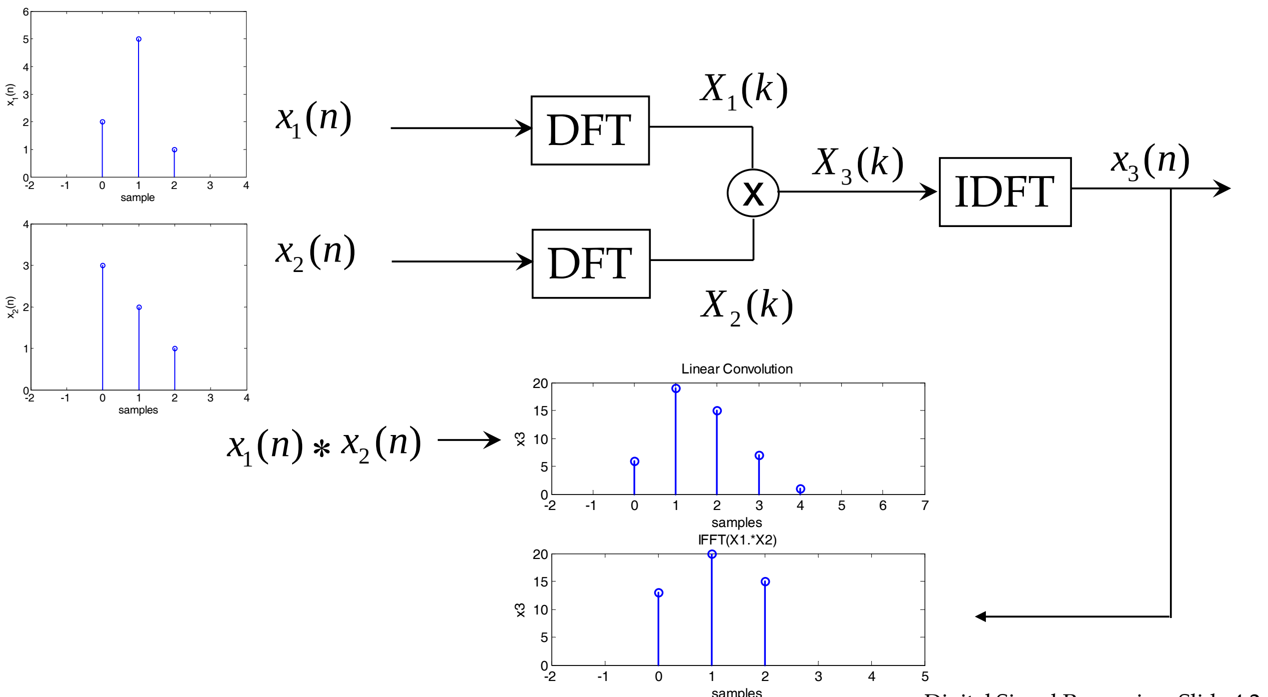
Module 3

Convolution

Aim

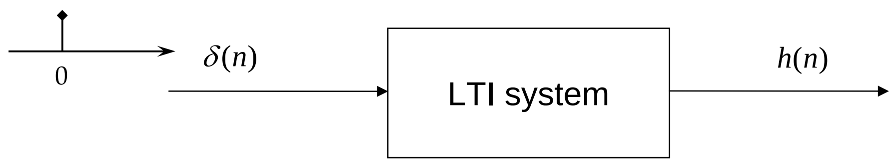
- How to perform convolution in real-time systems efficiently?
- Is convolution in time domain equivalent to multiplication of the transformed sequence?



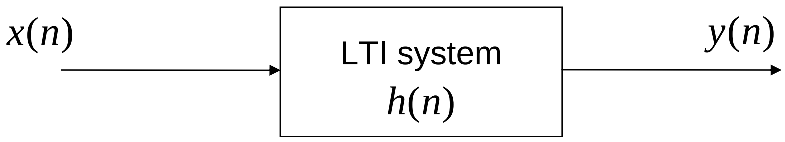
Contents

- Review of convolution techniques
- Linear convolution and circular convolution
  - relationship to filtering
- Fast algorithms for digital filters
  - block filtering

Introduction



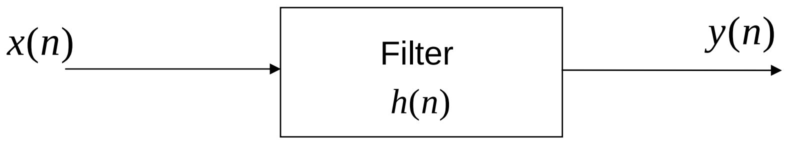
- Let $\delta(n)$ be the unit impulse sequence $\delta(\mathrm{n}) = \begin{cases} 1, & n = 0 \\ 0, & \text{otherwise} \end{cases}$
- We can write $x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$
  - represent $x(n)$ as sum of delayed and weighted impulse
- Let $h(n)$ be the output for $\delta(n)$ input
  - $h(n)$ is the impulse response
- Then $h(n-k)$ is the output for $\delta(n-k)$ input
- LTI – linear time invariance



- Then by superposition $y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$

- $y(n)$ is sum of lots of delayed impulse responses
- This equation is called the CONVOLUTION SUM

Linear Convolution



- Consider unit step input: $x(n) = u(n)$
- And example filter: $h(n) = a^n u(n)$
- Task: find $y(n)$

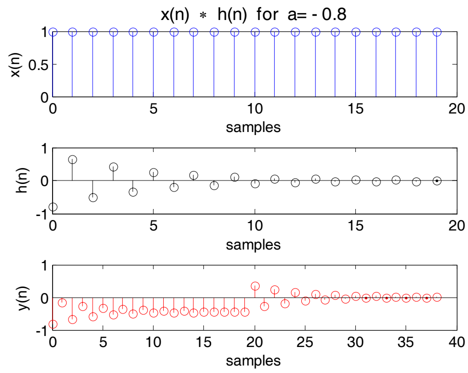
- Filtering is the operation of convolving a signal with the filter's impulse response.

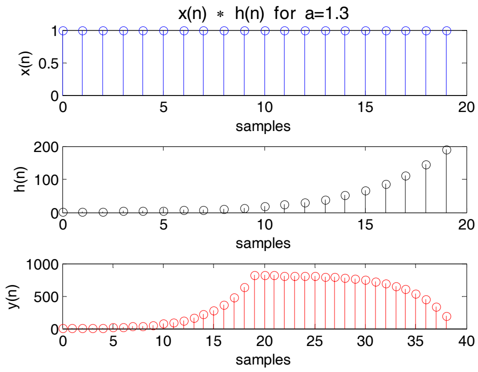
$$y(n) = x(n) * h(n)$$
$$= \sum_{m=-\infty}^{\infty} x(m)h(n-m)$$

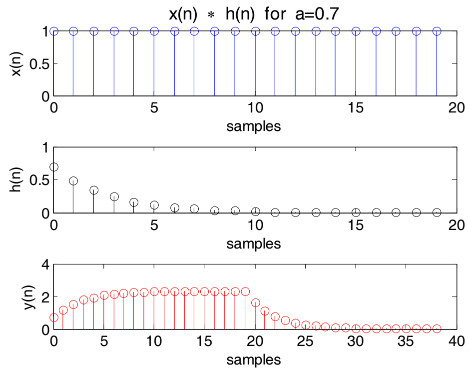
- For this example:

$$y(n) = 0, \quad n < 0$$
$$y(0) = x(0)h(0) = 1 \qquad \text{(all other terms are zero)}$$

$$y(n) = 1 + a + a^2 + \ldots + a^3, \quad n > 0$$
$$= \frac{1 - a^{n+1}}{1 - a} u(n)$$





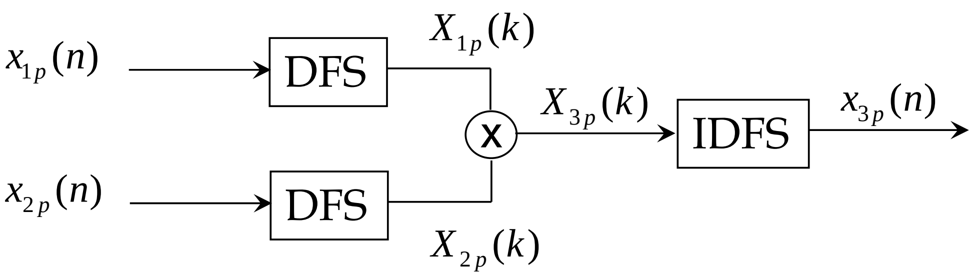


Periodic and Circular Convolution

- Develop ideas in terms of periodic signals and DFS, then extend to aperiodic signals.

- Given 2 periodic signals $x_{1p}(n)$ and $x_{2p}(n)$
  with discrete Fourier series coefficients given by $X_{1p}(k)$ and $X_{2p}(k)$
  what is $x_{3p}(n)$ formed from

$$x_{3p}(n) = \text{IDFS}\left\{X_{3p}(k) = X_{1p}(k)X_{2p}(k)\right\}$$



Write

$$X_{3p}(k) = \sum_{l=0}^{N-1}\sum_{r=0}^{N-1} x_{1p}(l)\, x_{2p}(r)\, e^{j\frac{2\pi}{N}(-lk-rk)}$$

then

$$x_{3p}(n) = \frac{1}{N}\sum_{k=0}^{N-1} X_{3p}(k)\, e^{j\frac{2\pi}{N}nk}$$

and finally (proof-tutorial question)

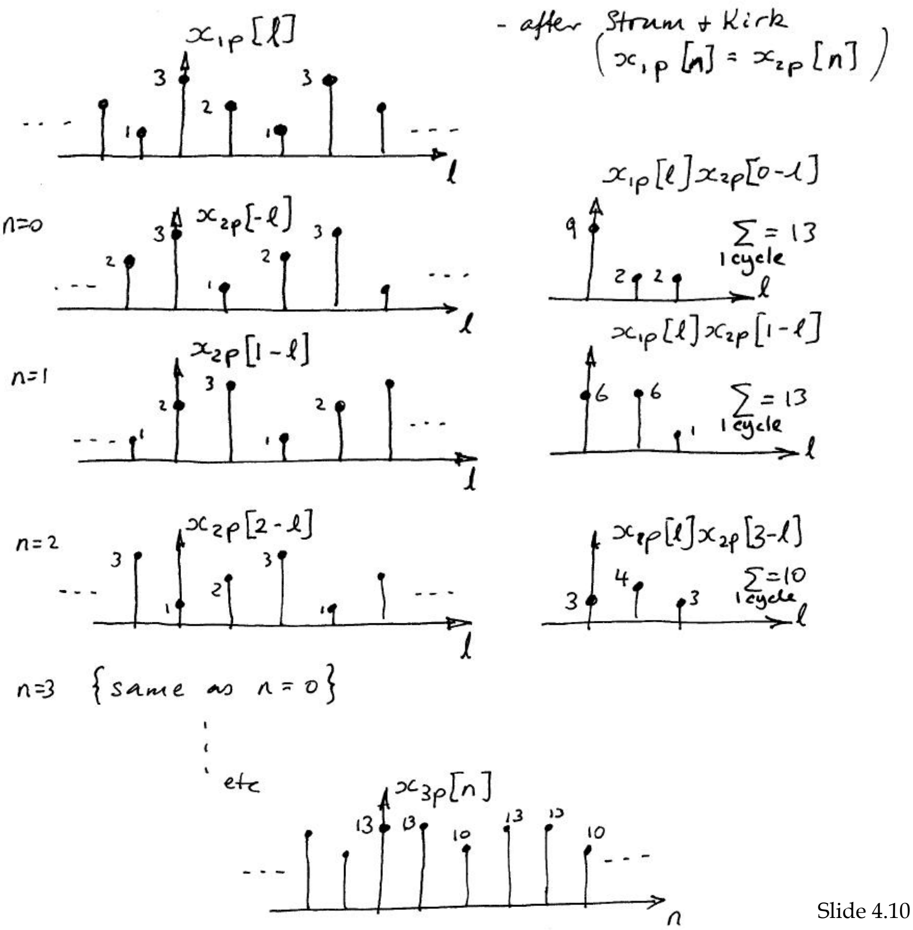
$$x_{3p}(n) = \sum_{l=0}^{N-1} x_{1p}(l) x_{2p}(n-l).$$

Compare to linear convolution

$$x_3(n) = \sum_{m=-\infty}^{\infty} x_1(m)\, x_2(n-m)$$

- Linear convolution:
  - Sum over infinite extent of signals
- Periodic convolution:
  - Sum over one period

Example A



Example B

- So far we have $x_{3p}(n)$ from a convolution of $x_{1p}(n)$ with $x_{2p}(n)$

- Since $X_{3p}(k) = X_{1p}(k) X_{2p}(k)$
  we can also obtain $x_{3p}(n)$ from the IDFS of $X_{3p}(k)$

$$X_{1p}(k) = \sum_{n=0}^{2} x_{1p}(n)\, e^{-j2\pi/3.nk}$$

$$= [6,\ \sqrt{3}e^{-j\pi/6},\ \sqrt{3}e^{+j\pi/6}]$$

Similarly $X_{2p}(k) = [6,\ \sqrt{3}e^{-j\pi/6},\ \sqrt{3}e^{j\pi/6}]$

$$\therefore \quad X_{3p}(k) = [36,\ 3e^{-j\pi/3},\ 3e^{j\pi/3}]$$

Hence $x_{3p}(n) = \frac{1}{3}\sum_{k=0}^{2} X_{3p}(k)\, e^{j2\pi/3.nk}$

$$= [13,\ 13,\ 10] \quad \text{- as before}$$

Periodic - Aperiodic

- So far we have considered periodic signals
  - Periodic convolution
    - Direct
    - Via DFS
- Now consider aperiodic signals
  - Circular convolution
  - Treat aperiodic signals as one period of periodic signals

Circular Convolution

- Consider aperiodic signals $x_1(n)$ and $x_2(n)$

  $x_{1p}(n)$ and $x_{2p}(n)$ are periodic extensions of $x_1(n)$ and $x_2(n)$.

  $$x_3(n) = \text{ one period of } \left\{ \sum_{l=0}^{N\text{-}1} x_{1p}(l)x_{2p}(n-l) \right\}$$

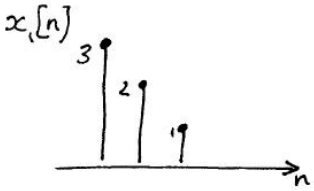
  $$x_3(n) = x_1(n) \;\otimes\; x_2(n)$$

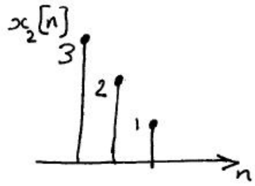
  $$DFT\{x_3(n)\} = DFT\{x_1(n) \;\otimes\; x_2(n)\} = X_1(k)X_2(k)$$

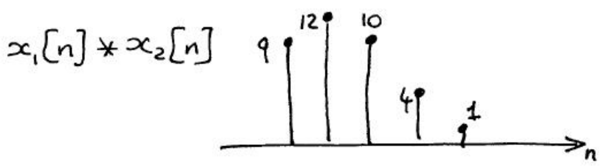
- The symbol $\otimes$ means circular convolution
  - Perform periodic extension
  - Take one period of the result of periodic convolution

Example

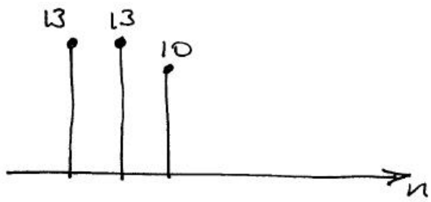
- Consider Linear convolution of $x_1(n) \;*\; x_2(n)$







- Consider Circular convolution of $x_1(n) \;\otimes\; x_2(n)$



  • different length

  • different values

Relationship between Linear Convolution and Circular Convolution

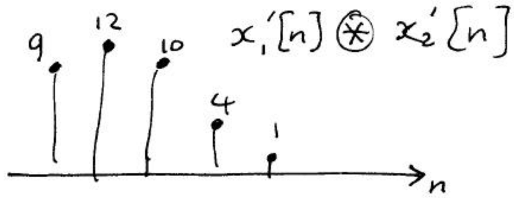
- Why are we interested?
  - Linear convolution is the filtering operation
  - Filters are a very important application of DSP
  - Would like to implement filters efficiently
  - FFT is a fast DFT
  - Maybe we can use FFTs to implement filters (?)

- We desire that circular and linear convolution give identical results, then we can use FFTs for fast filtering
- This can be achieved by applying zero-padding to the signals before performing circular convolution

- For a signal $x_1(n)$ of length N_1 and signal $x_2(n)$ of length N_2
- Zero-pad $x_1(n)$ with $N_2 - 1$ zeros
- Zero-pad $x_2(n)$ with $N_1 - 1$ zeros

Example

- For a signal $x_1(n)$ of length N_1 and signal $x_2(n)$ of length N_2
- Zero-pad $x_1(n)$ with $N_2 - 1$ zeros to give $x_1^{'}(n)$
- Zero-pad $x_2(n)$ with $N_1 - 1$ zeros to give $x_2^{'}(n)$



Equivalent to linear convolution

Summary

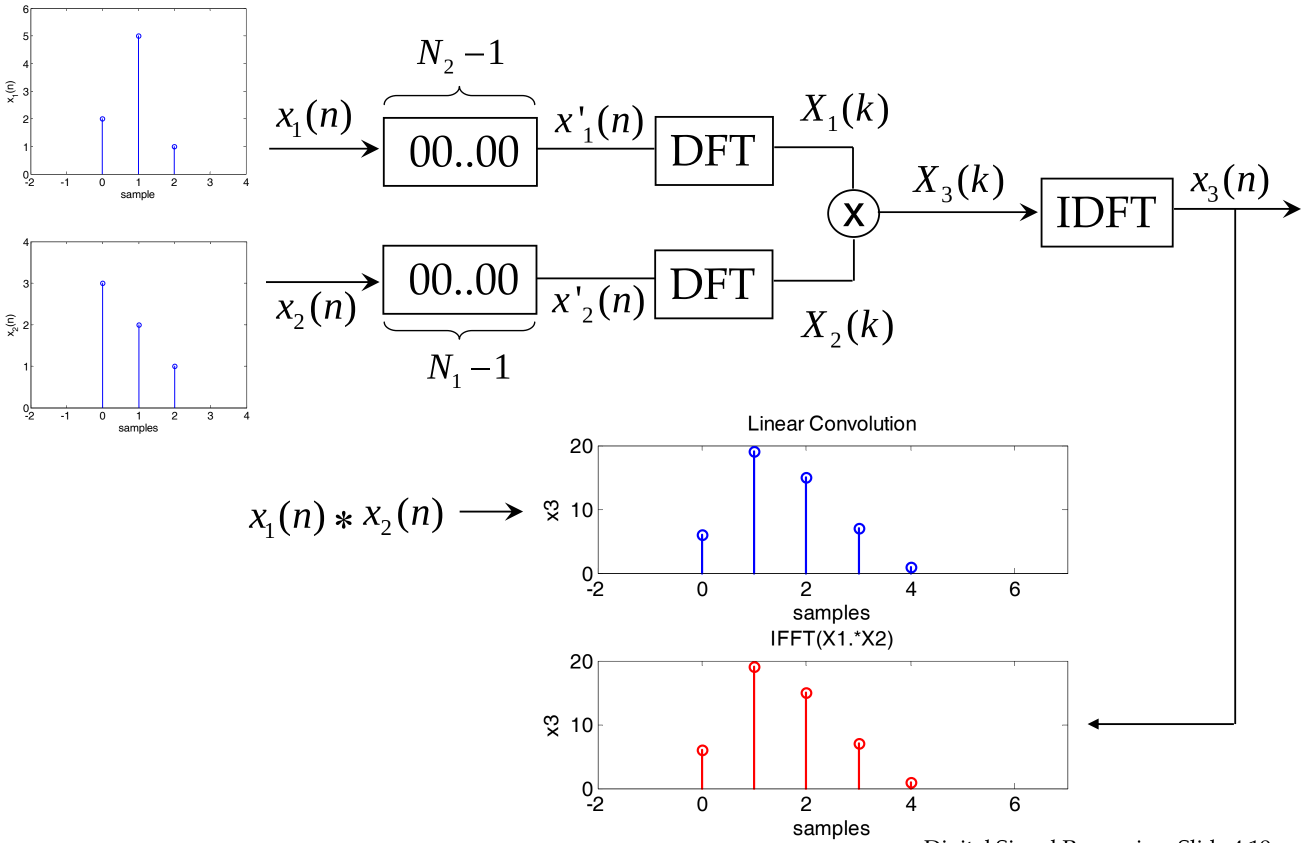
$$x_1(n) * x_2(n) = x_1^{'}(n) \otimes x_2^{'}(n) = \mathrm{IDFT}\left\{X_1^{'}(k) X_2^{'}(k)\right\}$$

given that $X_1(k) = DFT\left\{x(n)\right\}$

and $x_1^{'}(n)$ is a zero-padded version of $x_1(n)$

$*$ means Linear Convolution

$\otimes$ means Circular Convolution



Complexity

- Direct implementation
  - N^2 multiplies

- DFT Implementation

$$4 \times [2 \times (2N)^2 + 2N + (2N)^2] = 48N^2 + 8N$$

for complex multiplications

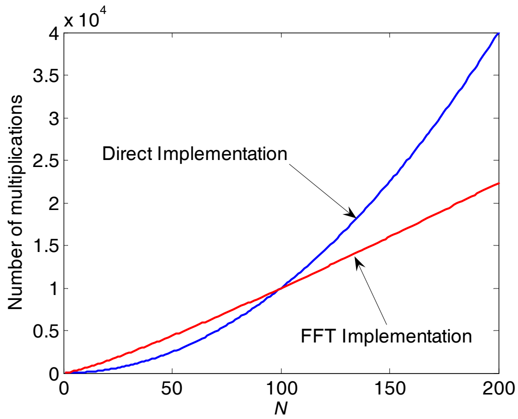
2 DFTs of 2N-1 points

Product of 2 sequences of length 2N

1 IDFT

- FFT Implementation

$$4 \times \left[2 \times \left(\frac{2N}{2} \log_2 2N \right) + 2N + \left(\frac{2N}{2} \log_2 2N \right) \right]$$
$$= 12N \log_2 2N + 8N$$



Block Filtering

- FFT-based convolution has advantage of lower computational complexity
  - But if data is long, must wait until all the data is captured
  - Long delay
- Solution
  - Use block filtering
  - Use FFT-based convolution on short blocks of data
  - Then join blocks together
- Sectioned convolution: two main methods
  - Overlap add
  - Overlap save

Overlap-add Procedure

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

$x(n)$: data; divide into blocks of length L

$h(n)$: impulse response; length M

- Perform linear convolution using FFT
- Each block of $x(n)$ padded with M-1 zeros
- Each block of $h(n)$ padded with L-1 zeros

$$y_1(n) = IFFT\left\{ X'_1(k)\, H'(k) \right\}$$

First block

Computed once and stored

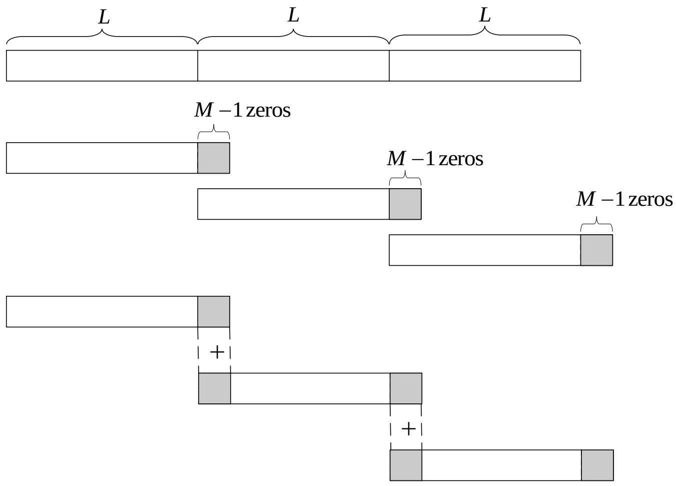
- Size of FFT: next integer power of 2 greater than or equal to L+M-1
  - M is fixed by the filter impulse response
  - Choose to obtain convenient size of FFT

- Note that the length of $y_1(n)$ is $L+M-1$
  - first L samples are true values
  - $M-1$ samples are in the overlap

$$y_2(n) = IFFT\{X'_2(k)H'(k)\}$$

second block

  - $y_2(n)$ is added to $y_1(n)$ with a shift of L samples



Overlap-save Procedure

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

$x(n)$: data; divide into blocks of length L

$h(n)$: impulse response; length M

- Perform linear convolution using FFT
- Each block of $x(n)$ consists of
  - the last M-1 data points from previous data block
  - follow by L new points
- Each block of $h(n)$ padded with L-1 zeros

$$y_1(n) = IFFT\{X'_1(k)\, H'(k)\}$$

First block　　Computed once and stored

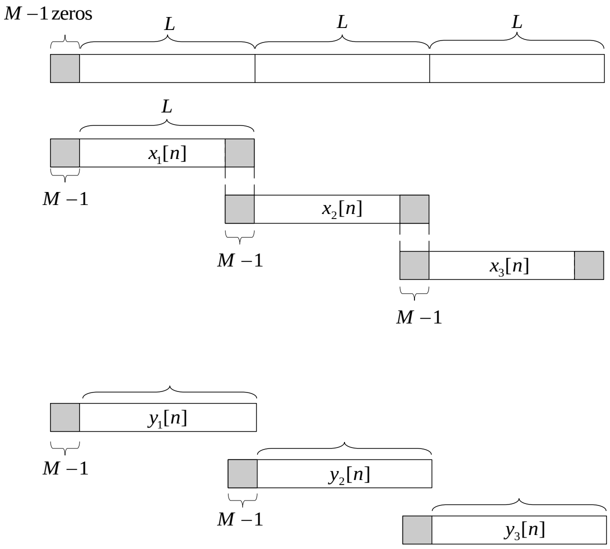
- Size of FFT: next integer power of 2 greater than or equal to $L+M-1$
  - M is fixed by the filter impulse response
  - Choose to obtain convenient size of FFT

- Note that the length of $y_1(n)$ is L+M-1
  - last L samples are true values
  - M-1 samples are in the overlap

$$y_2(n) = IFFT\left\{X'_2(k)H'(k)\right\}$$

second block

  - Discard the first M-1 samples, only last L samples are saved



Correlation

- Operation to determine some measure of similarity between two signals

- Correlation of two signals $x(n)$ and $y(n)$ is defined as

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l)\ , \quad l = 0, \pm 1, \pm 2, \ldots$$

l is the index of the correlation function $r_{xy}(l)$ and is referred to as "lag"

- Autocorrelation is the special case when $x(n) = y(n)$

or the
lock

- Relationship between correlation and convolution
  - Convolution

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

Time reversed and delayed samples of $y[m]$

  - Correlation

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l)$$
$$= \sum_{n=-\infty}^{\infty} x(n)y(-(l-n))$$
$$= x(l) * y(-l)$$

Delayed samples of $y[n]$

- Therefore, correlation can be implemented by a convolution algorithm providing one of the inputs is given time-inversed

Property of Correlation

- We can write $r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l)$

$$= \sum_{n=-\infty}^{\infty} x(n+l)y(n), \ \ l = 0, \pm 1, \pm 2 \ldots$$

- Similarly, $r_{yx}(l) = \sum_{n=-\infty}^{\infty} y(n)x(n-l)$

$$= \sum_{n=-\infty}^{\infty} y(n+l)x(n), \ \ l = 0, \pm 1, \pm 2 \ldots$$

- From which, we can deduce that

$$r_{xy}(l) = r_{yx}(-l)$$

- For the autocorrelation function, we obtain an even function

$$r_{xx}(l) = r_{xx}(-l)$$