Filtering in the frequency domain

Alessandro Ilic Mezza

February 2022

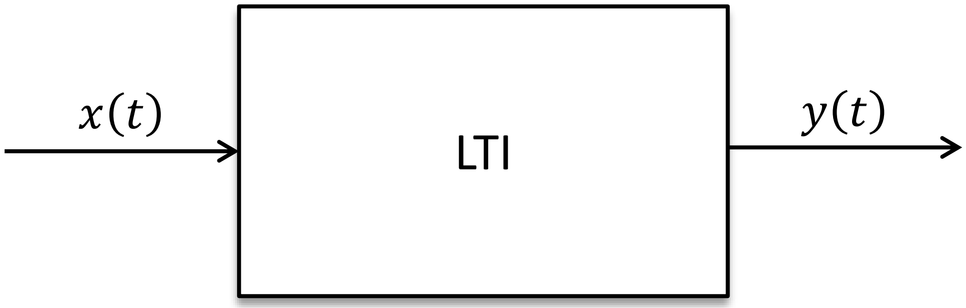
Filters

- A filter is a system that performs mathematical operations on an input signal to modify, reduce or enhance certain components or features of that signal
- Many ways to characterize different kinds of filters can be found in the signal processing literature
- In the following, we will focus on Linear Time-Invariant filters

Linear Time-Invariant Systems
Continuous Time

- Let's start reviewing the continuous-time case

- Let $x: \mathbb{R} \to \mathbb{R}$ be the input signal of a time-continuous Linear Time-Invariant (LTI) system



LTI Properties

- LTI systems are defined by two properties:

  - Linearity: if an input $x_1(t)$ produces an output $y_1(t)$ and an input $x_2(t)$ produces an output $y_2(t)$, then the input $a_1x_1(t) + a_2x_2(t)$ produces the output $a_1y_1(t) + a_2y_2(t)$

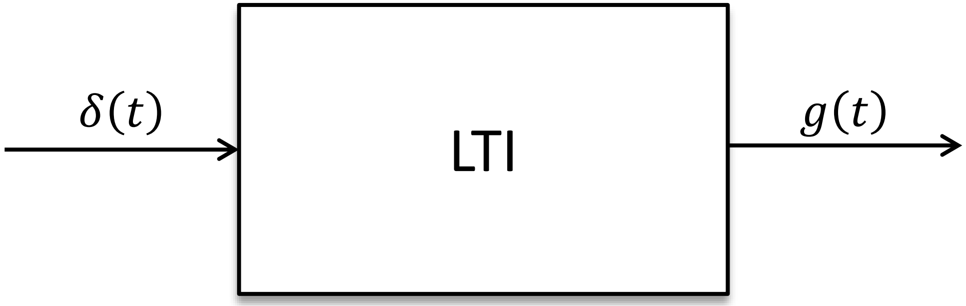
  - Time Invariance: if an input $x(t)$ produces an output $y(t)$, then $x(t - t_0)$ produces the output $y(t - t_0)$

- where a_1, a_2, t_0 are real-valued scalars

Impulse Response

- These two properties determine that any LTI system can be fully characterized by its impulse response $g(t)$

- i.e. the output of the system due to the input $x(t) = \delta(t)$



Dirac Delta Function

- $\delta(t)$ is known as Dirac delta function

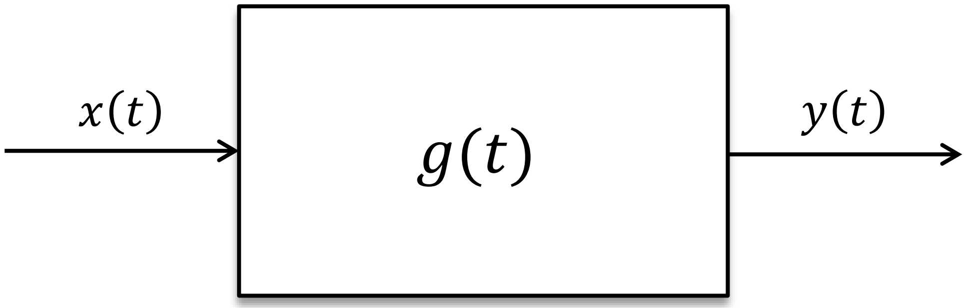
$$\delta(t) = \begin{cases} +\infty, & t = 0 \\ 0, & t \neq 0 \end{cases}$$

$$\int_{-\infty}^{\infty} \delta(t)dt = 1$$

- Note that this is not a function in the proper sense, but that's a rather intuitive way of expressing the analytical properties of $\delta(t)$

Linear Time-Invariant Systems
Continuous Time

- LTI systems are usually defined by means of their Impulse Response (IR)



- Depending on $g(t)$, an LTI system can be, e.g., a low-pass/high-pass filter, a reverberant room, a guitar cabinet...

- Given an input signal $x(t)$, the output of an LTI system $y(t)$ can be obtained through convolution

$$y(t) = (x * g)(t) := \int_{-\infty}^{\infty} x(u)g(t-u)\mathrm{d}u$$

- Note that convolution is commutative

$$(x * g)(t) = (g * x)(t)$$

- We express both functions by means of a dummy variable $u \in \mathbb{R}$ (corresponding to the time axis)

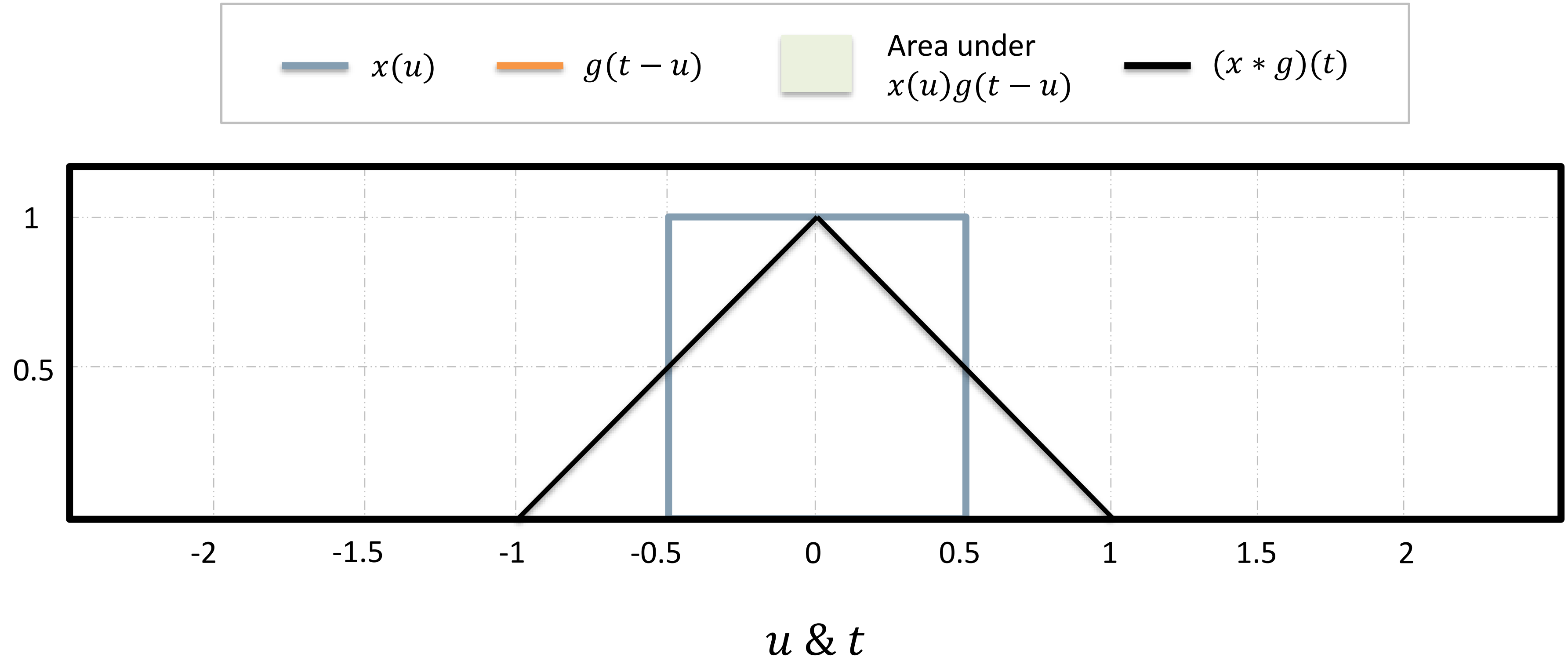
- We reverse one of the function in time, e.g.,
$$g(u) \rightarrow g(-u)$$

- We shift the time-reversed function by a value t, i.e.,
$$g(-u) \ \rightarrow g(-u+t)$$

- The value of the convolution for a given time-offset t is computed by integrating over the dummy variable the product the two functions

- Intuitively:
  - Fix one of the two function in time, flip the other.
  - Let the time-reversed function slide over the fixed one, i.e., let the time-offset t vary from $-\infty$ and $+\infty$,
  - For each t, compute the area under the product of the overlapping sections.
- Note: convolution $(x * g)(t)$ is a function of the time-offset t

Convolution
An example



- Notice that, since we have depicted both the convolved functions and the result of the convolution in the previous figures, the abscissae represents both the dummy variable u and the time-offset variable t

- Notice also that, if the two functions have finite support, the result of the convolution is zero for all those time-offsets t for which the time-reversed “sliding” function does not overlap with the other

Convolution Theorem
Continuous Time

- Given two square-integrable functions $x(t)$ and $g(t)$, let $\mathcal{F}$ be the Fourier transform operator

- The convolution theorem states that

$$\mathcal{F}\{(x * g)(t)\} = \mathcal{F}\{x(t)\} \cdot \mathcal{F}\{g(t)\}$$

- and equivalently

$$\mathcal{F}\{x(t) \cdot g(t)\} = \mathcal{F}\{x(t)\} * \mathcal{F}\{g(t)\}$$

- Convolution in the time domain corresponds to a product in the frequency domain and vice versa!

Convolution Theorem
Continuous Time

- Namely, let $X(f) = \mathcal{F}\{x(t)\}$ and $G(f) = \mathcal{F}\{g(t)\}$ be the Fourier transforms of $x(t)$ and $g(t)$, respectively.

- Then,

$$y(t) = (x * g)(t) \overset{\mathcal{F}}{\leftrightarrow} Y(f) = X(f) \cdot G(f)$$

$$y(t) = x(t) \cdot g(t) \overset{\mathcal{F}}{\leftrightarrow} Y(f) = (X * G)(f)$$

- s.t.

$$y(t) = \mathcal{F}^{-1}\{Y(f)\}$$

- Therefore, signal filtering can be performed in the frequency domain instead of computing the time-domain convolution

$$y(t) = (x * g)(t) \xrightarrow{\mathcal{F}} Y(f) = X(f) \cdot G(f)$$

$$x(t) \rightarrow \boxed{\mathcal{F}} \xrightarrow{X(f)} \boxed{X(f)G(f)} \xrightarrow{Y(f)} \boxed{\mathcal{F}^{-1}} \rightarrow y(t)$$

- As an advantage, the integral required to compute the convolution becomes a simple multiplication

Towards Digital Filters

- We have seen how to filter continuous-time signals both in the frequency domain and via time-domain convolution

- In order to move towards digital filtering, we need a few basic signal processing concepts:

  1. The properties of the Dirac delta function
  2. The train of impulses (Dirac comb)
  3. Sampling a time-domain signal
  4. Sampling the DTFT

Properties of the Dirac Delta Function

- Multiplication by a Dirac delta:

$$x(t) \cdot \delta(t - t_0) = x(t_0) \cdot \delta(t - t_0)$$

- Sifting property of the Dirac delta:

$$\int_{-\infty}^{\infty} x(t)\delta(t - t_0)\mathrm{d}t = x(t_0) \in \mathbb{R}$$

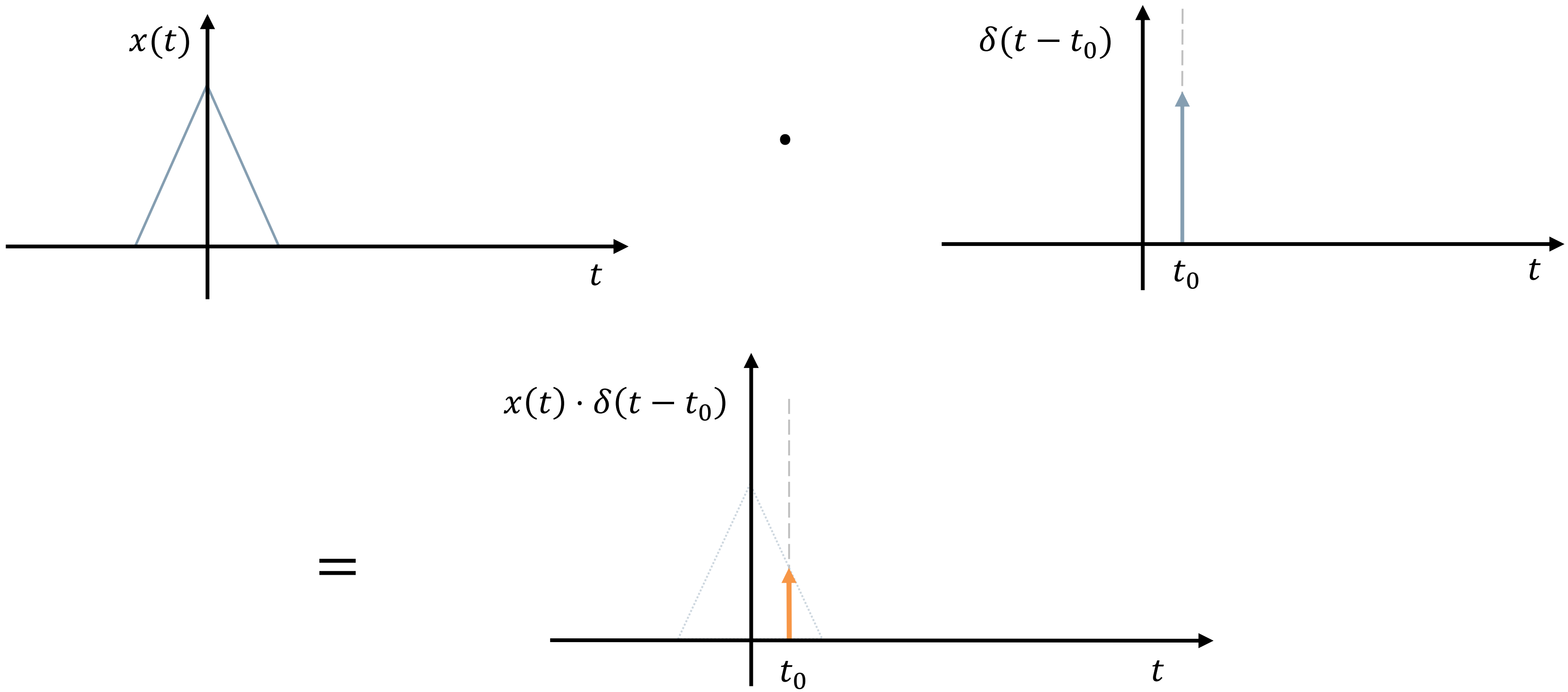
- Convolution by a Dirac delta:

$$x(t) * \delta(t - t_0) = x(t - t_0)$$

- where $t_0 \in \mathbb{R}$

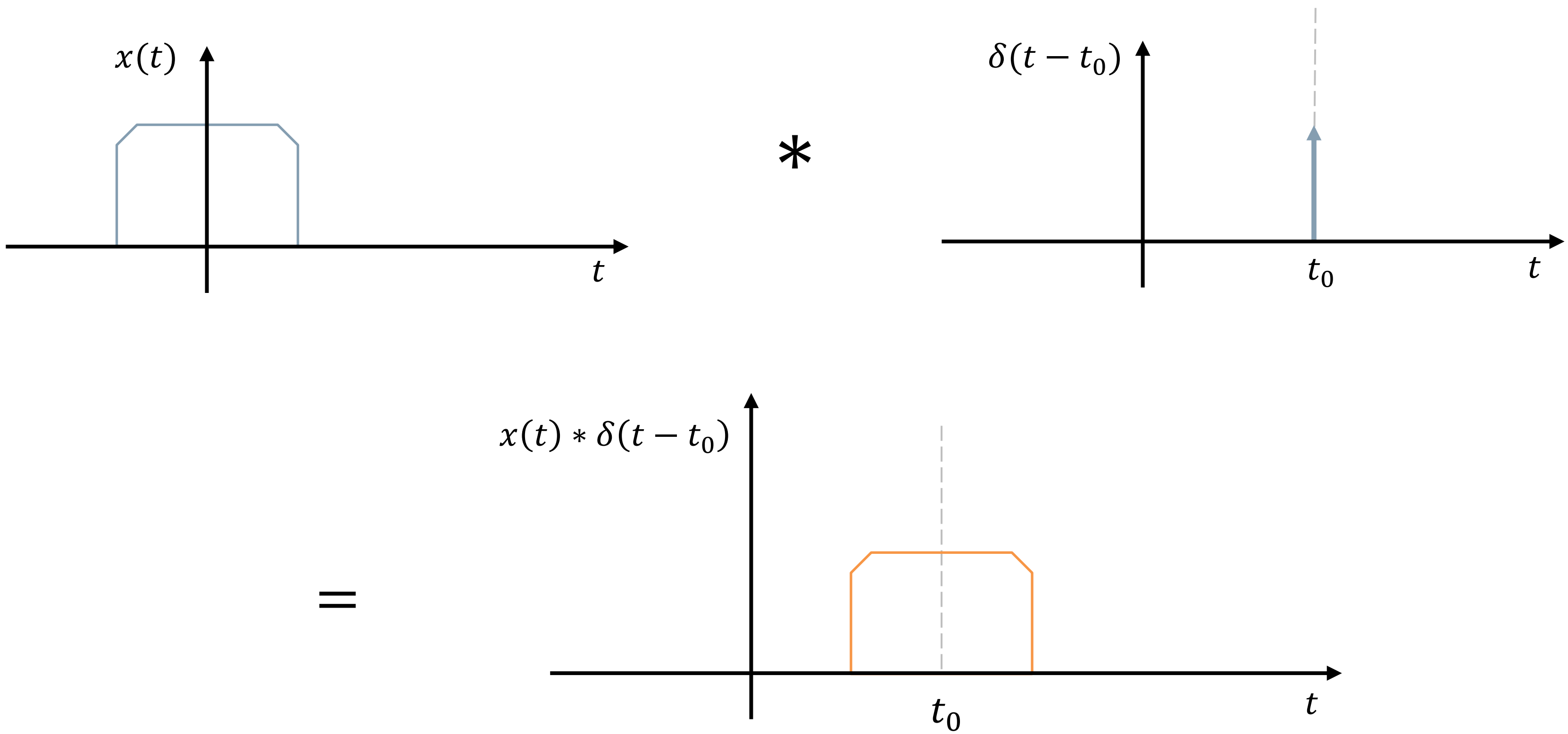
Properties of the Dirac Delta Function
Product

- Product by a Dirac delta function:



Properties of the Dirac Delta Function
Convolution

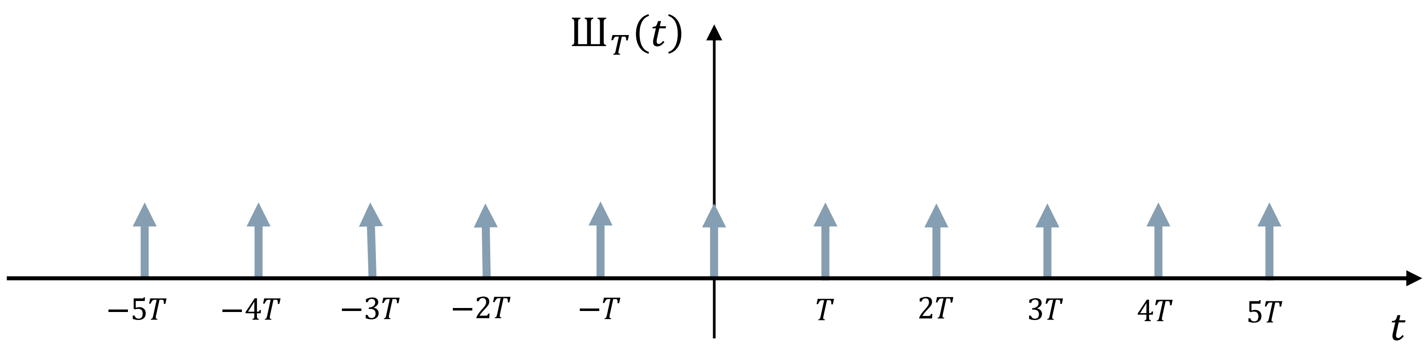
- Convolution by a Dirac delta function:



Train of Impulses

- A train of impulses (also known as Dirac Comb) is defined as

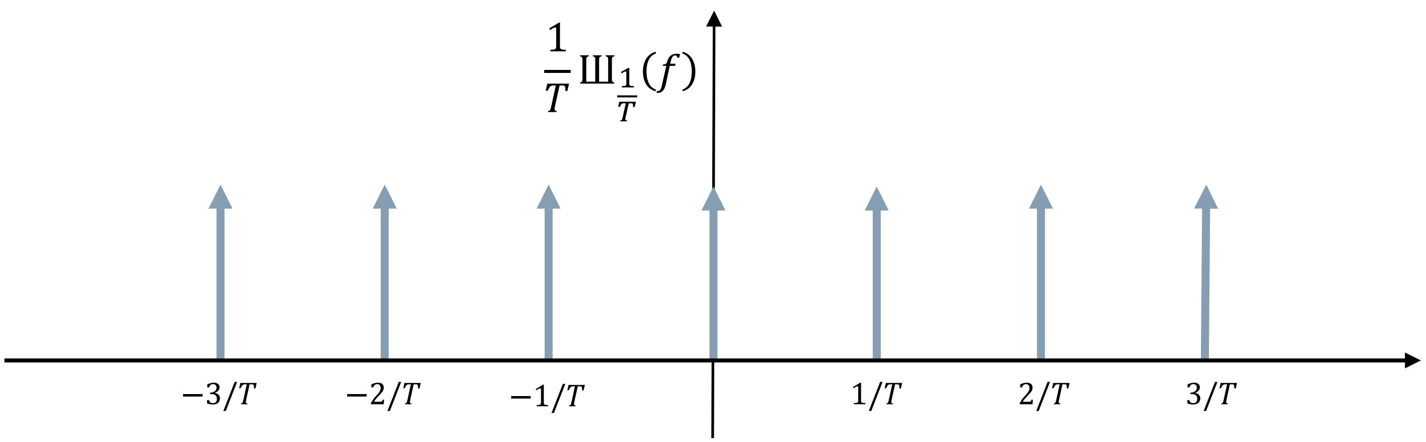
$$Ш_T(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$



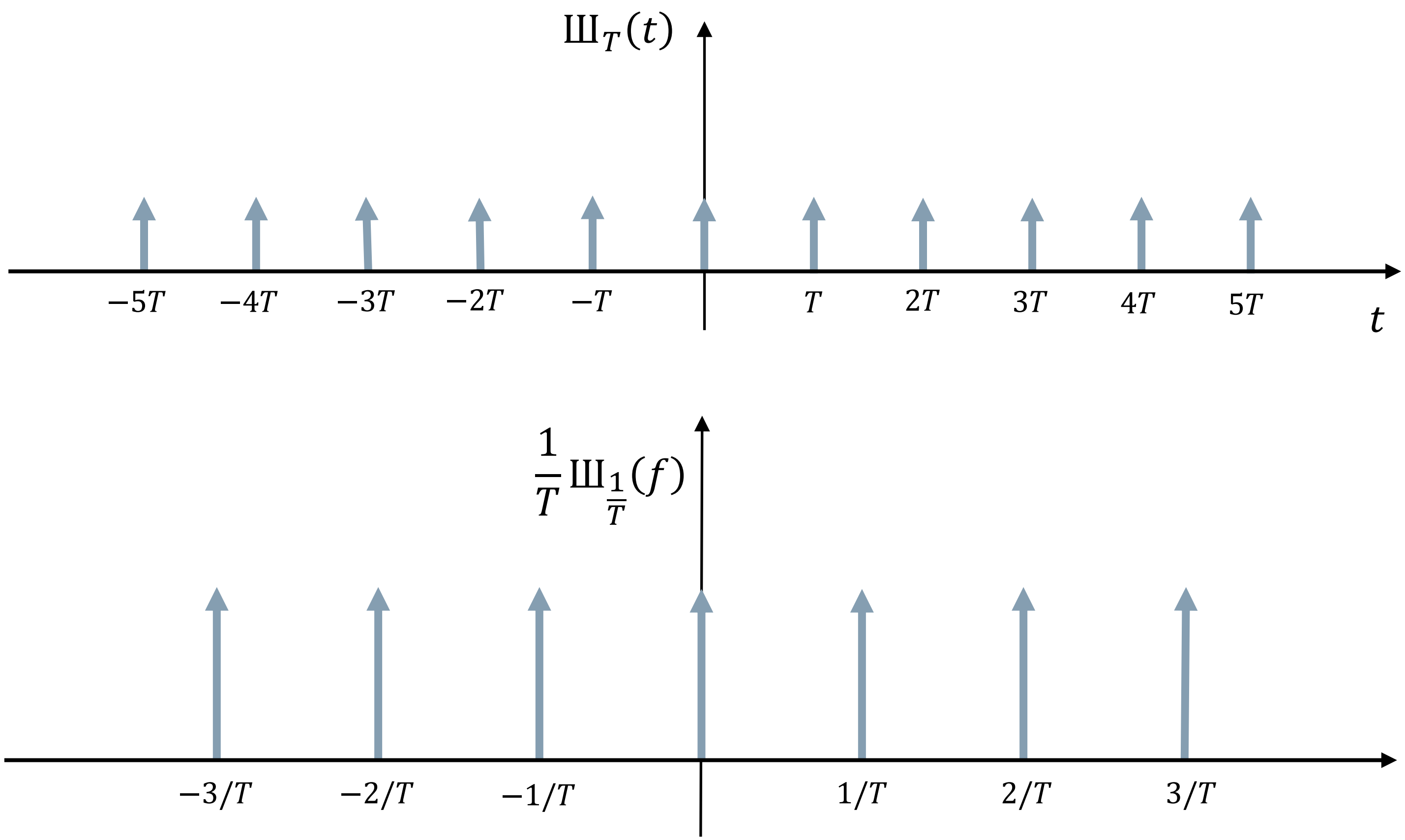
Train of Impulses
Self-transforming Property

- Self-transforming property: The Fourier transform of a Dirac comb is still a Dirac Comb

$$Ш_T(t) \xrightarrow{\mathcal{F}} \frac{1}{T} Ш_{1/T}(f) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \delta\left(f - n\frac{1}{T}\right)$$



Train of Impulses
Visualization



Kronecker Delta

- The discrete-time equivalent of the Dirac delta is known as the Kronecker delta:

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

- Sifting property of the Kronecker delta:

$$\sum_{n=-\infty}^{\infty} x(n)\delta(n-m) = x(m) \in \mathbb{R}$$

Discrete-time Signals

- The train of impulses gives us an analytical way to express discrete-time signals

- In general, a discrete-time signal is a function $x\colon \mathbb{Z} \rightarrow \mathbb{R}$

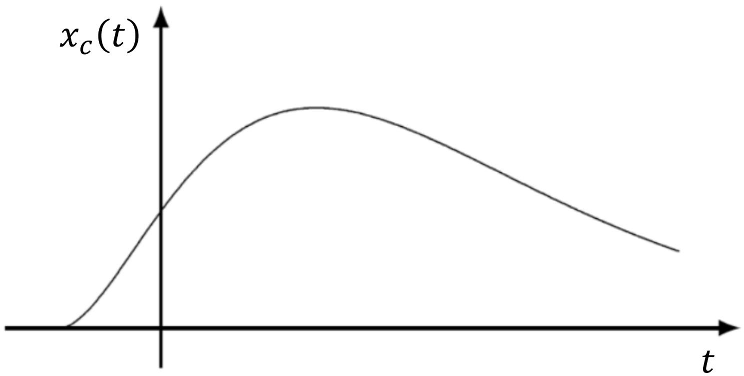
- However, it can also be seen as a sequence of values $x(nT)$, $n \in \mathbb{Z}$, obtained by sampling a continuous-time signal $x_c(t)$ at regular intervals, i.e., every T seconds

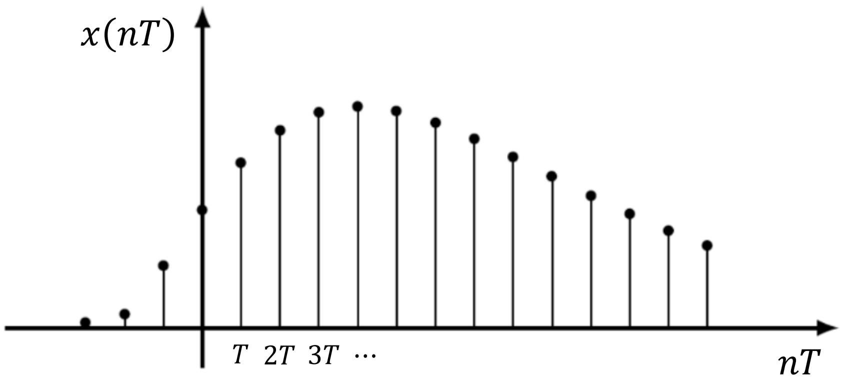
- $T \in \mathbb{R}$ is called sampling period

Sampling in the Time Domain

- This allow us to define sampling in time domain as

$$x(nT) = x_c(t) \cdot Ш_T(t)$$





- $T \in \mathbb{R}$ (sampling period)
- $\mathrm{F}_s = 1/T$ (sampling frequency)

Convolution Theorem
Discrete Time

- The convolution theorem for discrete-time signals is

$$y(nT) = (x * g)(nT) \xleftrightarrow{\text{DTFT}} Y(f) = X(f) \cdot G(f)$$

$$y(nT) = x(nT) \cdot g(nT) \xleftrightarrow{\text{DTFT}} Y(f) = (X * G)(f)$$

- The latter relationship and the self-transforming property yields

$$x(nT) = x_c(t) \cdot \text{Ш}_T(t) \xrightarrow{\text{DTFT}} X(f) = \frac{1}{T}\left(X_c * \text{Ш}_{1/T}\right)(f)$$

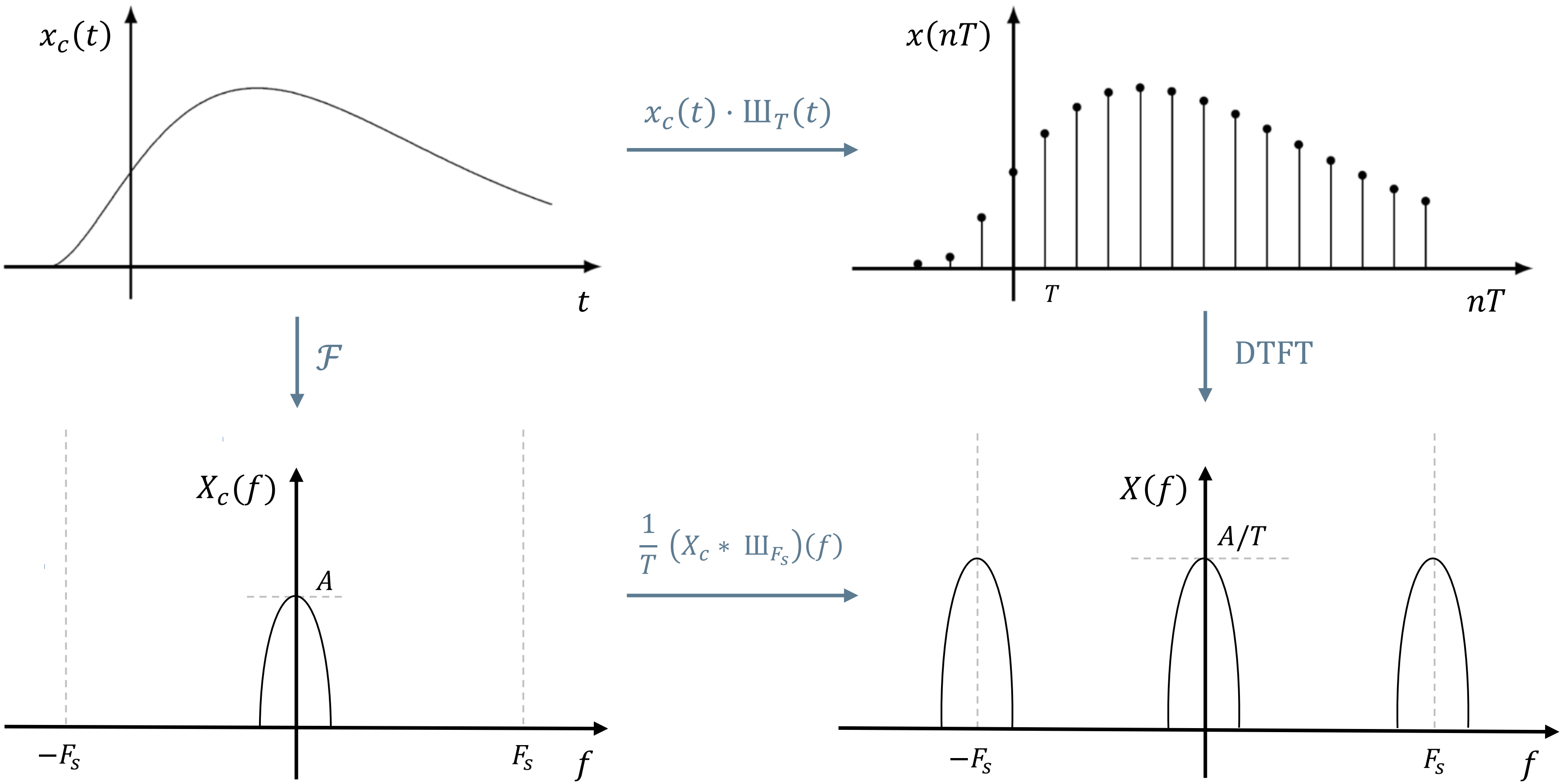
Effects of Sampling in the Time Domain

- Since convolution with a delta function $\delta(f - n/T)$ is equivalent to shifting a function by n/T, convolution with the Dirac comb corresponds to the replication or periodic summation of the spectrum $X_c(f)$

$$\frac{1}{T}\left(X_c * Ш_{1/T}\right)(f) = \frac{1}{T}\sum_{n=-\infty}^{\infty} X_c\left(f - \frac{n}{T}\right)$$

- In other words, the spectrum $X_c(f)$ is replicated infinitely many times and each replica is shifted in the position of one of the Dirac delta functions in the comb, i.e., every integer multiple of $F_s = 1/T$

Effects of Sampling in the Time Domain



Effects of Sampling in the Time Domain

- The resulting spectrum of the discrete-time signal is periodic with period $\mathrm{F}_s = 1/T$
- This leads to an intuitive understanding of the inherent periodicity of the DTFT in the frequency domain…
- …and to a natural formulation of the Nyquist-Shannon sampling theorem!

Effects of Sampling
Dualities

- Discretizing in the time domain corresponds to introducing a periodicity in the frequency domain…
- …and vice versa: sampling in the frequency domain corresponds to introducing a periodicity in the time domain
- This aspect is critical as computers cannot deal with continuous frequency and thus the DTFT cannot be implemented in practice

Sampling the DTFT

- Recall the definition of the DTFT – periodic with period $F_s = 1/T$

$$X(f) = \sum_{n=-\infty}^{\infty} x(nT)e^{-j2\pi f nT}$$

- We can now sample $X(f)$ with a sampling step equal to $1/NT$

$$X(f)\Big|_{f=\frac{k}{NT}} = \sum_{n=-\infty}^{\infty} x(nT)e^{-j2\pi\frac{k}{NT}(nT)}$$

Sampling the DTFT

- Note: $f = \frac{k}{NT}$ is so that when $k = N$, then $f = 1/T$, i.e., equal to the periodicity of $X(f)$
- In fact, each period of the DTFT is sampled with N equidistant points: $f \in [0, F_s) \rightarrow k = 0, \ldots, N-1$
- Notably, due to discretization in the frequency domain, the inverse transform of the sampled DTFT ends up being a periodic signal with period NT

Discrete Fourier Transform (DFT)

- By assuming $T = 1$ and limiting the sum to a single period both in the time and the frequency domain, we obtain the definition of the Discrete Fourier Transform (DFT)

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

- Often, the substitution $\omega_k \coloneqq 2\pi k/N$ is applied. We can thus write the DTF of $x(n)$ as

$$X(\omega_k) = \sum_{n=0}^{N-1} x(n) e^{-j\omega_k n}$$

The Effects of the DFT
In short

- In short, the need for a numerical computation of the Fourier transform requires discrete time and discrete frequency → DFT
- Because of sampling, we end up dealing with a periodic repetition of the spectrum in the frequency domain…
- …as well as with a periodic repetition of the signal in time domain

The Effects of the DFT
On Aperiodic Signals

- However, not all discrete-time signals are periodic!

- Periodicity is a (necessary) nuisance introduced by the computation of the DFT itself

- Imagine having a discrete-time signal $x(n)$ of length N (i.e., nonzero only for $n \in [0, N-1]$) obtained with a sampling frequency $F_s = 1/T$

- Note that such a signal is indeed aperiodic!

The Effects of the DFT
On Aperiodic Signals

- If we compute the DFT of this signal, however, we should cope with the fact that:
  - the resulting spectrum is F_s-periodic
  - the signal has implicitly become NT-periodic in the time domain

- Therefore, the Inverse DFT (IDFT) yields a periodic repetition of the original signal

- Fortunately, we can usually limit the result of the IDFT to just a single period – i.e., the original N samples – as within such interval the periodic repetition corresponds to the original signal (deperiodization)

- However, if we perform some processing in the frequency domain, we should be rather careful in order to obtain the desired time-domain signal via IDFT in spite of the periodicity we have introduced

- In particular, we are going to see which precautions to take if one wants to exploit the convolution theorem to perform filtering in the frequency domain similarly to what we saw for continuous-time signals

- Note: in the following, to stress that the results are independent on the sampling rate, we will consider $T = 1$ without loss of generality

- In general, the response of a digital filter with finite impulse response $g(n)$ can be obtained by means of direct convolution

$$y(n) = (x * g)(n) = \sum_{m=-\infty}^{\infty} x(m)g(n-m)$$

- If $x(n)$ is nonzero only for $n \in [0, N-1]$ (as it is the case for most real-life signals) then the formula becomes

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

Complexity

- In the latter case, the complexity of computing the direct convolution in time domain is O(N^2)

- Thus, to save up on computation time, we would like to perform the convolution as a multiplication in the frequency domain as we have seen for continuous-time signals…

- Indeed, a fast implementation known as Fast Fourier Transform (FFT) allows to compute the DFT with a complexity of O($N\log N$)

- Idea

  - Compute the FFT of both $x(n)$ and $g(n)$ → $\mathrm{O}(N \log N)$
  - Compute the product $X(\omega_k) \cdot G(\omega_k)$ → $\mathrm{O}(N)$
  - Compute the IFFT of $Y(\omega_k)$ → $\mathrm{O}(N \log N)$

- If N is big enough ($N \geq 128$ is a good a rule of thumb), it is thus preferable to perform the filtering via FFT instead of the direct convolution

Circular Convolution Theorem

- However, the discrete version of the convolution theorem (also known as circular convolution theorem) is defined as

$$y(n) = (x \circledast g)(n) \leftrightarrow Y(\omega_k) = X(\omega_k) \cdot G(\omega_k)$$

- Where $\circledast$ denotes the cyclic convolution

$$(x \circledast g)(n) = \sum_{m=0}^{N-1} x_N(m) g(n-m)$$

- where $x_N(n)$ is a periodic signal of period N
- and $g(n)$ is the filter impulse response

Circular Convolution Theorem

- By recalling that computing the DFT implicitly introduces a time-domain periodicity of N samples, it is readily clear why the circular convolution theorem is given in terms of cyclic convolution

- Having to deal with such a periodicity (“circularity”) means that we should pay more attention in the discrete case with respect to the continuous case!

Filtering in the Frequency Domain
Convolution Length

- The result of a convolution is longer than the convolved signal $x(n)$
- Imagine a short sound (e.g., a hand clap) fed to a digital reverb (long impulse response, e.g., 10000 samples)
- The result of such filtering will be way longer than the original sound!

$g(n)$

$x(n) \longrightarrow$

$\longrightarrow y(n)$

Zero padding

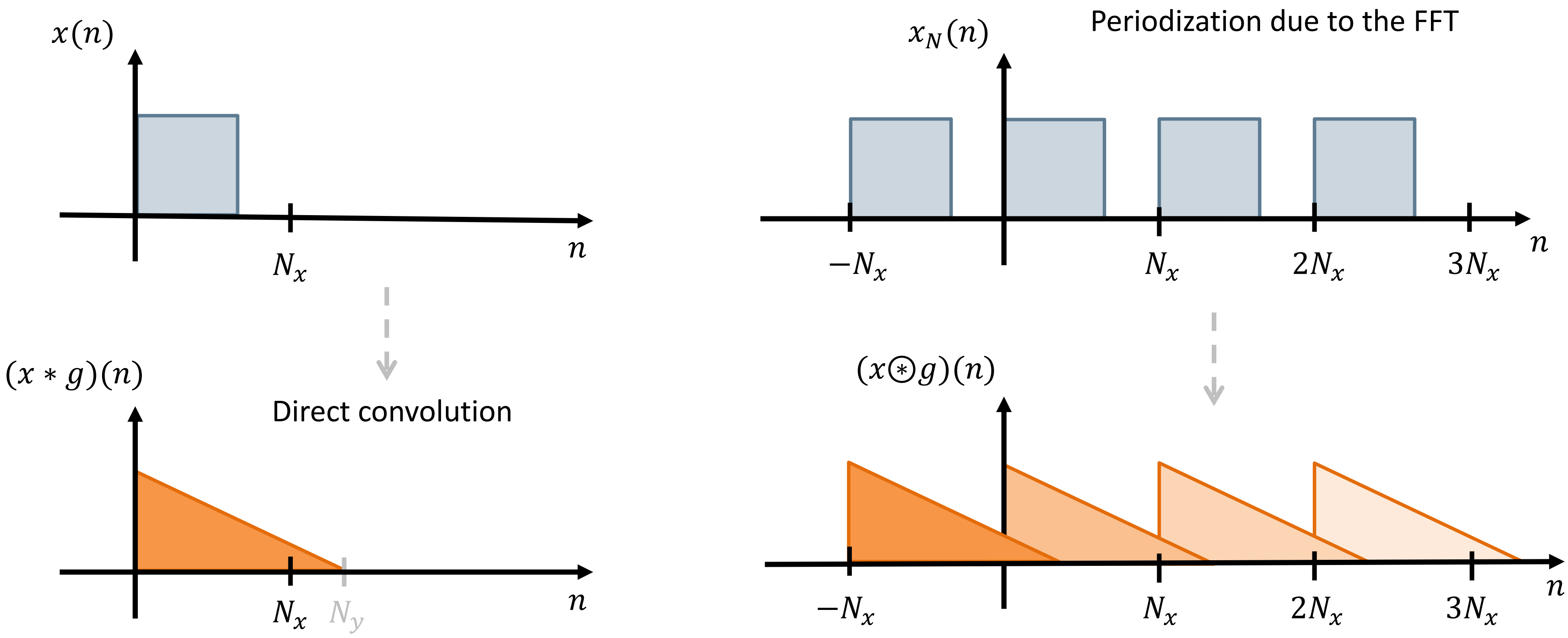
- Namely, if the signal $x(n)$ and the impulse response $g(n)$ are N_x and N_g samples long, respectively
- Then the convolution $y(t)$ will be of length $N_y = N_x + N_g - 1$
- Therefore, we must add enough zeros (zero-padding) to $x(n)$ and $g(n)$ so that their length is N_y or longer (usually a power of 2 in order to exploit the efficiency of the FFT)
- If we don't do that, some of our convolution terms "wrap around" and add back upon the others!

- Zero-padding $x(n)$ up to a length N_y before computing the FFT implies that the time-domain periodicity we are introducing is actually of period N_y instead of N_x

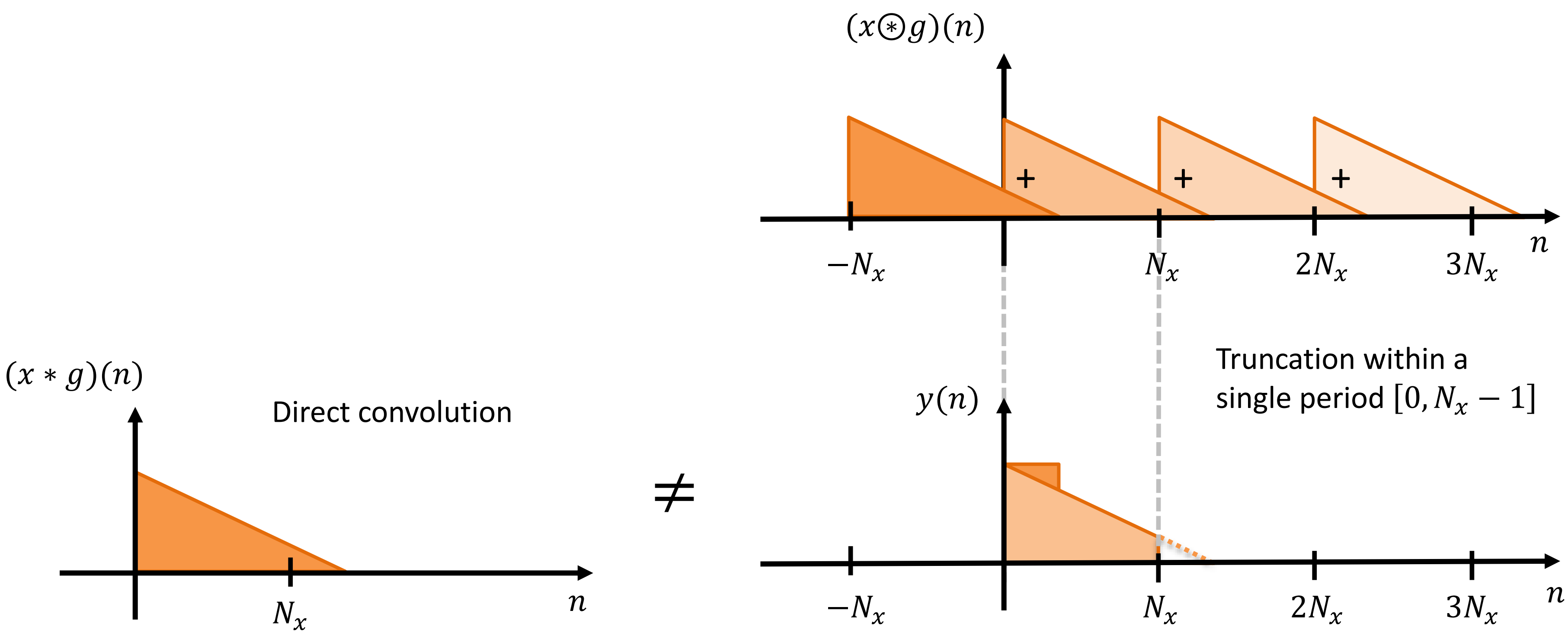
- Then, having computed $\mathrm{Y}(\omega_k) = X(\omega_k) \cdot G(\omega_k)$ via a simple multiplication, we can now take the IFFT and truncate the resulting $y(n)$ within a single period of length N_y

- This way, we obtain the same result as the one of a direct convolution while avoiding unwanted “artifacts” and saving up on computation time

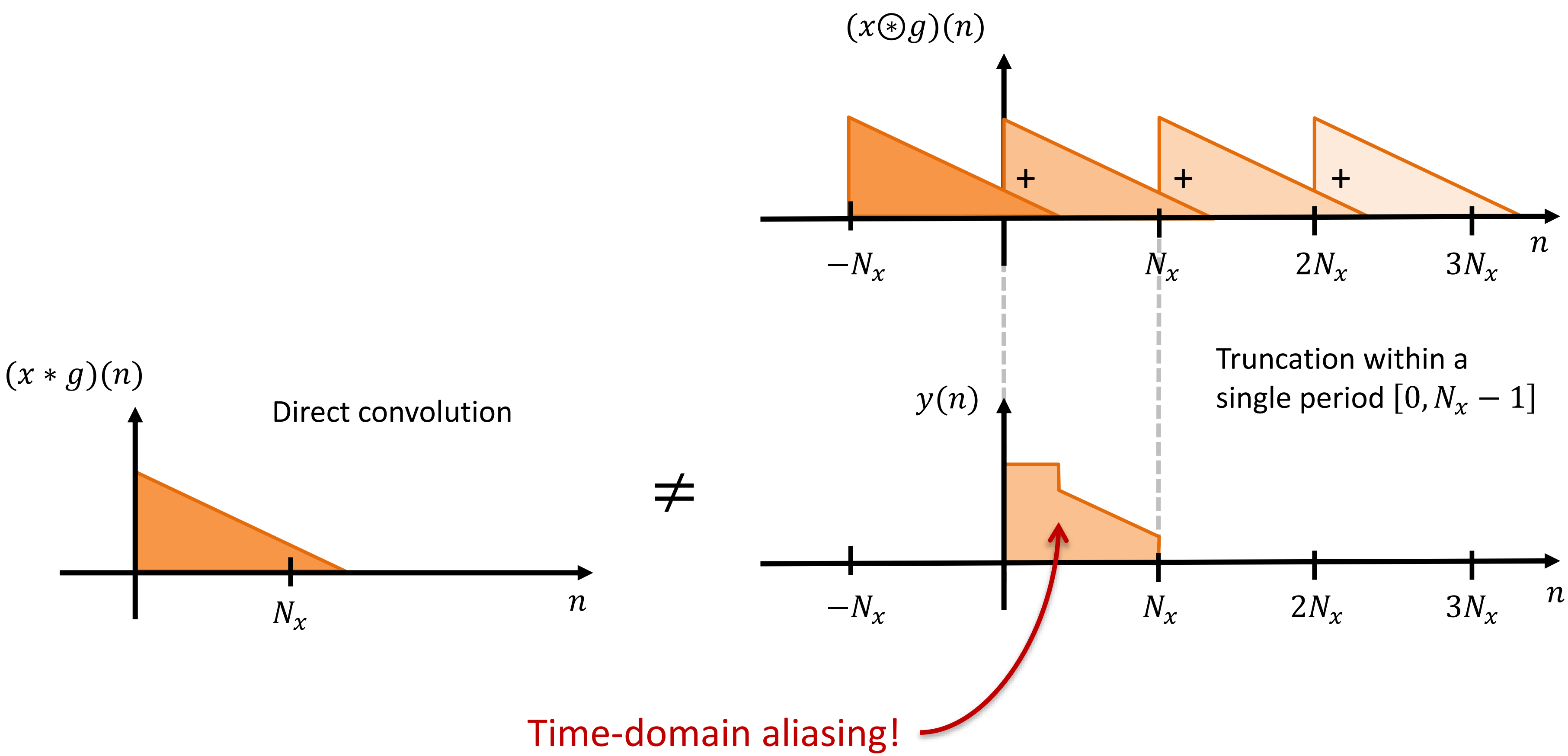
Example of Time-domain Aliasing
No Zero-padding



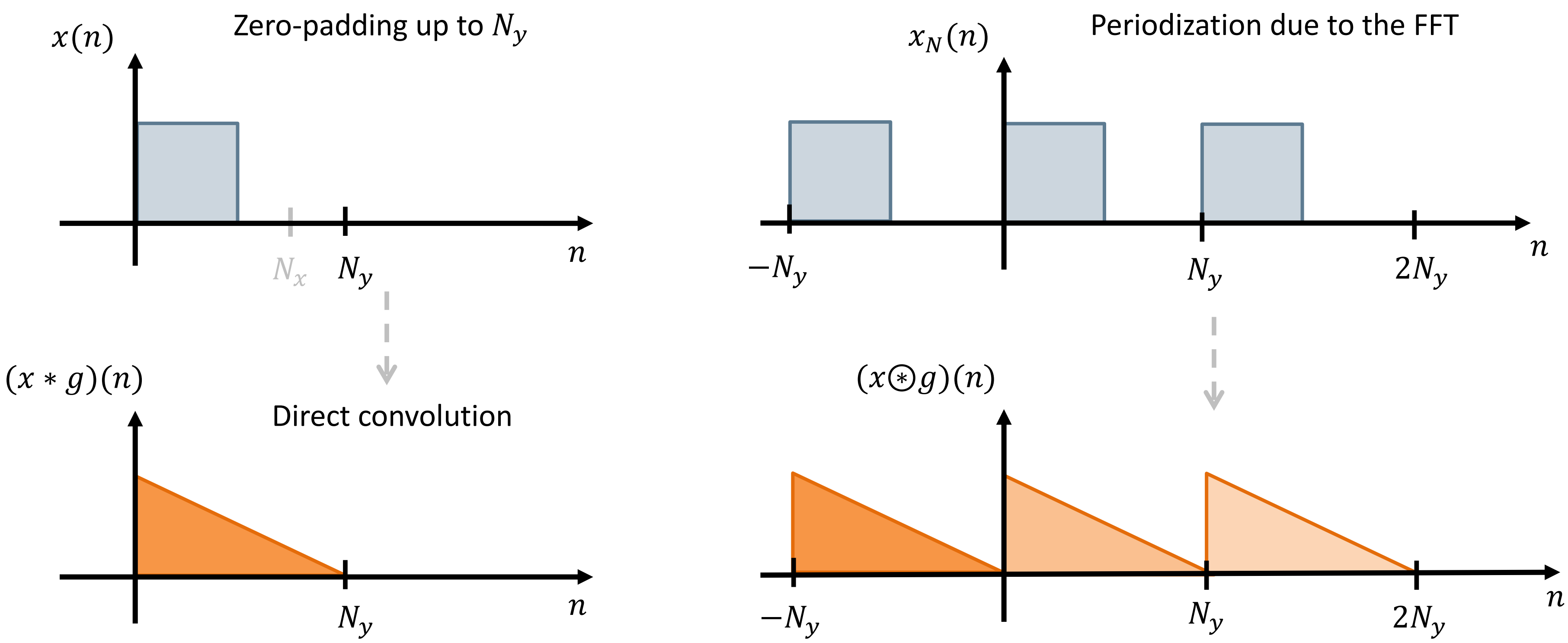
Example of Time-domain Aliasing
No Zero-padding



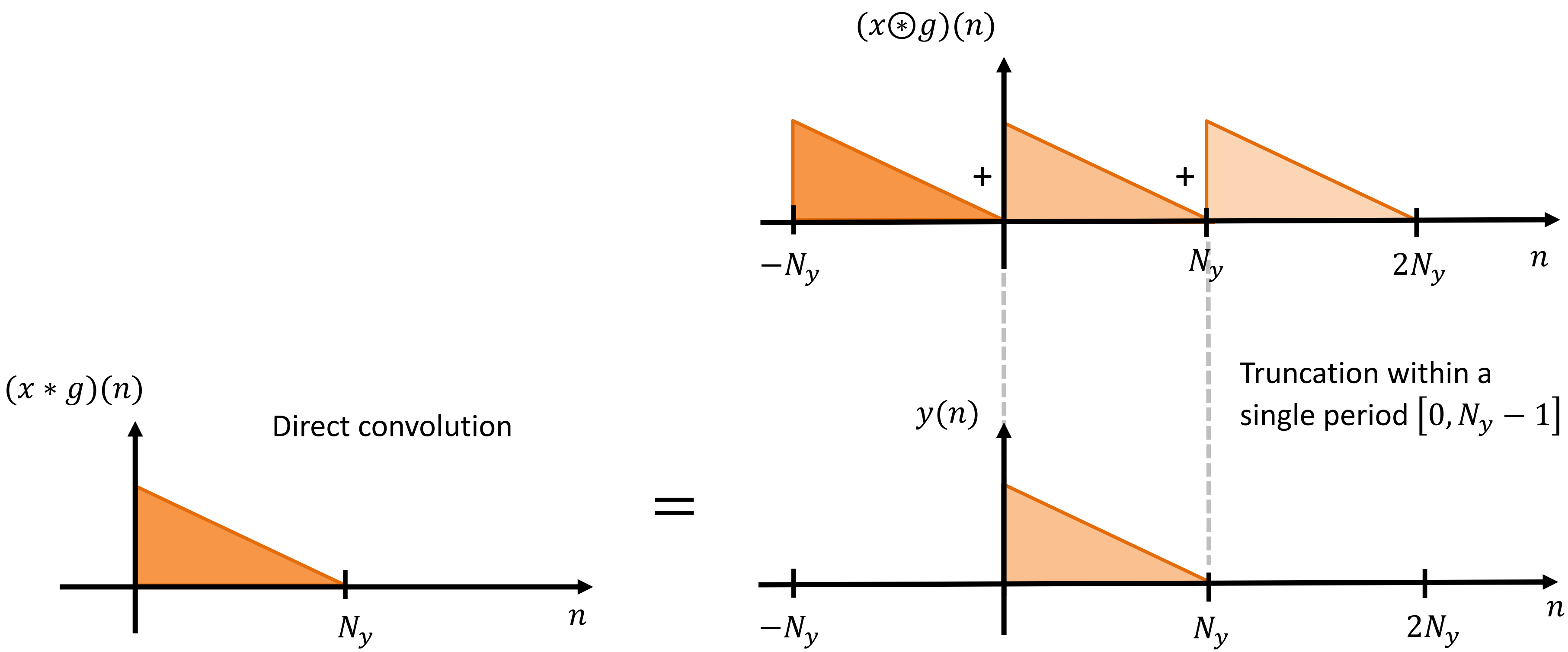
Example of Time-domain Aliasing
No Zero-padding



Avoid Time-domain Aliasing
Zero-padding



Avoid Time-domain Aliasing
Zero-padding



No time-domain aliasing!

Filtering in the Frequency Domain
Time-domain Aliasing

- These “artifacts” can be thought as time-domain aliasing

- In other terms, enlarging the period from N_x up to N_y in time domain corresponds to having more samples (closer spacing) in the discrete frequency domain

- This can be thought of as a higher “sampling rate” in the frequency domain – note that we are now sampling the frequency axis with a sampling interval $\frac{1}{N_y} < \frac{1}{N_x}$ (since $N_y > N_x$)

- If we have a high enough sampling rate, we can avoid time-domain aliasing – analogously to the Nyquist-Shannon sampling theorem!

- Problem: There are some situations where it is not practical to perform the convolution of two signals using a single FFT:
  - N_x is very large
  - Real-time applications
  - We can’t wait until the signal ends

- Solution: The Overlap-And-Add (OLA) algorithm!

Overlap-And-Add (OLA)

- Idea: Process the signal one block at a time

- Algorithm:
  1. Chop up the input signal $x(n)$ by windowing
  2. Perform FFT convolution on each block separately
  3. Make sure we put it all back together correctly!

- The latter aspect is ensured by the so-called Constant Overlap-And-Add (COLA) condition

Overlap-And-Add (OLA)
Windowing

- Let $w(n)$ be a zero-phase window of length M
- The block extraction is performed as follows:

$$x_m(n) \coloneqq x(n) \cdot w(n - mR)$$

- where $m \in \mathbb{Z}$ is the frame index and $R \in \mathbb{N}$ is the hop size
- Intuitively, we let the window $w(n)$ "slide" over the signal $x(n)$
- The hop size R corresponds to the number of samples the m-th window is shifted with respect to the $(m-1)$-th

Overlap-And-Add (OLA)
Perfect Reconstruction

- For this frame-by-frame spectral processing to work, we must be able to reconstruct $x(n)$ from the individual overlapping frames

- This can be written as

$$x(n) = \sum_{m=-\infty}^{\infty} x_m(n) = \sum_{m=-\infty}^{\infty} x(n) \cdot w(n - mR) = x(n) \cdot \sum_{m=-\infty}^{\infty} w(n - mR)$$

- It follows that

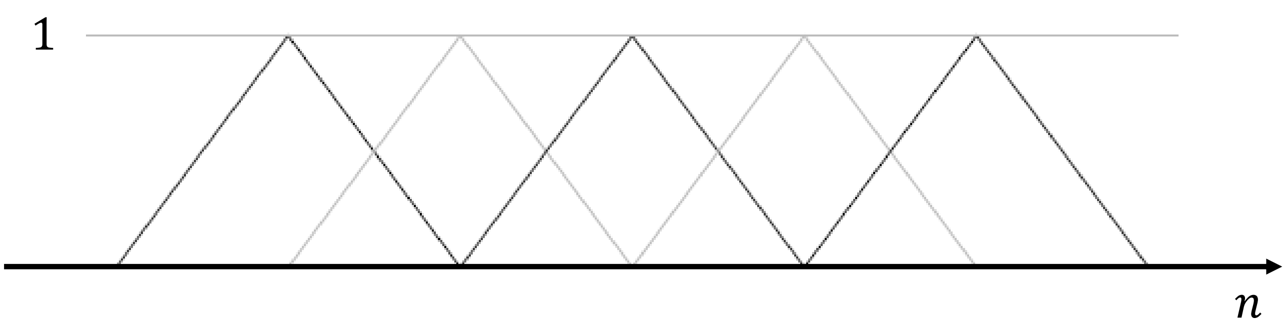
$$x(n) = \sum_{m=-\infty}^{\infty} x_m(n) \overset{\text{if and only if}}{\Longleftrightarrow} \sum_{m=-\infty}^{\infty} w(n - mR) = 1$$

COLA Condition

- The right-hand term is the so-called Constant Overlap-and-Add (COLA) condition on the analysis window $w(n)$

$$\sum_{m=-\infty}^{\infty} w(n - mR) = 1$$

- Example: Triangular window with $R = M/2$ (50% overlap)



COLA Condition

- There is no constraint on the window type, provided that the window overlap-adds to a constant for the hop size used

- Examples:
  - Rectangular window at 0% overlap ($R = M$)
  - Triangular (Bartlett) window at 50% overlap* ($R = M/2$)
  - Hann or Hamming window at 50% overlap* ($R = M/2$)
  - Hann or Hamming window at 75% overlap* ($R = M/4$)
  - …
  - Any window with $R = 1$ (“sliding FFT”)

* M even

Filtering in Frequency Domain via OLA

- We would like to perform convolution in the frequency domain via FFT using OLA
- This is achieved by applying the convolution theorem to each windowed block $x_m(n)$
- As discussed above, to avoid time-domain aliasing, we need to zero-pad each block up to a length of $N_y = N_g + M - 1$
- Where N_g is the length of the filter response and M is the length of the window

The OLA algorithm

- Each m-th block $x_m(n)$ is translated back to the origin by shifting it by mR

$$\tilde{x}_m(n) \coloneqq x_m(n + mR)$$

- We zero-pad $\tilde{x}_m(n)$ and $g(n)$ up to a total length of $N_g + M - 1$ samples

- We compute the convolution as the product of the FFTs

$$\tilde{Y}_m(\omega_k) = \tilde{X}_m(\omega_k) \cdot G(\omega_k)$$

- We take the IFFT of the result

$$\tilde{y}_m(n) = \mathrm{IFFT}\{\tilde{Y}_m(\omega_k)\}$$

- Then, $\tilde{y}_m(n)$ is shifted back by $-mR$

$$y_m(n) = \tilde{y}_m(n - mR)$$

- Finally, the filtered signal is reconstructed as the sum of every overlapping frames obtained as described above

$$y(n) = \sum_{m=-\infty}^{\infty} y_m(n)$$

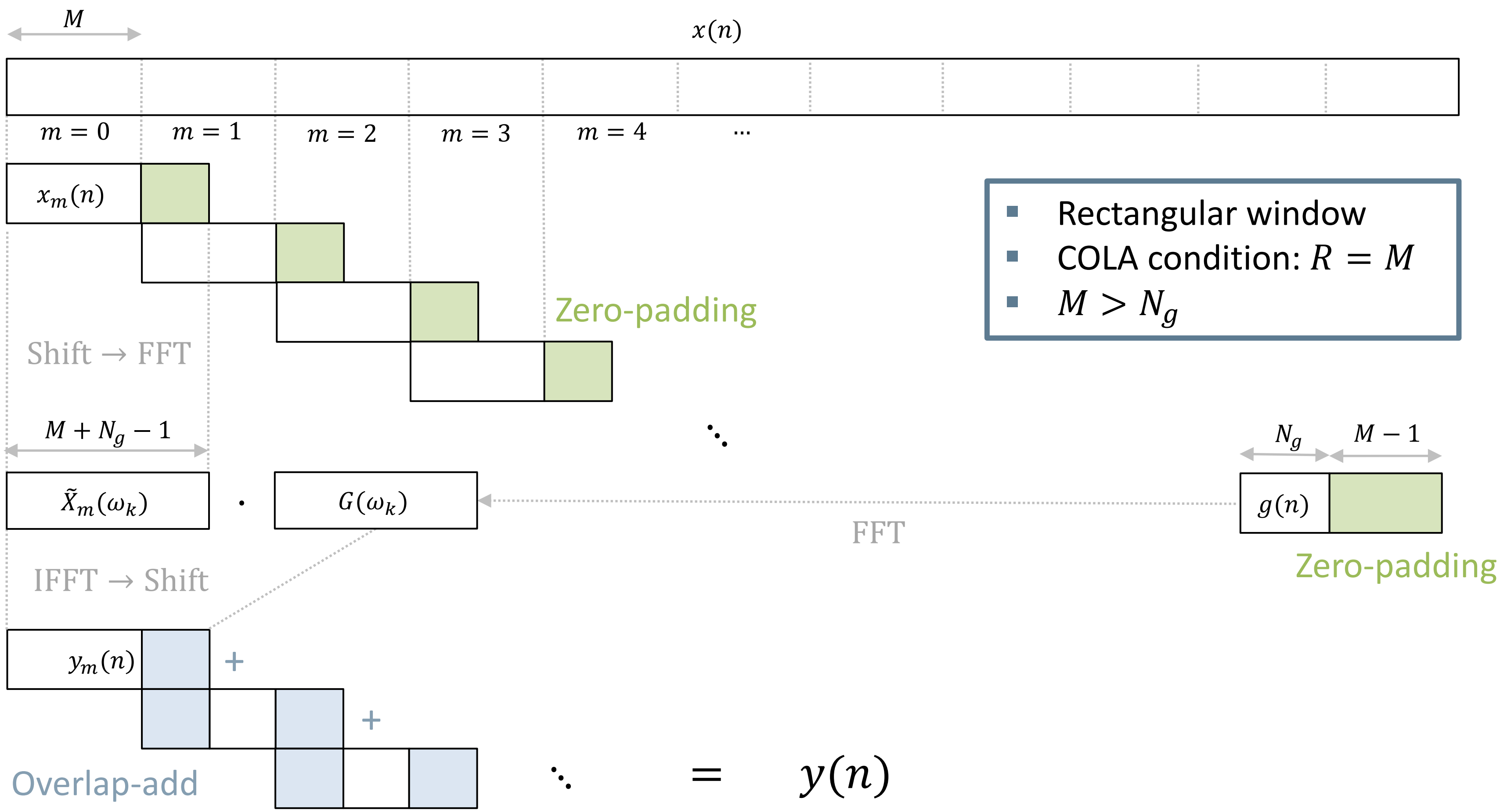
Filtering in Frequency Domain via OLA
Remarks

- The reconstruction is perfect as long as we have satisfied
  - The COLA condition on $w(n)$
  - The anti-aliasing (zero-padding) condition on $\tilde{x}_m(n)$ and $g(n)$

- Additionally, notice that $m \in \mathbb{Z}$ in $y(n) = \sum_{m=-\infty}^{\infty} y_m(n)$

- However, for casual signals with limited support, the values of m are usually limited to those for which the sliding window and the signal support do overlap $\rightarrow$ $m = 0, 1, 2, \ldots, \left\lceil \frac{N_x - M}{R} + 1 \right\rceil$

Filtering in Frequency Domain via OLA
Example

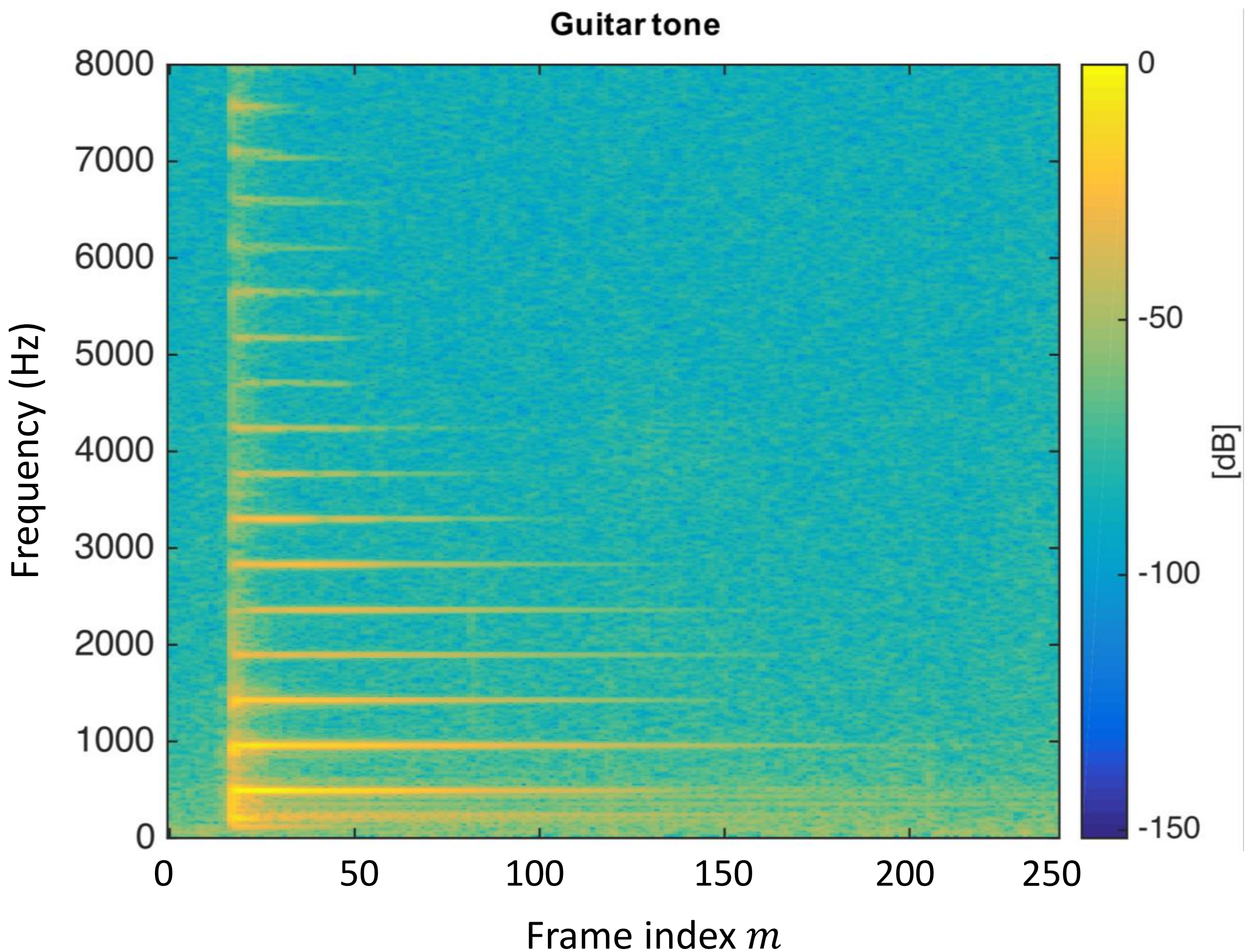


- Rectangular window
- COLA condition: $R = M$
- $M > N_g$

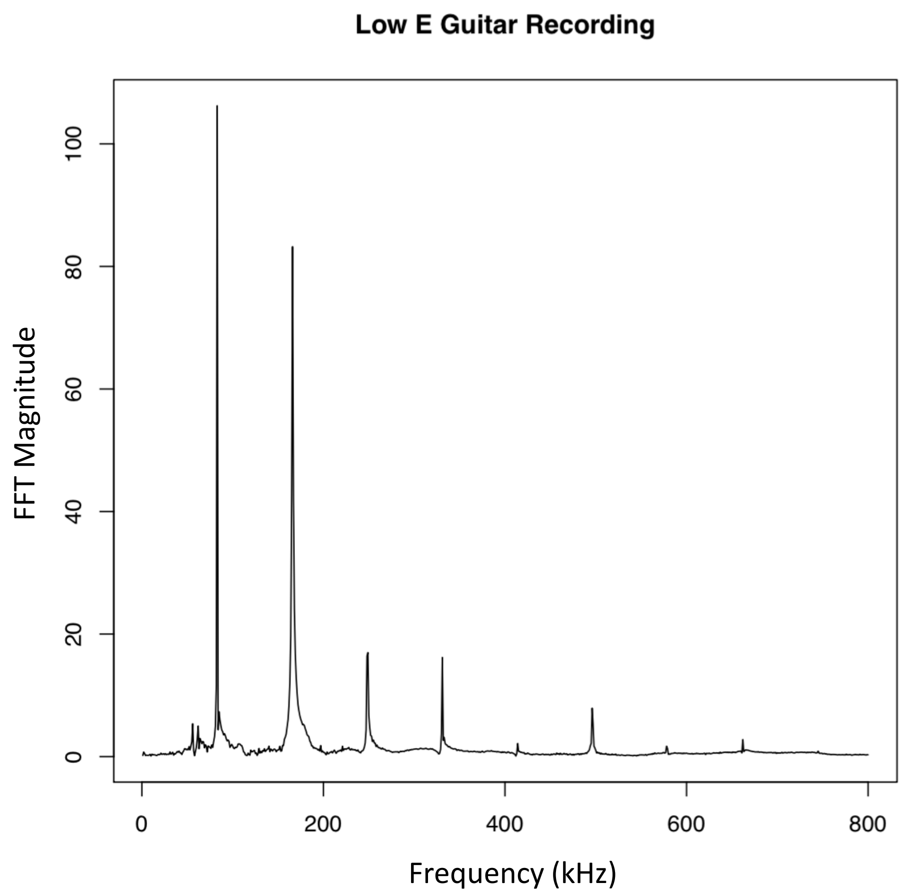
Short-time Analysis

- OLA offers a good insight into a broader family of techniques known as short-time analysis or time-frequency analysis
- These techniques study a signal in both the time and frequency domains *simultaneously*
- The main idea is to window a signal in time and then to compute the spectrum of each frame in order to analyze the temporal evolution of the frequency content or to perform frame-wise spectral processing
- One of the most basic forms of time-frequency analysis is the short-time Fourier transform (STFT)

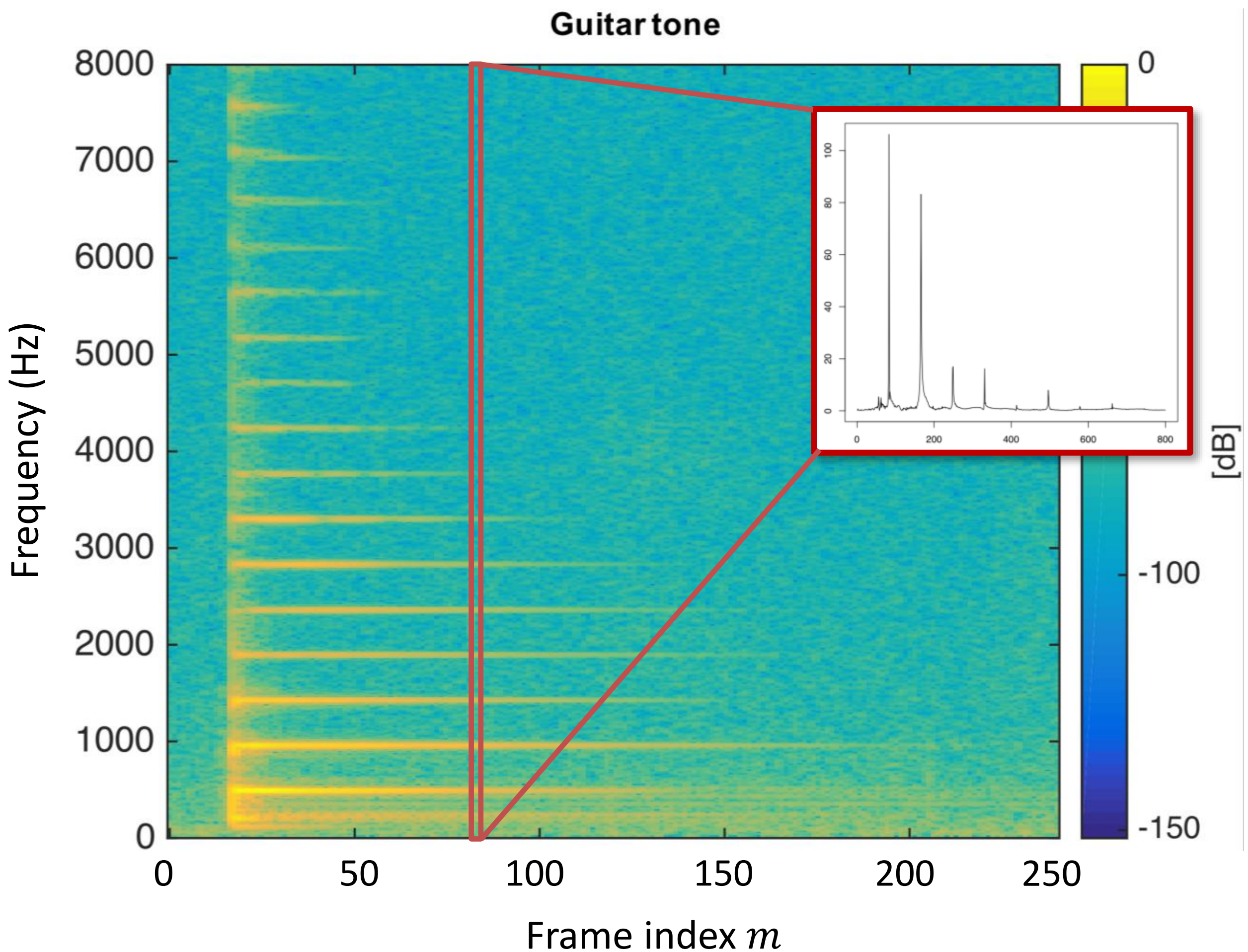
Short-Time Fourier Transform
Example: Guitar Tone



Short-Time Fourier Transform
Example: Guitar Tone



Short-Time Fourier Transform
Example: Guitar Tone



References

- J.O. Smith, Lecture notes: https://www-ccrma.standford.edu/~jos/sasp/
- In particular, the section on FIR filters: https://ccrma.stanford.edu/~jos/sasp/Audio_FIR_Filters.html
- Meinard Müller, “Fundamentals of Music Processing”, *Springer,* 2015, https://doi.org/10.1007/978-3-319-21945-5 - see Chapter 2 for a great review of Fourier Analysis.