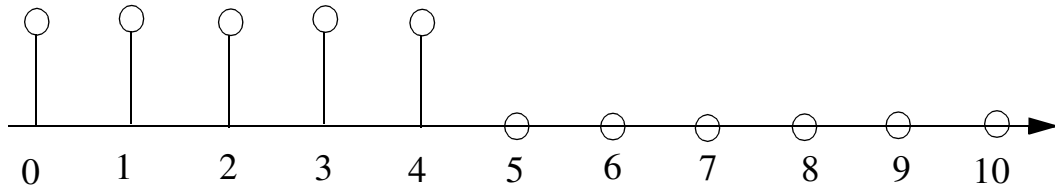


Lecture 5: Discrete Fourier Transform

Reading: Sec. 8.0 - 8.6.

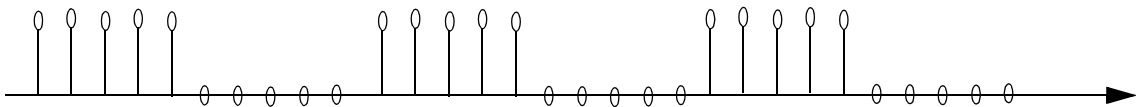
- Discrete Fourier transform (DFT) is different from discrete-time Fourier transform (DTFT) in that it deals with discrete samples not only in the time domain, but also in the frequency domain, a concept often referred to as frequency sampling.

Example: Suppose that $x(n) = u(n-5) - u(n)$:



$$\text{Its DTFT is } H(e^{j\omega}) = \sum_{n=0}^4 e^{-j\omega n} = (e^{-j2\omega}) \frac{\sin 5\frac{\omega}{2}}{\sin \frac{\omega}{2}}.$$

Q: If $H(e^{j\omega})$ is sampled at $\omega = 2\pi k/10$, what would the corresponding time domain sequence be?



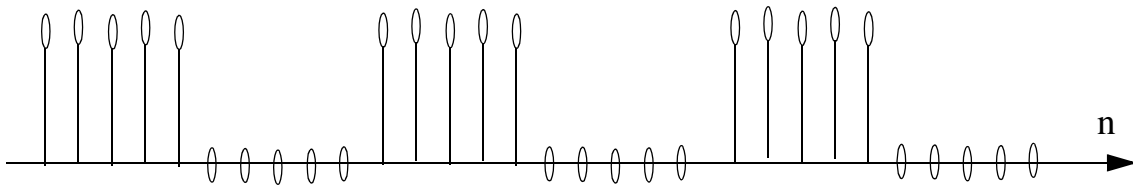
- Discrete Fourier Series:** For a periodic discrete-time signal $\tilde{x}(n)$, we can define its discrete Fourier series (DFS) $\tilde{X}[k]$ as follows:

$$\tilde{X}[k] = \sum_{n=0}^{N-1} \tilde{x}(n) e^{-j\left(\frac{2\pi}{N}\right)kn}, \text{ where } N \text{ is number of samples per period in time.}$$

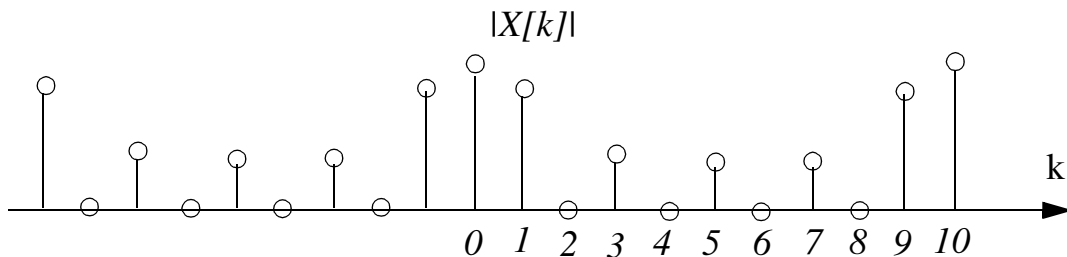
$$\tilde{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[k] e^{j\left(\frac{2\pi}{N}\right)kn}, \text{ where } N \text{ is the number of samples from } 0 \text{ to } 2\pi.$$

- The DFS pair is interesting in that this is the first time that we don't need an integral in defining a Fourier-like transform, and both the summations are finite! We can therefore **compute** them without using numerical approximation. You might ask if this is such a big deal because most discrete-time signals are **not** periodic. Well, what we will do is to break up infinite sequences into finite blocks and recombine the block results as if we had processed the original sequence (later).

Example: $x(n)$



$$\tilde{X}[k] = \sum_{n=0}^4 e^{-j\frac{2\pi}{10}kn} = \left(e^{-j2\frac{2\pi k}{10}} \right) \frac{\sin \frac{\pi k}{2}}{\sin \frac{\pi k}{10}} .$$



- Now let's look at the relationship between any $x(n)$ and $\tilde{x}(n)$, a periodic sequence constructed from sampling the DTFT of $x(n)$.
- Frequency sampling:** If we take N equally spaced samples of $X(e^{j\omega})$, the DTFT of $x(n)$, and call these samples $\tilde{X}[k]$, then we have

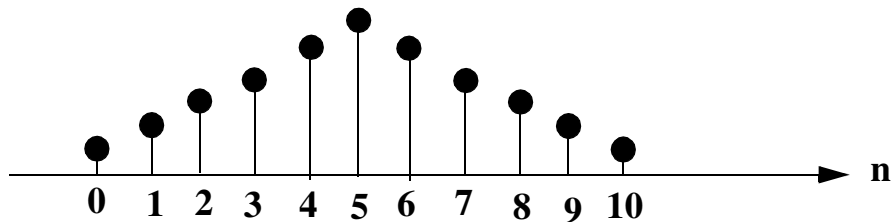
$$\tilde{X}[k] = X(e^{j\omega}) \Big|_{\omega = \frac{2\pi}{N}k} = X(z) \Big|_{z = e^{j\left(\frac{2\pi}{N}\right)k}} .$$

To see what the corresponding $\tilde{x}(n)$ looks like, we can put $\tilde{X}[k]$ back into the DFS pair defined earlier:

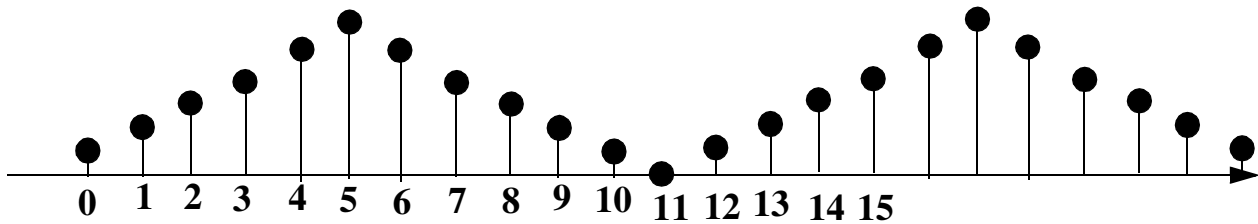
$$\begin{aligned}
 \tilde{x}(n) &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn} \\
 &= \frac{1}{N} \sum_{k=0}^{N-1} \left[\sum_{m=-\infty}^{\infty} x(m) e^{(-j)\frac{2\pi}{N}km} \right] e^{j\frac{2\pi}{N}kn} \\
 &= \sum_{m=-\infty}^{\infty} x(m) \frac{1}{N} \left[\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}k(n-m)} \right] \\
 &= \sum_{m=-\infty}^{\infty} x(m) \left[\sum_{r=-\infty}^{\infty} \delta(n-m+rN) \right] \\
 &= x(n) \otimes \sum_{r=-\infty}^{\infty} \delta(n+rN) \\
 &= \sum_{r=-\infty}^{\infty} x(n+rN)
 \end{aligned}$$

- $\tilde{x}(n)$ is nothing more than an aliased version of $x(n)$! How often the aliases occur depends on N , the number of samples taken in the frequency domain. The duality between sampling in time and sampling in frequency is obvious--both produce aliases on the other domain. You can check that the technique we used to describe sampling in time also works for predicting $\tilde{x}(n)$ from its DFS in the frequency domain.

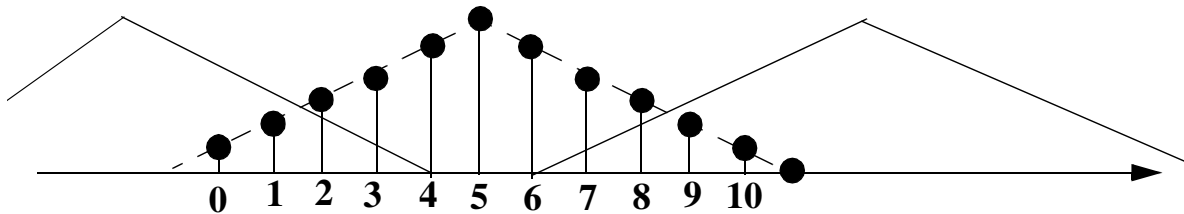
Example: Suppose that $x(n)$ takes the following form:



Q: What does $\tilde{x}(n)$ look like if $N = 12$?



Q: What does $\tilde{x}(n)$ look like if $N = 7$?



Remember $\tilde{x}(n)$ is just an aliased version of $x(n)$, with aliases repeating at every N samples.

- Discrete Fourier transform (DFT) is a Fourier representation of finite duration sequences. You can think of DFT as an approximation of $H(e^{j\omega})$ by sampling at several frequency points. In fact all computer-based signal processing computation has made this approximation, including the routines in Matlab. We only need to make sure that enough samples are taken with sufficient frequency resolution, similar to the sampling theory in the time domain, so that the aliases will not corrupt the samples of interest.
- Oppenheim said that we simply take one period of $\tilde{x}(n)$ and call it $x(n)$, defined only between $0 \leq n \leq N-1$. Of this artificial sequence we calculate its DFS and generate the corresponding $X[k]$, defined only between $0 \leq k \leq N-1$. Note that even though $x(n)$ and $X[k]$ are of finite duration, as constructed by taking only one period of a periodic sequence, they are defined in terms of the corresponding DFS. (I'd prefer not to make any difference between the DFS and DFT, because it doesn't make much sense to talk about a sequence defined only between $0 \leq n \leq N-1$. Thinking of DFT as a periodic function helps in understanding some subtle issues in later lectures.)

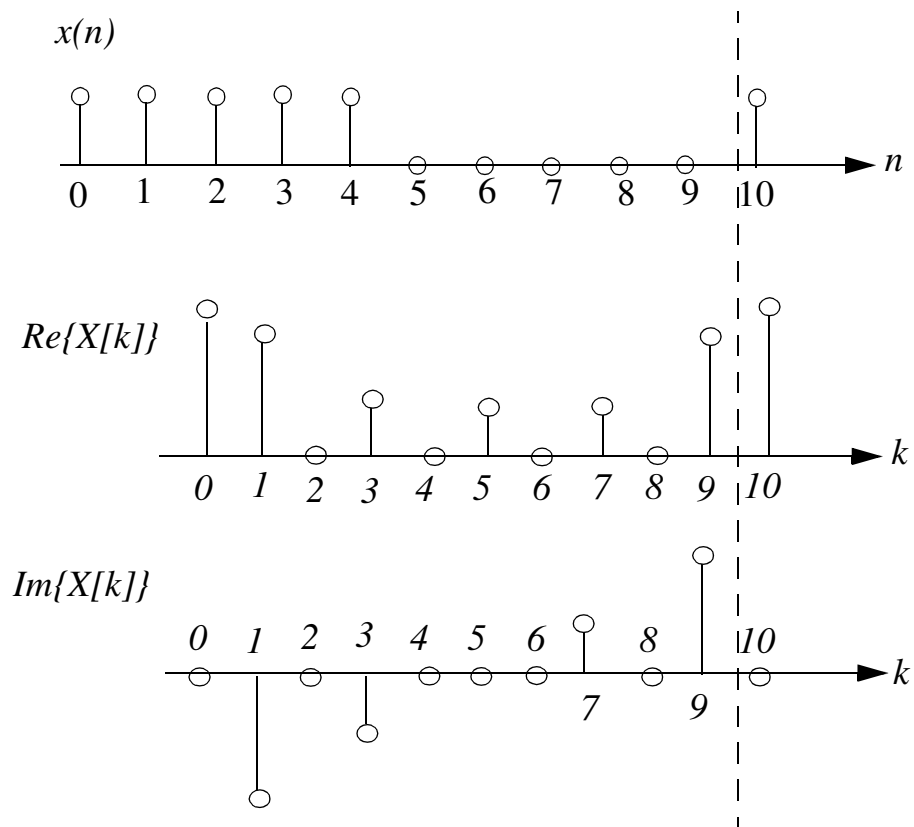
- Linearity: If $x_1(n)$ has length N_1 and $x_2(n)$ has length N_2 where $N_2 > N_1$, then the DFT of $ax_1(n) + bx_2(n)$ can be obtained by zero-padding $x_1(n)$ to a length of N_2 . $DFT[ax_1(n) + bx_2(n)] = a DFT[x_1(n)] + b DFT[x_2(n)]$.
- Circular shift: If $x_1(n) = x((n-m)_{mod N})$, a circular shift of m samples assumes that $x(n)$ is periodic, then $X_1[k] = e^{-j\left(\frac{2\pi}{N}\right)km} X[k]$.

- Duality:

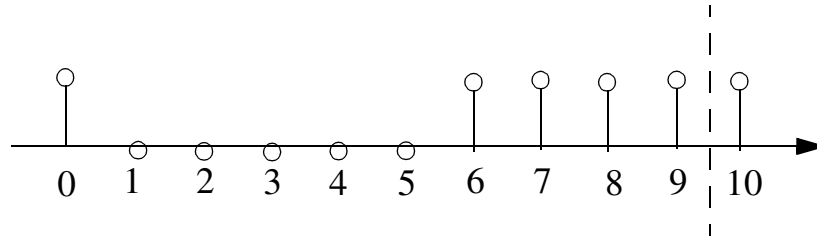
$$x(n) \leftrightarrow X[k]$$

$$X[k] \leftrightarrow Nx((-k)_{mod N})$$

Example:



If we let $x'(n) = X[n]$, its DFT is $10x((-k)_{mod 10})$:



As an exercise, show that the duality relationship holds.

- Symmetry property: If $x(n) \leftrightarrow X[k]$, then

$$\text{Re}\{x(n)\} \leftrightarrow \frac{1}{2}[X[k] + X^*[(-k)_{mod N}]]$$

$$j\text{Im}\{x(n)\} \leftrightarrow \frac{1}{2}[X[k] - X^*[(-k)_{mod N}]]$$

We define a sequence $x(n)$ to be of conjugate symmetry by $x(n) = x^*(-n)$.

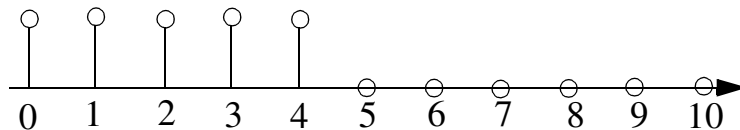
- Circular convolution: If $x_3(n)$ is the circular convolution of $x_1(n)$ and $x_2(n)$:

$$x_3(n) = \sum_{m=0}^{N-1} x_1(m)x_2((n-m)_{mod N}),$$

then $X_3[k] = X_1[k] X_2[k]$.

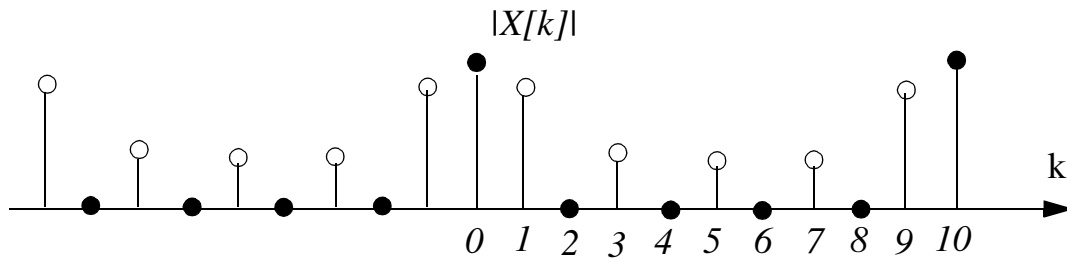
- **Zero padding:** Increasing the sampling resolution in the frequency domain is equivalent to zero-padding the time-domain sequence.

Example:



Pick $N = 5$, then $X[k] = N \delta(k)$.

$$\text{Pick } N = 10, \text{ then } X[k] = \sum_{n=0}^4 e^{-j\frac{2\pi}{10}kn} = \left(e^{-j2\frac{2\pi k}{10}} \right) \frac{\sin \frac{\pi k}{2}}{\sin \frac{\pi k}{10}}$$

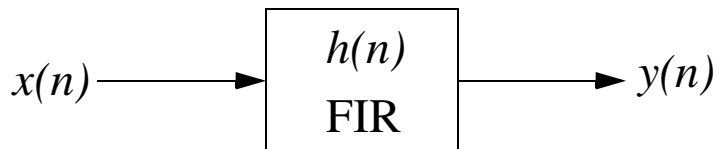


To show that padding zeros at the end of a time-domain sequence is equivalent to increasing sampling resolution in the frequency domain, we take a look at both the DTFT and the DFS equations. In DTFT, adding a few zero samples at the end of a finite-length sequence $x(n)$ is not going to have any effect on its $X(e^{j\omega})$. The N samples of DFT, therefore, are taken from the same $X(e^{j\omega})$ but with a higher resolution. The resulting $\tilde{x}(n)$ has the same $x(n)$ as one period, but with zeros padded in between neighboring periods to accommodate a longer N .

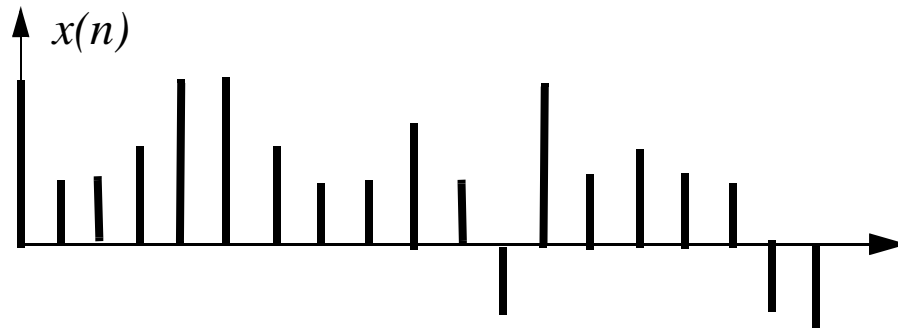
Linear Convolution Using DFT

Reading: Section 8.9.

- DFT, or rather its fast implementation FFT, can sometimes be used to calculate the output of an FIR filter more efficiently than the direct time-domain convolution.



- Represent $x(n) = \sum_{r=0}^{\infty} x_r[n - rL]$, assuming causality.



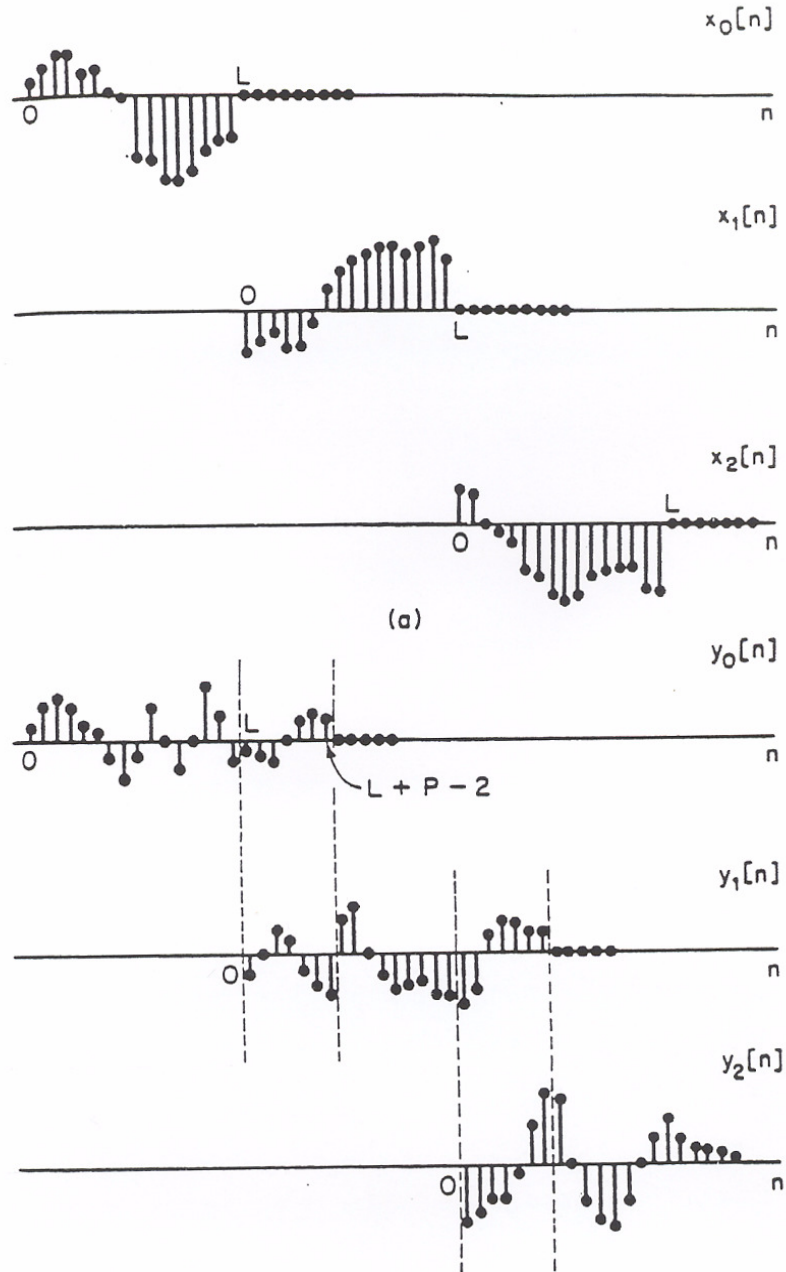
Suppose that these finite sequences have length L and $h(n)$ has length P (relatively shorter than L).

$$y(n) = x(n) \otimes h(n) = \sum_r x_r(n) \otimes h(n) = \sum_r y_r(n)$$

- Recall that the length of the output sequence after linear convolution of two finite sequences is longer than either one of the input sequences, i.e. the length of the output sequence, N , is equal to $L+P-1$. In other words, if both $x_r(n)$ and $h(n)$ are zero-padded to be of length N , then linear convolution can be performed by FFT in the frequency domain.

Overlap-and-Add Method

- Typically we select $N=L+P-1$.

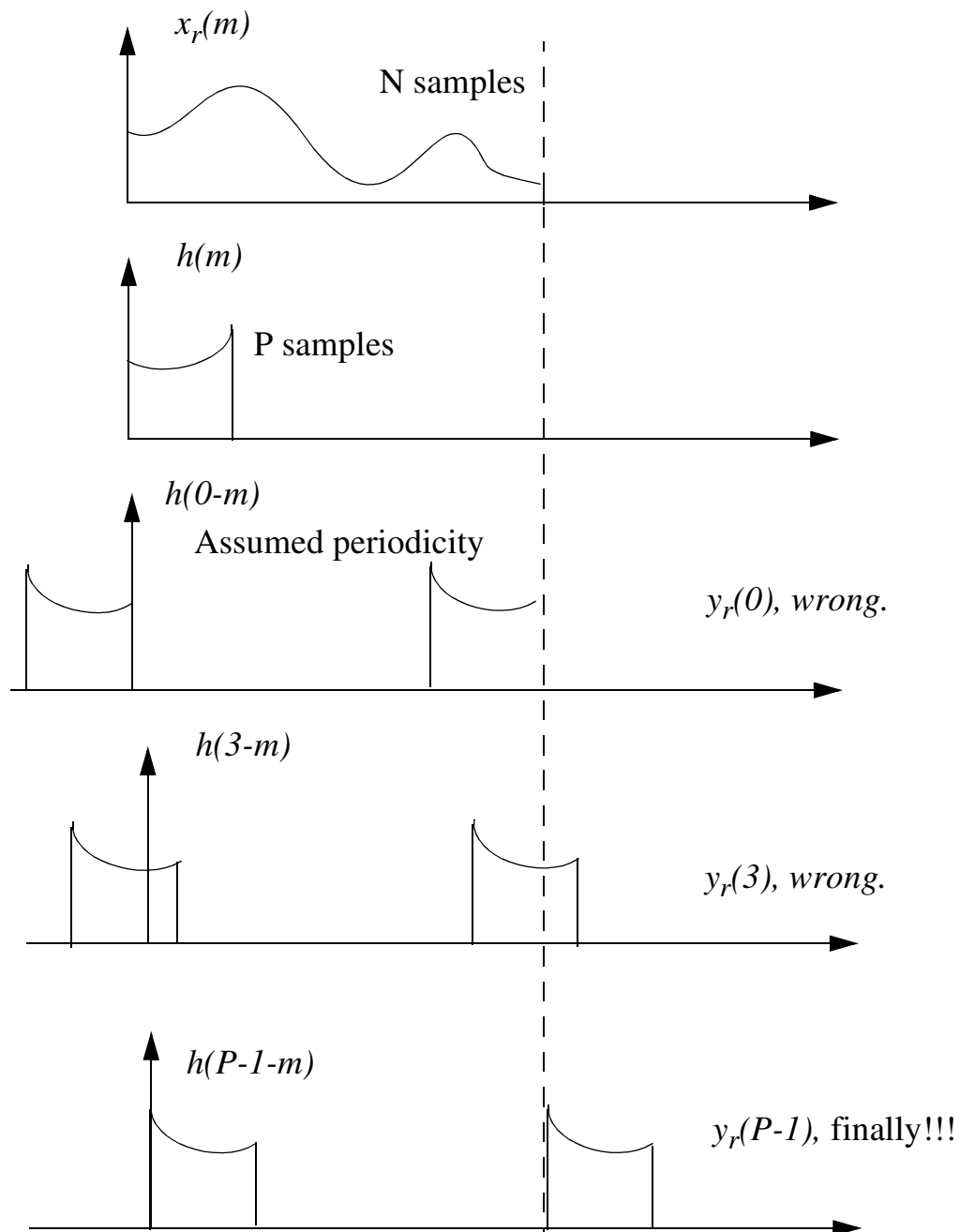


Example: If $P=14$ and we chose to use a 256-point FFT, then $L=243$. What is the average number of multiplications per sample in implementing this filter? What if $P=100$?

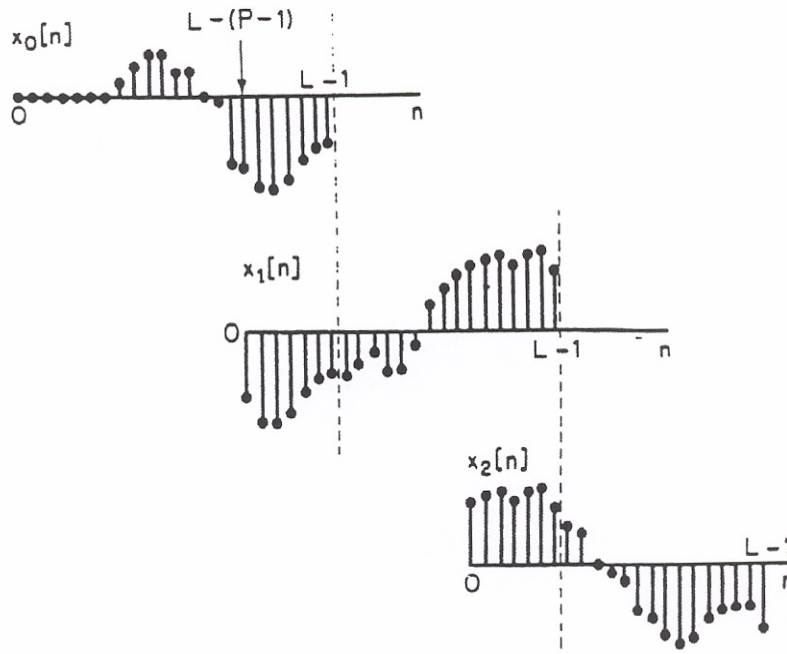
Overlap-and-Save Method

- The overlap-and-save method deals with the problem of using circular convolution to calculate linear convolution by throwing away garbage samples as a result of the assumed periodicity.

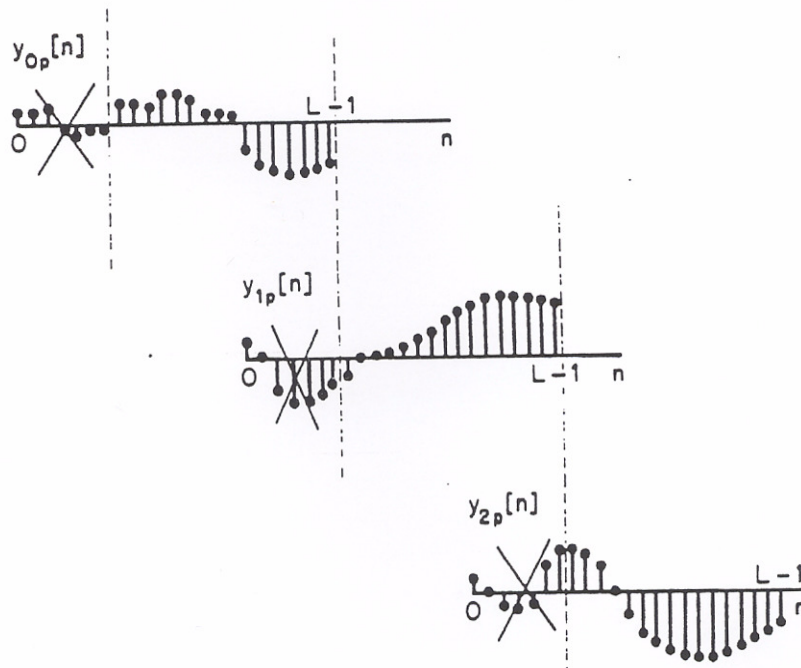
$$y_r(n) = \sum_{m=-P+1}^{N-1} x_r(m)h(n-m) \quad \text{where } N > P \text{ and } n = 0, 1, \dots, N-1.$$



- All the samples between $y_r(P-1)$ and $y_r(N-1)$ will be identical to the corresponding samples calculated using linear convolution.



(a)



- It seems that the overlap-and-save method requires less computation (there is no need for the final additions); why is that we still talk about it.