Overlap-Save and Overlap-Add for Real-time Filtering

Professor Deepa Kundur

University of Toronto

Overlap-Save and Overlap-Add for Real-time Processing

Reference:

Section 7.3 of

John G. Proakis and Dimitris G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th edition, 2007.

The Discrete Fourier Transform Pair

- DFT and inverse-DFT (IDFT):

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi k \frac{n}{N}}, \quad k = 0, 1, \ldots, N-1$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi k \frac{n}{N}}, \quad n = 0, 1, \ldots, N-1$$

Important DFT Properties

Property	Time Domain	Frequency Domain
Notation:	$x(n)$	$X(k)$
Periodicity:	$x(n) = x(n+N)$	$X(k) = X(k+N)$
Linearity:	$a_1 x_1(n) + a_2 x_2(n)$	$a_1 X_1(k) + a_2 X_2(k)$
Time reversal	$x(N-n)$	$X(N-k)$
Circular time shift:	$x((n-l))_N$	$X(k)e^{-j2\pi kl/N}$
Circular frequency shift:	$x(n)e^{j2\pi ln/N}$	$X((k-l))_N$
Complex conjugate:	$x^*(n)$	$X^*(N-k)$
Circular convolution:	$x_1(n) \otimes x_2(n)$	$X_1(k)X_2(k)$
Multiplication:	$x_1(n)x_2(n)$	$\frac{1}{N} X_1(k) \otimes X_2(k)$
Parseval's theorem:	$\sum_{n=0}^{N-1} x(n)y^*(n)$	$\frac{1}{N}\sum_{k=0}^{N-1} X(k)Y^*(k)$

DTFT Theorems and Properties

Property	Time Domain	Frequency Domain
Notation:	$x(n)$	$X(\omega)$
	$x_1(n)$	$X_1(\omega)$
	$x_2(n)$	$X_1(\omega)$
Linearity:	$a_1x_1(n) + a_2x_2(n)$	$a_1X_1(\omega) + a_2X_2(\omega)$
Time shifting:	$x(n-k)$	$e^{-j\omega k}X(\omega)$
Time reversal	$x(-n)$	$X(-\omega)$
Convolution:	$x_1(n) * x_2(n)$	$X_1(\omega)X_2(\omega)$
Correlation:	$r_{x_1x_2}(l) = x_1(l) * x_2(-l)$	$S_{x_1x_2}(\omega) = X_1(\omega)X_2(-\omega)$
		$= X_1(\omega)X_2^*(\omega)$ [if $x_2(n)$ real]
Wiener-Khintchine:	$r_{xx}(l) = x(l) * x(-l)$	$S_{xx}(\omega) = \lvert X(\omega)\rvert^2$

Circular and Linear Convolution

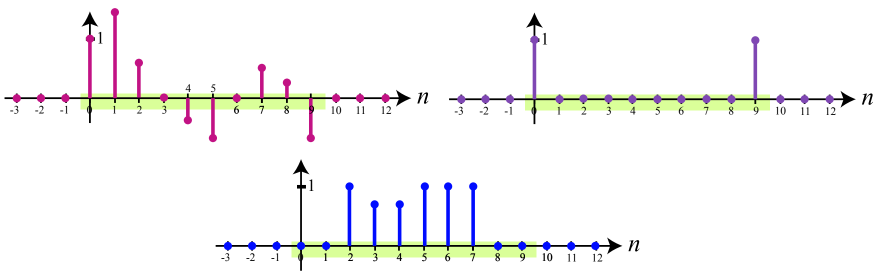
$$x_1(n) \otimes x_2(n) \overset{\mathcal{F}}{\longleftrightarrow} X_1(k)X_2(k)$$
$$x_1(n) * x_2(n) \overset{\mathcal{F}}{\longleftrightarrow} X_1(\omega)X_2(\omega)$$

Q: What is circular convolution?

Circular Convolution

Assume: $x_1(n)$ and $x_2(n)$ have support $n = 0, 1, \dots, N-1$.

Examples: $N = 10$ and support: $n = 0, 1, \dots, 9$



Circular Convolution

Assume: $x_1(n)$ and $x_2(n)$ have support $n = 0, 1, \dots, N-1$.

$$\begin{aligned} x_1(n) \otimes x_2(n) &= \sum_{k=0}^{N-1} x_1(k)x_2((n-k))_N \\ &= \sum_{k=0}^{N-1} x_2(k)x_1((n-k))_N \end{aligned}$$

where $(n)_N = n \mod N =$ remainder of n/N.

Modulo Indices and Periodic Repetition

$$\boxed{(n)_N = n \mod N = \text{remainder of } n/N}$$

Example: $N = 4$

n	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
$(n)_4$	0	1	2	3	0	1	2	3	0	1	2	3	0

$$\frac{n}{N} = \text{integer} + \frac{\boxed{\text{nonneg integer} < N}}{N}$$

$$\frac{5}{4} = 1 + \frac{1}{4} \qquad \frac{-2}{4} = -1 + \frac{2}{4}$$

Overlap-Save and Overlap-Add for Real-time Processing Circular and Linear Convolution

Modulo Indices and Periodic Repetition

$$\boxed{(n)_N = n \mod N = \text{remainder of } n/N}$$

Example: $N = 4$

n	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
$(n)_4$	0	1	2	3	0	1	2	3	0	1	2	3	0

$x((n))_4$ will be periodic with period 4. The repeated pattern will be consist of: $\{x(0), x(1), x(2), x(3)\}$.

Thus, $x((n))_N$ is a periodic signal comprised of the following repeating pattern: $\{x(0), x(1), \cdots x(N-2), x(N-1)\}$.

Overlap During Periodic Repetition

A periodic repetition makes an aperiodic signal $x(n)$, periodic to produce $x_p(n)$.

$$x_p(n) = \sum_{l=-\infty}^{\infty} x(n - lN)$$

Overlap-Save and Overlap-Add for Real-time Processing Circular and Linear Convolution

Overlap During Periodic Repetition

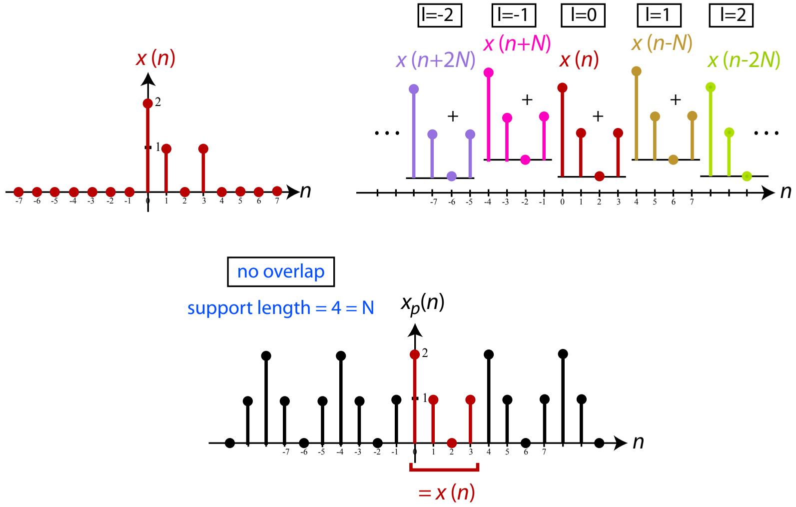
A periodic repetition makes an aperiodic signal $x(n)$, periodic to produce $x_p(n)$.

There are two important parameters:

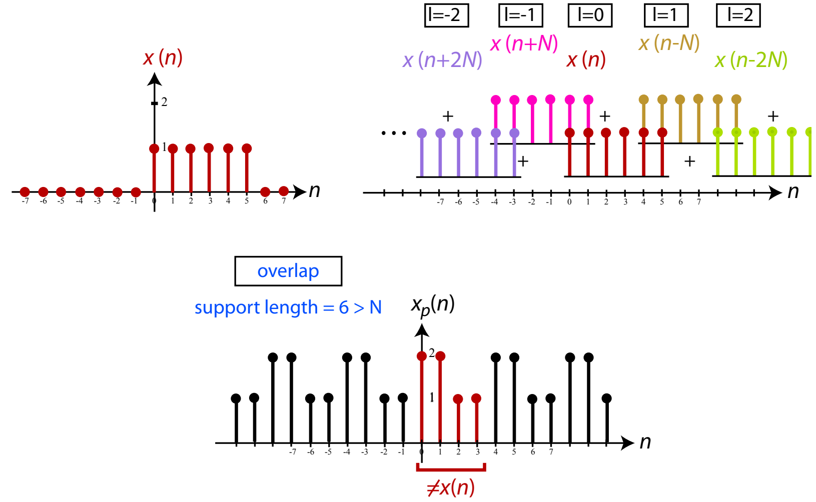
1. smallest support length of the signal $x(n)$
2. period N used for repetition that determines the period of $x_p(n)$

- smallest support length $>$ period of repetition
  - there will be $\boxed{\text{overlap}}$
- smallest support length $\leq$ period of repetition
  - there will be $\boxed{\text{no overlap}}$
    $\Rightarrow x(n)$ can be recovered from $x_p(n)$

Periodic Repetition: Example $N = 4$



Period Repetition: Example $N = 4$



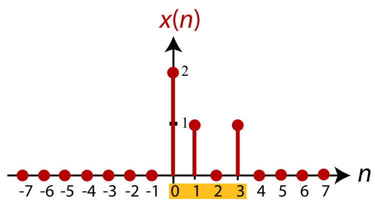
Modulo Indices and the Periodic Repetition

Assume: $x(n)$ has support $n = 0, 1, \ldots, N - 1$.

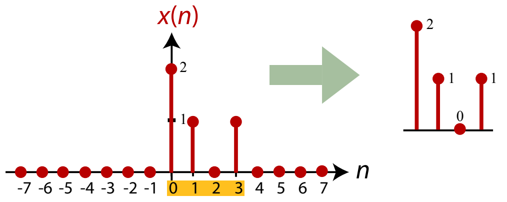
$$x((n))_N = x(n \bmod N) = x_p(n) = \sum_{l=-\infty}^{\infty} x(n - lN)$$

Note: Because the support size and period size are the same, there is no overlap when taking the periodic repetition $x((n))_N$.

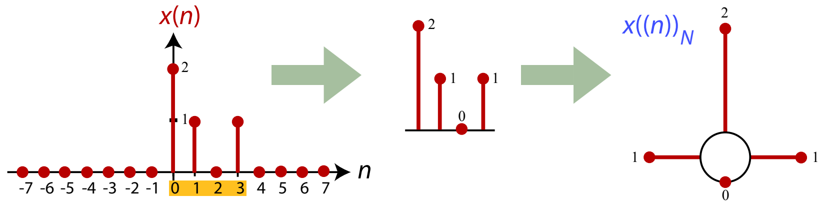
Modulo Indices and the Periodic Repetition



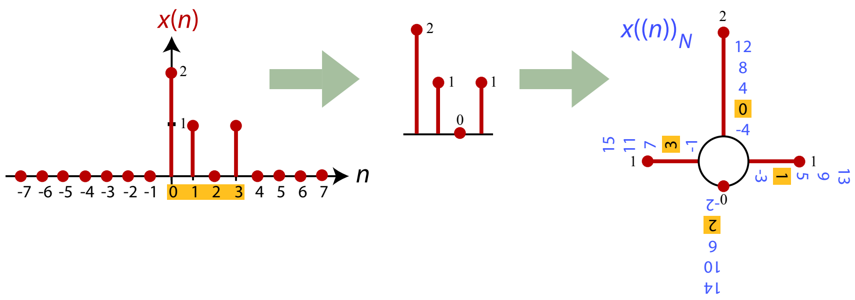
Modulo Indices and the Periodic Repetition



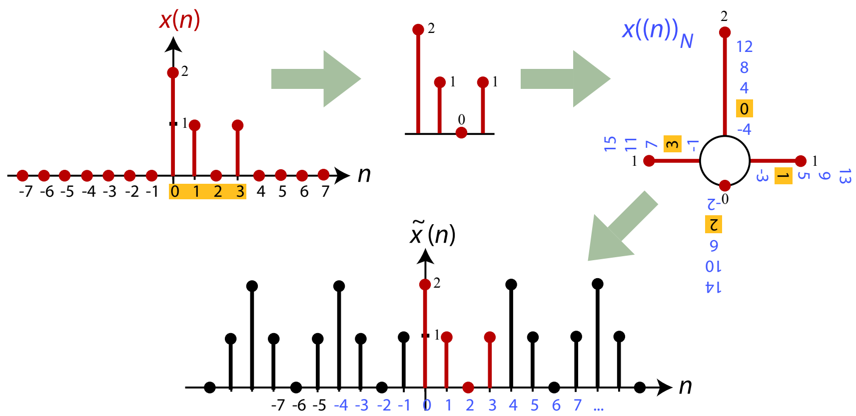
Modulo Indices and the Periodic Repetition



Modulo Indices and the Periodic Repetition



Modulo Indices and the Periodic Repetition



Therefore $x((n))_N = x_p(n)$.

Circular Convolution: One Interpretation

Assume: $x_1(n)$ and $x_2(n)$ have support $n = 0, 1, \ldots, N-1$.

To compute $\sum_{k=0}^{N-1} x_1(k)x_2((n-k))_N$ (or $\sum_{k=0}^{N-1} x_2(k)x_1((n-k))_N$):

1. Take the periodic repetition of $x_2(n)$ with period N:

$$x_{2p}(n) = \sum_{l=-\infty}^{\infty} x_2(n - lN)$$

2. Conduct a standard linear convolution of $x_1(n)$ and $x_{2p}(n)$ for $n = 0, 1, \ldots, N-1$:

$$x_1(n) \otimes x_2(n) = x_1(n) * x_{2p}(n) = \sum_{k=-\infty}^{\infty} x_1(k)x_{2p}(n-k) = \sum_{k=0}^{N-1} x_1(k)x_{2p}(n-k)$$

Note: $x_1(n) \otimes x_2(n) = 0$ for $n < 0$ and $n \geq N$.

Circular Convolution: One Interpretation

$$\sum_{k=0}^{N-1} x_1(k) \boxed{x_2((n-k))_N} = \sum_{k=0}^{N-1} x_1(k) \boxed{x_{2p}(n-k)}$$

... which makes sense, since $x((n))_N = x_p(n)$.

Circular Convolution: Another Interpretation

Assume: $x_1(n)$ and $x_2(n)$ have support $n = 0, 1, \ldots, N-1$.

To compute $\sum_{k=0}^{N-1} x_1(k)x_2((n-k))_N$ (or $\sum_{k=0}^{N-1} x_2(k)x_1((n-k))_N$):

1. Conduct a linear convolution of $x_1(n)$ and $x_2(n)$ for all n:

$$x_L(n) = x_1(n) * x_2(n) = \sum_{k=-\infty}^{\infty} x_1(k)x_2(n-k) = \sum_{k=0}^{N-1} x_1(k)x_2(n-k)$$

2. Compute the periodic repetition of $x_L(n)$ and window the result for $n = 0, 1, \ldots, N-1$:

$$x_1(n) \otimes x_2(n) = \sum_{l=-\infty}^{\infty} x_L(n - lN), \qquad n = 0, 1, \ldots, N-1$$

Using DFT for Linear Convolution

Therefore, circular convolution and linear convolution are related as follows:

$$x_C(n) = x_1(n) \otimes x_2(n) = \sum_{l=-\infty}^{\infty} x_L(n - lN)$$

for $n = 0, 1, \ldots, N-1$

Q: When can one recover $x_L(n)$ from $x_C(n)$?
When can one use the DFT to compute linear convolution?

A: When there is no overlap in the periodic repetition of $x_L(n)$.
When support length of $x_L(n) \leq N$.

Using DFT for Linear Convolution

Let $x(n)$ have support $n = 0, 1, \ldots, L-1$.
Let $h(n)$ have support $n = 0, 1, \ldots, M-1$.

We can set $N \geq L + M - 1$ and zero pad $x(n)$ and $h(n)$ to have support $n = 0, 1, \ldots, N-1$.

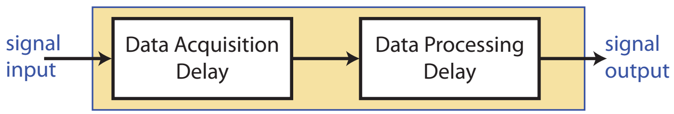
1. Take N-DFT of $x(n)$ to give $X(k)$, $k = 0, 1, \ldots, N-1$.
2. Take N-DFT of $h(n)$ to give $H(k)$, $k = 0, 1, \ldots, N-1$.
3. Multiply: $Y(k) = X(k) \cdot H(k)$, $k = 0, 1, \ldots, N-1$.
4. Take N-IDFT of $Y(k)$ to give $y(n)$, $n = 0, 1, \ldots, N-1$.

Filtering of Long Data Sequences

- The input signal $x(n)$ is often very long especially in real-time signal monitoring applications.
- For linear filtering via the DFT, for example, the signal must be limited size due to memory requirements.

Filtering of Long Data Sequences

- All N-input samples are required simultaneously by the DFT operator.
- If N is too large as for long data sequences, then there is a significant delay in processing that precludes real-time processing.



Filtering of Long Data Sequences

- Strategy:
  1. Segment the input signal into fixed-size blocks prior to processing.
  2. Compute DFT-based linear filtering of each block separately.
  3. Fit the output blocks together in such a way that the overall output is equivalent to the linear filtering of $x(n)$ directly.
- Main advantage: samples of the output $y(n) = h(n) * x(n)$ will be available real-time on a block-by-block basis.

Filtering of Long Data Sequences

- Goal: FIR filtering: $y(n) = x(n) * h(n)$
- Two approaches to real-time linear filtering of long inputs:
  - Overlap-Add Method
  - Overlap-Save Method
- Assumptions:
  - FIR filter $h(n)$ length $= M$
  - Block length $= L \gg M$

Overlap-Add Method

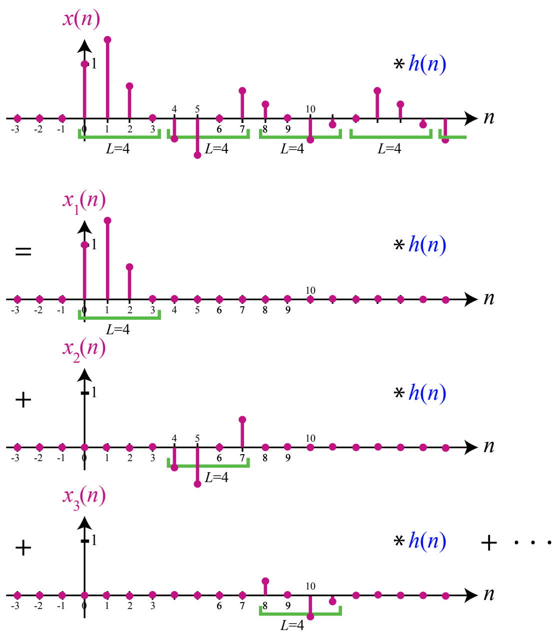
Overlap-Add Method

Overlap-Add Method

Deals with the following signal processing principles:

- The linear convolution of a discrete-time signal of length L and a discrete-time signal of length M produces a discrete-time convolved result of length $L + M - 1$.
- Addititvity:

$$(x_1(n)+x_2(n)) * h(n) = x_1(n) * h(n)+x_2(n) * h(n)$$



Input $x(n)$ is divided into non-overlapping blocks $x_m(n)$ each of length L.

Each input block $x_m(n)$ is individually filtered as it is received to produce the output block $y_m(n)$.

Overlap-Add Filtering Stage

- makes use of the N-DFT and N-IDFT where: $N = L + M - 1$
  - Thus, zero-padding of $x(n)$ and $h(n)$ that are of length $L, M < N$ is required.
  - The actual implementation of the DFT/IDFT will use the fast Fourier Transform (FFT) for computational simplicity.

Using DFT for Linear Convolution

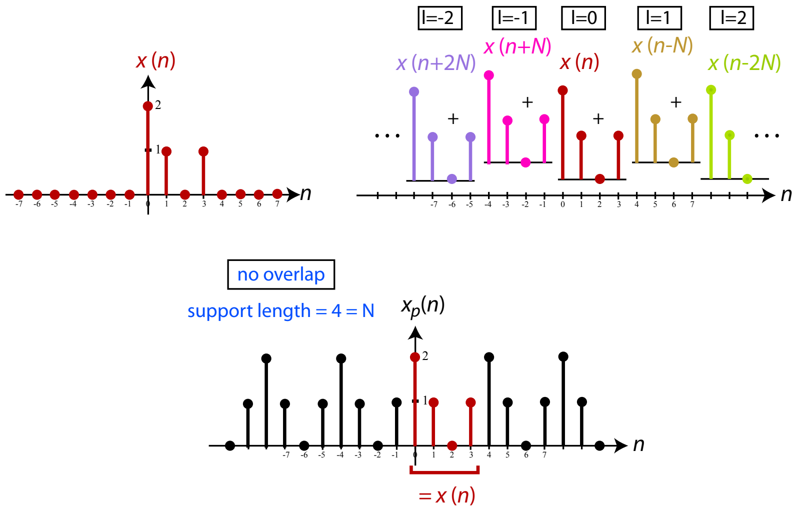
Let $x_m(n)$ have support $n = 0, 1, \ldots, L-1$.
Let $h(n)$ have support $n = 0, 1, \ldots, M-1$.

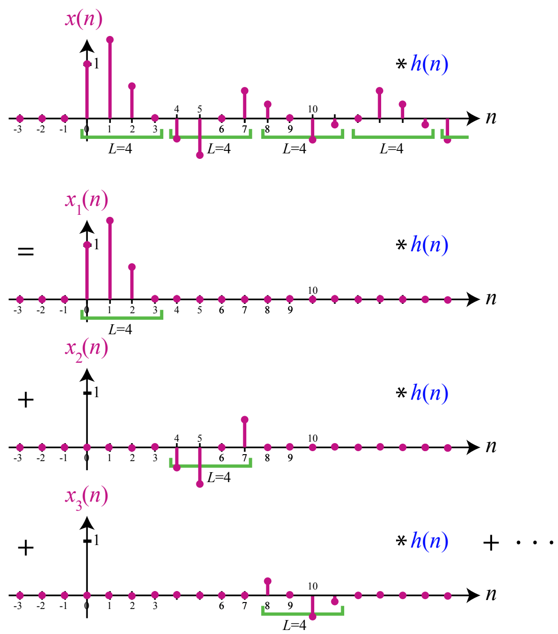
We set $N \geq L + M - 1$ (the length of the linear convolution result) and zero pad $x_m(n)$ and $h(n)$ to have support $n = 0, 1, \ldots, N-1$.

1. Take N-DFT of $x_m(n)$ to give $X_m(k)$, $k = 0, 1, \ldots, N-1$.
2. Take N-DFT of $h(n)$ to give $H(k)$, $k = 0, 1, \ldots, N-1$.
3. Multiply: $Y_m(k) = X_m(k) \cdot H(k)$, $k = 0, 1, \ldots, N-1$.
4. Take N-IDFT of $Y_m(k)$ to give $y_m(n)$, $n = 0, 1, \ldots, N-1$.

Linear Convolution via the DFT

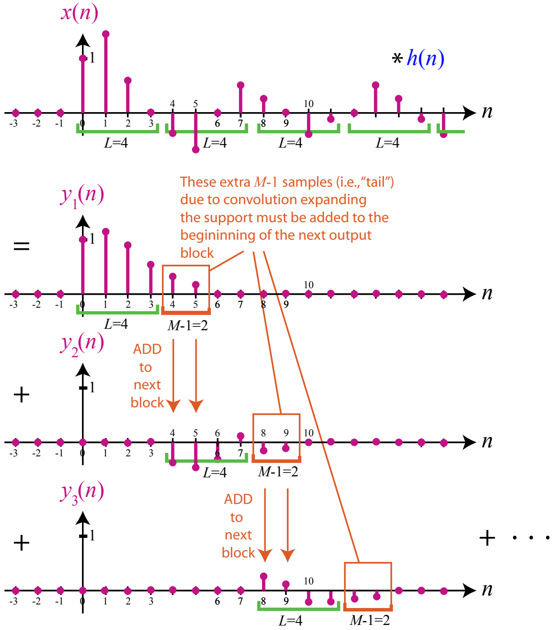
Length of linear convolution result = Length of DFT





Input $x(n)$ is divided into non-overlapping blocks $x_m(n)$ each of length L.

Each input block $x_m(n)$ is individually filtered as it is received to produce the output block $y_m(n)$.



Output blocks $y_m(n)$ must be fitted together appropriately to generate:

$$y(n) = x(n) * h(n)$$

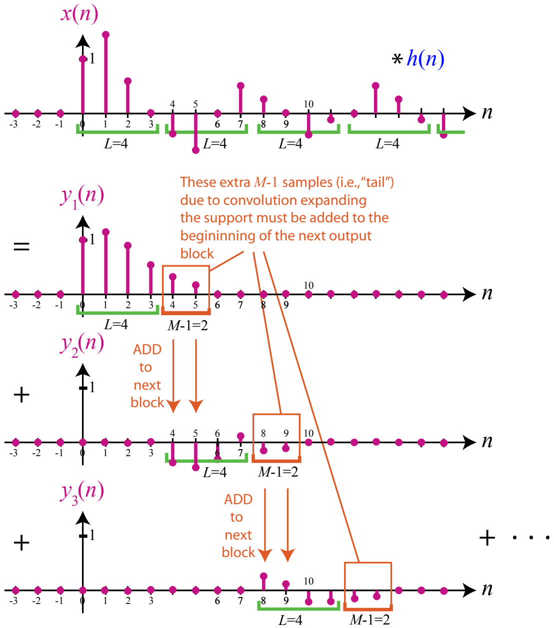
The support overlap amongst the $y_m(n)$ blocks must be accounted for.

Overlap-Add Addition Stage

From the Addititvity property, since:

$$
\begin{aligned}
x(n) &= x_1(n) + x_2(n) + x_3(n) + \cdots = \sum_{m=1}^{\infty} x_m(n) \\
x(n) * h(n) &= (x_1(n) + x_2(n) + x_3(n) + \cdots) * h(n) \\
&= x_1(n) * h(n) + x_2(n) * h(n) + x_3(n) * h(n) + \cdots \\
&= \sum_{m=1}^{\infty} x_m(n) * h(n) = \sum_{m=1}^{\infty} y_m(n)
\end{aligned}
$$

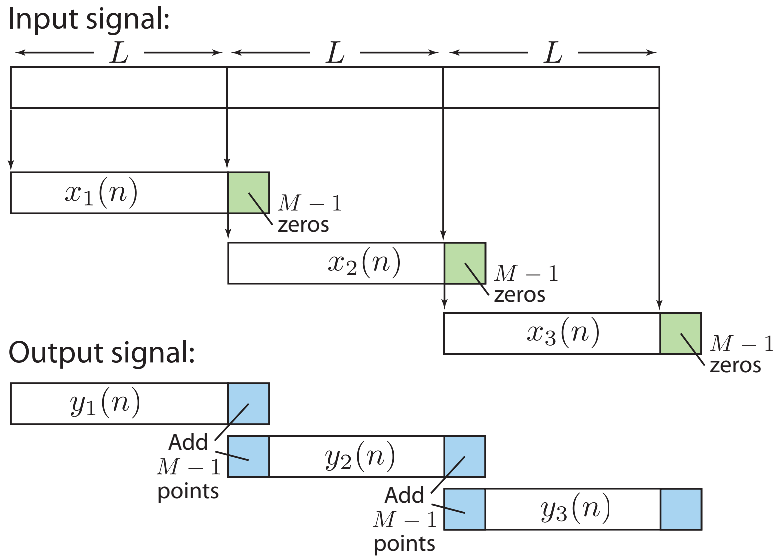
Overlap-Save and Overlap-Add for Real-time Processing Overlap-Add Method



Output blocks $y_m(n)$ must be fitted together appropriately to generate:

$$y(n) = x(n) * h(n)$$

The support overlap amongst the $y_m(n)$ blocks must be accounted for.



Overlap-Add Method

1. Break the input signal $x(n)$ into non-overlapping blocks $x_m(n)$ of length L.
2. Zero pad $h(n)$ to be of length $N = L + M - 1$.
3. Take N-DFT of $h(n)$ to give $H(k)$, $k = 0, 1, \ldots, N-1$.
4. For each block m:
   - 4.1 Zero pad $x_m(n)$ to be of length $N = L + M - 1$.
   - 4.2 Take N-DFT of $x_m(n)$ to give $X_m(k)$, $k = 0, 1, \ldots, N-1$.
   - 4.3 Multiply: $Y_m(k) = X_m(k) \cdot H(k)$, $k = 0, 1, \ldots, N-1$.
   - 4.4 Take N-IDFT of $Y_m(k)$ to give $y_m(n)$, $n = 0, 1, \ldots, N-1$.
5. Form $y(n)$ by overlapping the last $M-1$ samples of $y_m(n)$ with the first $M-1$ samples of $y_{m+1}(n)$ and adding the result.

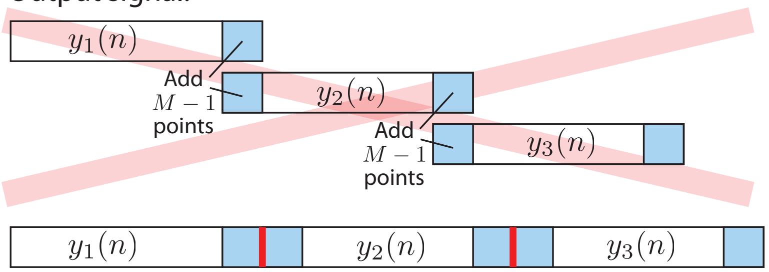
Overlap-Add Method: Cautionary Note

If you DO NOT overlap and add, but only append the output blocks $y_m(n)$ for $m = 1, 2, \ldots$, then you will not get the true $y(n)$ sequence.

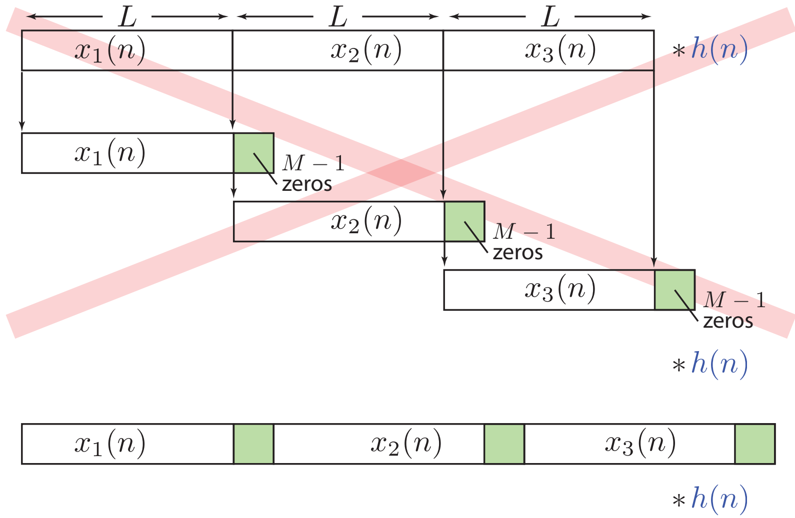
Q: What sequence will you obtain instead?

Overlap-Add Method: Cautionary Note

Output signal:



Overlap-Add Method: Cautionary Note



Overlap-Save Method

Overlap-Save Method

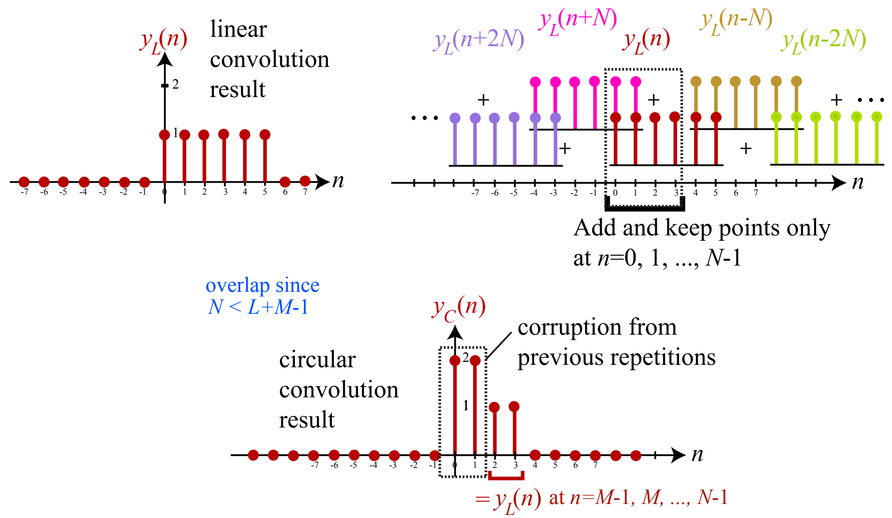
Overlap-[Discard] Method

Overlap-Save Method

Deals with the following signal processing principles:

- The $N = (L + M - 1)$-circular convolution of a discrete-time signal of length N and a discrete-time signal of length M using an N-DFT and N-IDFT.
- Time-Domain Aliasing:

$$x_C(n) = \sum_{l=-\infty}^{\infty} \underbrace{x_L(n - lN)}_{\text{support}=M+N-1}, \quad n = 0, 1, \ldots, N-1$$



Overlap-Save Method

- Convolution of $x_m(n)$ with support $n = 0, 1, \ldots, N-1$ and $h(n)$ with support $n = 0, 1, \ldots, M-1$ via the N-DFT will produce a result $y_{C,m}(n)$ such that:

$$y_{C,m}(n) = \begin{cases} \text{aliasing corruption} & n = 0, 1, \ldots, M-2 \\ y_{L,m}(n) & n = M-1, M, \ldots, N-1 \end{cases}$$

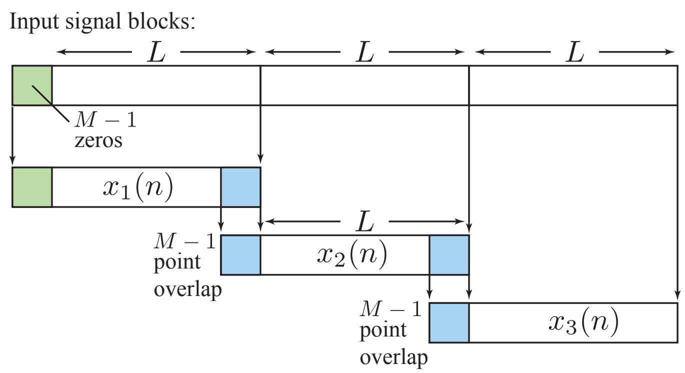
where $y_{L,m}(n) = x_m(n) * h(n)$ is the desired output.

- The first $M-1$ points of a the current filtered output block $y_m(n)$ must be discarded.
- The previous filtered block $y_{m-1}(n)$ must compensate by providing these output samples.

Overlap-Save Input Segmentation Stage

1. All input blocks $x_m(n)$ are of length $N = (L + M - 1)$ and contain sequential samples from $x(n)$.
2. Input block $x_m(n)$ for $m > 1$ overlaps containing the first $M - 1$ points of the previous block $x_{m-1}(n)$ to deal with aliasing corruption.
3. For $m = 1$, there is no previous block, so the first $M - 1$ points are zeros.

Overlap-Save Input Segmentation Stage



Overlap-Save Input Segmentation Stage

$$
\begin{aligned}
x_1(n) &= \{\underbrace{0, \quad 0, \quad \ldots \quad 0,}_{M-1 \text{ zeros}} \; x(0), x(1), \ldots, x(L-1)\} \\
x_2(n) &= \{\underbrace{x(L-M+1), \ldots x(L-1)}_{\text{last } M-1 \text{ points from } x_1(n)}, x(L), \ldots, x(2L-1)\} \\
x_3(n) &= \{\underbrace{x(2L-M+1), \ldots x(2L-1)}_{\text{last } M-1 \text{ points from } x_2(n)}, x(2L), \ldots, x(3L-1)\} \\
&\vdots
\end{aligned}
$$

The last $M - 1$ points from the previous input block must be saved for use in the current input block.

Overlap-Save Filtering Stage

- makes use of the N-DFT and N-IDFT where: $N = L + M - 1$
  - Only a one-time zero-padding of $h(n)$ of length $M \ll L < N$ is required to give it support $n = 0, 1, \ldots, N - 1$.
  - The input blocks $x_m(n)$ are of length N to start, so no zero-padding is necessary.
  - The actual implementation of the DFT/IDFT will use the fast Fourier Transform (FFT) for computational simplicity.

Using DFT for Circular Convolution

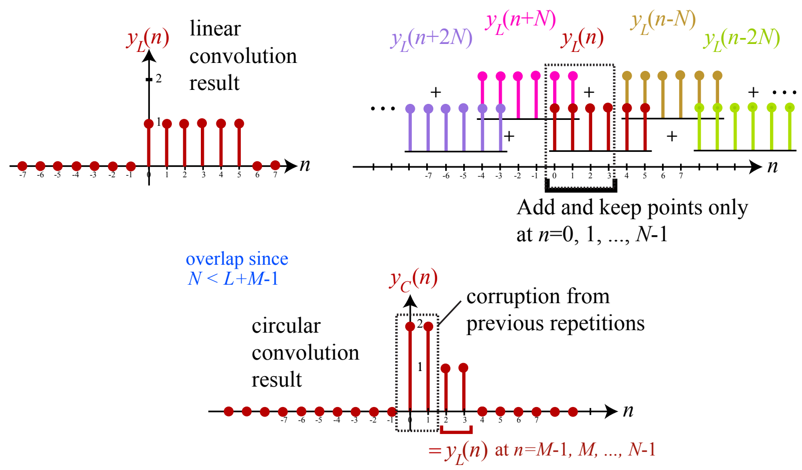
$N = L + M - 1$.
Let $x_m(n)$ have support $n = 0, 1, \ldots, N-1$.
Let $h(n)$ have support $n = 0, 1, \ldots, M-1$.

We zero pad $h(n)$ to have support $n = 0, 1, \ldots, N-1$.

1. Take N-DFT of $x_m(n)$ to give $X_m(k)$, $k = 0, 1, \ldots, N-1$.
2. Take N-DFT of $h(n)$ to give $H(k)$, $k = 0, 1, \ldots, N-1$.
3. Multiply: $Y_m(k) = X_m(k) \cdot H(k)$, $k = 0, 1, \ldots, N-1$.
4. Take N-IDFT of $Y_m(k)$ to give $y_{C,m}(n)$, $n = 0, 1, \ldots, N-1$.

Circular Convolution via the DFT

Length of linear convolution result $>$ Length of DFT



Overlap-Save Output Blocks

$$y_{C,m}(n) = \begin{cases} \text{aliasing} & n = 0, 1, \ldots, M-2 \\ y_{L,m}(n) & n = M-1, M, \ldots, N-1 \end{cases}$$

where $y_{L,m}(n) = x_m(n) * h(n)$ is the desired output.

Overlap-Save [Discard] Output Blocks

$$y_1(n) = \{\underbrace{y_1(0), \quad y_1(1), \quad \ldots \quad y_1(M-2)}_{M-1 \text{ points corrupted from aliasing}}, \; y(0), \ldots, y(L-1)\}$$

$$y_2(n) = \{\underbrace{y_2(0), \quad y_2(1), \quad \ldots \quad y_2(M-2)}_{M-1 \text{ points corrupted from aliasing}}, y(L), \ldots, y(2L-1)\}$$

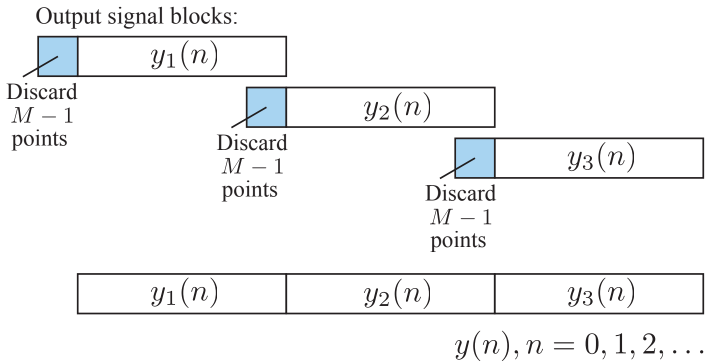
$$y_3(n) = \{\underbrace{y_3(0), \quad y_3(1), \quad \ldots \quad y_3(M-2)}_{M-1 \text{ points corrupted from aliasing}}, y(2L), \ldots, y(3L-1)\}$$

where $y(n) = x(n) * h(n)$ is the desired output.

The first $M-1$ points of each output block are discarded.

The remaining L points of each output block are appended to form $y(n)$.

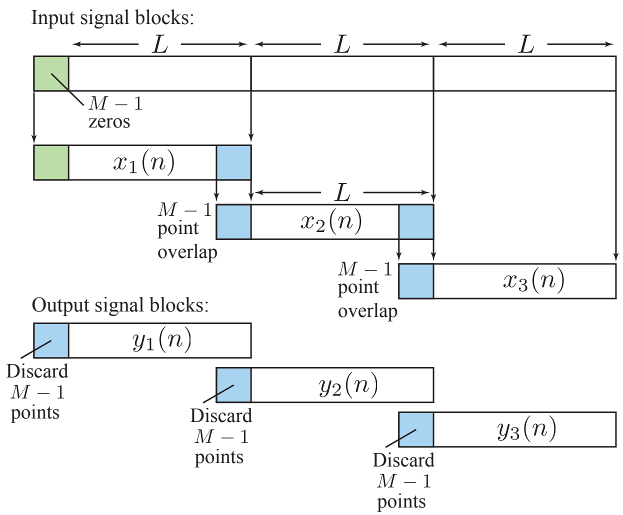
Overlap-Save Output Stage



Overlap-Save Method

1. Insert $M-1$ zeros at the beginning of the input sequence $x(n)$.
2. Break the padded input signal into overlapping blocks $x_m(n)$ of length $N = L + M - 1$ where the overlap length is $M - 1$.
3. Zero pad $h(n)$ to be of length $N = L + M - 1$.
4. Take N-DFT of $h(n)$ to give $H(k)$, $k = 0, 1, \ldots, N-1$.
5. For each block m:
   - 5.1 Take N-DFT of $x_m(n)$ to give $X_m(k)$, $k = 0, 1, \ldots, N-1$.
   - 5.2 Multiply: $Y_m(k) = X_m(k) \cdot H(k)$, $k = 0, 1, \ldots, N-1$.
   - 5.3 Take N-IDFT of $Y_m(k)$ to give $y_m(n)$, $n = 0, 1, \ldots, N-1$.
   - 5.4 Discard the first $M-1$ points of each output block $y_m(n)$.
6. Form $y(n)$ by appending the remaining (i.e., last) L samples of each block $y_m(n)$.

Overlap-Save Method



■