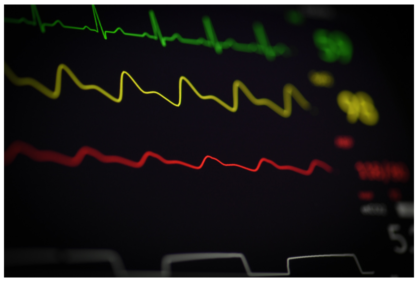
Grade 11/12 Math Circles

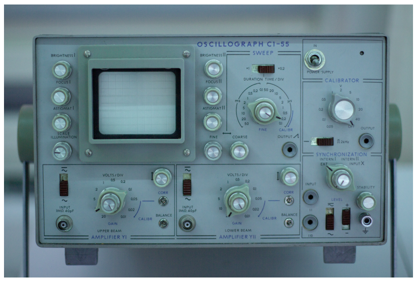
October 4 2023

Digital Signal Processing

What is a Signal?

Put simply, *signals* are a way of conveying information that changes over time. In order to measure signals, we need sensors - we can think of a signal as the data output from some sort of a sensor, i.e. the temperature from a thermometer, electrical heart activity measured by an ECG, the voltage or current measured from an electrical circuit, etc... We can even think of our bodies as a type of sensor which receive and react to signals - our ears receive and process sound waves which allow us to hear things, and our eyes take in visual data which allows us to see the world around us.



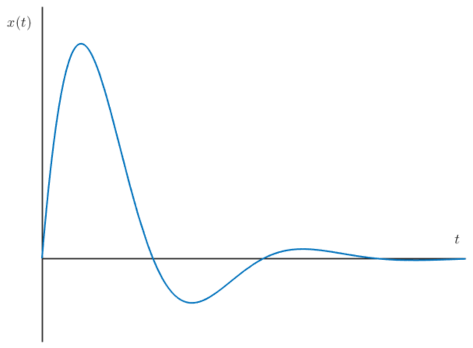


Signal processing refers to the analysis and manipulation of signals. Signal processing systems are present in most (if not all) of the technologies we use today. Signal processing can refer to a number of different applications, including: signal enhancement (such as filtering signals to remove noise), data compression (such as compressing audio or image files for storage), and things like speech recognition (i.e. the talk to text feature on your smartphone).

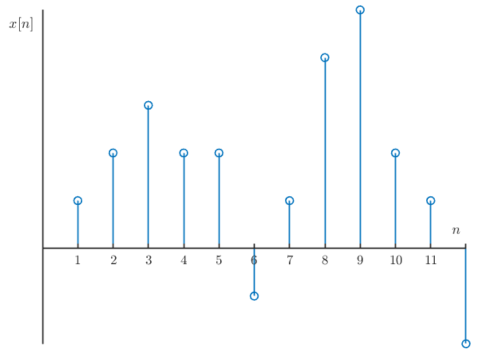
Analog vs. Digital Signals

A signal can be either *analog* or *digital*. An analog signal is a *continuous-time* signal. These are usually electrical signals, or signals produced by some physical process, such as the sound waves generated by someone speaking.

A digital signal is a *discrete-time* signal, which means that we can only measure the signal at some finite number of time intervals (discrete time points), instead of continuously over time. Most signals we come across today are digital signals, because computers can only measure things in discrete time. For example, if we wanted to record an audio signal (say someone singing) so that we could listen to it on our computer, the original analog signal would need to be converted into a digital signal, that is it would need to be *digitized*.



(a) An analog signal, $x(t)$.



(b) A digital signal, $x[n]$.

Figure 1: The difference between an analog and a digital signal.

Stop and Think

Can you think of some more examples of digital signals? What about analog signals?

Digitizing Analog Signals

An analog signal is digitized by *sampling* the original signal at discrete time points using something called an *analog-to-digital converter (ADC)*.

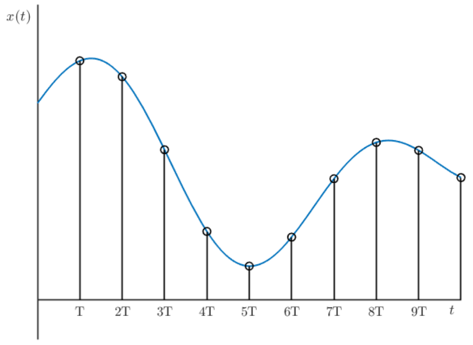


Figure 2: Sampling of an analog signal, $x[n] = x(nT)$

From the continuous-time analog signal $x(t)$ we define the discrete time signal $x[n]$, where n is an integer, by sampling at evenly spaced intervals. That is, $x[n] = x(nT)$, where T is the *sampling interval*. The *sampling rate*, or *sampling frequency*, is $f_s = \frac{1}{T}$.

Exercise 1

Consider the continuous-time signal defined by $x(t) = t$, $0 \leq t \leq 12$.

Compute and sketch the discrete-time (digital) signal $x[n]$ with sampling intervals:

a) $T = 1$, and

b) $T = 2$.

Mathematical Operations on Digital Signals

In order to discuss digital signal processing, we first need to define some of the basic mathematical operations which can be performed on digital signals.

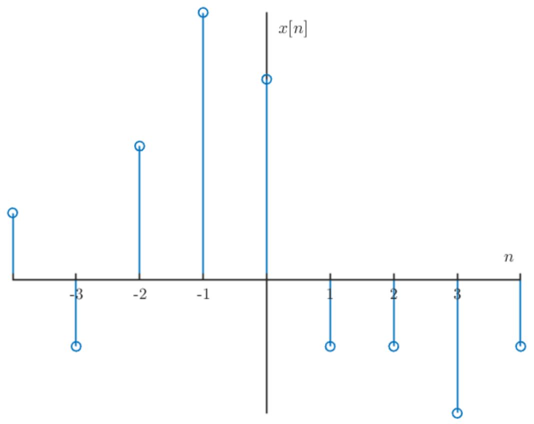
Digital signals are represented by *sequences* of numbers, denoted by $x[n]$, where n is an integer. We can think of this as representing a table of values for n and $x[n]$, like the following:

n	-3	-2	-1	0	1	2	3
$x[n]$	1	2	-1	3	4	-2	-1

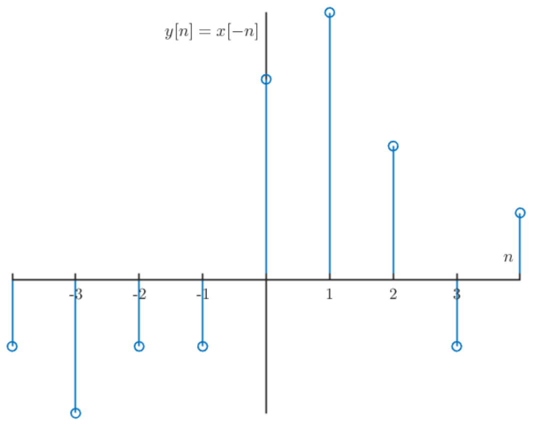
The signal above is *finite*, in that it is only defined for a finite number of values of n. In order to deal with signals of different lengths, we assume that $x[n] = 0$ everywhere that it is not defined. This is called *zero-padding*.

The basic transformations we can apply to a signal are flipping, scaling, and shifting.

Flipping: Given a digital signal $x[n]$ we can define a new signal $y[n] = x[-n]$ by flipping the signal across the $n = 0$ axis.



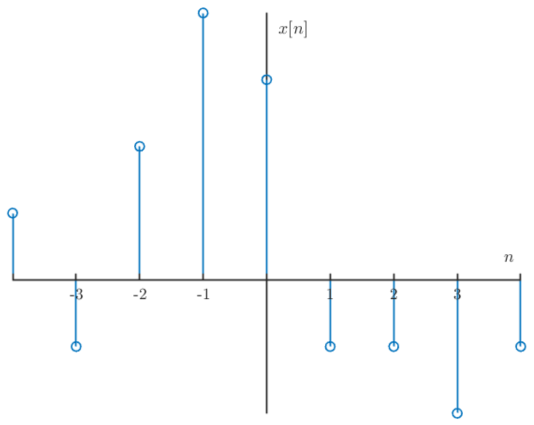
(a) A digital signal $x[n]$.



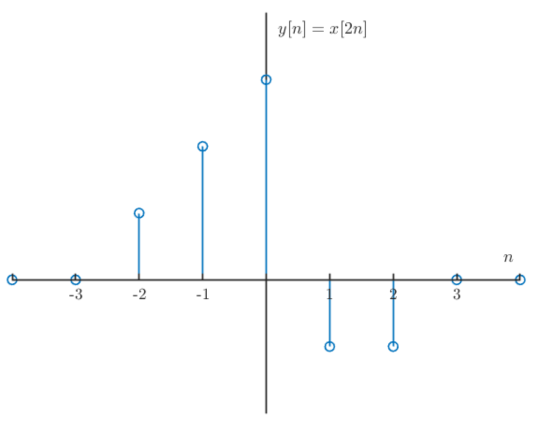
(b) The flipped signal $y[n] = x[-n]$.

Figure 3: Flipping a digital signal.

Scaling: We can also stretch and compress signals by scaling the argument. First, let's determine $y[n] = x[2n]$. Notice that when we compute $y[n] = x[2n]$ we lose some samples from the original signal $x[n]$.



(a) A digital signal $x[n]$.



(b) The scaled (compressed) signal $y[n] = x[2n]$.

Figure 4: Scaling (compressing) a digital signal.

Now, let's compute $y[n] = x[\frac{n}{3}]$. If n is not a multiple of 3, for example $n = 1$, we don't have a value for $y[1] = x[\frac{1}{3}]$ (this is not defined), so we fill in these values with zeros.

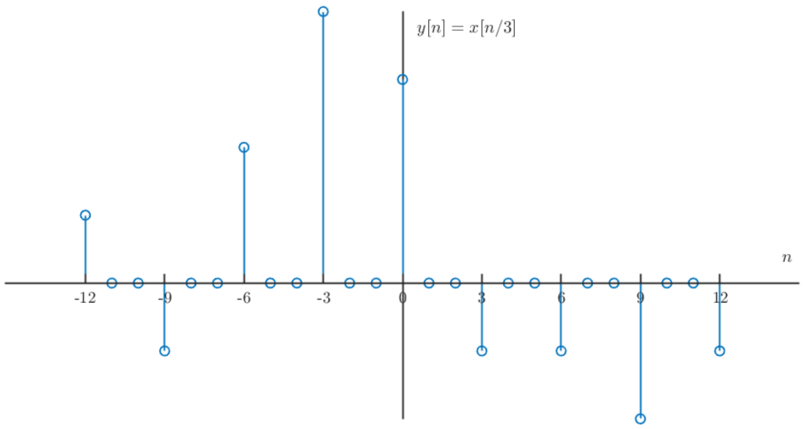
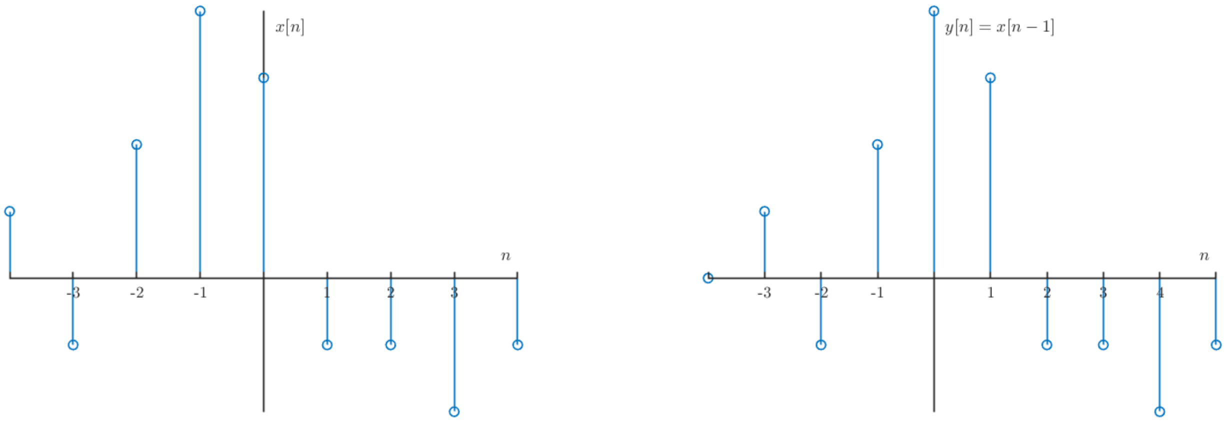


Figure 5: The scaled (stretched) signal $y[n] = x[n/3]$.

Shifting: We can shift signals forward and backward in time by shifting the argument. The shifted signal $y[n] = x[n-1]$ is a *time-delayed* version of $x[n]$.



(a) A digital signal $x[n]$.

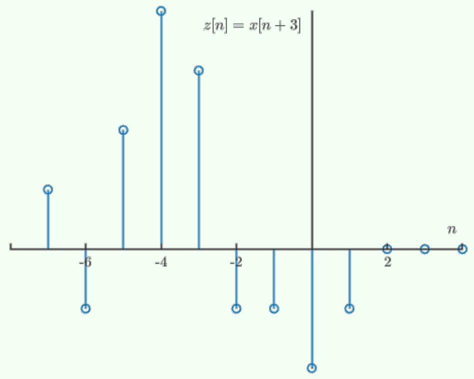
(b) The shifted signal $y[n] = x[n-1]$.

Figure 6: Shifting a digital signal.

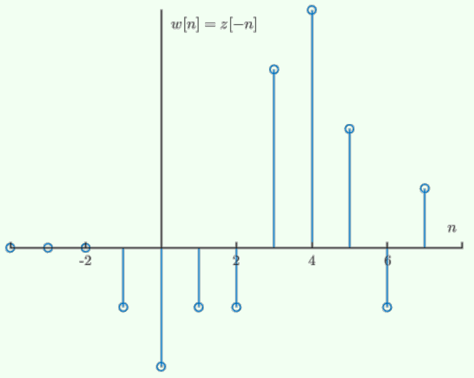
Finally, we can combine all of these operations. Just like when we apply transformations to continuous-time functions - we need to be careful about the order in which we apply each operation!

Example 1

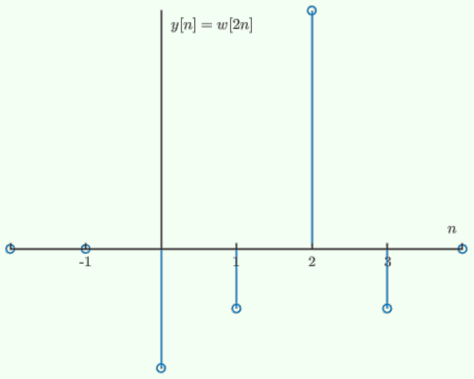
Let's determine the signal $y[n] = x[-2n+3]$. First, we **shift** the signal to compute $z[n] = x[n+3]$.



Next, we **flip** the signal to compute $w[n] = z[-n] = x[-n+3]$.



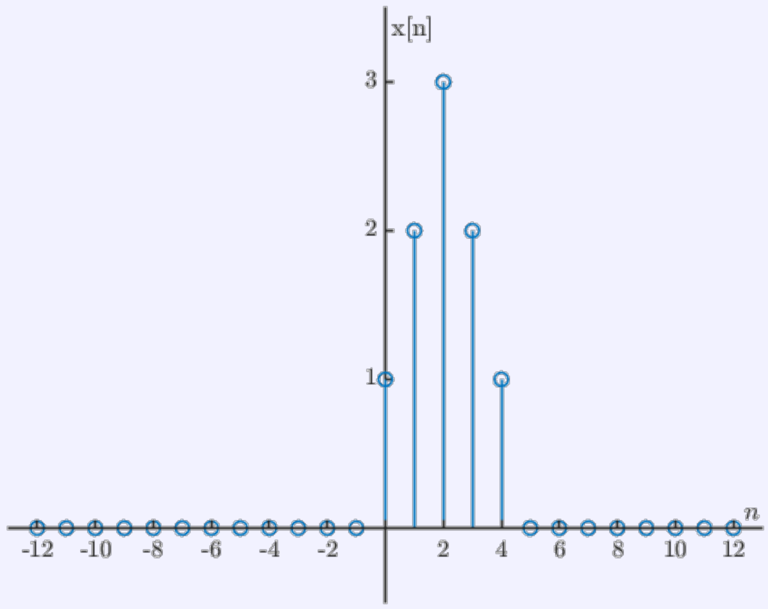
And finally, we **scale** the signal so that $y[n] = w[2n] = z[-2n] = x[-2n+3]$.



Now it's your turn!

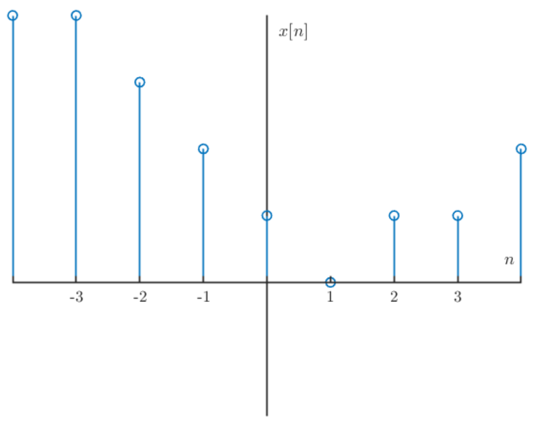
Exercise 2

Let $x[n]$ be the following signal.

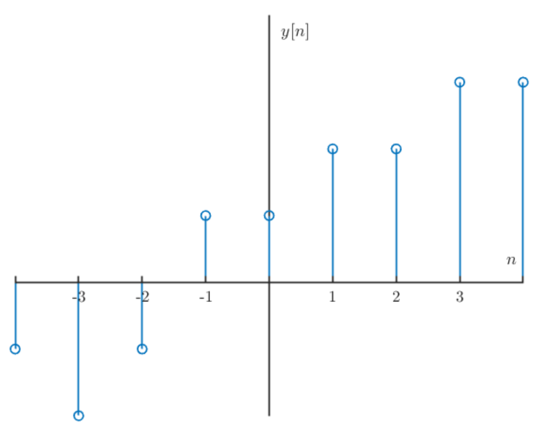


Sketch the transformed signal $y[n] = x[\frac{-n}{2} - 2]$ by shifting, flipping, and then scaling the original signal. It may be helpful to sketch each intermediate step.

Superposition: In addition to operations on individual signals, we can also add (and subtract) signals with one another. We add (and subtract) signals "element-by-element", i.e. for each index n, we evaluate $z[n] = x[n] + y[n]$.



(a) A digital signal $x[n]$.



(b) Another digital signal $y[n]$.

Figure 7: Two digital signals, $x[n]$ and $y[n]$.

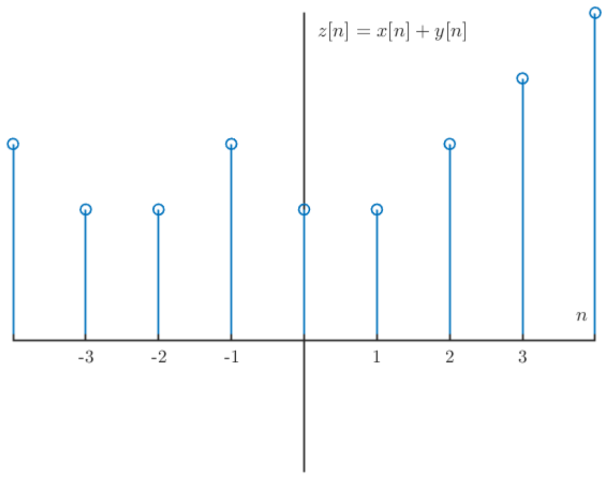


Figure 8: The sum $z[n] = x[n] + y[n]$.

Special Signals

There are two special signals which are often used in digital signal processing, *the unit step function*, and *the delta function (unit impulse)*.

The Unit Step Function

The unit step function, $u[n]$, is defined as:

$$u[n] = \begin{cases} 0 & \text{if } n < 0 \\ 1 & \text{if } n \geq 0. \end{cases}$$

We can think of this as a signal which "turns on" when $n = 0$ and never turns off, like a light that is switched on (and never turned off!).

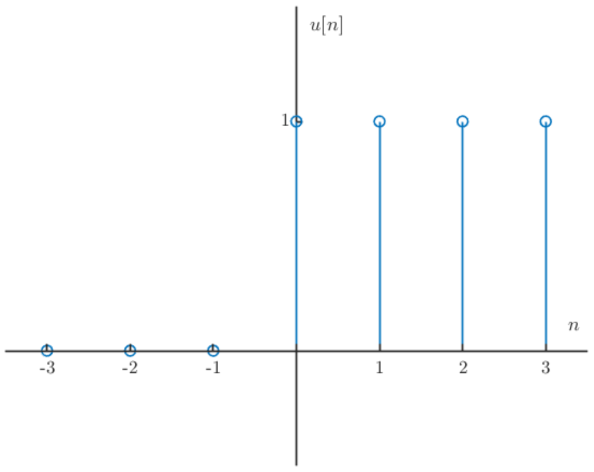


Figure 9: The unit step function, $u[n]$.

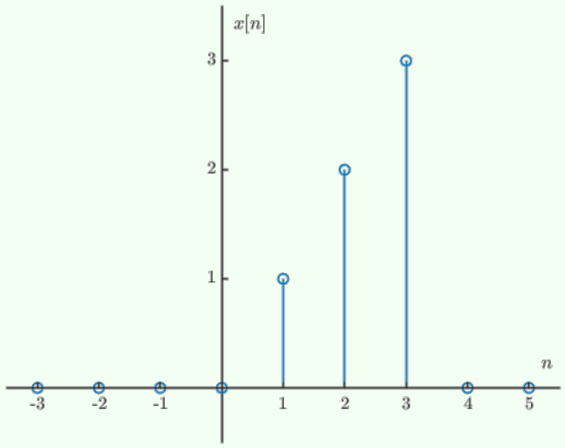
We can use the unit step function to write down finite signals which turn on (and off) at prescribed times in a nice mathematical way.

Example 2

Consider the signal

$$x[n] = \begin{cases} 0 & \text{if } n < 1 \\ n & \text{if } 1 \leq n < 4 \\ 0 & \text{if } n \geq 4 \end{cases}$$

which looks like:



We see that $x[n]$ "turns on" at $n = 1$, and "turns off" at $n = 4$. We can express this in terms of the unit step function as follows:

$$\begin{aligned} x[n] &= n \cdot u[n-1] - n \cdot u[n-4] \\ &= n(u[n-1] - u[n-4]). \end{aligned}$$

The Delta Function (a.k.a. The Unit Impulse)

The delta function, commonly referred to as the unit impulse, is denoted $\delta[n]$, and is defined as:

$$\delta[n] = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n \neq 0. \end{cases}$$

The delta function looks like a spike of height one at $n = 0$:

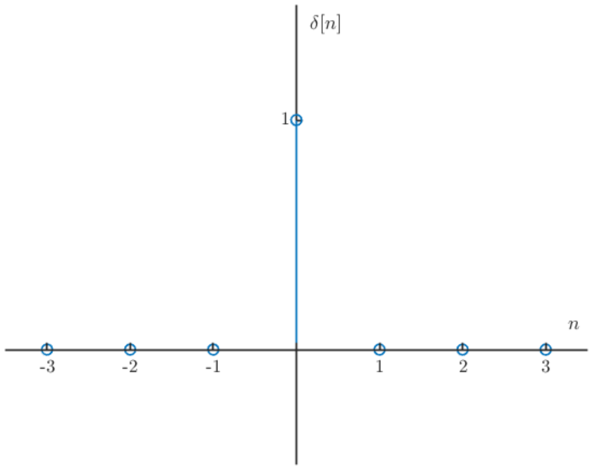


Figure 10: The unit impulse, $\delta[n]$.

Exercise 3
Write the delta function, $\delta[n]$, in terms of a sum (or difference) of shifted unit step functions.

The delta function has a number of nice properties which make it useful in the study of digital signal processing, including the fact that we can write any digital signal as a weighted sum of shifted delta functions.

Before we do this, let's introduce some (maybe new) notation for writing sums!

Summation (Sigma) Notation

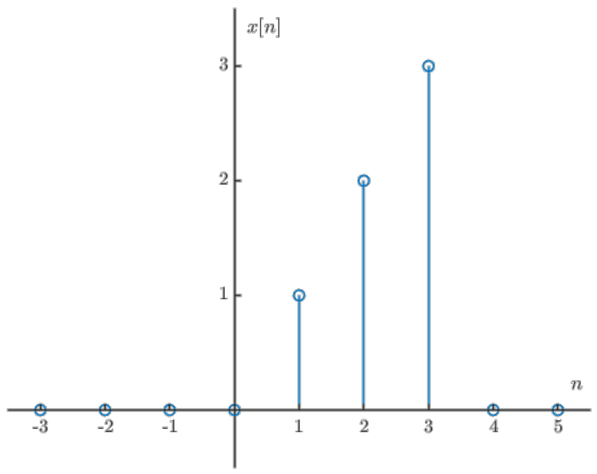
The symbol $\sum$ (sigma) is used to represent sums. For example, the sum

$$1+2+3+4+5+6+7+8+9+10=\sum_{k=1}^{10} k.$$

Using sigma notation we have a nice way to represent infinite sums as well, for instance

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\ldots=\sum_{k=1}^{\infty}\frac{1}{k}.$$

Given the signal $x[n]$ from before,



we see that $x[n]=1\cdot\delta[n-1]+2\cdot\delta[n-2]+3\cdot\delta[n-3]$. In general we can write

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

This may not seem all that useful right now, but in our next session we will see why it is useful to write signals in this way.

Digital Filters

A digital filter is a system which performs mathematical operations on a digital signal in order to reduce or enhance certain aspects of that signal. We can represent digital filters using *block diagrams*, like the one shown below.

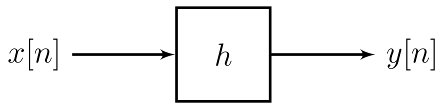


Figure 11: The signal $x[n]$ is processed using filter h to produce the output signal $y[n]$.

One of the simplest, and also most well-known digital filters is called a *moving average filter*. The moving average filter is a *smoothing* or a *de-noising* filter.

What is noise? In the context of signals and signal processing, noise refers to random fluctuations in experimental measurements. There are many sources of noise in physical measurements, such as building vibrations, electrical power fluctuations, stray radiation from nearby electrical equipment, static electricity, interference from radio and TV transmissions, turbulence in the flow of gases or liquids, random thermal motion of molecules, among other things. A de-noising filter is designed to filter out or remove some of the noise, while (mostly) leaving the original signal intact.

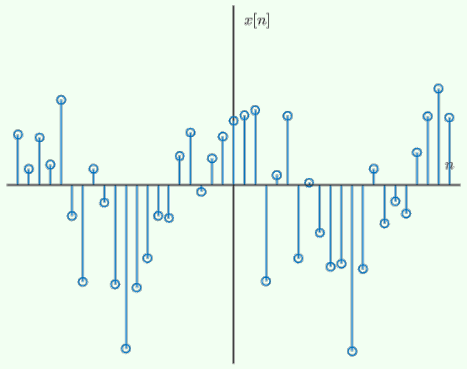
The output of the moving average filter is defined by

$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} x[n-k].$$

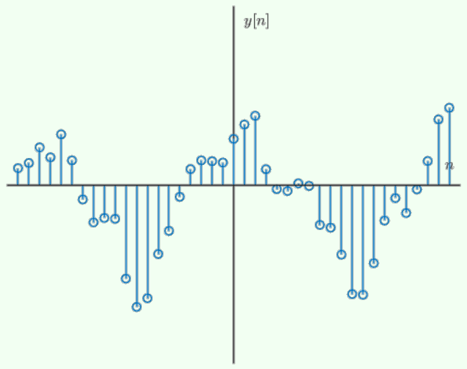
Each output value is defined as the average of the current input and the $N-1$ prior samples. The number of samples included in the average, N, is called the *window size*. The key assumption behind the design of the moving average filter is that the noise is random and zero mean (i.e. averages to zero).

Example 3

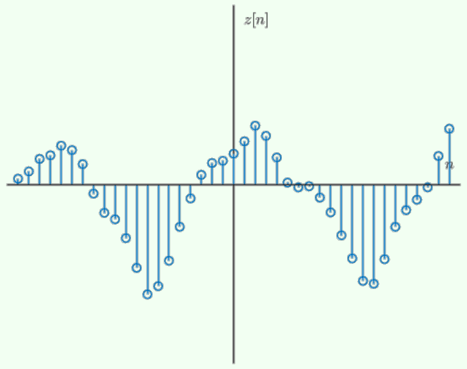
Consider the signal $x[n]$:



Applying the moving average filter with $N = 3$ results in $y[n]$:



Applying the moving average filter a second time results in $z[n]$:



Another type of average is the *median*. The median of a set of numbers is the middle number when the numbers are sorted from smallest to largest, i.e.

$$\text{median}(5, 2, 3) = \text{median}(2, 3, 5) = 3.$$

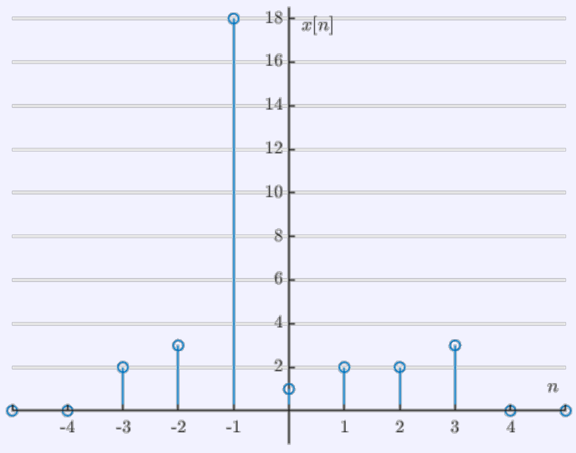
Like the moving average filter, the median filter is defined by

$$y[n] = \text{median}\left(x[n], x[n-1], ..., x[n-(N-1)]\right).$$

The output value is the median of the current input and the $N-1$ prior inputs. Once again, N defines the window size of the filter.

Exercise 4

Let $x[n]$ be the following signal.



Sketch the result of

a) applying the moving average filter to $x[n]$ with $N = 3$, and

b) applying the median filter to $x[n]$ with $N = 3$.

What differences do you notice?

The Impulse Response

Filters are often characterized by their *impulse response*, meaning the output of the filter when the input signal is the delta function (or unit impulse). The impulse response of the moving average filter is

$$\begin{aligned} h[n] &= \frac{1}{N}\sum_{k=0}^{N-1} \delta[n-k] \\ &= \begin{cases} \frac{1}{N} & \text{if } 0 \leq n \leq N-1 \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Exercise 5

An exponential moving average filter is defined by

$$y[n] = \alpha x[n] + (1-\alpha)y[n-1]$$

where $y[n]$ is the current output, $y[n-1]$ is the previous output, and $x[n]$ is the current input. The parameter α is a number between 0 and 1.

Assuming that $x[n] = 0$ when $n < 0$,

a) Find an expression for the output $y[n]$ in terms of only the previous input values, i.e. the values $x[n-k]$.

b) Using your result from a), determine the impulse response of the exponential moving average filter.