



2. *Linear Convolution*. Linearly convolve the signals shown in Figure 6 to determine  $y(n) = x(n) * h(n)$ . Let the length of  $x(n)$  and  $h(n)$  be  $L = 5$  and  $M = 3$ , respectively. Verify in fact that the length of  $y(n)$  is equal to  $L+M-1$ . You will have to use your past knowledge from signals and systems to graphically convolve this.

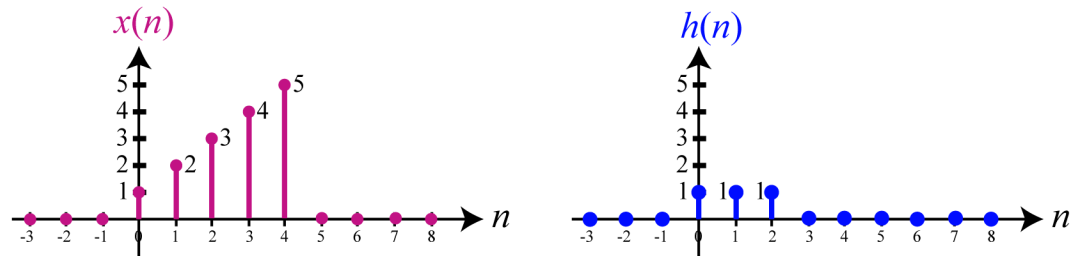


Figure 1: Two example signals to convolve.

3. *Circular Convolution.* The following digital signal processing is applied to  $x(n)$  and  $h(n)$  of Figure 1. Please assume appropriate zero-padding where necessary.

*Step 1:* Compute the  $N$ -FFT of  $x(n)$  to produce  $X(k)$  for  $k=0, 1, \dots, N-1$ .

*Step 2:* Compute the  $N$ -FFT of  $h(n)$  to produce  $H(k)$  for  $k=0, 1, \dots, N-1$ .

*Step 3:* Multiply  $Z(k) = X(k) H(k)$  for  $k=0, 1, \dots, N-1$ .

*Step 4:* Compute the  $N$ -IFFT of  $Z(k)$  to produce  $z(n)$  for  $k=0, 1, \dots, N-1$ .

- a. Draw  $z(n)$  for  $N = 16$ .

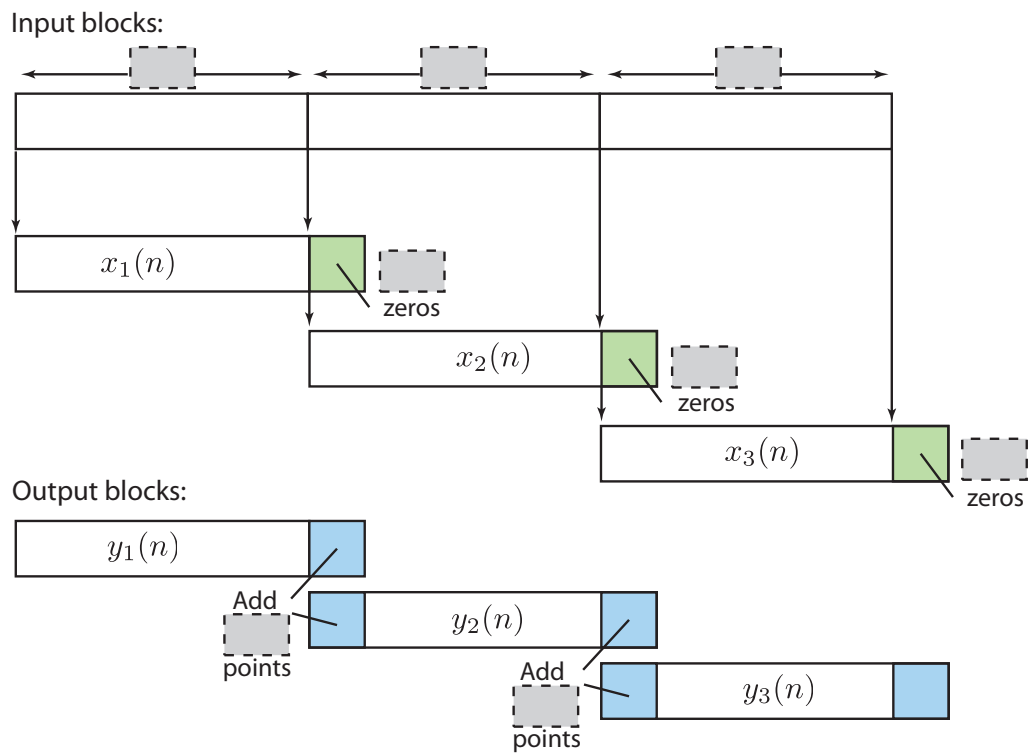
- b. Draw  $z(n)$  for  $N = 8$ .

- c. Draw  $z(n)$  for  $N = 4$ .

4. *Overlap-Add*. You would like to FIR filter in real-time an input stream  $x(n)$  to compute  $y(n)=x(n)*h(n)$  using the Overlap-Add approach. The FIR filter has impulse response  $h(n)$  of length  $M=5$ . You have available to you 128-FFT chips for processing.

a. What is the block length of the input?

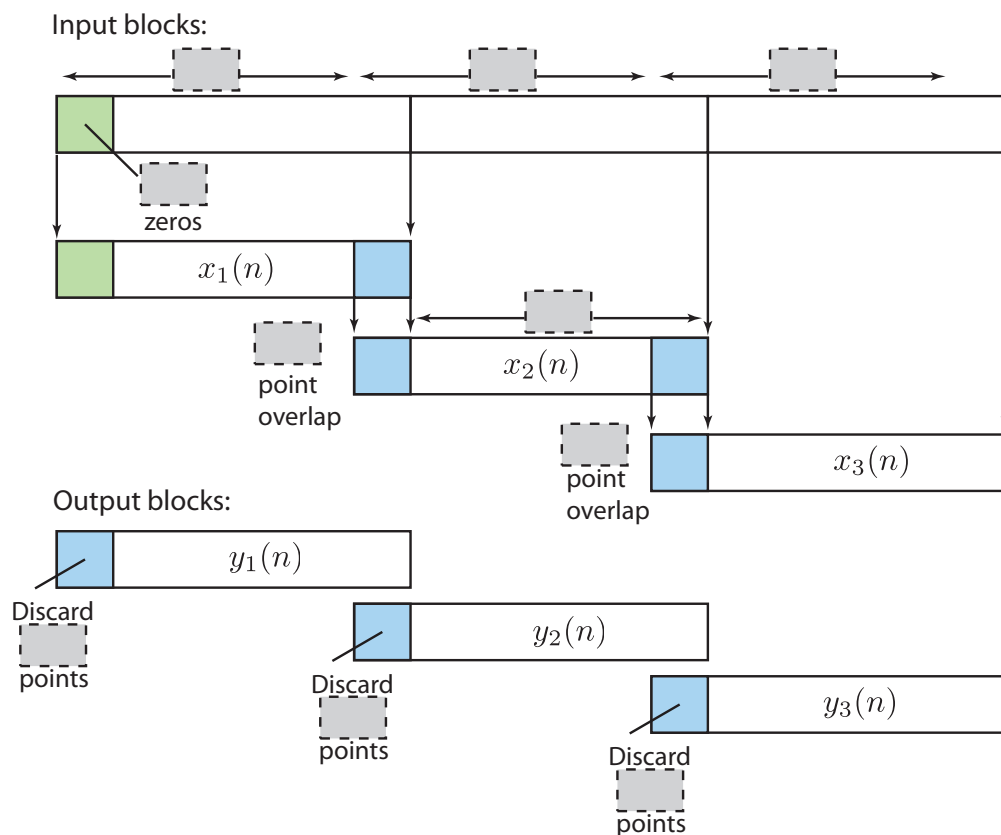
b. Consider the diagram below. Please assign the appropriate lengths inside the dashed rectangles to show how the input and output blocks will overlap.



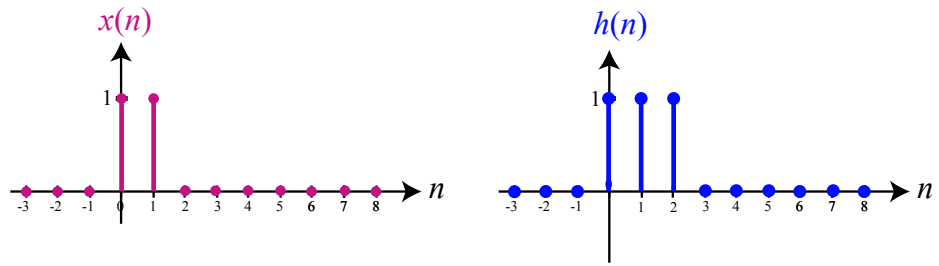
5. *Overlap-Save*. You would like to FIR filter in real-time an input stream  $x(n)$  to compute  $y(n)=x(n)*h(n)$  using the *Overlap-Save* approach. The FIR filter has impulse response  $h(n)$  of length  $M=5$ . You have available to you 128-FFT chips for processing.

a. What is the block length of the input?

b. Consider the diagram below. Please assign the appropriate lengths inside the dashed rectangles to show how the input and output blocks will overlap.



6. Consider the following signals  $x(n]$  and  $h(n]$ .



- a. Linearly convolve the signals shown in the following figure to determine  $y_L(n) = x(n) * h(n)$ . Let the length of  $x(n)$  and  $h(n)$  be  $L=2$  and  $M=3$ , respectively.

b. Determine the value of  $L+M-1$ . Verify that the support of  $y_L(n)$  is equal to  $L+M-1$ .

c. Assume the following digital signal processing is applied to the signals  $x(n)$  and  $h(n)$  this question. Please assume that a *radix-2 FFT* is employed and appropriate zero-padding where necessary is applied.

Step 1: Compute the  $N$ -FFT of  $x(n)$  to produce  $X(k)$  for  $k=0, 1, \dots, N-1$ .

Step 2: Compute the  $N$ -FFT of  $h(n)$  to produce  $H(k)$  for  $k=0, 1, \dots, N-1$ .

Step 3: Multiply  $Z(k) = X(k) H(k)$  for  $k=0, 1, \dots, N-1$ .

Step 4: Compute the  $N$ -IFFT of  $Z(k)$  to produce  $z(n)$  for  $k=0, 1, \dots, N-1$ .

What are the *three* smallest values of  $N$  such that  $z(n)$  will be exactly equal to the linear convolution result  $y_L(n)$ ?

7. *Overlap-Add versus Overlap-Save.* The purpose of this question is to give you some “food for thought” in comparing the complexity of the Overlap-Add and Overlap-Save methods. The Overlap-Add and Overlap-Save methods differ in some algorithmic details. What are these details and do they affect the complexity or memory of each method? Discuss the complexity and memory pros and cons of each method. Under what (possibly extreme) conditions is one lower complexity than the other? *Hint:* one way to start to answer this question would be to list the algorithmic steps of each method and then discuss how each stage is higher or lower complexity than the equivalent (if it exists) in the other method.