the filter:

Example for $n = M - 1$, "***normal***" filter:

$v(M-1)$	$v(M-2)$	$v(M-3)$	...	...	...	...	$v(0)$

$W_M^{-1\mu_0}$	$W_M^{-2\mu_0}$	$W_M^{-3\mu_0}$	...	...	...	...	$W_M^{-M\mu_0}$

Example for $n = M - 1$, "***extended***" filter:

$v(M-1)$	$v(M-2)$	$v(M-3)$	...	...	...	...	$v(0)$	0	0	...

$W_M^{-1\mu_0}$	$W_M^{-2\mu_0}$	$W_M^{-3\mu_0}$	...	...	...	...	$W_M^{-M\mu_0}$	$W_M^{-1\mu_0}$	$W_M^{-2\mu_0}$	...

Fast Computation of the DFT: The FFT – Insertion Part 5

The Goertzel Algorithm

Some more specific look on the filter realization:

$$H(z) = \sum_{i=0}^{\infty} h_i \, z^{-i}$$

... inserting the definition of the filter coefficients $h_i = W_M^{-(i+1)\mu_0}$ *...*

$$= \sum_{i=0}^{\infty} W_M^{-(i-1)\mu_0} \, z^{-i}$$

... exclude one of the twiddle factors ...

$$= W_M^{\mu_0} \cdot \sum_{i=0}^{\infty} W_M^{-i\mu_0} \, z^{-i}$$

... multiply with a "complicatedly written 1"...

$$= W_M^{\mu_0} \cdot \frac{1 - W_M^{-\mu_0} z^{-1}}{1 - W_M^{-\mu_0} z^{-1}} \cdot \sum_{i=0}^{\infty} W_M^{-i\mu_0} \, z^{-i}$$

Fast Computation of the DFT: The FFT – Insertion Part 6

The Goertzel Algorithm

Some more specific look on the filter realization (continued):

$$H(z) = W_M^{\mu_0} \cdot \frac{1 - W_M^{-\mu_0} z^{-1}}{1 - W_M^{-\mu_0} z^{-1}} \cdot \sum_{i=0}^{\infty} W_M^{-i\mu_0} z^{-i}$$

… multiply the numerator with the sum …

$$= W_M^{\mu_0} \cdot \frac{\sum_{i=0}^{\infty} W_M^{-i\mu_0} z^{-i} - \sum_{i=0}^{\infty} W_M^{-(i+1)\mu_0} z^{-(i+1)}}{1 - W_M^{-\mu_0} z^{-1}}$$

… exploit that only the sums differ only in one element …

$$= W_M^{\mu_0} \cdot \frac{W_M^0 z^0}{1 - W_M^{-\mu_0} z^{-1}}$$

… simplify the numerator …

$$= W_M^{\mu_0} \cdot \frac{1}{1 - W_M^{-\mu_0} z^{-1}}$$

Fast Computation of the DFT: The FFT – Insertion Part 7

The Goertzel Algorithm

Some more specific look on the filter realization (continued):

$$
\begin{aligned}
H(z) &= W_M^{\mu_0} \cdot \frac{1}{1 - W_M^{-\mu_0}\, z^{-1}} \\
&\quad \textit{... multiply with a “complicatedly written 1”...} \\
&= W_M^{\mu_0} \cdot \frac{1 - W_M^{\mu_0}\, z^{-1}}{1 - W_M^{\mu_0}\, z^{-1}} \cdot \frac{1}{1 - W_M^{-\mu_0}\, z^{-1}} \\
&\quad \textit{... combine both denominators ...} \\
&= W_M^{\mu_0} \cdot \frac{1 - W_M^{\mu_0}\, z^{-1}}{1 - 2\cos(\frac{2\pi}{M}\mu_0)\, z^{-1} + z^{-2}} \\
&\quad \textit{... split the filter into three subfilter ...} \\
&= \frac{1}{1 - 2\cos(\frac{2\pi}{M}\mu_0)\, z^{-1} + z^{-2}} \cdot \left(1 - W_M^{\mu_0}\, z^{-1}\right) \cdot W_M^{\mu_0}
\end{aligned}
$$

Fast Computation of the DFT: The FFT – Insertion Part 8

The Goertzel Algorithm

Some more specific look on the filter realization (continued):

$$H(z) = \underbrace{\frac{1}{1 - 2\cos(\frac{2\pi}{M}\mu_0)\, z^{-1} + z^{-2}}}_{H_0(z)} \cdot \underbrace{\left(1 - W_M^{\mu_0}\, z^{-1}\right)}_{H_1(z)} \cdot \underbrace{W_M^{\mu_0}}_{H_2(z)}$$

First filter, computed from $n = 0$ to $n = M - 1$:

$$H_0(z) = \frac{1}{1 - \underbrace{2\,\cos(\frac{2\pi}{M}\mu_0)}_{b_1}\, z^{-1} + z^{-2}}$$

$$y_0(z) = v(n) + b_1\, y_0(n-1) - y_0(n-2)$$

Second filter, computed only for $n = M - 1$:

$$H_1(z) = \left(1 - \underbrace{W_M^{\mu_0}}_{a_1}\, z^{-1}\right)$$

$$y_1(z) = y_0(n) - a_1\, y_0(n-1)$$

Fast Computation of the DFT: The FFT – Insertion Part 9

The Goertzel Algorithm

Some more specific look on the filter realization (continued):

$$H(z) = \underbrace{\frac{1}{1 - 2\cos(\frac{2\pi}{M}\mu_0)\, z^{-1} + z^{-2}}}_{H_0(z)} \cdot \underbrace{\left(1 - W_M^{\mu_0}\, z^{-1}\right)}_{H_1(z)} \cdot \underbrace{W_M^{\mu_0}}_{H_2(z)}$$

Third filter, computed only for $n = M - 1$:

$$H_1(z) = W_M^{\mu_0} \qquad\qquad y(z) = W_M^{\mu_0}\, y_1(n)$$

Fast Computation of the DFT: The FFT – Insertion Part 10

The Goertzel Algorithm

Some more specific look on the filter realization (continued):

$$H(z) = \underbrace{\frac{1}{1 - 2\cos(\frac{2\pi}{M}\mu_0)\, z^{-1} + z^{-2}}}_{H_0(z)} \cdot \underbrace{\left(1 - W_M^{\mu_0}\, z^{-1}\right)}_{H_1(z)} \cdot \underbrace{W_M^{\mu_0}}_{H_2(z)}$$

The ***first filter*** requires one real multiplication per sample, in total N real multiplications.

The ***second filter*** requires two real multiplication (only for the final sample).

The ***third filter*** requires two real multiplication (only for the final sample).

Fast Computation of the DFT: The FFT – Part 3

Radix-2 approach (decimation in time) – Part 1

For fast (efficient) realizations of the DFT some properties of the so-called twiddle factors $W_M^{\mu n} = e^{-j\mu\frac{2\pi}{M}n}$ can be used.
In particular we can utilize

- the conjugate complex ***symmetry***

$$W_M^{-\mu n} = \left[W_M^{\mu n}\right]^*$$

- and the ***periodicity*** both for n and for μ

$$W_M^{\mu n} = W_M^{\mu(n+\lambda M)} = W_M^{(\mu+\lambda M)n}.$$

For a so-called ***radix 2 realization*** of the DFT we ***decompose*** the input series ***into two series of half length***:

$$\begin{aligned} v_1(n) &= v(2n), \\ v_2(n) &= v(2n+1), \\ &\text{each with } n \in \{0, 1, ..., \frac{M}{2} - 1\}. \end{aligned}$$

Fast Computation of the DFT: The FFT – Part 4

Radix-2 approach (decimation in time) – Part 2

If we assume that the orignal series has an ***even length***, we can decompose it according to

$$V_M(\mu) = \sum_{n=0}^{M-1} v(n) \overbrace{e^{-j\frac{2\pi}{M}\mu n}}^{W_M^{\mu n}}, \quad \mu \in \{0, 1, ..., M-1\}$$

... inserting the definition of the twiddle factors ...

$$= \sum_{n=0}^{M-1} v(n)\, W_M^{\mu n}$$

... splitting the sum into even and odd indices ...

$$= \sum_{n=0}^{M/2-1} \underbrace{v(2n)}_{v_1(n)} W_M^{2\mu n} + \sum_{n=0}^{M/2-1} \underbrace{v(2n+1)}_{v_2(n)} W_M^{\mu(2n+1)}$$

... inserting the (signal) definitions from the last slide ...

$$= \sum_{n=0}^{M/2-1} v_1(n) \underbrace{W_M^{2\mu n}}_{W_{M/2}^{\mu n}} + W_M^{\mu} \sum_{n=0}^{M/2-1} v_2(n) \underbrace{W_M^{2\mu n}}_{W_{M/2}^{\mu n}}$$

Fast Computation of the DFT: The FFT – Part 5

Radix-2 approach (decimation in time) – Part 3

DFT decomposition (continued)

$$V_M(\mu) = \sum_{n=0}^{M/2-1} v_1(n) \underbrace{W_M^{2\mu n}}_{W_{M/2}^{\mu n}} + W_M^{\mu} \sum_{n=0}^{M/2-1} v_2(n) \underbrace{W_M^{2\mu n}}_{W_{M/2}^{\mu n}}$$

… inserting the definition of a DFT (of half length) …

$$= V_{M/2,1}(\mu) + W_M^{\mu} V_{M/2,2}(\mu).$$

DFT of length $\frac{M}{2}$ *for the signal* $v_2(n)$

DFT of length $\frac{M}{2}$ *for the signal* $v_1(n)$

Please note that this decomposition is – due to the periodicity of $V_{M/2,1}(\mu)$ and $V_{M/2,2}(\mu)$ – also true for $\mu \geq M/2$.

Fast Computation of the DFT: The FFT – Part 6

Radix-2 approach (decimation in time) – Part 4

For the ***computational complexity*** we obtain:

- ***Before the decomposition***:

  1 DFT of order M $\longrightarrow$ M^2 operations.

- ***After the decomposition***:

  2 DFTs of order $M/2$ $\longrightarrow$ $2\left(\frac{M}{2}\right)^2 = \frac{2\,M^2}{4} = \frac{M^2}{2}$ operations and

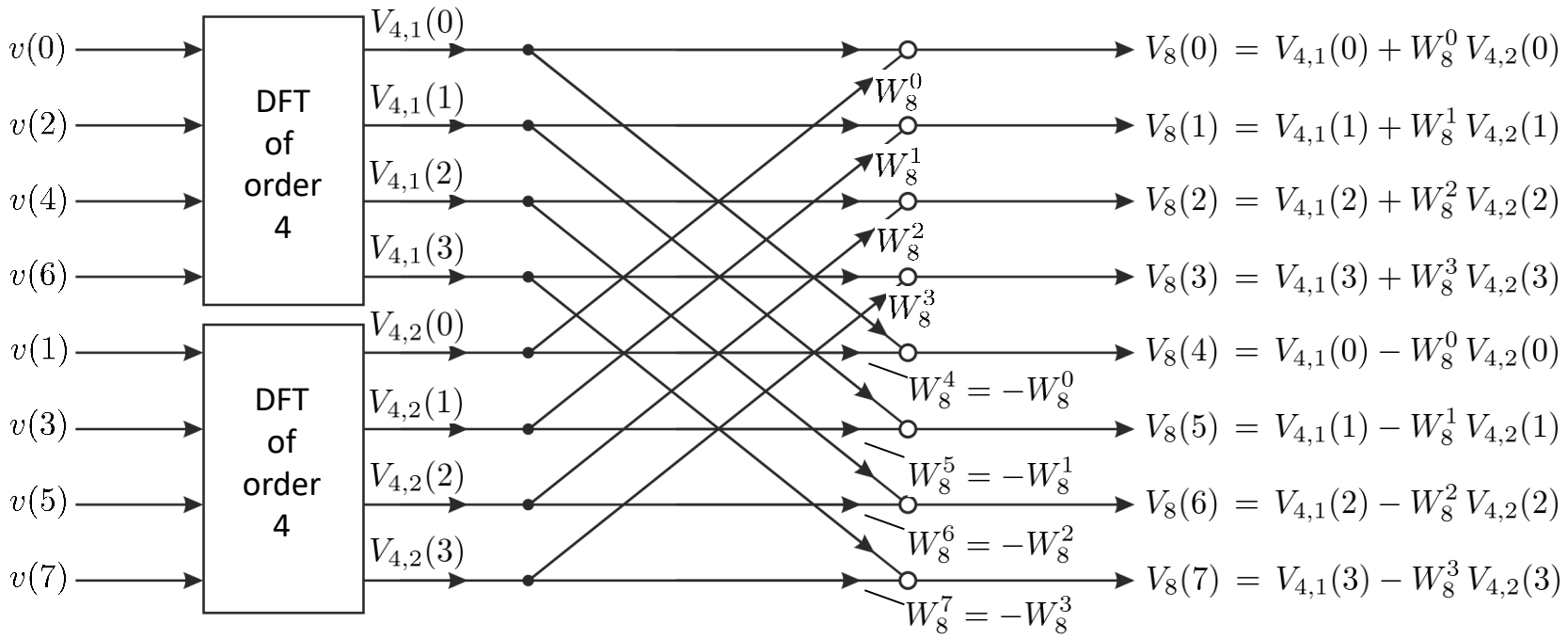
  combining the results $\longrightarrow$ M operations.

Using this „splitting" operation we were able to reduce the complexity form M^2 down to $\frac{1}{2}M^2 + M$. For large orders M this is about a half.

Fast Computation of the DFT: The FFT – Part 7

Radix-2 approach (decimation in time) – Part 5

Graphical explanation of (the first stage of) the decomposition for $M = 8$:



Fast Computation of the DFT: The FFT – Part 8

Radix-2 approach (decimation in time) – Part 6

This ***principle*** can be ***applied again***. Therefore, $M/2$ has to be even again. Then we get four DFTs of length $M/4$ and another M operations for combining those four DFTs such that we get the desired outputs. Together with the operations necessary for combining the low order DFTs we obtain a ***complexity*** of

$$\frac{M^2}{4} + 2M \text{ complex operations.}$$

We can apply this principle until we reach a „minimum order“ DFT of length 2. This can be achieved if we have

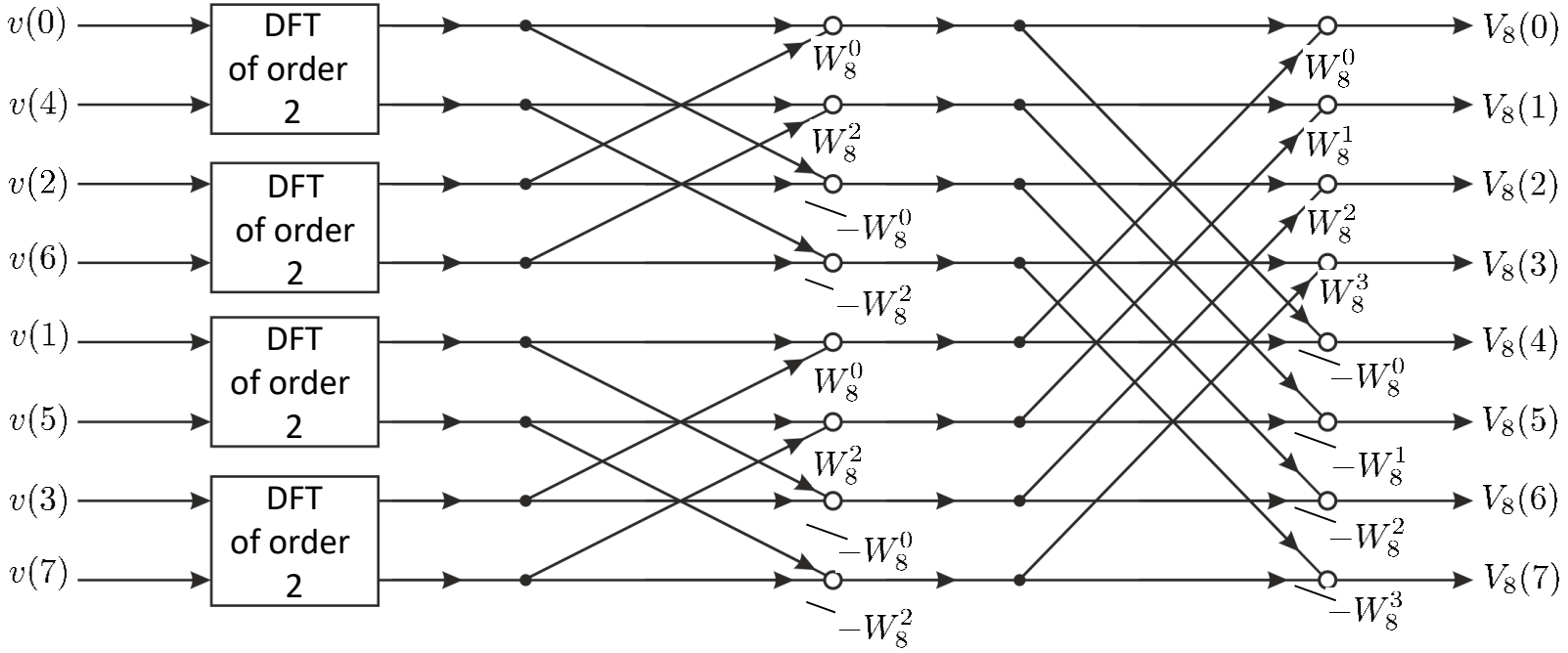
$$M = 2^m, \quad \text{with } m \in \mathbb{N}.$$

As a result M has to be a ***power of two***.

Fast Computation of the DFT: The FFT – Part 9

Radix-2 approach (decimation in time) – Part 7

Graphical explanation of (the second stage of) the decomposition for $M = 8$:



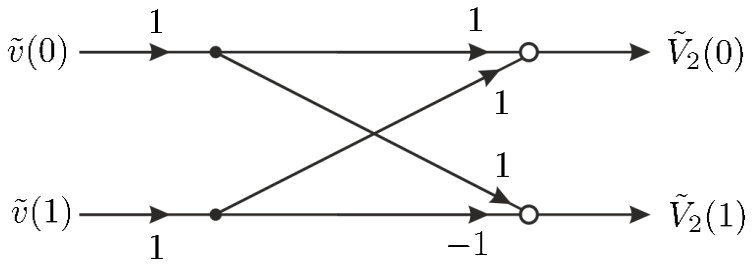
Fast Computation of the DFT: The FFT – Part 10

Radix-2 approach (decimation in time) – Part 8

As a last step we have to compute a DFT of length 2. This is achieved by:

$$\begin{aligned}\tilde{V}_2(0) &= \tilde{v}(0) + W_2^0\,\tilde{v}(1) = \tilde{v}(0) + \tilde{v}(1),\\ \tilde{V}_2(1) &= \tilde{v}(0) + W_2^1\,\tilde{v}(1) = \tilde{v}(0) - \tilde{v}(1).\end{aligned}$$

As we can see, also over here we need the same ***basic scheme*** that we have used also in the previous decompositions:

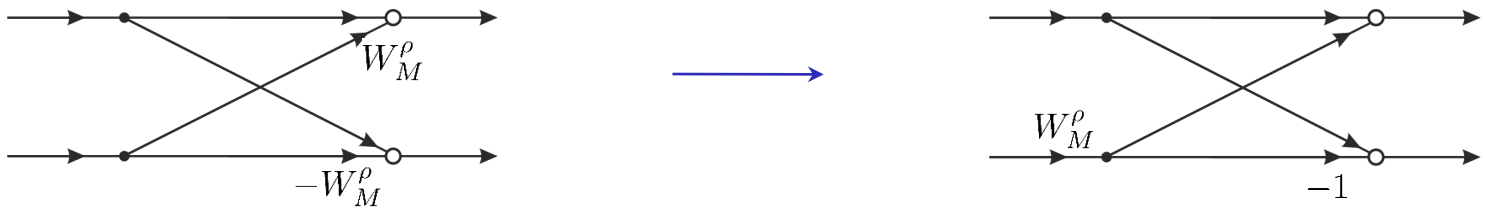


This basic scheme is called „***butterfly***“ of a radix-2 FFT. The abbreviation ***FFT*** stands for ***Fast Fourier Transform***.

Fast Computation of the DFT: The FFT – Part 11

Radix-2 approach (decimation in time) – Part 9

When computing the individual butterfly operations we can exploit that the twiddle factors W_M^ρ and $-W_M^\rho$ ***differ only in terms of their sign***. Thus, we can apply the following simplification:



Pathes without variables using a factor of 1!

This leads to a ***further reduction in terms of multiplications*** (only 50 % of them are really required).

In total we were able to reduce the required operations for computing a DFT from M^2 down to $M \log_2 M$ by using efficient radix-2 approaches.

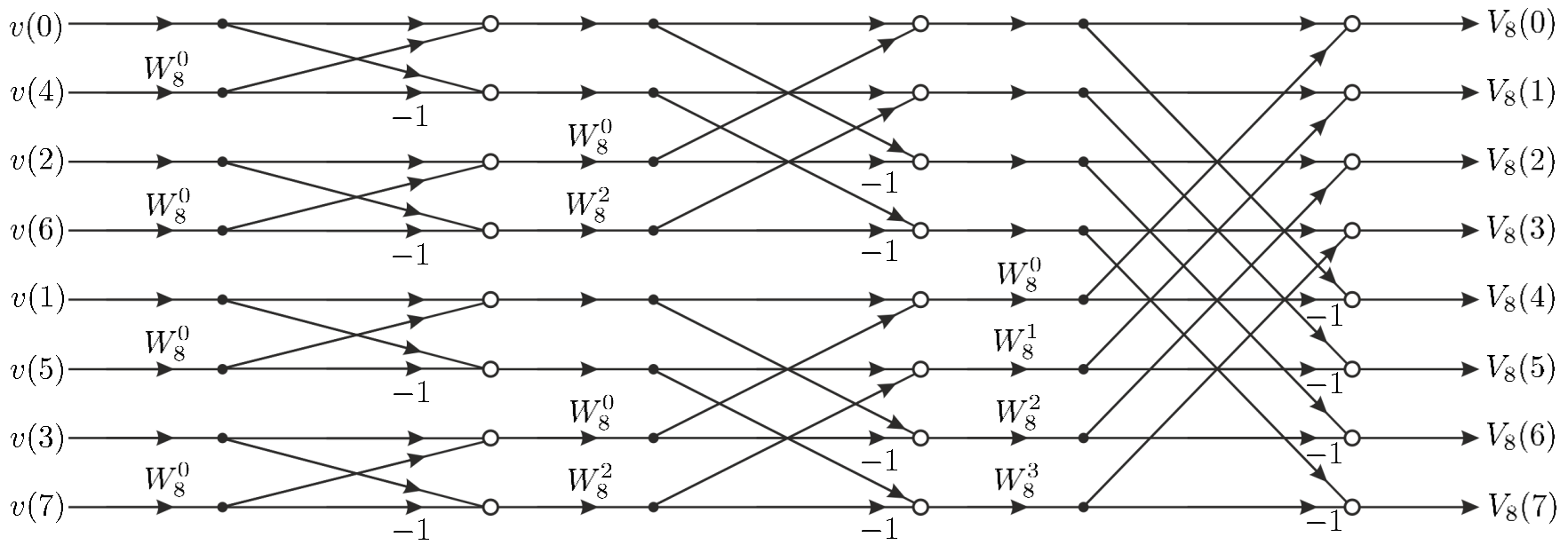
Examples:

$$M = 32 \quad \rightarrow \quad M^2 \approx 1000,\ M \log_2 M \approx 160 \quad \rightarrow \quad \text{Factor } 6,$$
$$M = 1024 \quad \rightarrow \quad M^2 \approx 10^6,\ \ M \log_2 M \approx 10^4 \quad \rightarrow \quad \text{Factor } 100.$$

Fast Computation of the DFT: The FFT – Part 12

Radix-2 approach (decimation in time) – Part 10

Graphical explanation of the decomposition for $M = 8$ ***with optimized butterfly structure***:

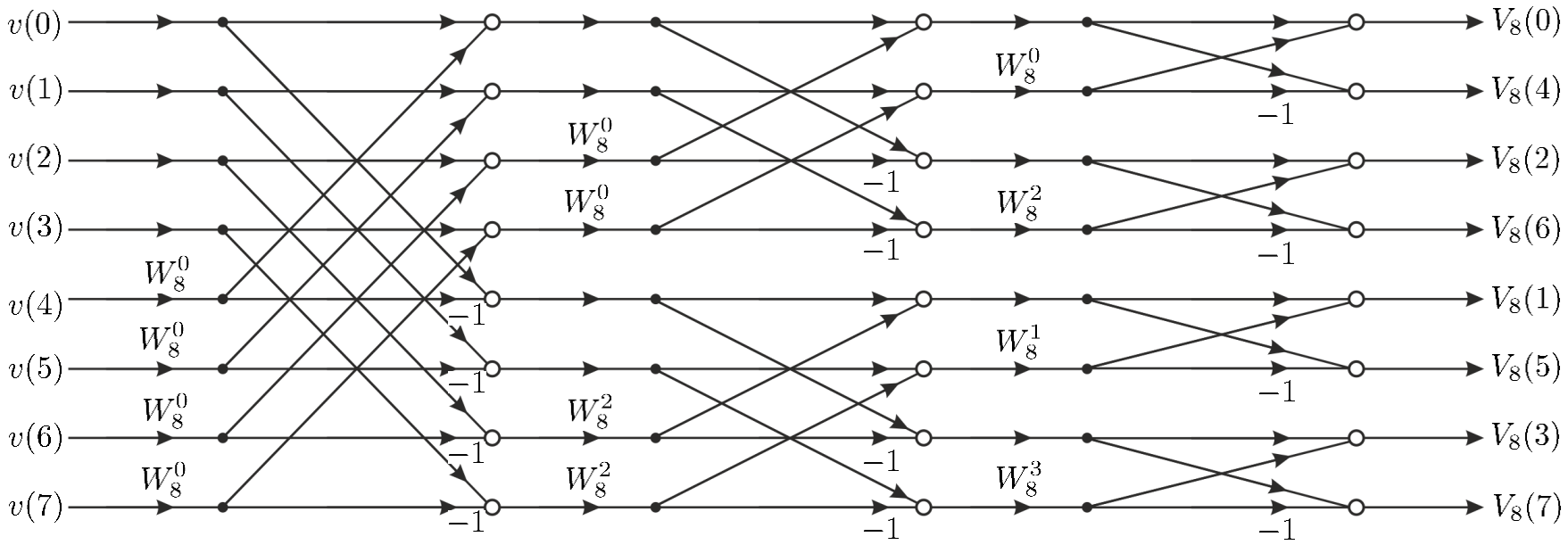


Please keep in mind that in each stage only „in-place operations“ are required.
This means that no now memory has to be allocated for a new stage!

Fast Computation of the DFT: The FFT – Part 13

Radix-2 approach (decimation in time) – Part 11

Graphical explanation of the decomposition for $M = 8$ (with keeping the „orientation“ of the input vector):



Fast Computation of the DFT: The FFT – Part 14

Radix-2 approach (decimation in time) – Part 12

Graphical explanation of the complexity reduction on the black board …

Fast Computation of the DFT: The FFT – Part 15

Radix-2-decimation-in-time FFT algorithms – Part 13

In-place computations

- The intermediate results $V_M^{(\ell)}(\mu_{1,2})$ in the l-th stage, $l = 0, \ldots, m-1$, are obtained as

$$V_M^{(\ell)}(\mu_1) = V_M^{(\ell-1)}(\mu_1) + W_M^{\rho}\, V_M^{(\ell-1)}(\mu_2),$$
$$V_M^{(\ell)}(\mu_2) = V_M^{(\ell-1)}(\mu_1) - W_M^{\rho}\, V_M^{(\ell-1)}(\mu_2)$$

  (butterfly computations) where $\mu_1, \mu_2, \rho \in \{0, \ldots, M-1\}$ vary from stage to stage.

- Only M storage cells are needed, which first contain the values $v(n)$, then the results form the individual stages and finally the values $V_M(\mu)$.

  $\Rightarrow$ ***In-place algorithm***

Fast Computation of the DFT: The FFT – Part 16

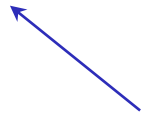
Radix-2-decimation-in-time FFT algorithms – Part 14

Bit-reversal

- The $v(n)$-values are ordered at the input of the decimation-in-time flow graph in permuted order.
- Example for $M = 8$, where the indices are written in binary notation:

# flow graph input	000	001	010	011
$v(n)$	$v(000)$	$v(100)$	$v(010)$	$v(110)$

# flow graph input	100	101	110	111
$v(n)$	$v(001)$	$v(101)$	$v(011)$	$v(111)$



Mirrored at the "center bit" in terms of binary counting!

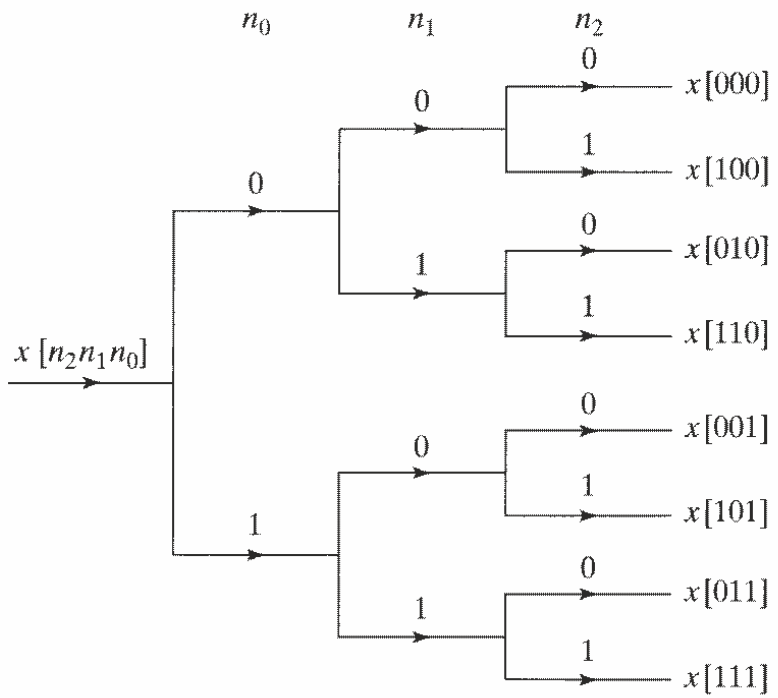
$\Rightarrow$ Input data is stored in ***bit-reversed*** order.

Fast Computation of the DFT: The FFT – Part 17

Radix-2-decimation-in-time FFT algorithms – Part 15

Bit-reversal

Bit-reversed order is due to the sorting in the even and odd indices in every stage, and thus is also necessary for in-place computation.



[$v \rightarrow x$, *Oppenheim, Schafer, 1999*]

Fast Computation of the DFT: The FFT – Part 18

Radix-2-decimation-in-time FFT algorithms – Part 16

Inverse FFT

The inverse DFT is defined as

$$v(n) = \frac{1}{M} \sum_{\mu=0}^{M-1} V_M(\mu)\, W_M^{-\mu n},$$

that is

$$v(-n) = \frac{1}{M} \sum_{\mu=0}^{M-1} V_M(\mu)\, W_M^{\mu n}, \quad \Longleftrightarrow$$

$$v(M-n) = \frac{1}{M} \mathrm{DFT}\{V_M(\mu)\}$$

$\Rightarrow$ With additional scaling and index permutations the IDFT can be calculated with the same FFT algorithm as the DFT.

Fast Computation of the DFT: The FFT – Part 19

FFT alternatives – Part 1

Alternative decimation-in-time (DIT) structures

Rearranging of the nodes in the signal flow graphs lead to FFTs with almost arbitrary permutation of the input and output sequence. Reasonable approaches are structures:

(a) without bit-reversal, or

(b) bit-reversal in the frequency domain.

The approach (a) has the disadvantage that it is a ***non-inplace*** algorithm, because the butterfly-structure does not continue past the first stage.

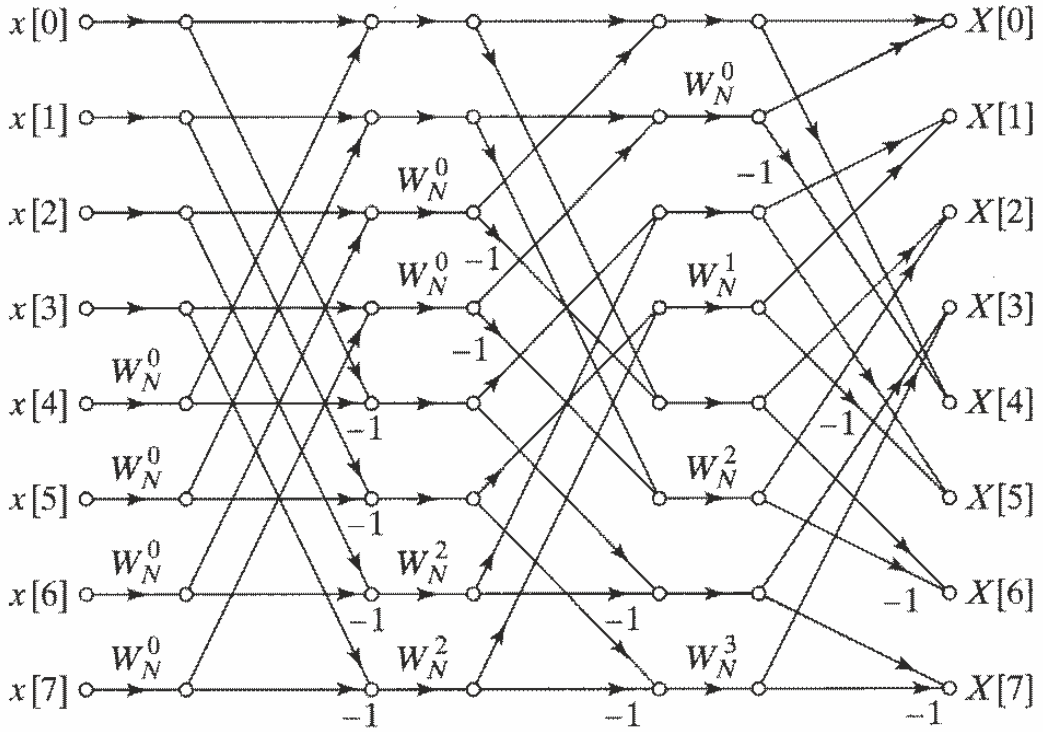
Next slides: the flow graphs for both approaches.

Fast Computation of the DFT: The FFT – Part 20

FFT alternatives – Part 2

Alternative decimation-in-time (DIT) structures (continued)

(a) Flow graph for $M = 8$ $(M \to N, v \to x, V \to X)$



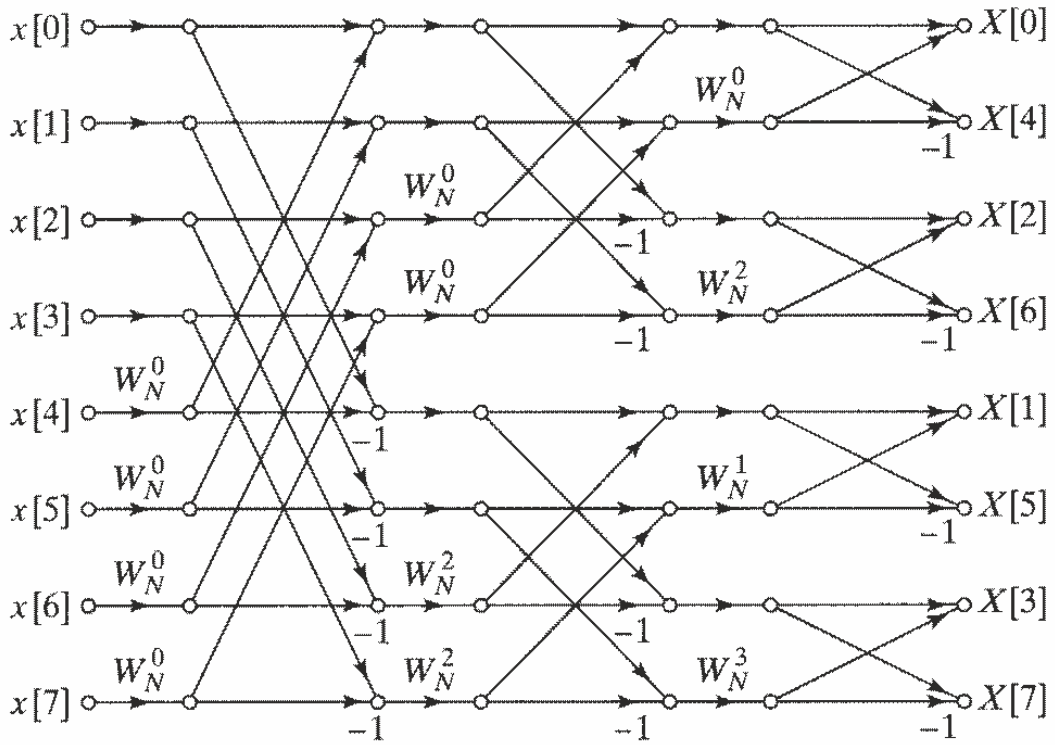
[Oppenheim, Schafer, 1999]

Fast Computation of the DFT: The FFT – Part 21

FFT alternatives – Part 3

Alternative decimation-in-time (DIT) structures (continued)

(b) Flow graph for $M = 8$ $(M \to N, v \to x, V \to X)$



[Oppenheim, Schafer, 1999]

Fast Computation of the DFT: The FFT – Part 22

FFT alternatives – Part 4

Decimation-in-frequency algorithms

Instead of applying the decomposition to time domain

$\rightarrow$ Starting the decomposition in the frequency domain

$\rightarrow$ The sequence of DFT coefficients $V_M(\mu)$ is decomposed into smaller sequences.

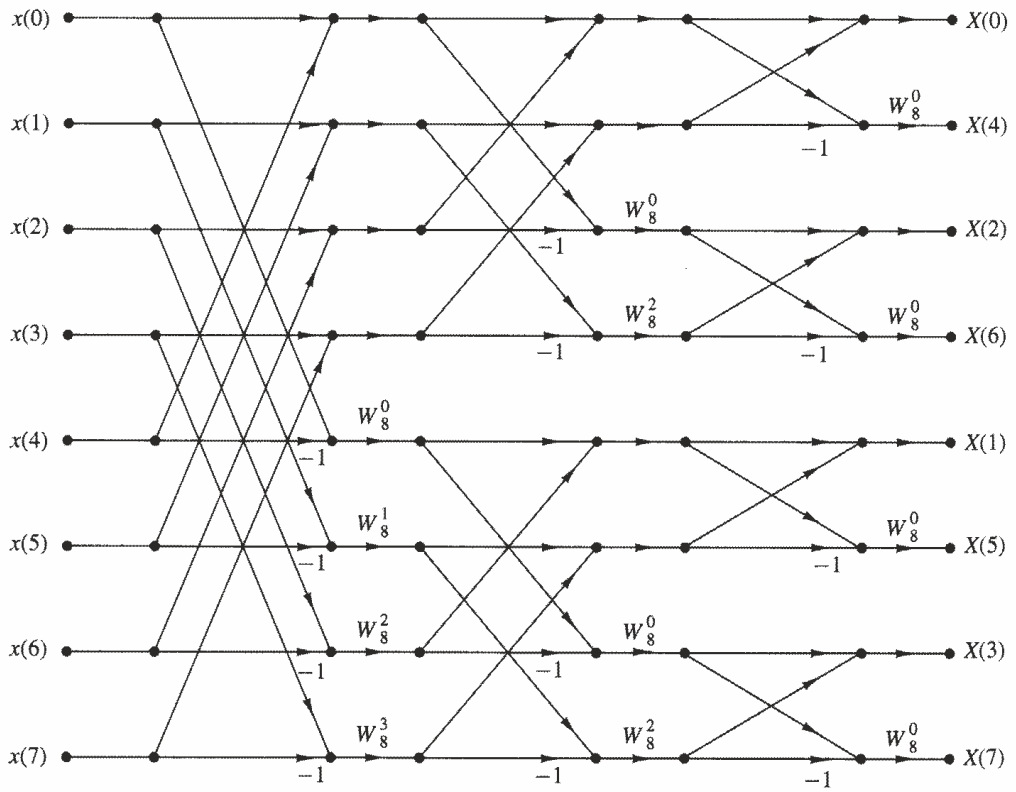
$\Rightarrow$ ***Decimation-in-frequency*** (DIF) FFT.

Fast Computation of the DFT: The FFT – Part 23

FFT alternatives – Part 5

Decimation-in-frequency algorithms (continued)

Signal flow graph for $M = 8$ $(v \to x, V \to X)$



[Proakis, Manolakis, 1996]

Fast Computation of the DFT: The FFT – Part 24

FFT alternatives – Part 6

Radix r and mixed-radix FFTs

When we generally use

$$M = r^m \quad \text{for } r \geq 2, \quad r, m \in \mathbb{N}$$

we obtain DIF or DIT decompositions with a radix r.

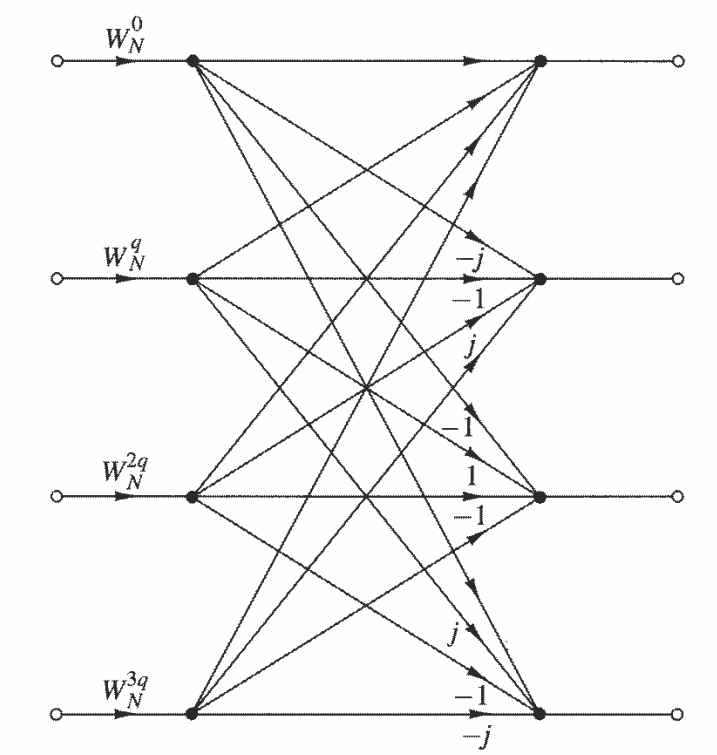
Besides $r = 2,\ r = 3$ and $r = 4$ are commonly used.

Fast Computation of the DFT: The FFT – Part 25

FFT alternatives – Part 7

Radix r and mixed-radix FFTs (continued)

Radix-4 butterfly $(q = 0, \ldots, M/4 - 1;\ M \to N)$:



[Proakis, Manolakis, 1999]

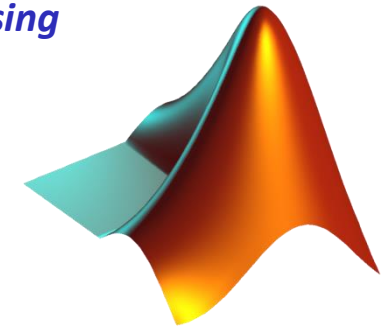
Fast Computation of the DFT: The FFT – Part 26

Convolution of a finite and an infinite sequence:

We will compare now the direct convolution in the time domain with it‘s DFT/FFT counterpart. For the DFT/FFT realization we will use the overlap-add technique.

For that purpose we will modify a music signal by means of amplifying the low and the very high frequencies of a music recording.

Realization using Matlab!



Fast Computation of the DFT: The FFT – Part 27

DFT and FFT:

Partner work – Please think about the following questions and try to find answers
(first group discussions, afterwards broad discussion in the whole group).

- What does “in-place” means and why is this property important for efficient algorithm?

  ……

  ……

- Which operations should be counted besides multiplications and additions when analyzing the efficiency of an algorithm?

  ……

  ……

- How would you realize an FFT of order 180?

  ……

  ……

Transformation of Real-Valued Sequences – Part 1

If $v(n) \in \mathbb{R} \Rightarrow$ FFT calculation for $v(n) = v_{\mathrm{re}}(n) + j\, v_{\mathrm{im}}(n) \in \mathbb{C}$ with $v_{\mathrm{im}}(n) = 0$.

$\Rightarrow$ inefficient due to arithmetic calculation with zero values

In the following: ***Methods for efficient usage of a complex FFT for real-valued data***.

DFT of two real sequences – Part 1

Given: $v_1(n),\ v_2(n) \in \mathbb{R}$ with $n = 0, \ldots, M-1$

Question: How can we efficiently obtain $V_{1,M}(\mu) = \mathrm{DFT}\{v_1(n)\}$ and $V_{2,M}(\mu) = \mathrm{DFT}\{v_2(n)\}$ by using the complex DFT?

Define

$$v(n) = v_1(n) + jv_2(n)$$

leading to the DFT

$$V_M(\mu) = \mathrm{DFT}\{v(n)\} = \underbrace{V_{1,M}(\mu)}_{\in \mathbb{C}} + j\ \underbrace{V_{2,M}(\mu)}_{\in \mathbb{C}}.$$

DFT of two real sequences – Part 2

How to separate $V_M(\mu)$ into $V_{1,M}(\mu)$ and $V_{2,M}(\mu)$?

Symmetry relations of the DFT:

$$v(n) = \underbrace{v_{\mathrm{re,even}}(n) + v_{\mathrm{re,odd}}(n)}_{=v_1(n)} + j\,\Big[\underbrace{v_{\mathrm{im,even}}(n) + v_{\mathrm{im,odd}}(n)}_{=v_2(n)}\Big]$$

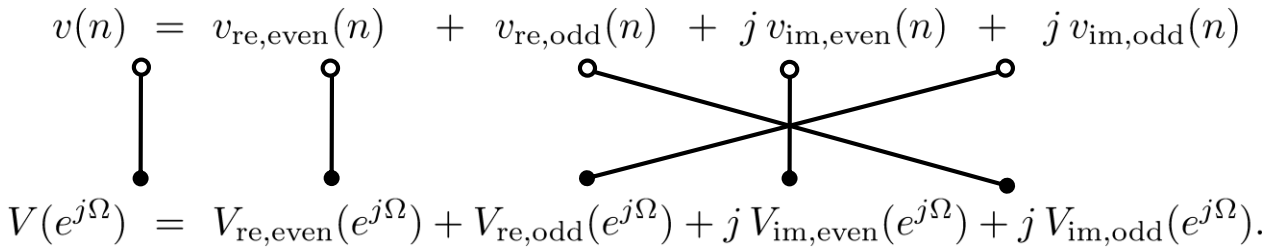
with the subscripts even denoting the even part and odd the odd part.

Corresponding DFTs:

$$\begin{aligned}
v_{\mathrm{re,even}}(n) &= \mathrm{IDFT}\{V_{M,\mathrm{re,even}}(\mu)\},\\
v_{\mathrm{re,odd}}(n) &= j\,\mathrm{IDFT}\{V_{M,\mathrm{im,odd}}(\mu)\},\\
j\,v_{\mathrm{im,even}}(n) &= j\,\mathrm{IDFT}\{V_{M,\mathrm{im,even}}(\mu)\},\\
j\,v_{\mathrm{im,odd}}(n) &= \mathrm{IDFT}\{V_{M,\mathrm{re,odd}}(\mu)\}.
\end{aligned}$$

DFT of two real sequences – Part 3

Repetition – symmetry scheme of Fourier transform:



… hope you remember ….

DFT of two real sequences – Part 4

Thus, we have

$$V_{1,M}(\mu) = \frac{1}{2}\Big[V_{M,\mathrm{re}}(\mu) + V_{M,\mathrm{re}}(M-\mu)\Big] + \frac{j}{2}\Big[V_{M,\mathrm{im}}(\mu) - V_{M,\mathrm{im}}(M-\mu)\Big],$$

with

$$V_{M,\mathrm{re,even}}(\mu) = \frac{1}{2}\Big[V_{M,\mathrm{re}}(\mu) + V_{M,\mathrm{re}}(M-\mu)\Big],$$

$$V_{M,\mathrm{im,odd}}(\mu) = \frac{1}{2}\Big[V_{M,\mathrm{im}}(\mu) - V_{M,\mathrm{im}}(M-\mu)\Big].$$

Likewise, we have for $V_{2,M}(\mu)$ the relation

$$V_{2,M}(\mu) = \frac{1}{2}\Big[V_{M,\mathrm{im}}(\mu) + V_{M,\mathrm{im}}(M-\mu)\Big] + \frac{j}{2}\Big[V_{M,\mathrm{re}}(\mu) - V_{M,\mathrm{re}}(M-\mu)\Big],$$

with

$$V_{M,\mathrm{im,even}}(\mu) = \frac{1}{2}\Big[V_{M,\mathrm{im}}(\mu) + V_{M,\mathrm{im}}(M-\mu)\Big],$$

$$V_{M,\mathrm{re,odd}}(\mu) = \frac{1}{2}\Big[V_{M,\mathrm{re}}(\mu) - V_{M,\mathrm{re}}(M-\mu)\Big].$$

DFT of two real sequences – Part 5

Rearranging both relations finally yields

$$V_{1,M}(\mu) = \frac{1}{2}\Big[V_M(\mu) + V_M^*(M-\mu)\Big],$$
$$V_{2,M}(\mu) = -\frac{j}{2}\Big[V_M(\mu) - V_M^*(M-\mu)\Big].$$

Due to the hermitean symmetry for real-valued sequences

$$V_{i,M}(\mu) = V_{i,M}^*(M-\mu)$$

only the values for $\mu \in \{0, ..., M/2\}$ have to be calculated.

The values for $\mu \in \{M/2+1, ..., M-1\}$ can be derived from the values for $\mu \in \{0, ..., M/2\}$.

Application:

Fast convolution of two real-valued sequences with the DFT/FFT!

DFT of a 2M-point real sequence – Part 1

Given: $v(n) \in \mathbb{R},\ n = 0, ..., 2M-1$

Wanted:

$$V_{2M}(\mu) = \mathrm{DFT}\{v(n)\} = \sum_{n=0}^{2M-1} v(n)\, W_{2M}^{\mu n}$$

with $\mu = 0, ..., 2M-1$.

Hermitian symmetry since $v(n) \in \mathbb{R}$ for all n:

$$V_{2M}(2M-\mu) = V_{2M}^{*}(\mu), \quad \mu = 0, \ldots, M$$

Define

$$\begin{aligned} \tilde{v}(n) &= v(2n) + jv(2n+1), \quad n = 0, ..., M-1, \\ &= v_1(n) + jv_2(n), \end{aligned}$$

where the even and odd samples of $v(n)$ are written alternatively into the real and imaginary part of $\tilde{v}(n)$.

DFT of a 2M-point real sequence – Part 2

We have a complex sequence $\tilde{v}(n)$ consisting of two real-valued sequences $v_1(n)$ and $v_2(n)$ of length M with the DFT

$$\widetilde{V}_M(\mu') \;=\; V_{1,M}(\mu') + j\, V_{2,M}(\mu'), \quad \mu' \;=\; 0,\, ...,\, M-1.$$

$V_{1,M}(\mu')$ and $V_{2,M}(\mu')$ can easily be obtained as

$$\begin{aligned} V_{1,M}(\mu') &= \frac{1}{2}\left[\widetilde{V}_M(\mu') + \widetilde{V}_M^*(M-\mu')\right], \\ V_{2,M}(\mu') &= -\frac{j}{2}\left[\widetilde{V}_M(\mu') - \widetilde{V}_M^*(M-\mu')\right] \end{aligned}$$

for $\mu' \;=\; 0,\, ...,\, M-1.$

In order to calculate $V_{2M}(\mu)$ from $V_{1,M}(\mu')$ and $V_{2,M}(\mu')$ we rearrange the expression for $\mathrm{DFT}\{v(n)\}$.

DFT of a 2M-point real sequence – Part 3

The rearranging leads to

$$\begin{aligned} V_{2M}(\mu) &= \sum_{n=0}^{M-1} \underbrace{v(2n)}_{=v_1(n)} W_{2M}^{2\mu n} + \sum_{n=0}^{M-1} \underbrace{v(2n+1)}_{=v_2(n)} W_{2M}^{(2n+1)\mu} \\ &= \sum_{n=0}^{M-1} v_1(n)\, W_M^{\mu n} + W_{2M}^{\mu} \sum_{n=0}^{M-1} v_2(n)\, W_M^{n\mu}. \end{aligned}$$

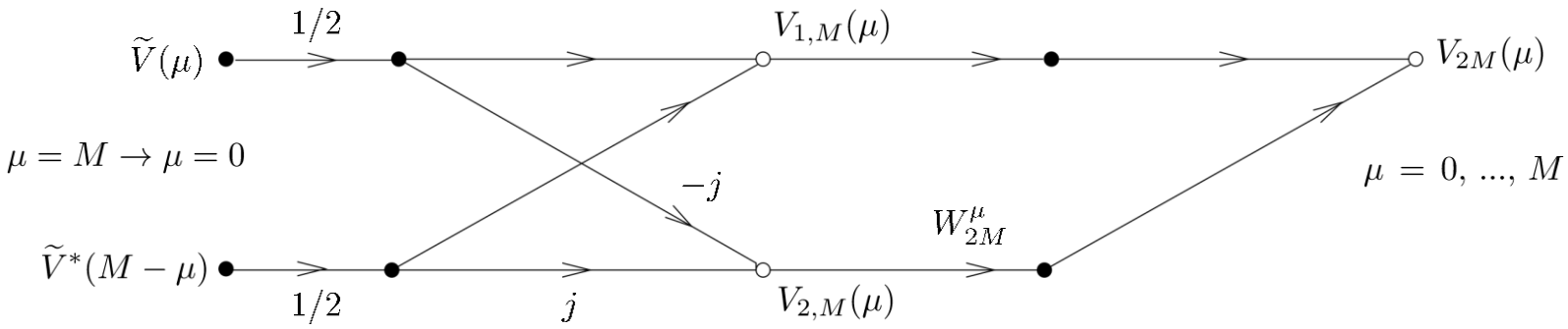
Finally we have:

$$V_{2M}(\mu) = V_{1,M}(\mu) + W_{2M}^{\mu}\, V_{2,M}(\mu), \quad \mu = 0, \ldots, 2M-1.$$

Due to the Hermitian symmetry $V_{2M}(\mu) = V_{2M}^*(2M-\mu)$, μ only needs to be evaluated from 0 to M with $V_{i,M}(M) = V_{i,M}(0)$.

DFT of a 2M-point real sequence – Part 4

Signal flow graph:



$\Rightarrow$ Computational savings by a factor of two compared to the complex-valued case since for real-valued input sequences only an M-point DFT is needed.

Transformation of Real-Valued Sequences – Part 10

DFT of a 2M-point real sequence – Part 5

Partner work – Please think about the following questions and try to find answers
(first group discussions, afterwards broad discussion in the whole group).

- How many memory elements do you need to store the result of a DFT of order M if the input sequence was real-valued?

  ……

  ……

- Why is it useful to operate with complex-valued sequences?

  ……

  ……

- (General question) Where can you use a DFT/FFT? Application examples?

  ……

  ……

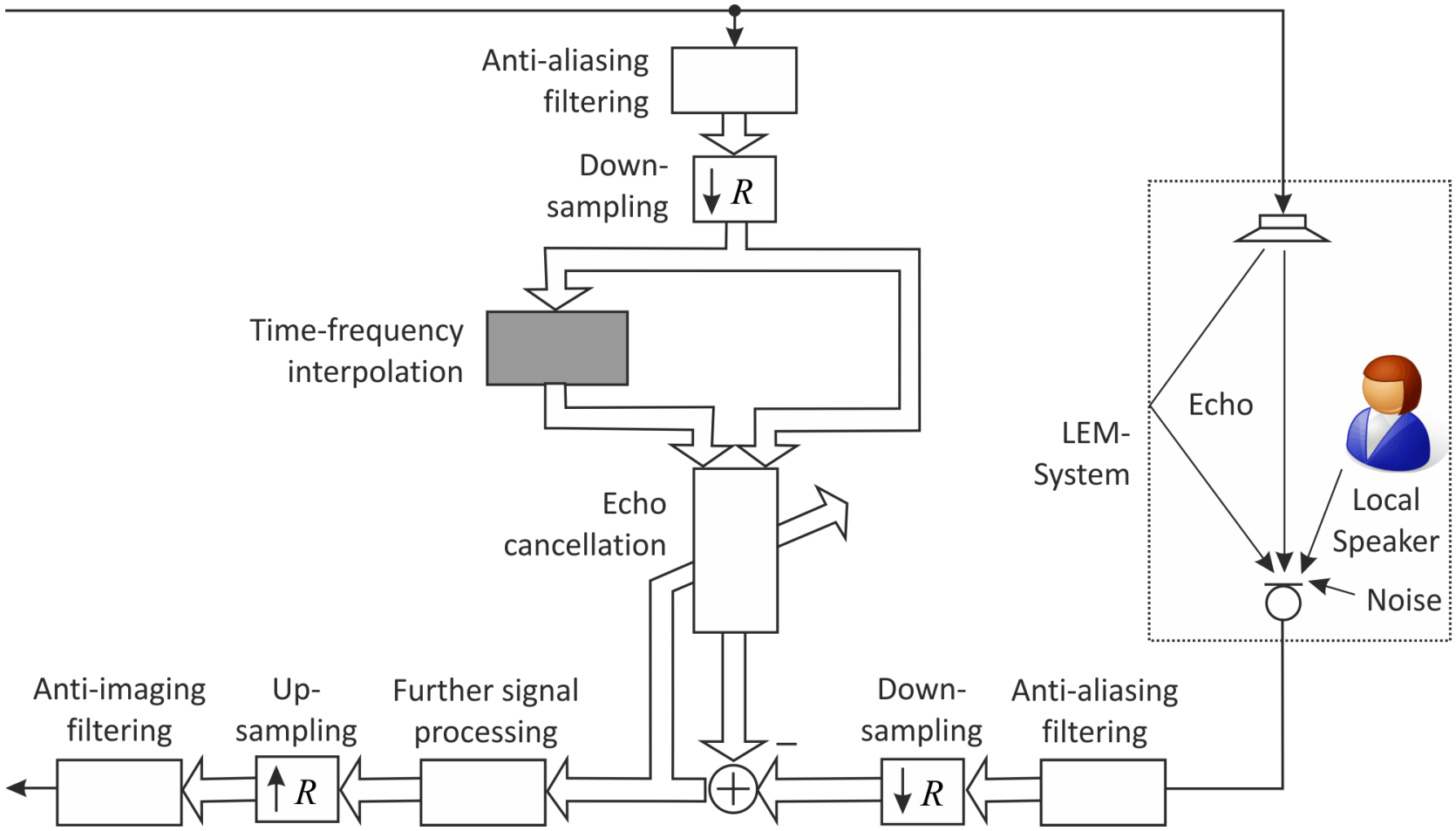
Summary

- Introduction
- Digital processing of continuous-time signals
- Efficient FIR structures
- ***DFT and FFT***
  - DFT and signal processing
  - Fast computation of the DFT: The FFT
  - Transformation of real-valued sequences
  - ***Spectral and temporal refinement***
- Digital filters
- Multi-rate digital signal processing

Spectral Refinement

Basic Idea – Part 1

The origin of it …

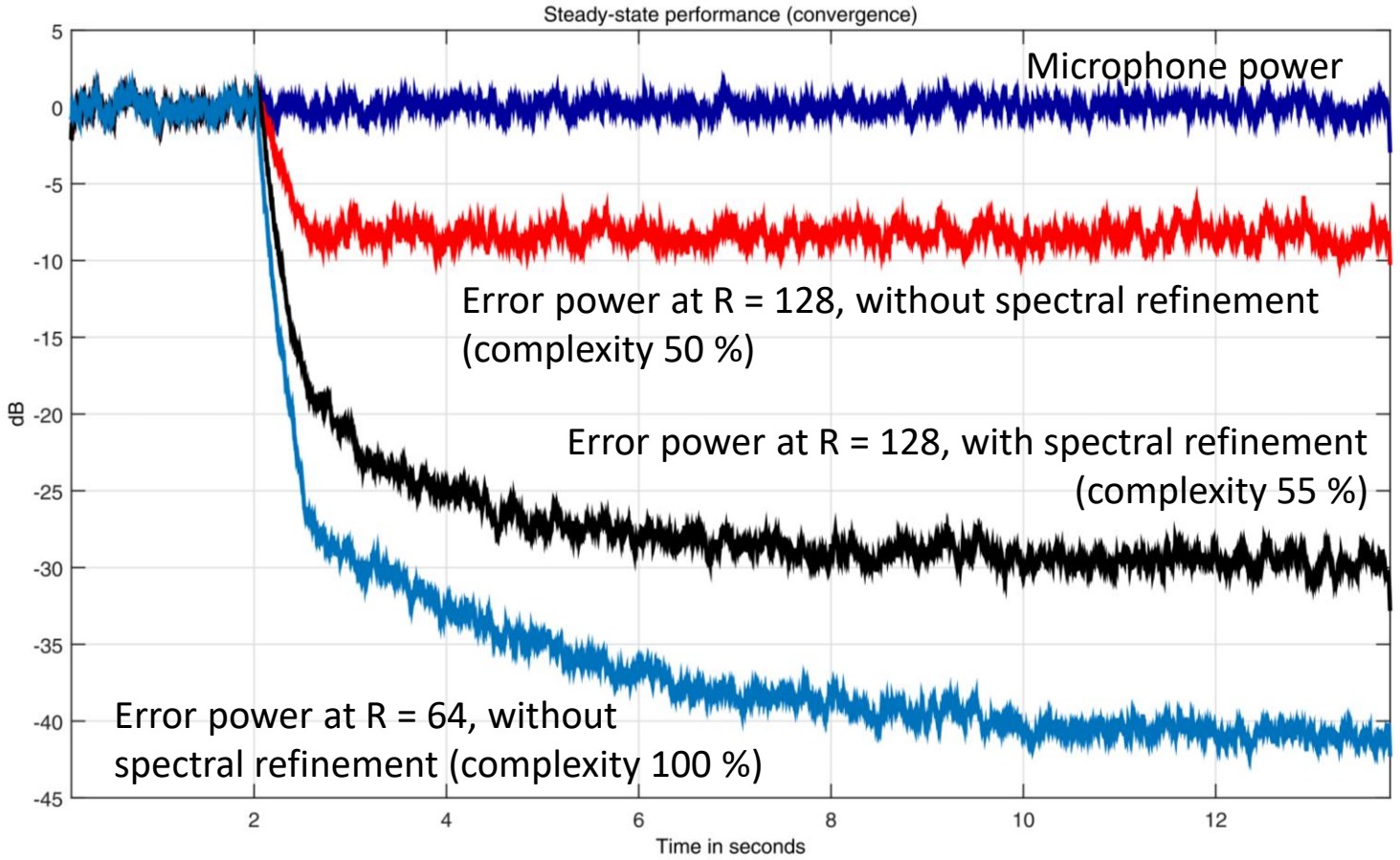


Spectral Refinement

Basic Idea – Part 2

Basis setup

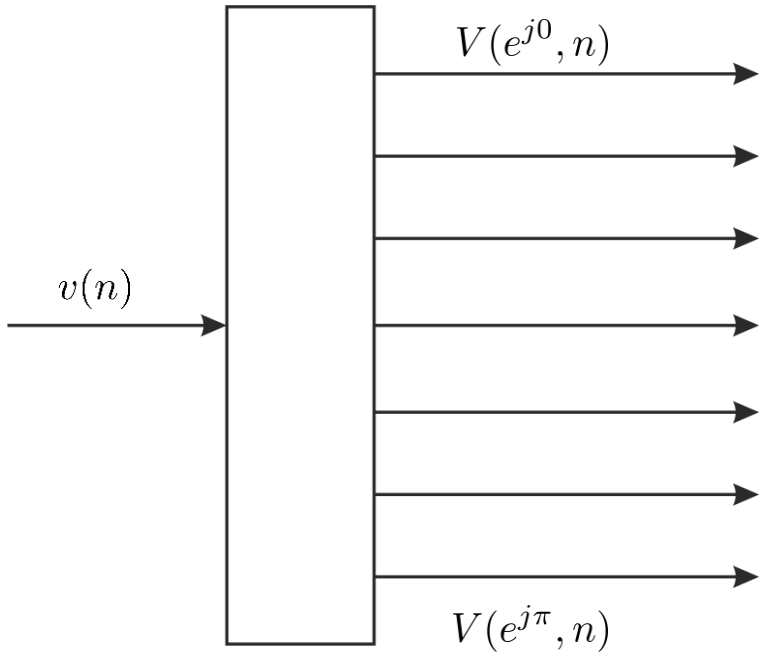
- Handsfree system with a DFT size of 256
- Hann window and with an overlap of 75 % (downsampling factor = 64)
- Processing power was about 30 % too high



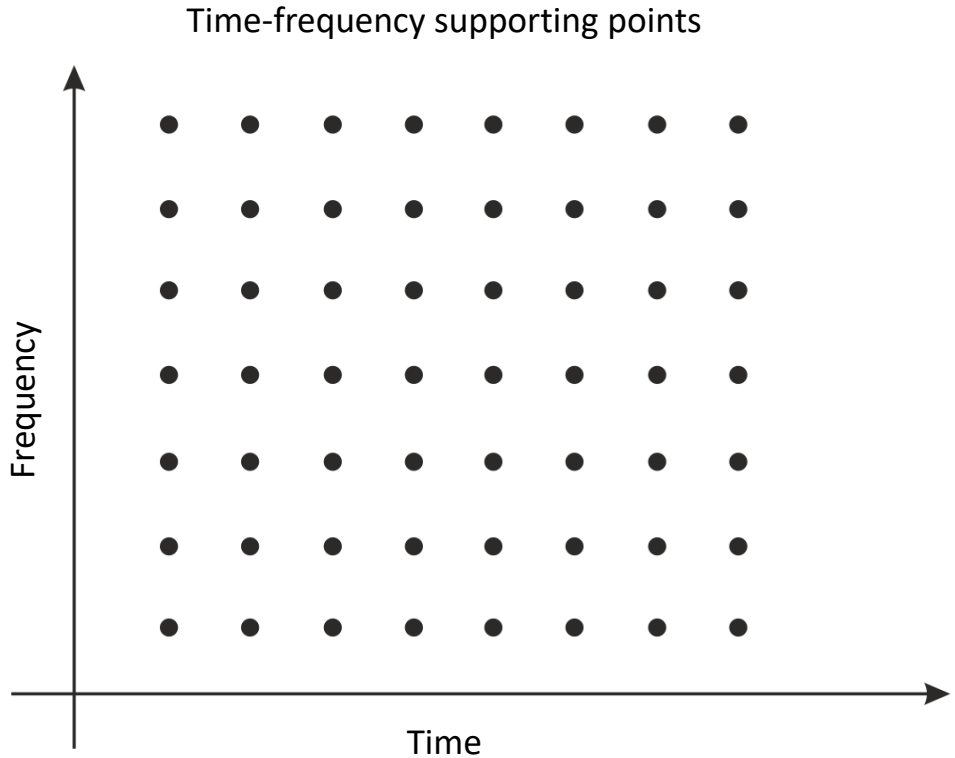
Spectral Refinement

Basic Idea – Part 3

A standard DFT …



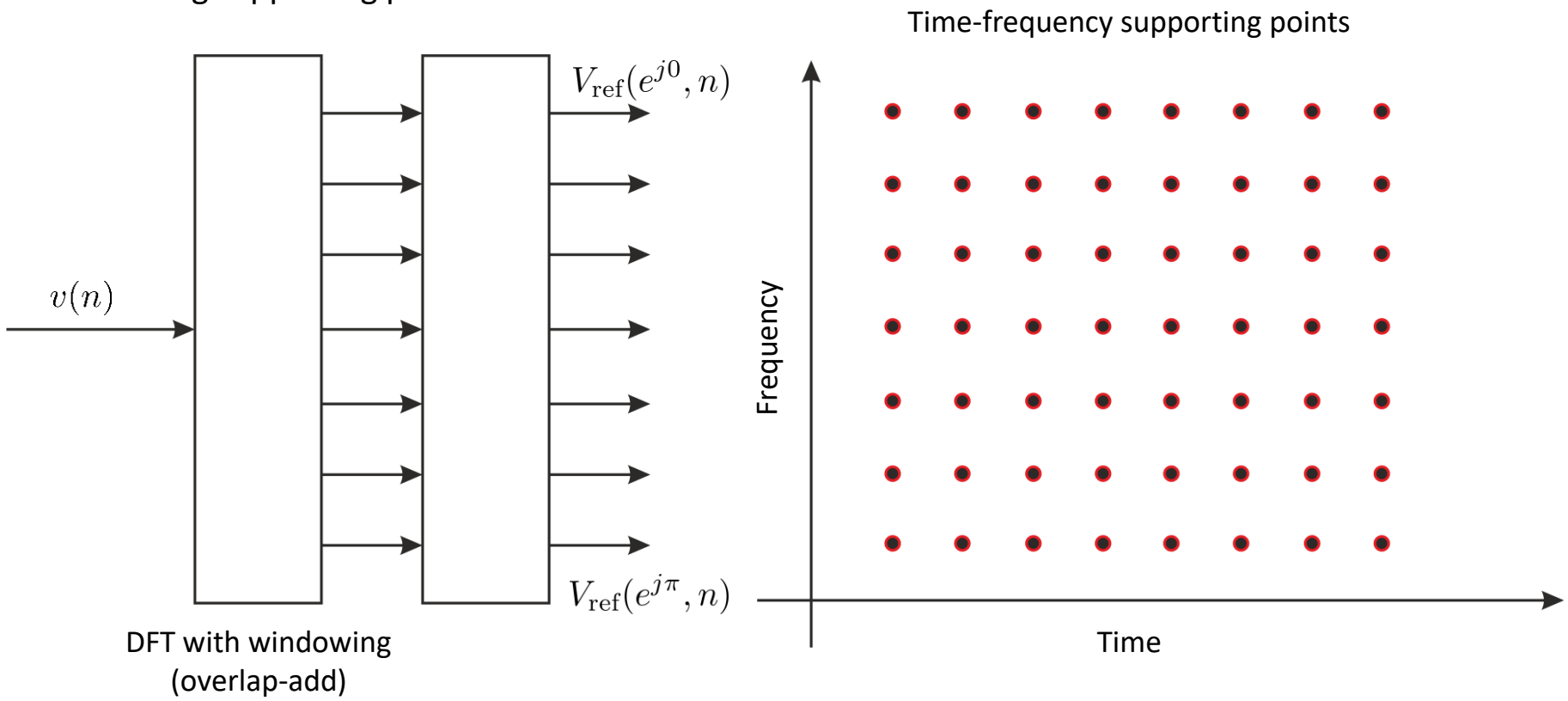
DFT with windowing
(overlap-add)



Spectral Refinement

Basic Idea – Part 4

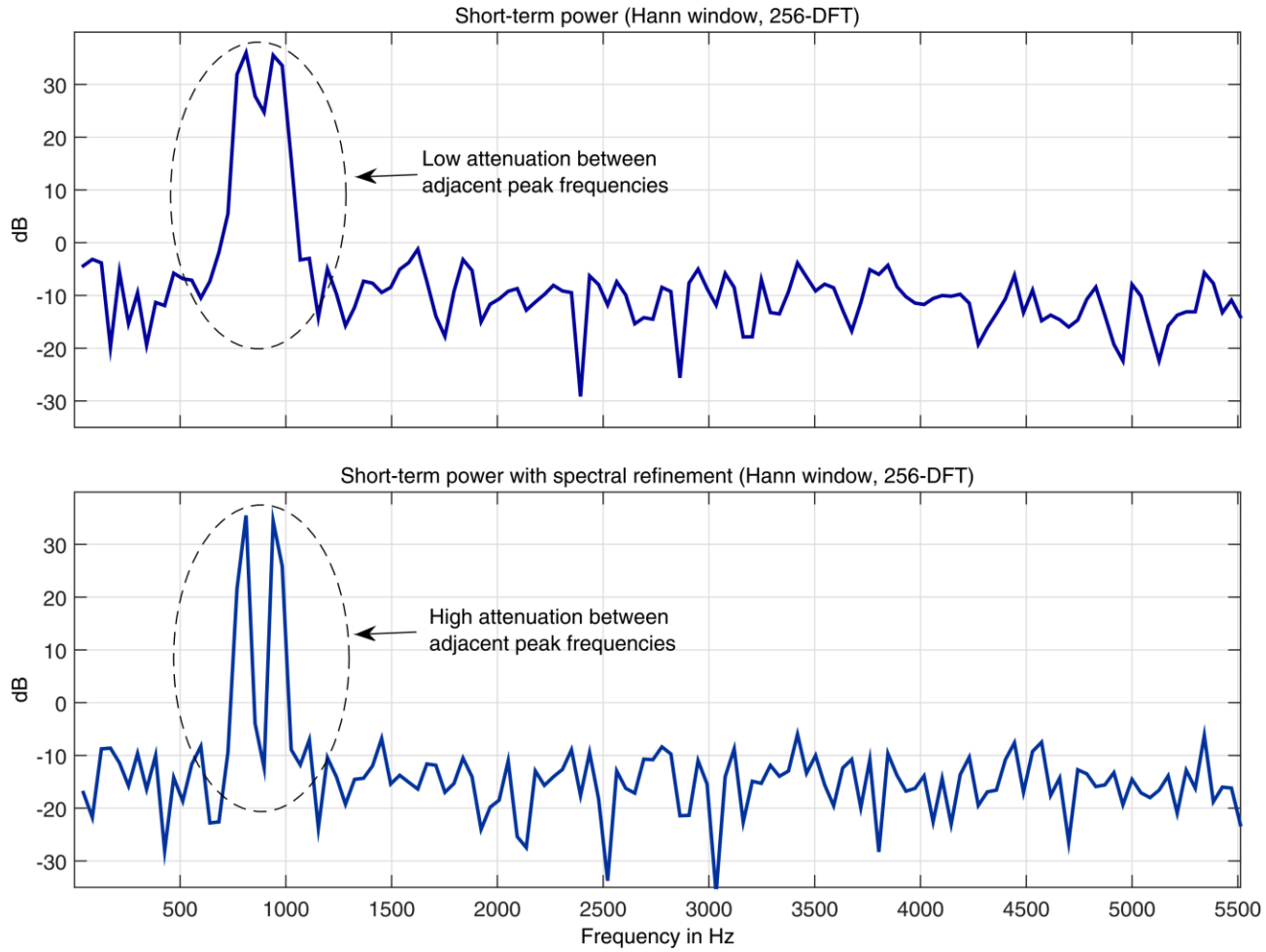
Refinement of existing supporting points



Spectral Refinement

Basic Idea – Part 5

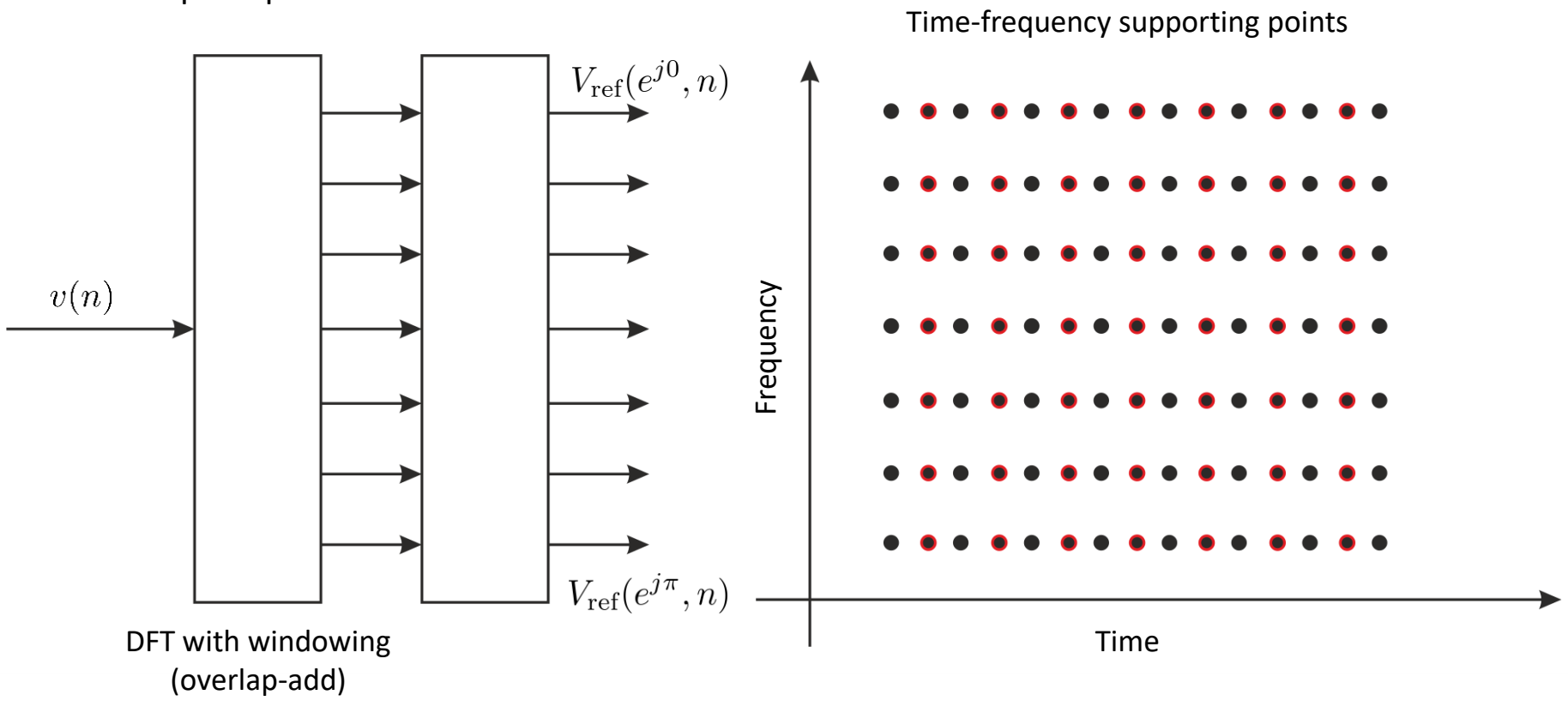
Example



Spectral Refinement

Basic Idea – Part 6

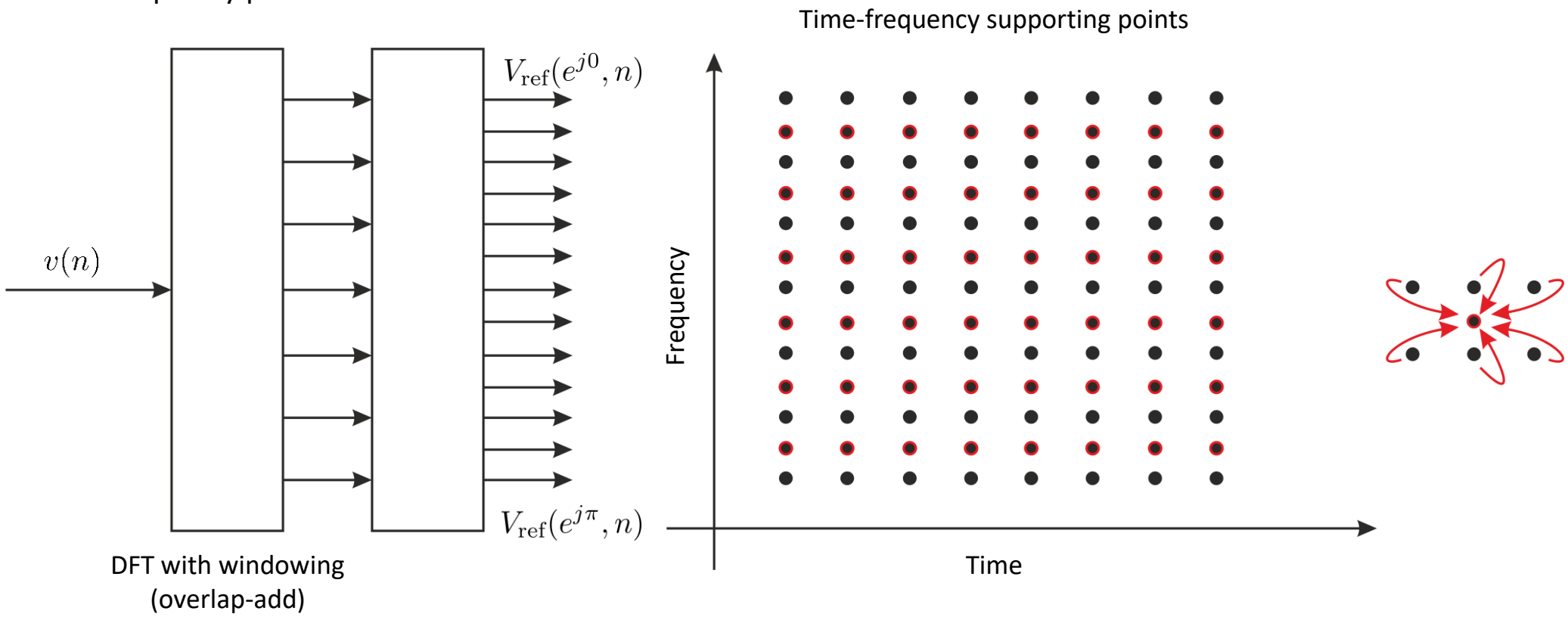
Creation of new temporal points



Spectral Refinement

Basic Idea – Part 1

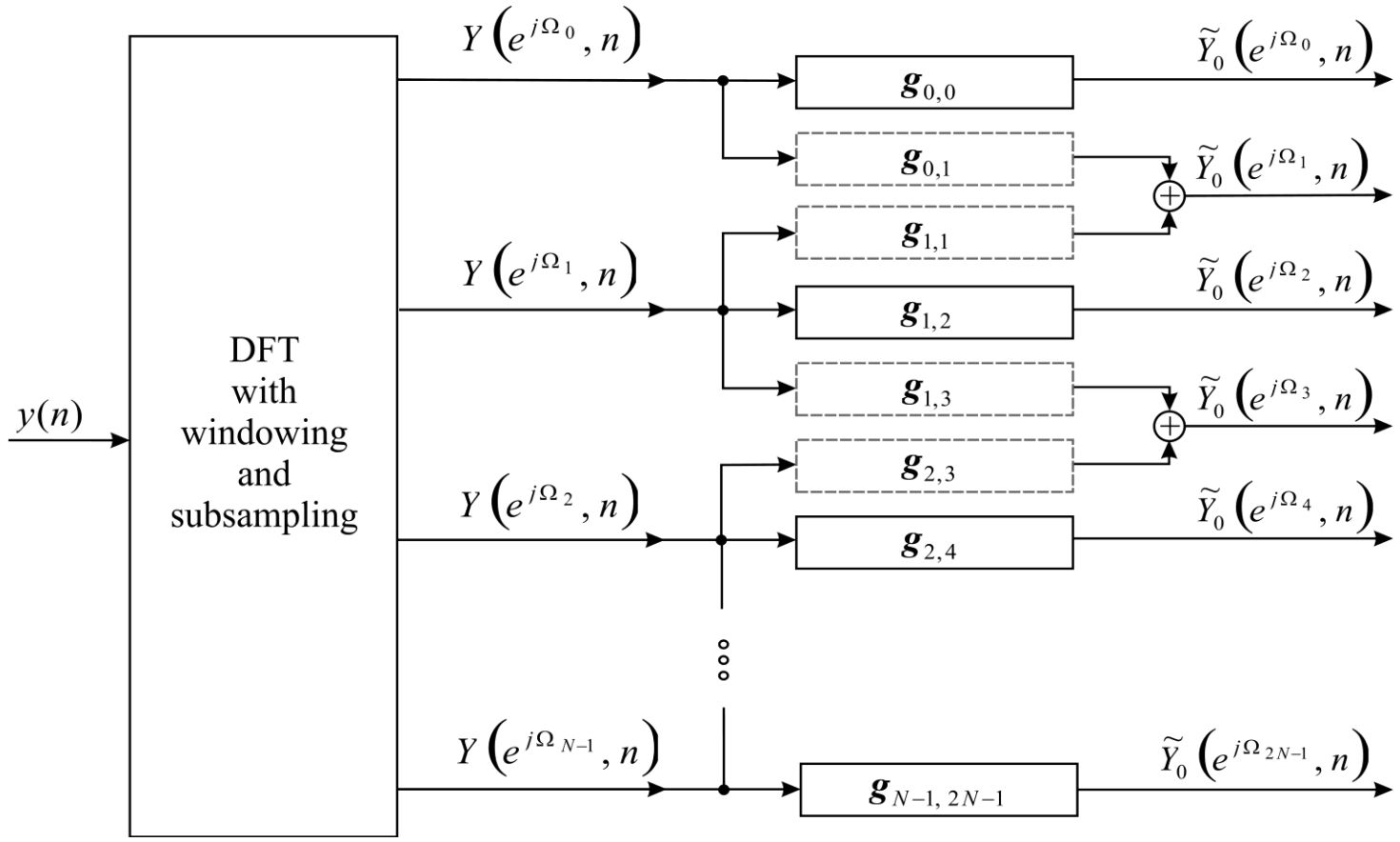
Creation of new frequency points



Spectral Refinement

Basic Idea – Part 7

Realization by FIR filters



Spectral Refinement

Different way to write a windowed DFT – Part 1

Input vector:

$$\boldsymbol{v}(nR) \;=\; \Big[v(nR),\, v(nR-1),\, ...,\, v(nR-M+1)\Big]^{\mathrm{T}}.$$

DFT (or FFT) of windowed input vector:

$$V(e^{j\Omega_\mu}, n) \;=\; \sum_{k=0}^{M-1} v(nR-k)\, h_k\, e^{-j\Omega_\mu k}.$$

Equidistantly selected frequency supporting points:

$$\Omega_\mu \;=\; \frac{2\pi}{M}\,\mu \quad \text{with} \quad \mu \;\in\; \{0,\, 1,\, ...,\, M-1\}.$$

Spectral Refinement

Different way to write a windowed DFT – Part 2

A “windowed DFT” in matrix-vector notation:

$$\left[V(e^{j\Omega_0}, n),\, V(e^{j\Omega_1}, n),\, ...,\, V(e^{j\Omega_{M-1}}, n)\right]^{\mathrm{T}} \;=\; \boldsymbol{V}(n) \;=\; \boldsymbol{D}_M\, \boldsymbol{H}\, \boldsymbol{v}(nR).$$

DFT matrix:

$$\boldsymbol{D}_M \;=\; \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & e^{-j\frac{2\pi}{M}1} & \cdots & e^{-j\frac{2\pi}{M}1(M-2)} & e^{-j\frac{2\pi}{M}1(M-1)} \\ 1 & e^{-j\frac{2\pi}{M}2} & \cdots & e^{-j\frac{2\pi}{M}2(M-2)} & e^{-j\frac{2\pi}{M}2(M-1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & e^{-j\frac{2\pi}{M}(M-1)} & \cdots & e^{-j\frac{2\pi}{M}(M-1)(M-2)} & e^{-j\frac{2\pi}{M}(M-1)(M-1)} \end{bmatrix}.$$

Window matrix:

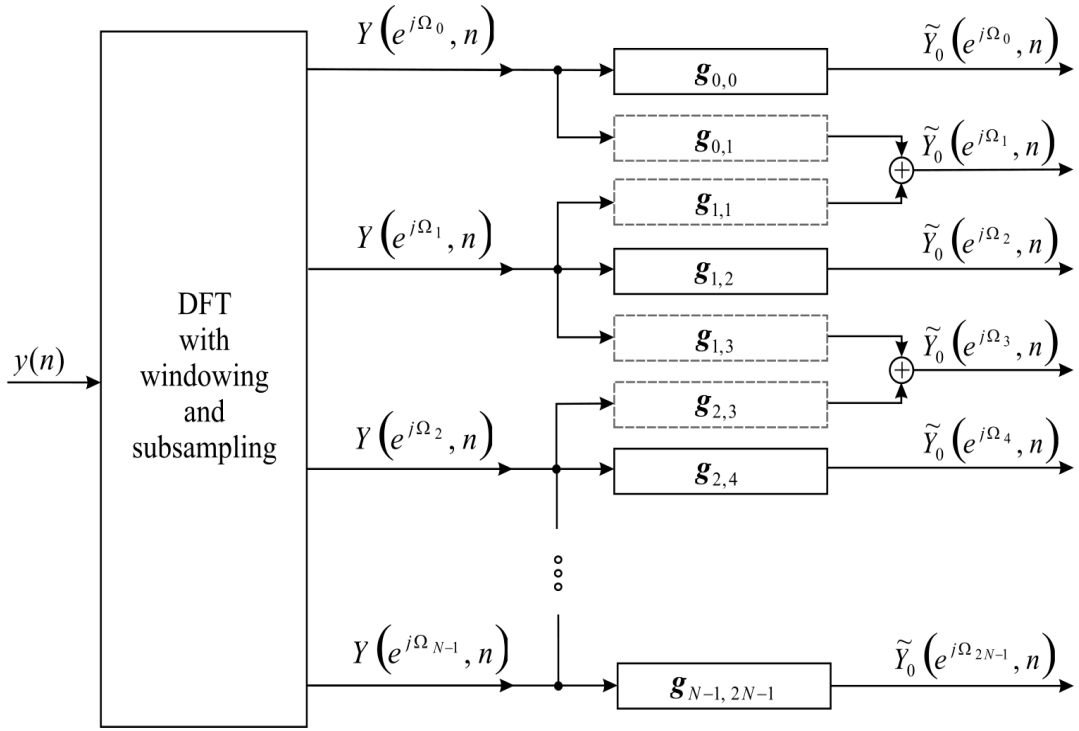
$$\boldsymbol{H} \;=\; \begin{bmatrix} h_0 & 0 & \cdots & 0 & 0 \\ 0 & h_1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & h_{M-1} \end{bmatrix}.$$

Spectral Refinement

Basic Idea of spectral refinement – Part 1

Basic idea of spectral refinement:

$$\begin{bmatrix} V_{\mathrm{ref}}(e^{j\Omega_0}, n) \\ V_{\mathrm{ref}}(e^{j\Omega_1}, n) \\ \vdots \\ V_{\mathrm{ref}}(e^{j\Omega_{M-1}}, n) \end{bmatrix} = \boldsymbol{V}_{\mathrm{ref}}(n) = \boldsymbol{S}_{\mathrm{ref}} \begin{bmatrix} \boldsymbol{V}(n) \\ \boldsymbol{V}(n-1) \\ \vdots \\ \boldsymbol{V}(n-K+1) \end{bmatrix}.$$



Spectral Refinement

Basic Idea of spectral refinement – Part 2

Basic idea of spectral refinement:

$$\begin{bmatrix} V_{\mathrm{ref}}(e^{j\Omega_0}, n) \\ V_{\mathrm{ref}}(e^{j\Omega_1}, n) \\ \vdots \\ V_{\mathrm{ref}}(e^{j\Omega_{M-1}}, n) \\ V_{\mathrm{int}}(e^{j\Omega_0}, n) \\ V_{\mathrm{int}}(e^{j\Omega_1}, n) \\ \vdots \\ V_{\mathrm{int}}(e^{j\Omega_{M-1}}, n) \end{bmatrix} = \begin{bmatrix} \boldsymbol{V}_{\mathrm{ref}}(n) \\ \boldsymbol{V}_{\mathrm{int}}(n) \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_{\mathrm{ref}} \\ \boldsymbol{S}_{\mathrm{int}} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}(n) \\ \boldsymbol{V}(n-1) \\ \vdots \\ \boldsymbol{V}(n-K+1) \end{bmatrix}.$$

Spectral Refinement

Basic Idea of spectral refinement – Part 3

Reordering (according to frequency) leads to :

$$\begin{bmatrix} V_{\mathrm{ref}}(e^{j\Omega_0},n) \\ V_{\mathrm{int}}(e^{j\Omega_0},n) \\ V_{\mathrm{ref}}(e^{j\Omega_1},n) \\ V_{\mathrm{int}}(e^{j\Omega_1},n) \\ \vdots \\ V_{\mathrm{ref}}(e^{j\Omega_{M-1}},n) \\ V_{\mathrm{int}}(e^{j\Omega_{M-1}},n) \end{bmatrix} = \begin{bmatrix} \widetilde{V}_{\mathrm{ref}}(e^{j\Omega_0},n) \\ \widetilde{V}_{\mathrm{int}}(e^{j\Omega_1},n) \\ \widetilde{V}_{\mathrm{ref}}(e^{j\Omega_2},n) \\ \widetilde{V}_{\mathrm{int}}(e^{j\Omega_3},n) \\ \vdots \\ \widetilde{V}_{\mathrm{ref}}(e^{j\Omega_{\widetilde{M}-2}},n) \\ \widetilde{V}_{\mathrm{int}}(e^{j\Omega_{\widetilde{M}-1}},n) \end{bmatrix} = \widetilde{\boldsymbol{V}}_{\mathrm{ref}}(n) = \underbrace{\boldsymbol{P}\begin{bmatrix} \boldsymbol{S}_{\mathrm{ref}} \\ \boldsymbol{S}_{\mathrm{int}} \end{bmatrix}}_{\widetilde{\boldsymbol{S}}} \begin{bmatrix} \boldsymbol{V}(n) \\ \boldsymbol{V}(n-1) \\ \vdots \\ \boldsymbol{V}(n-K+1) \end{bmatrix}.$$

Spectral Refinement

Basic Idea of spectral refinement – Part 4

The supervector of the DFT spectra can be reformulated:

$$\begin{bmatrix} \boldsymbol{V}(n) \\ \boldsymbol{V}(n-1) \\ \vdots \\ \boldsymbol{V}(n-K+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}_M \, \boldsymbol{H} \, \boldsymbol{v}(nR) \\ \boldsymbol{D}_M \, \boldsymbol{H} \, \boldsymbol{v}\big((n-1)R\big) \\ \vdots \\ \boldsymbol{D}_M \, \boldsymbol{H} \, \boldsymbol{v}\big((n-K+1)R\big) \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}_M & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{0}^{M\times M} \\ \boldsymbol{0}^{M\times M} & \boldsymbol{D}_M & \cdots & \boldsymbol{0}^{M\times M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times M} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{D}_M \end{bmatrix} \begin{bmatrix} \boldsymbol{H} \, \boldsymbol{v}(nR) \\ \boldsymbol{H} \, \boldsymbol{v}\big((n-1)R\big) \\ \vdots \\ \boldsymbol{H} \, \boldsymbol{v}\big((n-K+1)R\big) \end{bmatrix}.$$

Furthermore, we insert a longer time-domain vector:

$$\tilde{\boldsymbol{v}}(nR) \;=\; \Big[v(nR),\, v(nR-1),\, ...,\, v(nR-M+1),\, ...,\, v(nR-M-(K-1)R+1\Big]^{\mathrm{T}},$$

Spectral Refinement

Basic Idea of spectral refinement – Part 5

That allows as to rewrite the equation system as follows:

$$\begin{bmatrix} \boldsymbol{V}(n) \\ \boldsymbol{V}(n-1) \\ \vdots \\ \boldsymbol{V}(n-K+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}_M & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{0}^{M\times M} \\ \boldsymbol{0}^{M\times M} & \boldsymbol{D}_M & \cdots & \boldsymbol{0}^{M\times M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times M} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{D}_M \end{bmatrix} \begin{bmatrix} \boldsymbol{H}\,\boldsymbol{v}(nR) \\ \boldsymbol{H}\,\boldsymbol{v}\big((n-1)R\big) \\ \vdots \\ \boldsymbol{H}\,\boldsymbol{v}\big((n-K+1)R\big) \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{D}_M & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{0}^{M\times M} \\ \boldsymbol{0}^{M\times M} & \boldsymbol{D}_M & \cdots & \boldsymbol{0}^{M\times M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times M} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{D}_M \end{bmatrix} \begin{bmatrix} \boldsymbol{H} & \boldsymbol{0}^{M\times R} & \cdots & \boldsymbol{0}^{M\times R} \\ \boldsymbol{0}^{M\times R} & \boldsymbol{H} & \cdots & \boldsymbol{0}^{M\times R} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times R} & \boldsymbol{0}^{M\times R} & \cdots & \boldsymbol{H} \end{bmatrix} \tilde{\boldsymbol{v}}(nR).$$

Spectral Refinement

Basic Idea of spectral refinement – Part 6

Combining everything lead to (for the refined and interpolated spectrum):

$$\widetilde{\boldsymbol{V}}_{\mathrm{ref}}(n) \;=\; \widetilde{\boldsymbol{S}}_{\mathrm{ref}} \begin{bmatrix} \boldsymbol{D}_M & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{0}^{M\times M} \\ \boldsymbol{0}^{M\times M} & \boldsymbol{D}_M & \cdots & \boldsymbol{0}^{M\times M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times M} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{D}_M \end{bmatrix} \begin{bmatrix} \boldsymbol{H} & \boldsymbol{0}^{M\times R} & \cdots & \boldsymbol{0}^{M\times R} \\ \boldsymbol{0}^{M\times R} & \boldsymbol{H} & \cdots & \boldsymbol{0}^{M\times R} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times R} & \boldsymbol{0}^{M\times R} & \cdots & \boldsymbol{H} \end{bmatrix} \tilde{\boldsymbol{v}}(nR).$$

Spectral Refinement

A spectrum with higher resolution – Part 1

Starting again with a longer time-domain vector:

$$\tilde{\boldsymbol{v}}(nR) \;=\; \Big[v(nR),\, v(nR-1),\, ...,\, v(nR-M+1),\, ...,\, v(nR-\widetilde{M}+1\Big]^{\mathrm{T}},$$

with $\widetilde{M} > M$ (e.g. $\widetilde{M} = 2M$) .

And furthermore assuming that

$$\widetilde{M} \;=\; M + (K-1)R \;=\; 2M.$$

A second (high resolution) spectrum can be obtained by

$$\widetilde{\boldsymbol{V}}(n) \;=\; \boldsymbol{D}_{\widetilde{M}}\, \widetilde{\boldsymbol{H}}\, \tilde{\boldsymbol{v}}(nR).$$

Spectral Refinement

A spectrum with higher resolution – Part 2

We will restrict the long window to be a sum of weighted and shifted short windows:

$$\boldsymbol{A}\left[\boldsymbol{h}^{\mathrm{T}},\, \boldsymbol{h}^{\mathrm{T}},\, ...,\, \boldsymbol{h}^{\mathrm{T}}\right] \;=\; \widetilde{\boldsymbol{h}}.$$

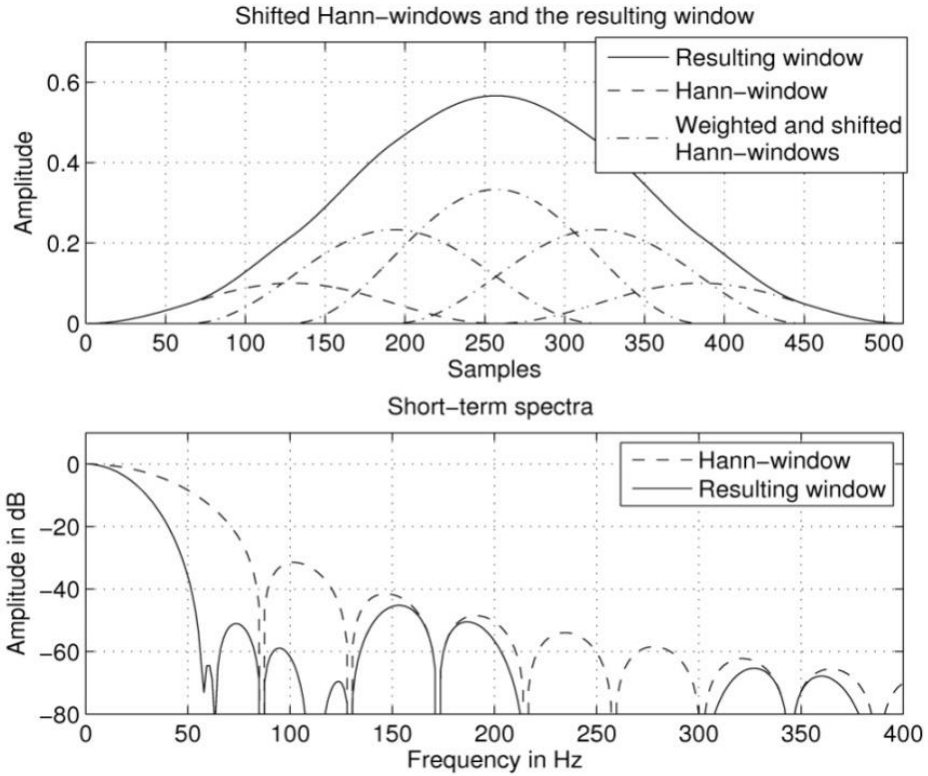
The weighting matrix is a sparse matrix of the form

$$\boldsymbol{A} \;=\; \left[\boldsymbol{A}_0,\, \boldsymbol{A}_1,\, ...,\, \boldsymbol{A}_{K-1}\right].$$

The first submatrix has the form

$$\boldsymbol{A}_0 \;=\; \begin{bmatrix} a_0\, \boldsymbol{I}^{M\times M} \\ \boldsymbol{0}^{(K-1)R\times M} \end{bmatrix}.$$

The remaining submatrices are equivalently defined.



Spectral Refinement

A spectrum with higher resolution – Part 3

Inserting these restrictions for the long window functions leads to:

$$\widetilde{\boldsymbol{V}}(n) = \boldsymbol{D}_{\widetilde{M}}\,\widetilde{\boldsymbol{H}}\,\tilde{\boldsymbol{v}}(nR).$$

$$= \boldsymbol{D}_{\widetilde{M}}\,\boldsymbol{A}\begin{bmatrix} \boldsymbol{H} & \boldsymbol{0}^{M\times R} & \cdots & \boldsymbol{0}^{M\times R} \\ \boldsymbol{0}^{M\times R} & \boldsymbol{H} & \cdots & \boldsymbol{0}^{M\times R} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times R} & \boldsymbol{0}^{M\times R} & \cdots & \boldsymbol{H} \end{bmatrix}\tilde{\boldsymbol{v}}(nR).$$

Comparing this with our refinement approach:

$$\widetilde{\boldsymbol{V}}_{\mathrm{ref}}(n) = \widetilde{\boldsymbol{S}}_{\mathrm{ref}}\begin{bmatrix} \boldsymbol{D}_M & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{0}^{M\times M} \\ \boldsymbol{0}^{M\times M} & \boldsymbol{D}_M & \cdots & \boldsymbol{0}^{M\times M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times M} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{D}_M \end{bmatrix}\begin{bmatrix} \boldsymbol{H} & \boldsymbol{0}^{M\times R} & \cdots & \boldsymbol{0}^{M\times R} \\ \boldsymbol{0}^{M\times R} & \boldsymbol{H} & \cdots & \boldsymbol{0}^{M\times R} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times R} & \boldsymbol{0}^{M\times R} & \cdots & \boldsymbol{H} \end{bmatrix}\tilde{\boldsymbol{v}}(nR).$$

Spectral Refinement

A spectrum with higher resolution – Part 4

Setting

$$\widetilde{\boldsymbol{V}}(n) \;=\; \widetilde{\boldsymbol{V}}_{\mathrm{ref}}(n)$$

leads to

$$\widetilde{\boldsymbol{S}}_{\mathrm{ref}} \begin{bmatrix} \boldsymbol{D}_M & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{0}^{M\times M} \\ \boldsymbol{0}^{M\times M} & \boldsymbol{D}_M & \cdots & \boldsymbol{0}^{M\times M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times M} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{D}_M \end{bmatrix} = \boldsymbol{D}_{\widetilde{M}}\,\boldsymbol{A}$$

$$\widetilde{\boldsymbol{S}}_{\mathrm{ref}} \;=\; \boldsymbol{D}_{\widetilde{M}}\,\boldsymbol{A} \begin{bmatrix} \boldsymbol{D}_M^{-1} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{0}^{M\times M} \\ \boldsymbol{0}^{M\times M} & \boldsymbol{D}_M^{-1} & \cdots & \boldsymbol{0}^{M\times M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times M} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{D}_M^{-1} \end{bmatrix}.$$

Spectral Refinement

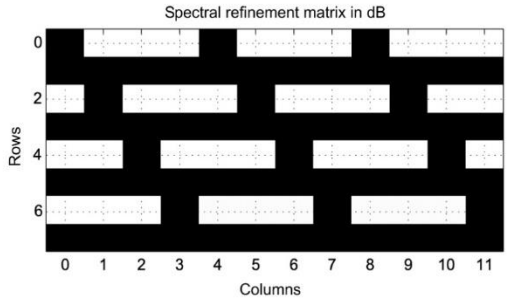
A spectrum with higher resolution – Part 4

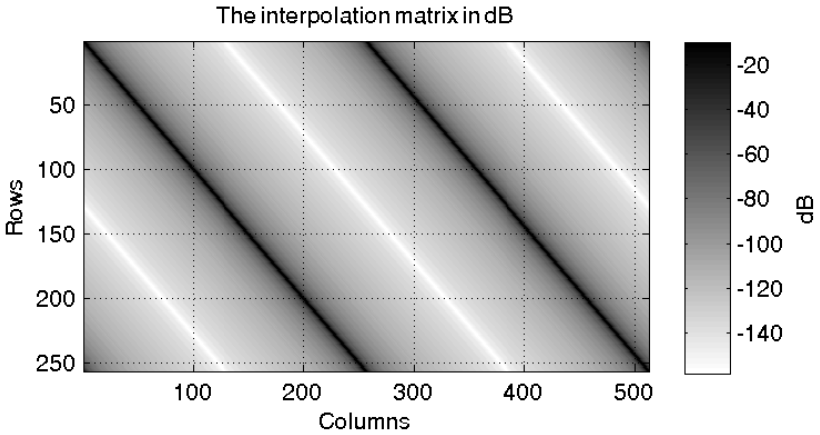
Setting

$$\widetilde{\boldsymbol{V}}(n) \;=\; \widetilde{\boldsymbol{V}}_{\mathrm{ref}}(n)$$

leads to

$$\widetilde{\boldsymbol{S}}_{\mathrm{ref}} \begin{bmatrix} \boldsymbol{D}_M & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{0}^{M\times M} \\ \boldsymbol{0}^{M\times M} & \boldsymbol{D}_M & \cdots & \boldsymbol{0}^{M\times M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times M} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{D}_M \end{bmatrix} = \boldsymbol{D}_{\widetilde{M}}\,\boldsymbol{A}$$





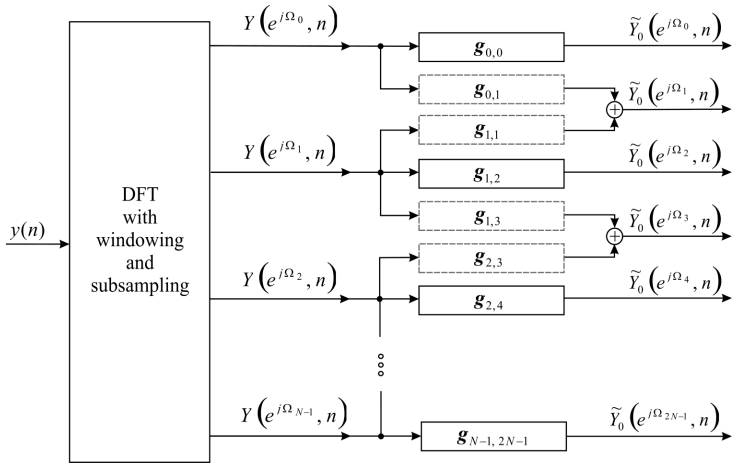
Can be approximated by a very sparse matrix.

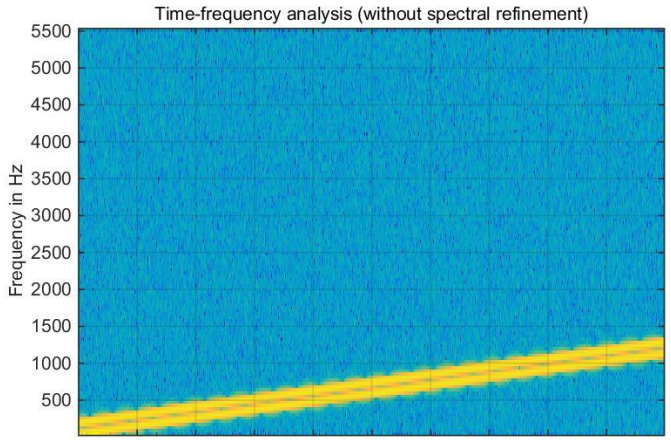
$$\widetilde{\boldsymbol{S}}_{\mathrm{ref}} \;=\; \boldsymbol{D}_{\widetilde{M}}\,\boldsymbol{A} \begin{bmatrix} \boldsymbol{D}_M^{-1} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{0}^{M\times M} \\ \boldsymbol{0}^{M\times M} & \boldsymbol{D}_M^{-1} & \cdots & \boldsymbol{0}^{M\times M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{M\times M} & \boldsymbol{0}^{M\times M} & \cdots & \boldsymbol{D}_M^{-1} \end{bmatrix}.$$

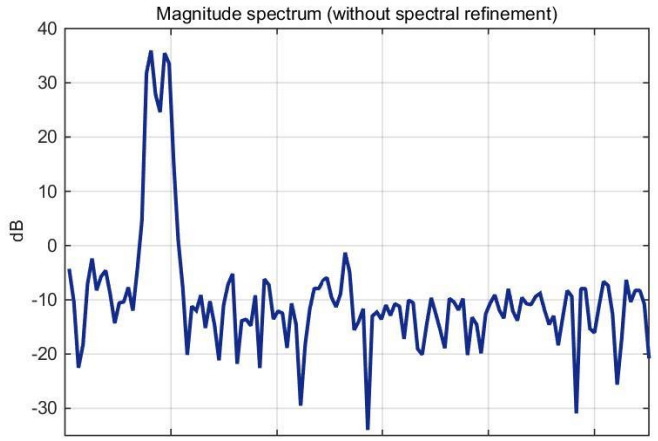
Spectral Refinement

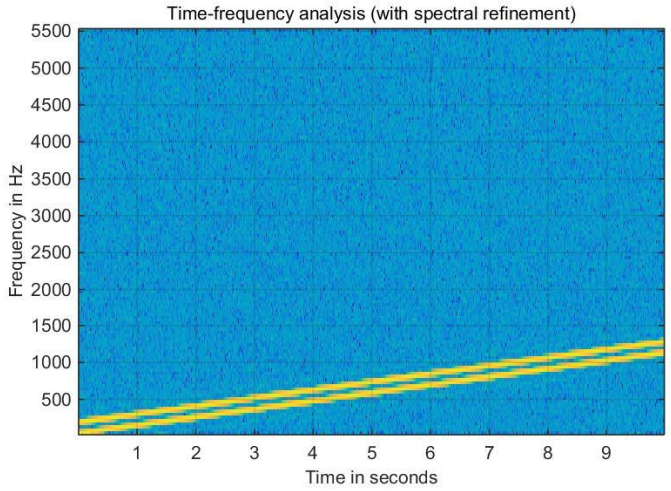
Realization – Part 1

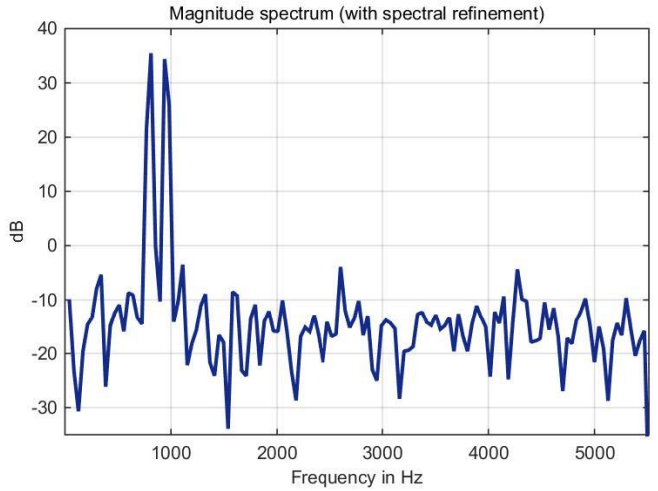
Example for a five tab refinement filter











Spectral Refinement

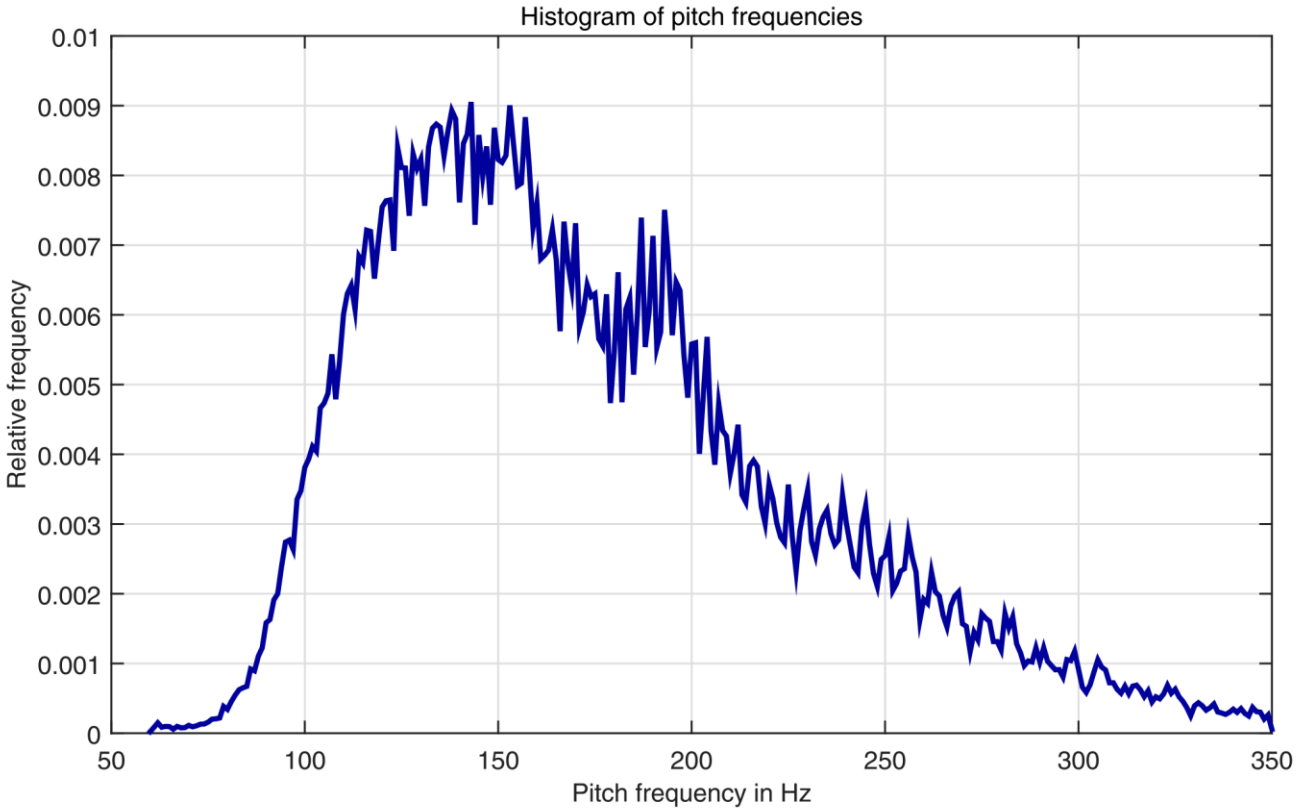
Results – Part 1

Pitch estimation – data base

- Clean speech laryngograph/microphone database
- 38 speakers (18 female and 20 male)
- Mixing to different typical automotive SNR conditions

Pitch estimation – error types

- < 3 % absolute difference
- < 10 % absolute difference
- < 20 % absolute difference



Spectral Refinement

Results – Part 2

Pitch estimation – results

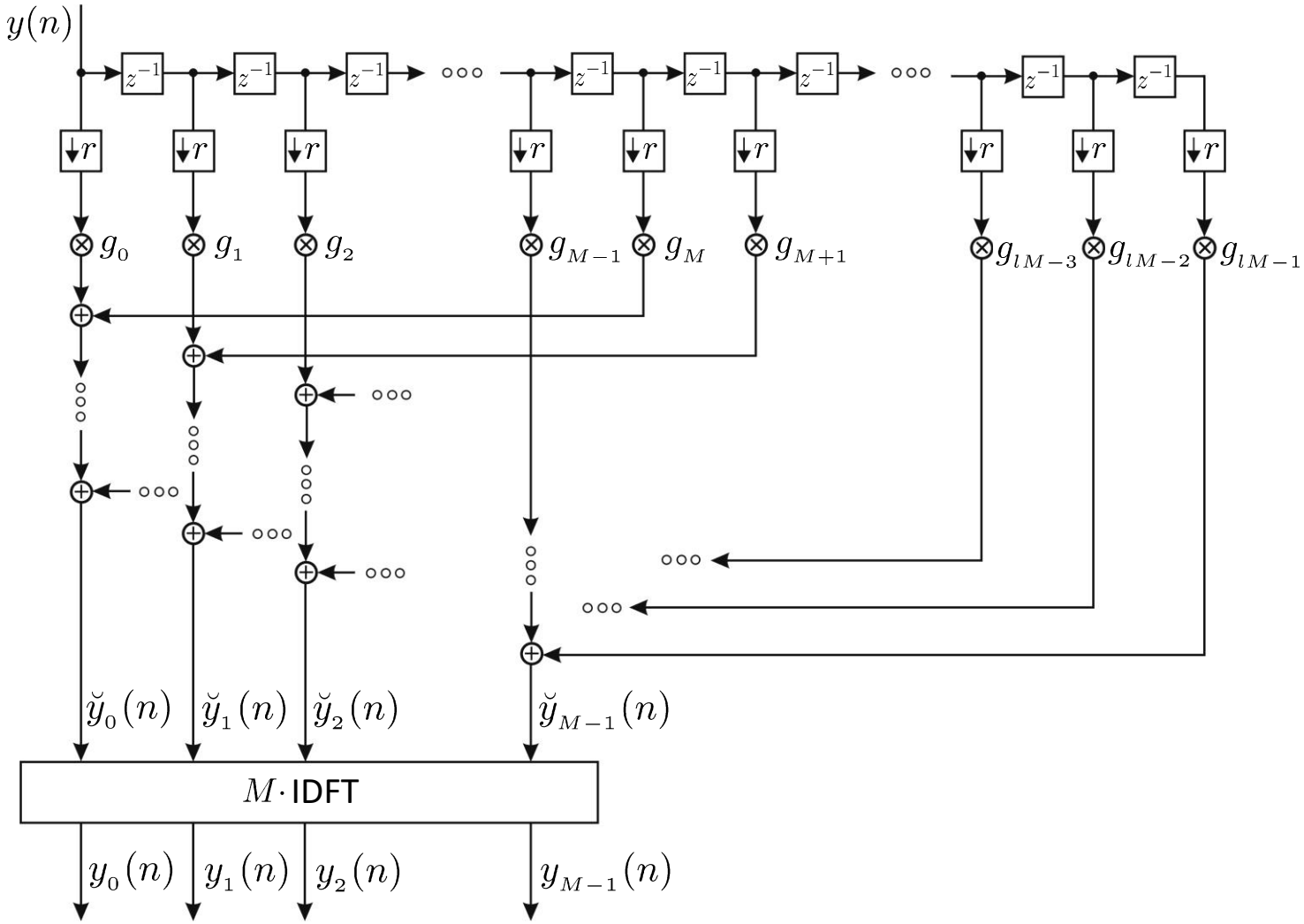
- Correctness of estimated fundamental frequencies without and with spectral refinement for different SNR and tolerance ranges

	Accepted tolerance	High SNR	Medium SNR	Low SNR
Standard method	< 3 %	62.1 %	70.1 %	47.2 %
	< 10 %	65.1 %	70.9 %	48.4 %
	< 20 %	65.5 %	71.4 %	49.3 %
Spectral refinement up to 1 kHz	< 3 %	82.1 %	80.3 %	55.9 %
	< 10 %	88.1 %	85.3 %	58.2 %
	< 20 %	88.8 %	86.2 %	58.7 %
Spectral refinement up to 3 kHz	< 3 %	83.2 %	80.1 %	53.4 %
	< 10 %	89.6 %	85.3 %	56.9 %
	< 20 %	90.6 %	86.3 %	57.5 %

Spectral Refinement

Outlook – Part 1

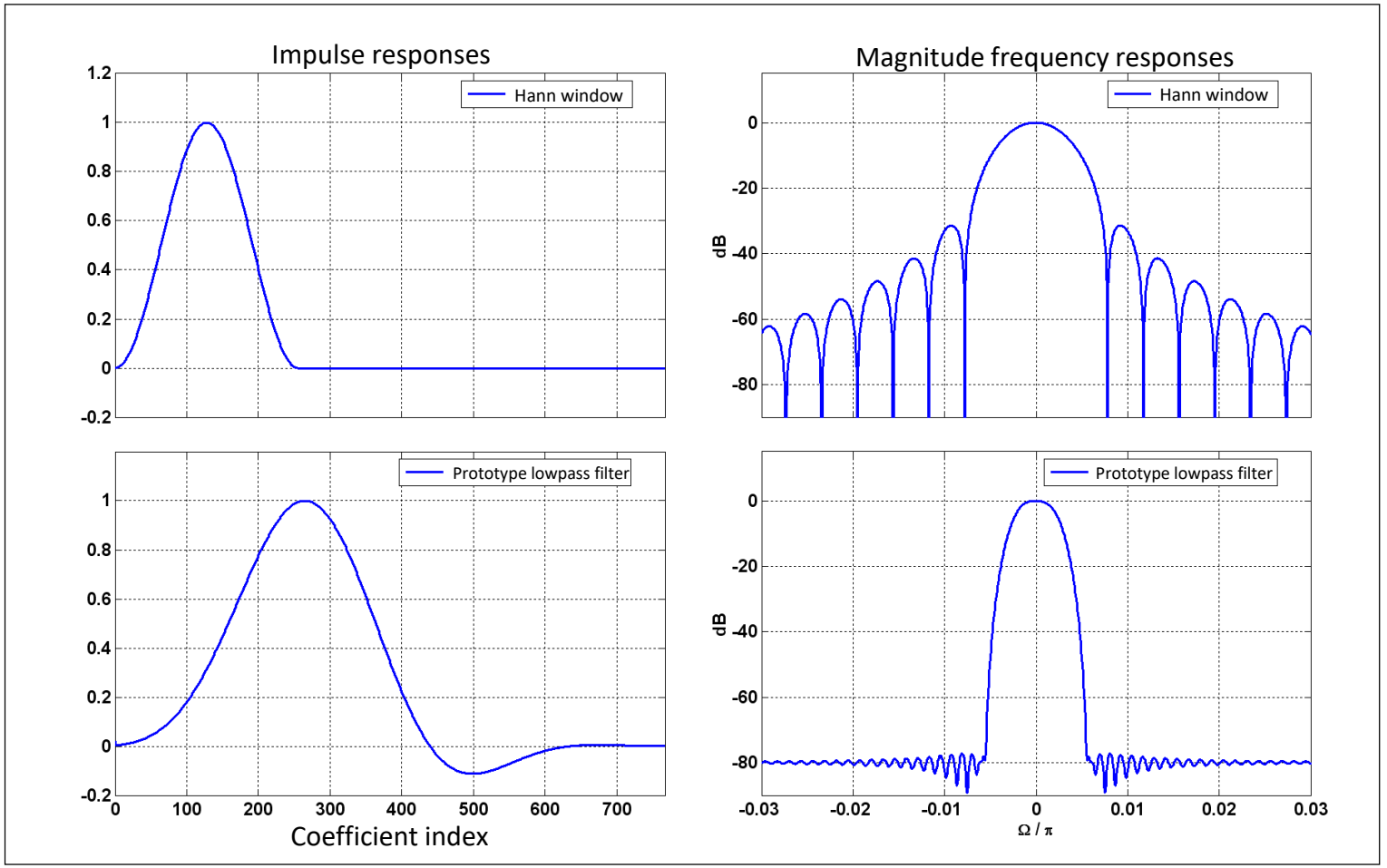
Polyphase filterbanks in the lecture “Adaptive Filters”



Spectral Refinement

Outlook – Part 2

Polyphase filterbanks in the lecture “Adaptive Filters”



Summary

- Introduction
- Digital processing of continuous-time signals
- Efficient FIR structures
- ***DFT and FFT***
  - ***DFT and signal processing***
  - ***Fast computation of the DFT: The FFT***
  - ***Transformation of real-valued sequences***
  - ***Spectral refinement***
- Digital filters
- Multi-rate digital signal processing