The Discrete Fourier Transform

Jose Krause Perin

Stanford University

August 8, 2018

Last lecture

- The linear combiner is the basis of adaptive systems and adaptive filtering
- We use the mean square error (MSE) as the performance metric
- The Wiener solution is the optimal set of weights that minimizes the MSE
- The LMS algorithm is a simple way to train the adaptive filter to approximate the Wiener solution
- The LMS algorithm uses the instantaneous error to obtain an estimate of the gradient
- This estimate is very noisy, but on average it converges to the Wiener solution
- We adjust the adaption constant to control how fast the LMS algorithm converges and how noisy the solutions near the Wiener solution (excess noise and misadjustment)

Outline

The discrete Fourier transform (DFT)

Definition

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn}, \quad k = 0, \ldots, N-1 \qquad \text{(direct transform)}$$

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{j(2\pi/N)kn}, \quad n = 0, \ldots, N-1 \qquad \text{(inverse transform)}$$

Notation: k indexes frequency, while n indexes time.

Important observations

- The direct and inverse transforms are periodic with period N:

$$x[n] = x[n+N], \forall\ n \qquad \text{and} \qquad X[k] = X[k+N], \forall\ k$$

- The direct and inverse transform can be computed efficiently with complexity $\mathcal{O}(N \log N)$ using fast Fourier transform (FFT) algorithms.

The discrete Fourier transform (DFT)

Another common notation

Represent complex exponentials by $W_N \equiv e^{-j2\pi/N}$

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, \ldots, N-1 \qquad \text{(direct transform)}$$

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] W_N^{-kn}, \quad n = 0, \ldots, N-1 \qquad \text{(inverse transform)}$$

This is equivalent to the previous slide, but with a more compact notation.

Relation between the DFT and the DTFT

Suppose we calculate the DTFT of some signal $x[n]$:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \qquad \text{(DTFT of } x[n]\text{)}$$

Now we sample $X(e^{j\omega})$ at frequencies $\omega_k = 2\pi/Nk$, $k = 0, \ldots, N-1$:

$$X(e^{j\omega})\Big|_{\omega=2\pi/Nk} = \sum_{n=-\infty}^{\infty} x[n]e^{-j(2\pi/N)kn} = \sum_{n=0}^{N-1} x[n]e^{-j(2\pi/N)kn} = X[k]$$

(ony if $x[n]$ is time-limited with duration $< N$)

The sampled DTFT $X(e^{j\omega_k})$ is identical to the DFT $X[k]$ only if $x[n]$ is time-limited with duration $\leq N$.
Computing the inverse DFT of $X[k] = X(e^{j2\pi/Nk})$ i.e., samples of DTFT:

$$\tilde{x}[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(e^{j(2\pi/N)k})e^{j(2\pi/N)kn} = \sum_{r=-\infty}^{\infty} x[n - rN] \qquad (1)$$

Proof of (1):

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

$$= \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j(2\pi/N)kn} \qquad \text{(since } X(e^{j\omega})\Big|_{\omega=2\pi/Nk} = X[k]\text{)}$$

$$= \frac{1}{N}\sum_{k=0}^{N-1}\left(\sum_{m=0}^{N-1} x[m]e^{-j(2\pi/N)km}\right)e^{j(2\pi/N)kn} \qquad \text{(defintion of DFT)}$$

$$= \sum_{m=0}^{N-1} x[m]\underbrace{\left(\frac{1}{N}\sum_{k=0}^{N-1} e^{j(2\pi/N)k(n-m)}\right)}_{\text{inverse DFT of impulse train}} \qquad \text{(interchanging summations)}$$

$$= \sum_{m=0}^{N-1} x[m]\sum_{r=-\infty}^{\infty}\delta[n-m-rN] = \sum_{r=-\infty}^{\infty}\sum_{m=0}^{N-1} x[m]\delta[n-m-rN]$$

(only non-zero when $m = n - rN$)

$$= \sum_{r=-\infty}^{\infty} x[n-rN]$$

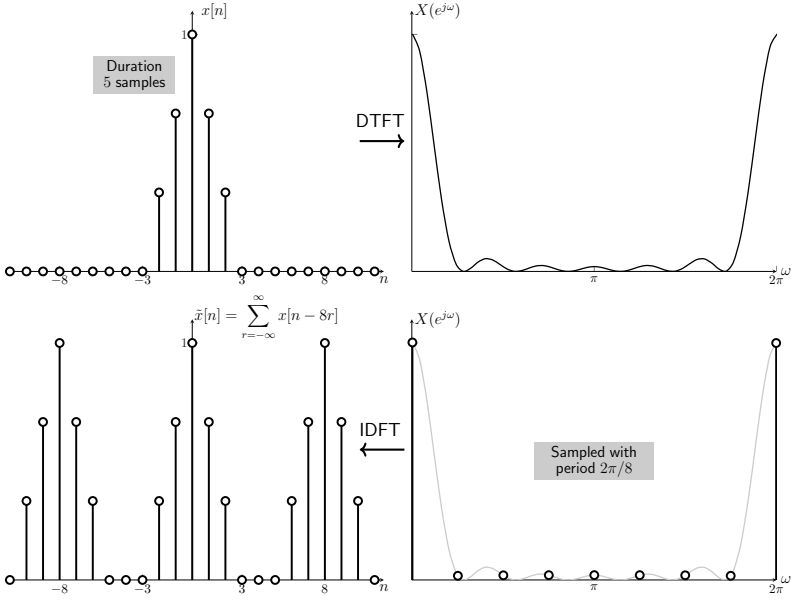
Relation between the DFT and the DTFT

Main conclusions from the previous derivation

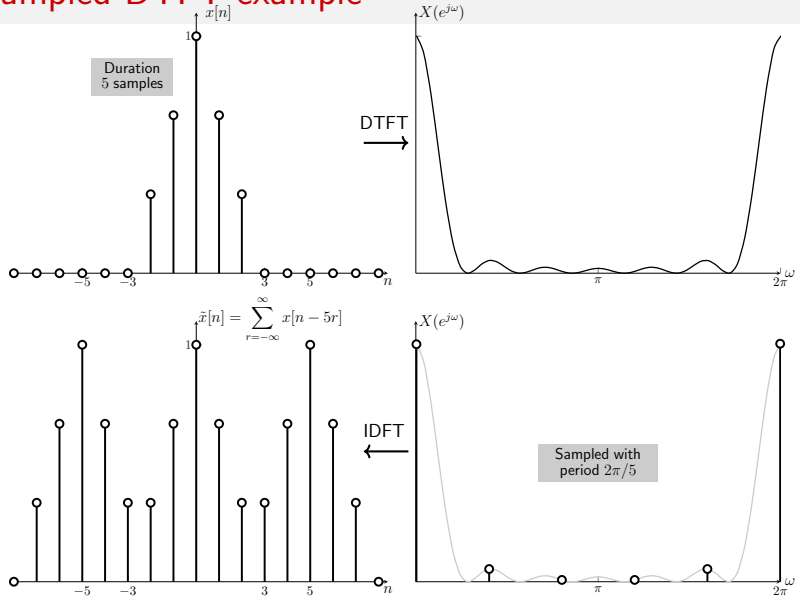
$$\tilde{x}[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(e^{j(2\pi/N)k})e^{j(2\pi/N)kn} = \sum_{r=-\infty}^{\infty} x[n-rN]$$

- The N-point DFT of $x[n]$ is only equal to the DTFT sampled with period $2\pi/N$ if $x[n]$ is time-limited with duration $\leq N$.
- The inverse DFT of the sampled DTFT produces a periodic signal $\tilde{x}[n]$ with period N, even though $x[n]$ is not periodic.
- $\tilde{x}[n]$ is called the **periodic extension** of $x[n]$
- For the N-point DFT and inverse DFT, all signals are periodic with period N

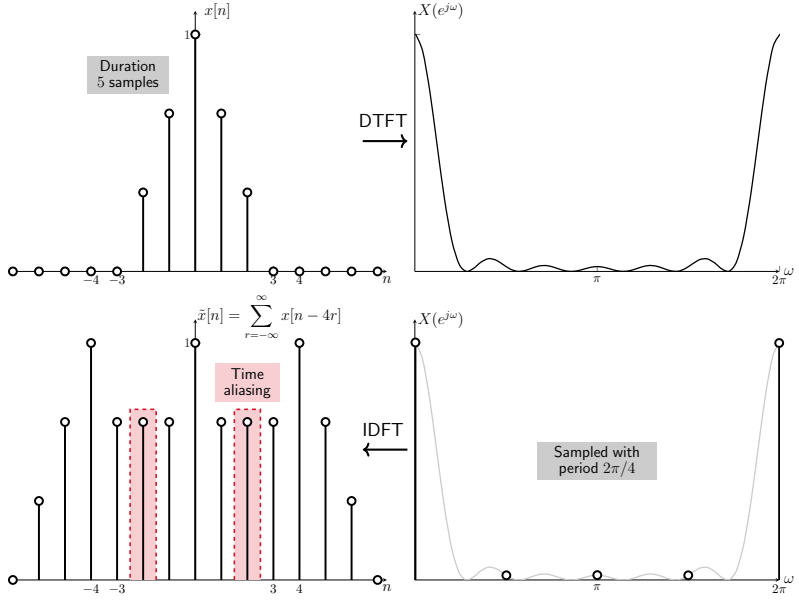
DFT as a sampled DTFT example



DFT as a sampled DTFT example



DFT as a sampled DTFT example



Relation between DFT and the DTFT

As another example, let's consider the infinitely-long signal

$$x[n] = \frac{1}{2}\text{sinc}\Big(\frac{n - N/2}{2}\Big) \Longleftrightarrow |X(e^{j\omega})| = \begin{cases} 1, & |\omega| \leq \pi/2 \\ 0, & \pi/2 < |\omega| \leq \pi \end{cases}$$

The DTFT of $x[n]$ is the ideal lowpass filter with cutoff frequency $\pi/2$.

Question: what about the N-point DFT of $x[n]$?
Instead of applying the direct transform, let's define a N-point truncated version of $x[n]$:

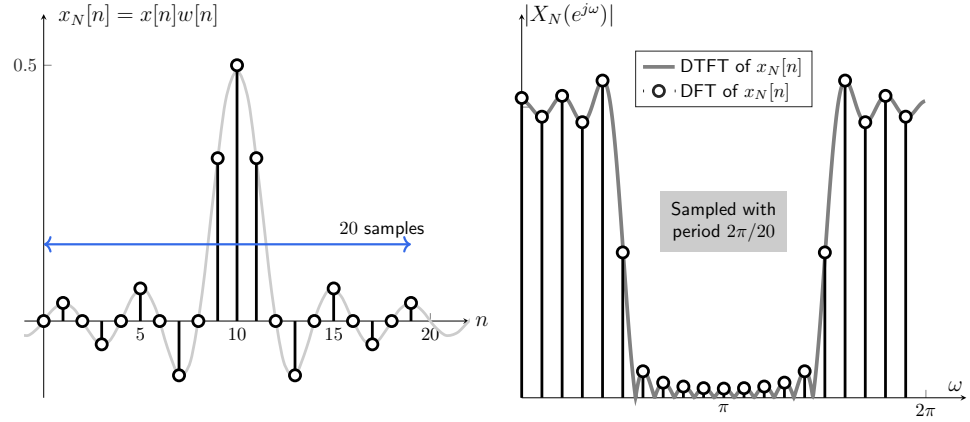
$$x_N[n] = x[n]w[n] \quad \text{where} \quad w[n] = \begin{cases} 1, & n = 0, \ldots, N-1 \\ 0, & \text{otherwise} \end{cases}$$

$w[n]$ is the **rectangular window**.
Note:

- The DFT of $x_N[n]$ is equal to the DFT of $x[n]$
- $x_N[n]$ is a time-limited sequence of duration N, hence the DFT of $x_N[n]$ is equal to the DTFT of $x_N[n]$ sampled with period $2\pi/N$.

Graphically

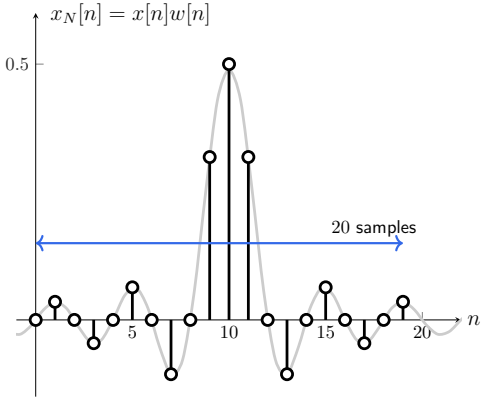
Consider the particular case of $N = 20$. The 20-point DFT of $x_N[n]$ is equal to the DTFT of $x_N[n]$ sampled with period $2\pi/20$.
The rectangular window causes significant *ringing*.

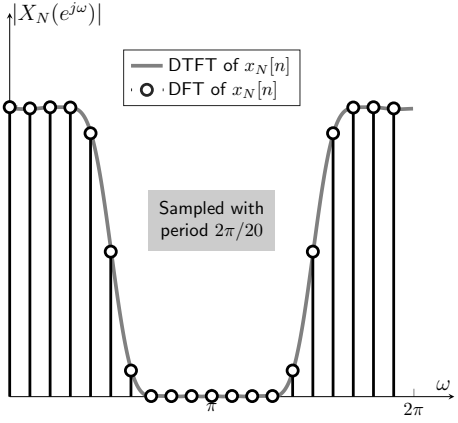


See code on Canvas/Files/Matlab/`DFT_sinc_example.m`

Graphically

Same scenario as before, but now $w[n]$ is the **Hamming window**. *Ringing* was reduced at the expense of slower roll-off.





See code on Canvas/Files/Matlab/`DFT_sinc_example.m`

Relation between DFT and the DTFT

One more example, now with a periodic signal

$$x[n] = \cos(n\pi/2) \Longleftrightarrow X(e^{j\omega}) = \pi\delta(\omega - \pi/2) + \pi\delta(\omega + \pi/2), \quad |\omega| \leq \pi$$

The DTFT of $x[n]$ (in the interval $[-\pi, \pi]$) is simply impulses at frequencies $\pm\pi/2$.

As in the previous example, define a N-point truncated version of $x[n]$:

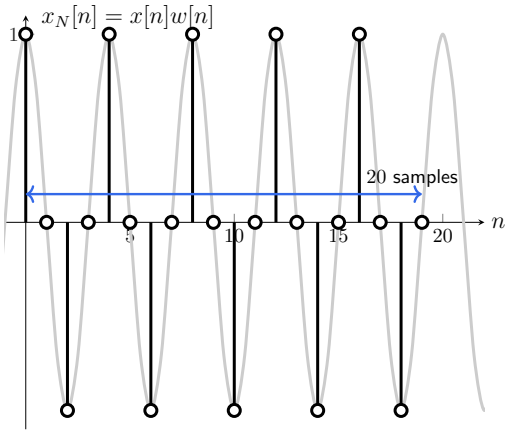
$$x_N[n] = x[n]w[n] \quad \text{where} \quad w[n] = \begin{cases} 1, & n = 0, \ldots, N-1 \\ 0, & \text{otherwise} \end{cases}$$

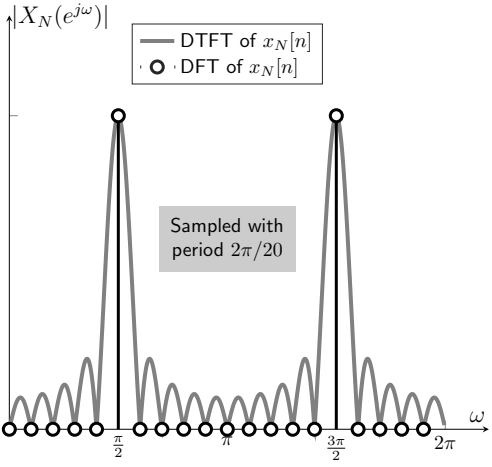
Once again, the DFT of $x_N[n]$ is equal to the DTFT of $x_N[n]$ sampled with period $2\pi/N$.

Graphically

Consider the particular case of $N = 20$.

$x_N[n]$ contains exactly 5 periods of $x[n]$. As a result, windowing had no effect on the DFT, as its samples fall at frequencies where the *windowed* DTFT is zero.

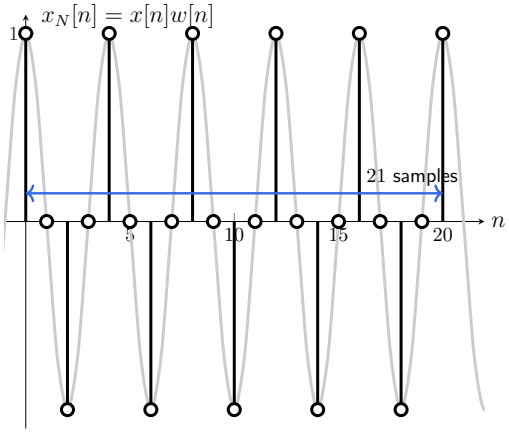


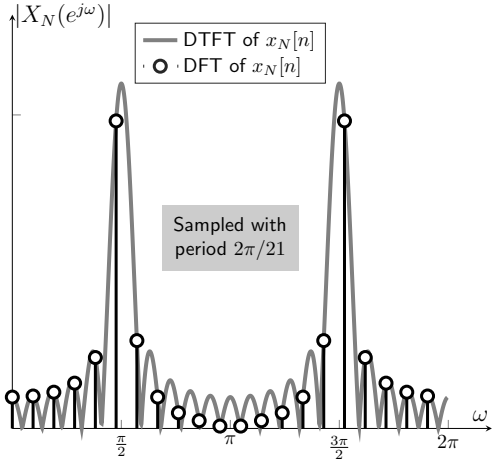


Graphically

Now $N = 21$.

Samples of DFT fall at frequencies where the *windowed* DTFT is non-zero.





Another interpretation

Let's look back at the sampling equation (1).
Sampling in the frequency domain results in signal replicas in time domain:

$$\tilde{x}[n] = \sum_{r=-\infty}^{\infty} x_N[n - rN]$$

- If N is an integer multiple of the period of $x[n]$, the replicas of $x_N[n]$ will perfectly reconstruct the periodic signal $x[n]$, i.e., $\tilde{x}[n] = x[n]$.
- If N is not an integer multiple of the period of $x[n]$, the replicas will lead to time-domain aliasing and consequently $\tilde{x}[n] \neq x[n]$.

Side note: Remember that $\cos(\omega_0 n)$ and $\sin(\omega_0 n)$ are only periodic if ω_0/π is rational. Hence, for ω_0 irrational we cannot find a value of N that would allow us to eliminate the effects of windowing.

Relation between DFT and the z-transform

For time-limited sequences of duration N:

DFT

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

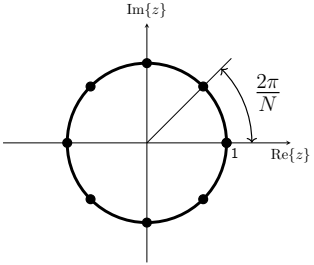
(direct transform)

z-transform

$$X(z) = \sum_{n=0}^{N-1} x[n]\, z^{-n}$$

(direct transform)

The DFT is equal to samples of the z-transform on the unit circle



Outline

DFT as a matrix-vector product

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, \ldots, N-1 \qquad \text{(direct transform)}$$

We can write the direct transform as a matrix-vector product:

$$X = Qx$$

$$\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} W_N^{0(1)} & W_N^{0(2)} & \cdots & W_N^{0(N-1)} \\ W_N^{1(1)} & W_N^{1(2)} & \cdots & W_N^{1(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ W_N^{(N-1)(1)} & W_N^{(N-1)(2)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

Similarly, for the inverse transform: $x = Q^H X$, where $Q^H = (Q^*)^T$ is the **Hermitian** of Q (conjugate transpose)

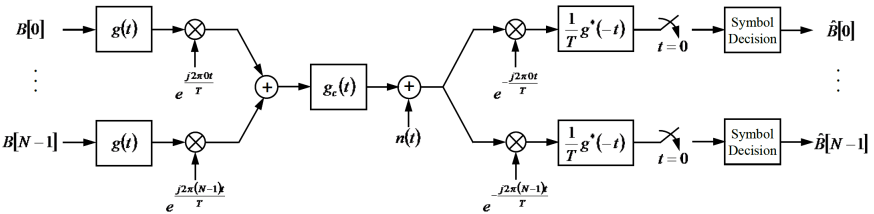
In Matlab: `>> Q = dftmtx(N)`.

Fast Fourier transform (FFT) algorithms

- Computing $x = Q^H X$ or $X = Qx$ has complexity $\mathcal{O}(N^2)$. That is, the number of computations grows quadratically for N large.
- **Fast Fourier transform (FFT)** refers to a collection of algorithms to compute the discrete Fourier transform (DFT) with complexity $\mathcal{O}(N \log N)$. That is, the number of computations grows at $N \log N < N^2$ for N large.
- This algorithm was known by Gauss back in 1800's, and it was rediscovered by Cooley and Tukey in 1965.

Application example: OFDM

Orthogonal frequency-division multiplexing (OFDM) is a widely used transmission technique in digital communications.
It is based on transmitting information on narrow-band and orthogonal sub-carriers

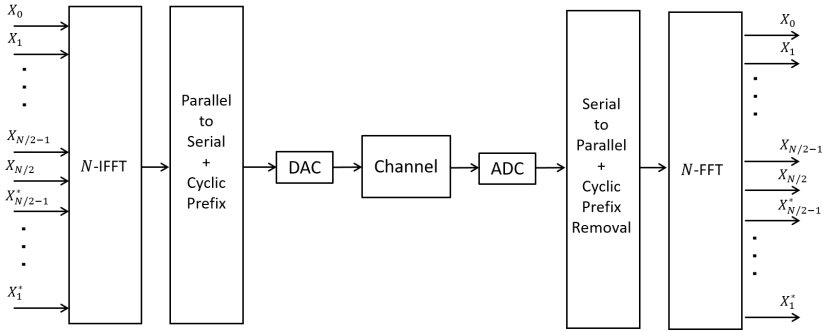


Analog implementation of OFDM. Diagram taken from EE 379 lecture notes.

OFDM was proposed in the 60's, but making such transmission in analog electronics was impossible, since the oscillators had to be perfectly synchronize to guarantee orthogonality

Application example: OFDM

With the advent of the FFT, it was realized that OFDM could be implemented using the IFFT/FFT:



FFT/IFFT-based implementation of OFDM.

OFDM is used in

- **Long-term evolution (LTE)**, IFFT/FFT size up to 2048
- **Wi-Fi**, IFFT/FFT size up to 256
- **Bluetooth**

FFT algorithms

FFT algorithms achieve dramatic reduction in computation by

1. Exploiting the **periodicity** and **symmetry** of complex exponentials W^{kn}:

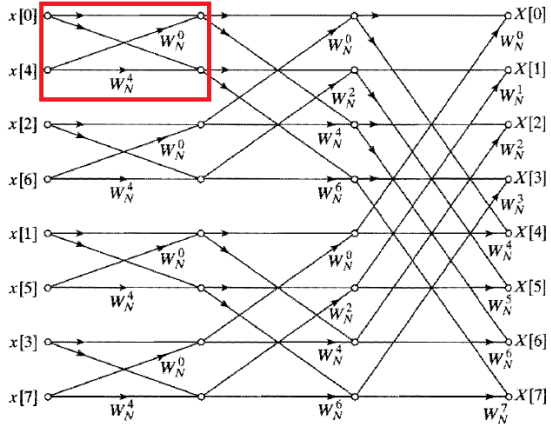
$$W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^* \qquad \text{(complex conjugate symmetry)}$$

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n} \qquad \text{(periodicity in } n \text{ and } k\text{)}$$

2. Decomposing the computation into successively smaller DFTs.
   - Decomposition of $x[n]$ into successively smaller subsequences is called **decimation in time**.
   - Decomposition of $X[k]$ into successively smaller subsequences is called **decimation in frequency**.
   - The flow graph of decimation-in-frequency decomposition can be obtained by **transposing** the flow graph of decimation-in-time decomposition, and vice-versa.

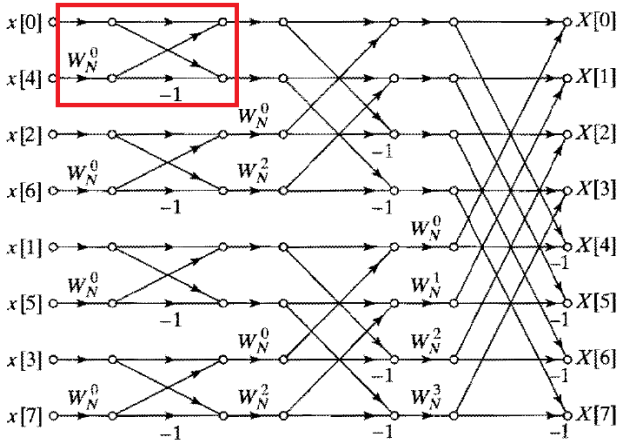
Decimation-in-time decomposition

Flow graph of complete decimation-in-time decomposition of an 8-point DFT computation. The red rectangle represents a 2-point DFT.



Decimation-in-time decomposition

Flow graph of complete decimation-in-time decomposition of an 8-point DFT computation. The red rectangle represents a 2-point DFT, now computed with just one multiplication by exploiting the periodicity and symmetry of W_N^{kn}.



FFT algorithms

FFT generalizations

- Radix R-algorithms

$$N = R^{\nu}$$

  DFTs are broken into factors of R

- Mixed-radix algorithms

$$N = N_1 N_2 \dots N_{\nu}$$

  DFTs are broken into factors of $\{N_1, N_2, \dots, N_{\nu}\}$.

- Prime-factor algorithms

$$N = N_1 N_2 \dots N_{\nu}$$

  DFTs are broken into prime factors $\{N_1, N_2, \dots, N_{\nu}\}$.

For more detailed information on FFT algorithms see Chapter 9 of the textbook.

For open-source implementation of FFT, see the Fastest Fourier Transform in the West (FFTW). FFTW is used in Matlab.

FFT and IFFT in Matlab

Useful commands:

Compute the N-point DFT of the vector `x`. If `N` is not passed, Matlab assumes `N = length(x)`.

```
>> X = fft(x, N)
```

Compute the N-point inverse DFT of the vector `X`. If `N` is not passed, Matlab assumes `N = length(X)`.

```
>> x = ifft(X, N)
```

To obtain the N-point DFT for frequencies in $[-\pi, \pi)$

```
>> fftshift(X)
```

To restore the N-point DFT for frequencies in $[0, 2\pi)$

```
>> ifftshift(X)
```

Outline

Properties of the DFT

- The DFT shares many properties with the DTFT and with the z-transform
- However, since the DFT is periodic in time and frequency, most properties will appear in a *circular* form
- For this reason, it is convenient to define a notation for **circular indexing**:

$$((n))_N \equiv n \text{ modulo } N$$

Examples: $X[0] = X[((7))_7]$, $X[1] = X[((6))_5]$, $X[8] = X[((17))_9]$.

- A complete list of properties is given in Table 8.2 of the textbook DTSP. This table is shown in the next slide
- We will cover in detail the circular time shift and circular convolution properties

Properties of the DFT

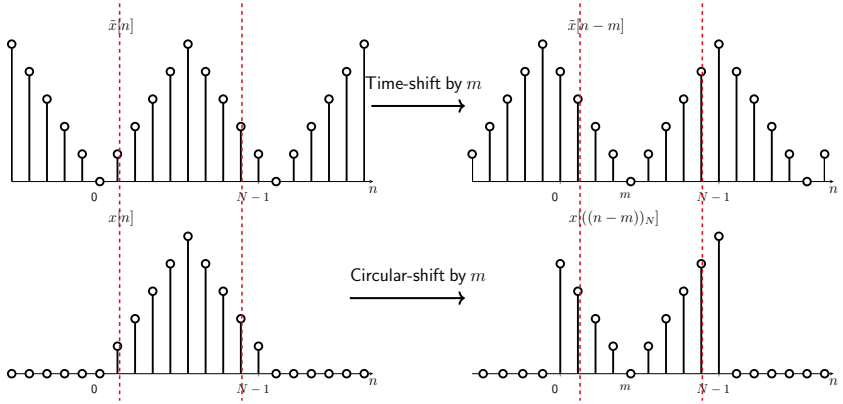
TABLE 8.2 SUMMARY OF PROPERTIES OF THE DFT

Finite-Length Sequence (Length N)	N-point DFT (Length N)
1. $x[n]$	$X[k]$
2. $x_1[n], x_2[n]$	$X_1[k], X_2[k]$
3. $ax_1[n] + bx_2[n]$	$aX_1[k] + bX_2[k]$
4. $X[n]$	$Nx[((-k))_N]$
5. $x[((n-m))_N]$	$W_N^{km} X[k]$
6. $W_N^{-\ell n} x[n]$	$X[((k-\ell))_N]$
7. $\sum_{m=0}^{N-1} x_1[m]x_2[((n-m))_N]$	$X_1[k]X_2[k]$
8. $x_1[n]x_2[n]$	$\frac{1}{N}\sum_{\ell=0}^{N-1} X_1[\ell]X_2[((k-\ell))_N]$
9. $x^*[n]$	$X^*[((-k))_N]$
10. $x^*[((-n))_N]$	$X^*[k]$
11. $\mathcal{R}e\{x[n]\}$	$X_{ep}[k] = \frac{1}{2}\{X[((k))_N] + X^*[((-k))_N]\}$
12. $j\mathcal{I}m\{x[n]\}$	$X_{op}[k] = \frac{1}{2}\{X[((k))_N] - X^*[((-k))_N]\}$
13. $x_{ep}[n] = \frac{1}{2}\{x[n] + x^*[((-n))_N]\}$	$\mathcal{R}e\{X[k]\}$
14. $x_{op}[n] = \frac{1}{2}\{x[n] - x^*[((-n))_N]\}$	$j\mathcal{I}m\{X[k]\}$
Properties 15–17 apply only when $x[n]$ is real.	
15. Symmetry properties	$\begin{cases} X[k] = X^*[((-k))_N] \\ \mathcal{R}e\{X[k]\} = \mathcal{R}e\{X[((-k))_N]\} \\ \mathcal{I}m\{X[k]\} = -\mathcal{I}m\{X[((-k))_N]\} \\ \lvert X[k]\rvert = \lvert X[((-k))_N]\rvert \\ \angle\lvert X[k]\rvert = -\angle\{X[((-k))_N]\} \end{cases}$
16. $x_{ep}[n] = \frac{1}{2}\{x[n] + x[((-n))_N]\}$	$\mathcal{R}e\{X[k]\}$
17. $x_{op}[n] = \frac{1}{2}\{x[n] - x[((-n))_N]\}$	$j\mathcal{I}m\{X[k]\}$

Circular time shift

$$x[((n-m))_N] \Longleftrightarrow W_N^{km} X[k] \qquad \text{(circular time shift)}$$

To understand circular time shift and other properties of the DFT, it is useful to work with the periodic extension of $x[n]$, $\tilde{x}[n]$



Circular convolution

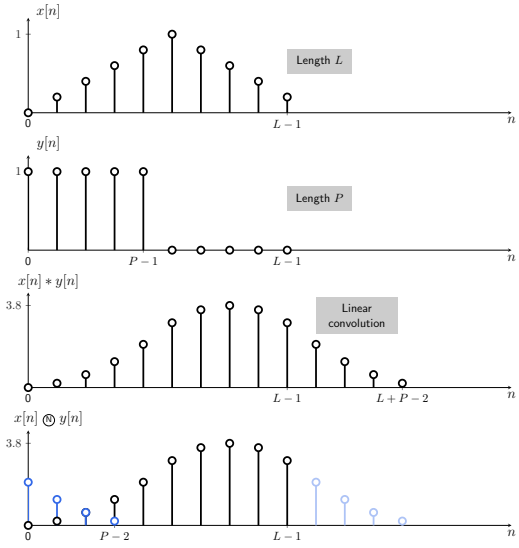
Product in the frequency domain means **circular convolution** in time domain

$$x[n] \textcircled{N} y[n] = \sum_{m=0}^{N-1} x[m]y[((n-m))_N] \Longleftrightarrow X[k]Y[k] \qquad \text{(circular convolution in time)}$$

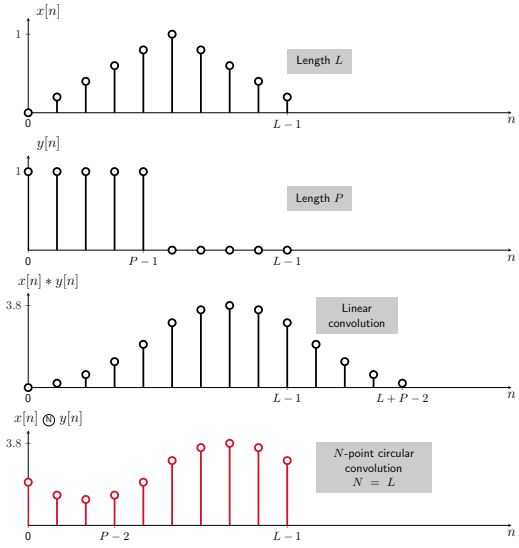
Similarly, product in time domain means **circular convolution** in frequency domain

$$x[n]y[n] \Longleftrightarrow X[k] \textcircled{N} Y[k] = \sum_{m=0}^{N-1} X[m]Y[((k-m))_N] \quad \text{(circular convolution in frequency)}$$

Understanding circular convolution



Understanding circular convolution

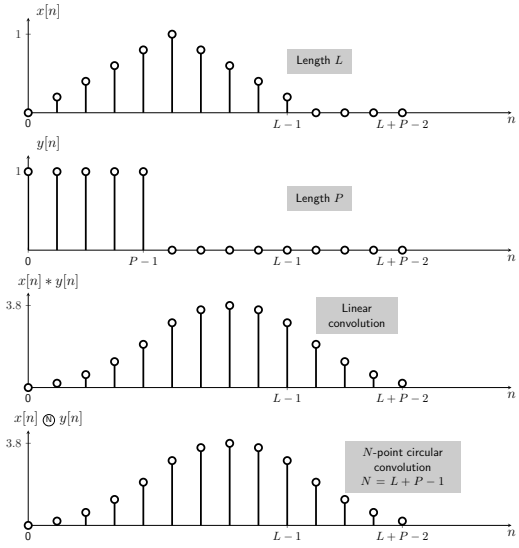


Understanding circular convolution

In this first example

- $x[n]$ has length L, while $y[n]$ has length P
- The result of linear convolution $x[n] * [y]$ has length $L + P - 1$
- For the N-point circular convolution, the $P - 1$ samples that fall beyond $N - 1$ are added to the beginning of the sequence
- Note that the N-point circular convolution and the linear convolution produce different results
- We can make the circular convolution equal to the linear convolution by zero-padding the sequences and performing the $(L + P - 1)$-point circular convolution

Understanding circular convolution



Linear convolution using circular convolution

By zero-padding and performing the $(L + P - 1)$-point circular convolution, we can calculate the linear convolution using the DFT

In Matlab:
The vector `x` has length L while `y` has length P

```
>> conv(x, y, 'full')
```
(linear convolution)

This will produce an output of length $N = L + P - 1$. The linear convolution has complexity $\mathcal{O}(N^2)$

```
>> ifft(fft(x, N).*fft(y, N))
```
(circular convolution)

The parameter `N` tells Matlab to zero-pad vectors `x` and `y` and compute the N-point DFT. The FFT/IFFT has complexity $\mathcal{O}(N \log N)$

Conclusion: for N large, it is more efficient to compute the linear convolution through the DFT using FFT algorithm.

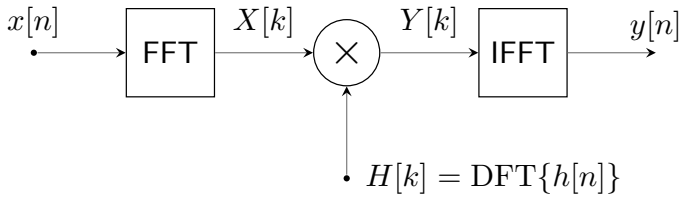
Linear convolution using circular convolution

Recall that for FIR filters, filtering is essentially a linear convolution between the input and the filter coefficients:

$$y[n] = x[n] * h[n] = \sum_{m=0}^{M} h[m]x[n-m]$$

where M is the filter order.

We can implement this filter using the DFT:



By using **block convolution** we process the samples in batches.

1. Overlap-add method
2. Overlap-save method

Overlap-add method

In the **overlap-add method** the incoming signal $x[n]$ is broken down into several non-overlapping segments $x_r[n]$, each of length L:

$$x_r[n] = \begin{cases} x[n + rL], & 0 \leq n \leq L - 1 \\ 0, & \text{otherwise} \end{cases}$$

For each segment, we compute the N-point circular convolution of $x_r[n]$ and the filter impulse response $h[n]$:

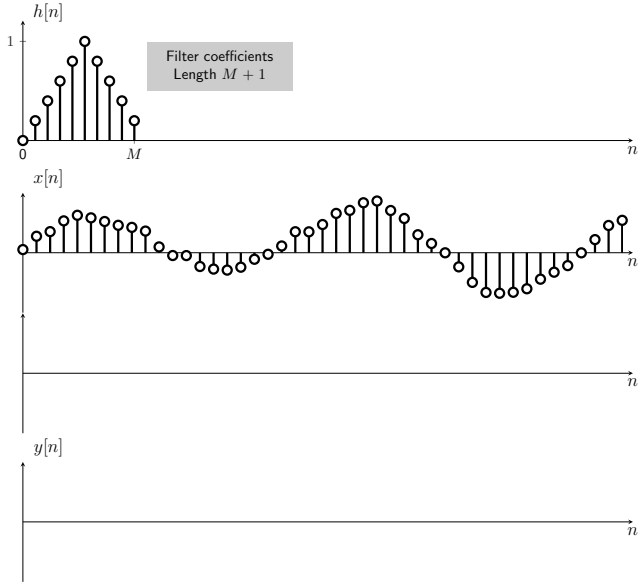
$$y_r[n] = h[n] \textcircled{N} x_r[n] = h[n] * x_r[n]$$

where $N \geq L + M$ so that $y_r[n]$ is equivalent to the linear convolution of $x_r[n]$ and $h[n]$. Note that M is the filter order. Hence, the filter has $M + 1$ coefficients.
Finally, the output is computed by adding the filtered segments

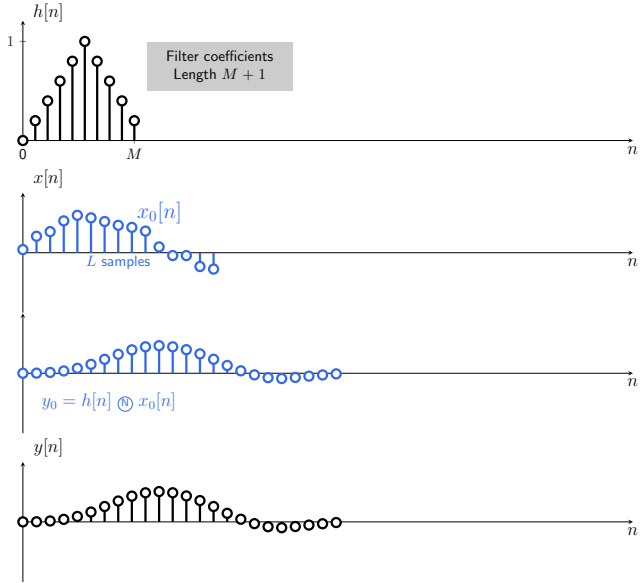
$$y_r[n] = \sum_{r=0}^{\infty} y_r[n - rL]$$

The sequences $y_r[n - rL]$ overlap by M samples.

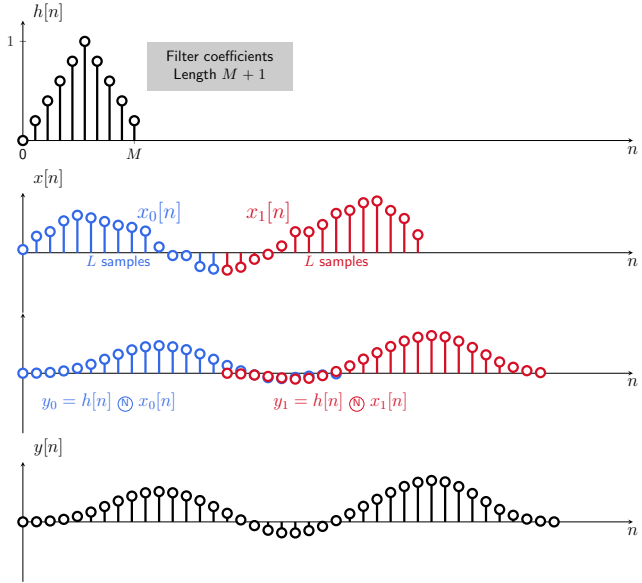
Overlap-add method



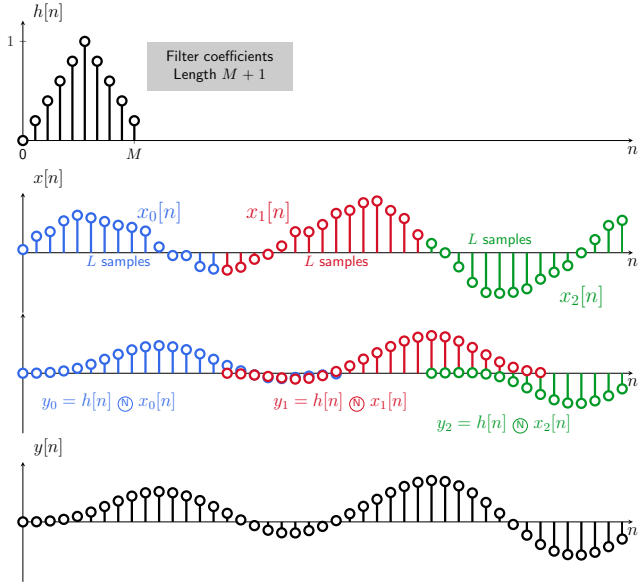
Overlap-add method



Overlap-add method



Overlap-add method



Overlap-save method

In the **overlap-save method** the incoming signal $x[n]$ is decomposed into several overlapping segments $x_r[n]$, each of length L:

$$x_r[n] = \begin{cases} x[n + r(L-M) - M], & 0 \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases}$$

The sequences $x_r[n]$ overlap by M samples. Then, we compute the L-point circular convolution of $h[n]$ and $x_r[n]$:

$$y_r[n] = h[n] \textcircled{L} x_r[n]$$

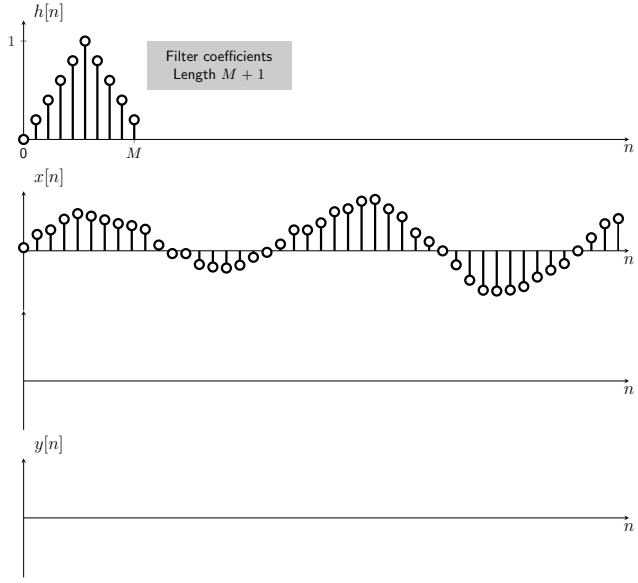
The first M samples of $y_r[n]$ are unusable, since they are not equal to the linear convolution $h[n] * x_r[n]$. Hence, we define the usable part of $y_r[n]$:

$$y_{r,u}[n] = \begin{cases} y_r[n], & M \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases}$$

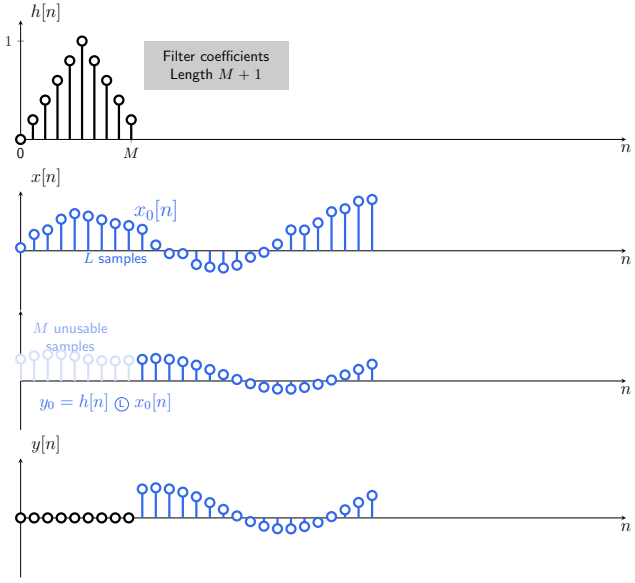
Finally, the output is computed by adding the usable parts of the filtered segments

$$y_r[n] = \sum_{r=0}^{\infty} y_{r,u}[n - r(L-M) + M]$$

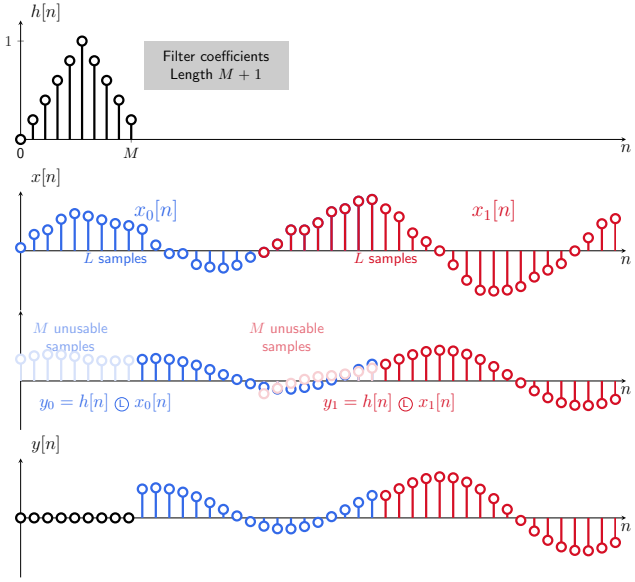
Overlap-save method



Overlap-save method



Overlap-save method



Summary

- Sampling the DTFT in frequency domain results in signal replicas in time domain
- The N-point DFT of $x[n]$ is equal to the DTFT of $x[n]$ sampled with period $2\pi/N$, only if $x[n]$ is time-limited with duration $\leq N$
- For sequences longer than N, the N-point DFT is equal to the samples of the windowed DTFT
- Fast Fourier transform (FFT) algorithms compute the DFT with complexity $\mathcal{O}(N \log_2 N)$
- We can use DFT to perform linear convolution (filtering) efficiently using block convolution
- In the overlap-add method, blocks are non-overlapping and the result of circular convolution of each block is added to produce the output signal
- In the overlap-save method, blocks do overlap and we have to discard samples that are unusable due to the circular convolution not being equal to the linear convolution at all points