Grade 11/12 Math Circles

October 18 2023

Digital Signal Processing

Properties of Digital Filters

Last session we discussed two important properties of digital filters: *linearity* and *causality*.

Let's review!

Causality: A digital filter is *causal* if the output at time n only depends on the input up to time n. For example, the filter defined by

$$y[n] = x[n] - 2 \cdot x[n-1]$$

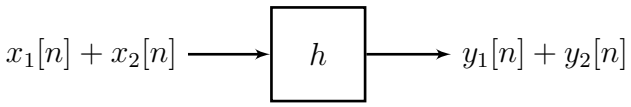
is a causal filter, whereas the filter defined by

$$y[n] = x[n+3]$$

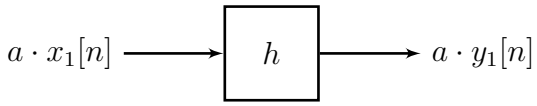
is not causal because $y[n]$ depends on the input at time $n+3$.

Linearity: A linear digital filter has the following two properties:

1. Additivity:

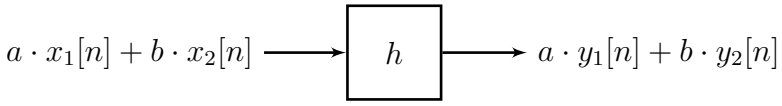


2. Homogeneity:



where a is a constant, $x_1[n]$ is a signal with corresponding output $y_1[n]$ and $x_2[n]$ is a signal with corresponding output $y_2[n]$.

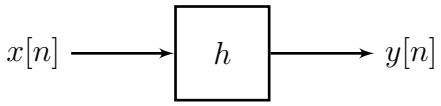
Combining these two properties, we say that a digital filter is linear if it satisfies the following:



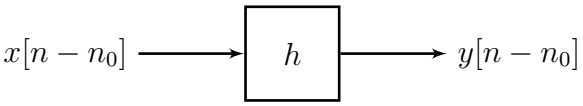
for *any* input signals $x_1[n]$ and $x_2[n]$ and any constants a and b.

Another important property of digital filters is something called *time-invariance*.

Time-Invariance: A filter is said time-invariant if it behaves the same way, regardless of when the input is applied. This means that if we have



then we also have



i.e. no matter when we start filtering the signal, the filter acts on the signal in the same way. In other words, if we delay the input signal by some amount of time n_0, we get the same output, just delayed (or shifted in time) by n_0. In order for a filter to be time-invariant, this needs to be true for all input signals $x[n]$, and all time delays n_0.

Example 1

Consider the filter defined by

$$y[n] = x[n] - 2 \cdot x[n-1].$$

Let's check if this filter is time-invariant.

Let $z[n] = x[n - n_0]$. When $z[n]$ is the input, the filter output is

$$\begin{aligned} z[n] - 2 \cdot z[n-1] &= x[n - n_0] - 2 \cdot x[n - 1 - n_0] \\ &= x[n - n_0] - 2 \cdot x[n - n_0 - 1] \\ &= y[n - n_0] \end{aligned}$$

Therefore the filter is time-invariant.

Now let's give it a try!

Exercise 1

Consider an input signal $x[n]$ and the corresponding time delayed signal $x[n - n_0]$.

Use these two signals to show that the digital filter defined by

$$y[n] = \frac{1}{2}(x[n] + x[n-1])$$

is time-invariant.

As we have seen, in order to show that a filter is time-invariant we need to show that the time-invariance property holds in the general case. On the other hand, in order to show that a filter is not time-invariant all we need to do is find at least one scenario where the time-invariance property is not true, i.e. find a counterexample.

Exercise 2

Consider the input signal $\delta[n]$ and the shifted signal $\delta[n+1]$.

Use these two signals as a counterexample to show that the digital filter defined by

$$y[n] = x[n^2]$$

is not time-invariant.

Linear Time-Invariant (LTI) Filters

A filter which is both linear and time-invariant is called a *linear time-invariant (LTI) filter*. Many real-world systems are modelled using LTI filters because

1. it is often a good/reasonable approximation, and
2. the analysis becomes very easy when working with LTI filters.

The Superposition Property of LTI Filters

As a result of being both linear and time-invariant, LTI filters have a useful property which is called the *superposition* property. This property is illustrated below.

$$a_0x[n] + a_1x[n-1] + a_2x[n-2] + \ldots \longrightarrow \boxed{LTI} \longrightarrow a_0y[n] + a_1y[n-1] + a_2y[n-2] + \ldots$$

We can see how this property might be useful if we write the input signal as a weighted sum of shifted delta functions. More on this later, first let's get some practice using the superposition property.

Exercise 3

Consider the LTI filter defined by

$$y[n] = 2x[n] + x[n-1]$$

and the signals $z[n] = \begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}$ and $z[n-1] = \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix}$. Compute $y[n]$ for the input signal $x[n] = z[n] + z[n-1]$ by

a) evaluating the filter response to $x[n]$ directly, and

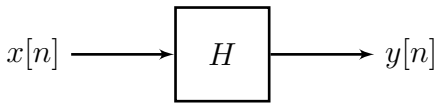
b) using the superposition property.

The Impulse Response of an LTI Filter

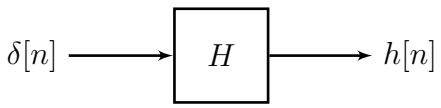
As a result of the superposition property, we can show that any LTI filter is completely characterized by its impulse response. What this means is that if we know the impulse response of the filter, we can use this to determine the output for any input signal.

Remember, the impulse response is the filter output when the input is the delta function (i.e. $\delta[n]$), and we call this $h[n]$.

Consider an LTI filter which we will name H, i.e.



which has impulse response $h[n]$, i.e.



We can write the action of filter H acting on the input signal $x[n]$ in more compact notation as

$$y[n] = H(x[n]).$$

Now, since H is an LTI filter, we know that it is both linear and time-invariant. In this new notation, this means

1. $H(ax_1[n] + bx_2[n]) = ay_1[n] + by_2[n]$ (linearity)
2. $H(x[n - n_0]) = y[n - n_0]$ (time-invariance)

where a and b are constants, and $y[n] = H(x[n])$, $y_1[n] = H(x_1[n])$, and $y_2[n] = H(x_2[n])$.

Now, recall from last session that we can write any signal as a weighted sum of shifted delta functions, i.e. we can write the input signal $x[n]$ as

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n - k].$$

What is the filter response to the signal $x[n]$?

$$\begin{aligned} y[n] = H(x[n]) &= H\left(\sum_{k=-\infty}^{\infty} x[k]\delta[n - k]\right) \\ &= \sum_{k=-\infty}^{\infty} x[k]H\left(\delta[n - k]\right) \quad \text{(by linearity)} \\ &= \sum_{k=-\infty}^{\infty} x[k]h[n - k] \quad \text{(by time-invariance)} \end{aligned}$$

This sum is called the *convolution* of $x[n]$ (the input) with $h[n]$ (the impulse response).

We write the convolution of $x[n]$ with $h[n]$ as

$$y[n] = x[n] * h[n].$$

where the $*$ symbol indicates the convolution (not multiplication).

Convolution

As we have just seen, the output of an LTI digital filter can be expressed quite nicely as the convolution of the input signal with the impulse response of the filter. The convolution operation is very important in digital signal processing, however it also comes up in many other applications, such as probability and statistics, analog signal processing (there is also a continuous time version of convolution), differential equations, and physics.

Using the output of an LTI filter as motivation, let's take a look at the different ways we can compute the convolution and see what it means to *convolve* two digital signals.

Consider the LTI filter defined by

$$y[n] = x[n] - 2x[n-1] + 3x[n-2].$$

What is the response to the input signal $x[n] = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$?

Remembering that in this notation, the first entry corresponds to $n = 0$ and that $x[n] = 0$ everywhere else we see that the signal $x[n]$ looks like:

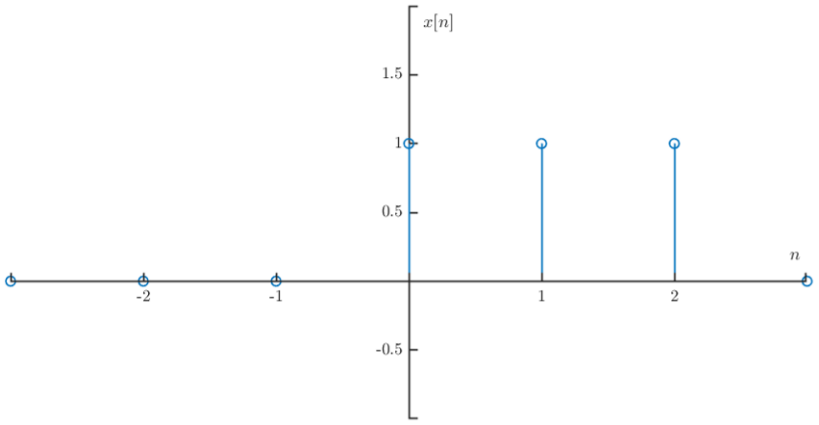


Figure 1: The signal $x[n]$.

The most obvious approach is to determine the filter output by *direct evaluation of* $y[n]$.

Direct Evaluation

Using the definition of the filter directly we find that

$$\begin{aligned}
y[0] &= x[0] - 2x[-1] + 3x[-2] = 1 - 0 + 0 = 1 \\
y[1] &= x[1] - 2x[0] + 3x[-1] = 1 - 2(1) + 3 = -1 \\
y[2] &= x[2] - 2x[1] + 3x[0] = 1 - 2(1) + 3(1) = 2 \\
y[3] &= x[3] - 2x[2] + 3x[1] = 0 - 2(1) + 3(1) = 1 \\
y[4] &= x[4] - 2x[3] + 3x[2] = 0 - 0 + 3(1) = 3
\end{aligned}$$

which could also be written as

$$y[n] = \begin{bmatrix} 1 & -1 & 2 & 1 & 3 \end{bmatrix}.$$

Another way we can determine the filter output is to determine the impulse response and directly evaluate the *convolution sum*.

Convolution Sum

From the previous section we know that the filter output is given by

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

where $x[n] = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ is the input signal from before and $h[n]$ is the impulse response of the filter.

As a result, in order to evaluate $y[n]$ using the convolution sum the first thing we need to do is determine $h[n]$ for our particular LTI filter.

Exercise 4

Determine $h[n]$, i.e. determine the impulse response of the filter defined by

$$y[n] = x[n] - 2x[n-1] + 3x[n-2].$$

CHALLENGE: Without doing any calculations, could you write down the impulse response of the filter $y[n] = ax[n] + bx[n-1] + cx[n-2]$, where a, b and c are constants?

Using the result from Exercise 4, we have

$$h[n] = \begin{bmatrix} 1 & -2 & 3 \end{bmatrix}.$$

Since $x[n]$ only has three non-zero entries, the convolution sum becomes

$$\begin{aligned} y[n] &= \sum_{k=-\infty}^{\infty} x[k]h[n-k] \\ &= x[0]h[n] + x[1]h[n-1] + x[2]h[n-2] \\ &= (1)h[n] + (1)h[n-1] + (1)h[n-2] \\ &= h[n] + h[n-1] + h[n-2]. \end{aligned}$$

We could draw out these three signals to add them together, or write this as

$$\begin{aligned} & \begin{bmatrix} 1 & -2 & 3 & 0 & 0 \end{bmatrix} \\ + & \begin{bmatrix} 0 & 1 & -2 & 3 & 0 \end{bmatrix} \\ + & \begin{bmatrix} 0 & 0 & 1 & -2 & 3 \end{bmatrix} \\ y[n] = & \begin{bmatrix} 1 & -1 & 2 & 1 & 3 \end{bmatrix} \end{aligned}$$

This agrees with what we found previously by directly evaluating the filter output.

The first two methods for computing the filter output (i.e. the convolution of $x[n]$ with $h[n]$) can be quite tedious for longer signals. The next method we will look at is a graphical method in which we *flip and slide* one signal.

Flip & Slide

Once again, let's consider the definition of the convolution sum

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k].$$

What do the signals $x[k]$ and $h[n-k]$ look like?

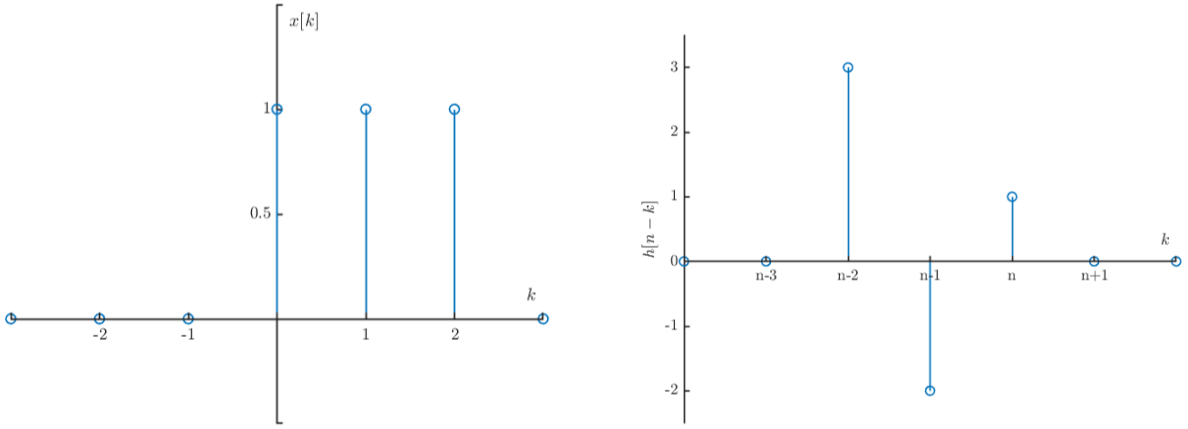


Figure 2: The signals $x[k]$ and $h[n-k]$.

The signal $h[n-k]$ is the impulse response flipped and shifted to the right by n (see Lesson 1 for how to combine mathematical operations on signals to convince yourself that this is true). For example, computing $y[0]$ looks like:

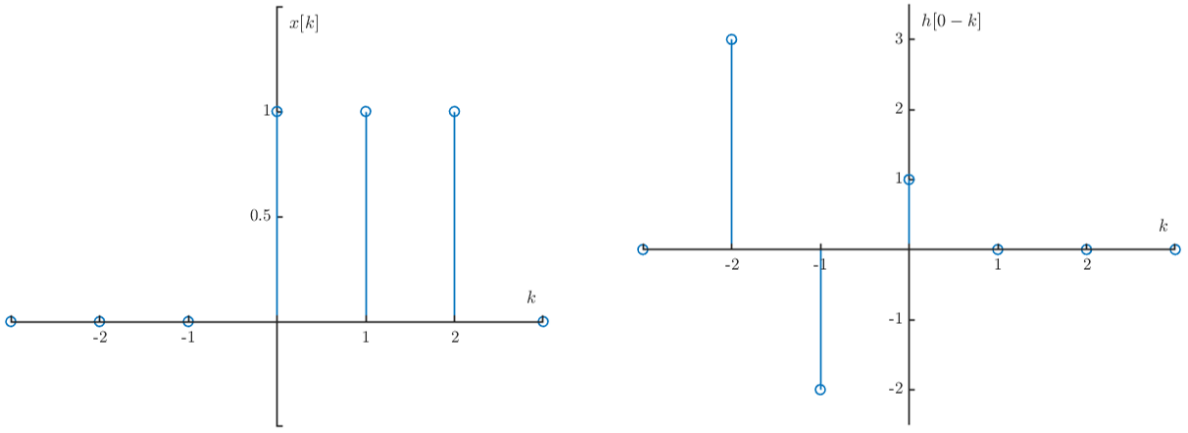


Figure 3: The signals $x[k]$ and $h[0-k]$.

and we find $y[0] = 1 \cdot 1 = 1$.

Similarly, computing $y[1]$ looks like:

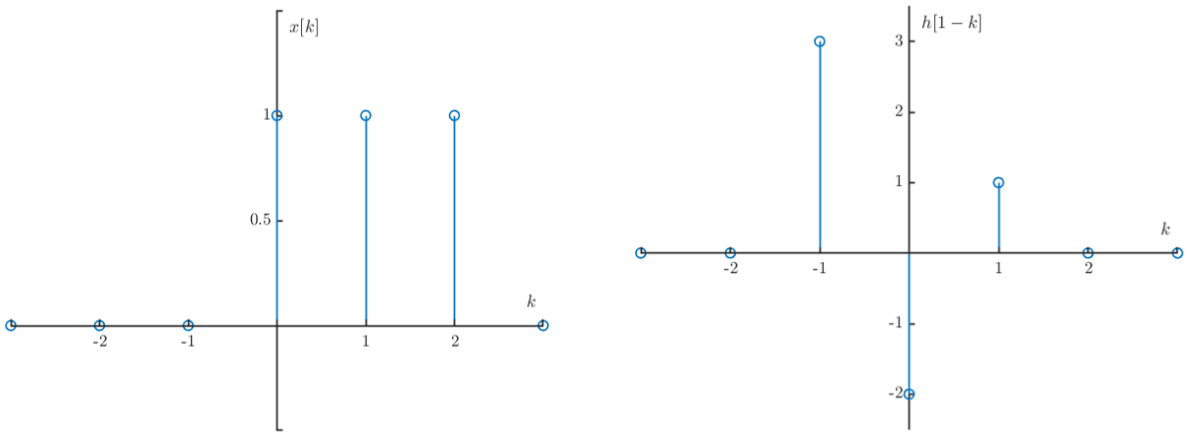


Figure 4: The signals $x[k]$ and $h[1-k]$.

and we find $y[1] = -2 \cdot 1 + 1 \cdot 1 = -1$. We continue this process, sliding h to the right by one each time until the two signals no longer overlap and we find

$$y[n] = \begin{bmatrix} 1 & -1 & 2 & 1 & 3 \end{bmatrix}$$

as before.

This video by 3Blue1Brown provides a nice animated visualization showing how the convolution is computed by flipping and sliding one signal.

The flip and slide method provides us with some nice intuition for what is happening when we evaluate a convolution, however it is still quite tedious to compute a convolution in this way.

We can use something called a *convolution array* to carry out these computations much faster.

Convolution Array

Once again, consider $x[n] = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ and $h[n] = \begin{bmatrix} 1 & -2 & 3 \end{bmatrix}$. The convolution $x[n] * h[n]$ can be computed by setting up the following table (i.e. the convolution array) with $x[n]$ on the top and $h[n]$ on the left.

	1	1	1
1			
-2			
3			

We fill in the entries of the table by computing products of $x[n]$ and $h[n]$, as follows

	1	1	1
1	**1**		
-2			
3			

$\rightarrow$

	1	**1**	1
1	1	**1**	
-2			
3			

$\rightarrow$

	1	1	1
1	1	1	
-2	**-2**		
3			

$\ldots$

Filling in the rest of the table we have:

	1	1	1
1	1	1	1
-2	-2	-2	-2
3	3	3	3

Now we can evaluate the convolution, i.e. $y[n] = x[n] * h[n]$ by summing over the diagonals of the table. For example, to compute $y[0]$ we look at the first diagonal

	1	1	1
1	**1**	1	1
-2	-2	-2	-2
3	3	3	3

to find $y[0] = 1$.

Next, to find $y[1]$ we look at the second diagonal

	1	1	1
1	1	**1**	1
-2	**-2**	-2	-2
3	3	3	3

to find that $y[1] = -2 + 1 = -1$. Repeating this process, we once again find that

$$y[n] = \begin{bmatrix} 1 & -1 & 2 & 1 & 3 \end{bmatrix}.$$

For finite signals, the convolution array is the fastest (and easiest) method for evaluating the convolution of two discrete-time signals.

Now it's your turn!

Exercise 5

Evaluate the convolution of

$$a[n] = \begin{bmatrix} 2 & -1 & 1 \end{bmatrix}$$

with

$$b[n] = \begin{bmatrix} 3 & 4 & 1 \end{bmatrix},$$

i.e. evaluate

$$a[n] * b[n].$$

a) graphically using the flip and slide method,

b) and using the convolution array method.

Which method do you prefer?