Discrete Convolution and Some Intuition Building for Digital Filters

This tech note concerns the ***convolution*** of a digital signal with a convolution kernel. We'll see that the kernel can perform many familiar operations.

1 Quick bit of theory

Let's start with a sampled digital sequence $x[n]$. This could be an audio recording, an electrical measurement, and optical one, mechanical one such as acceleration, you name it. Let's take the latter as a motivating example, where the acceleration measurements $x[n]$ come from an accelerometer on a FitBit of simliar device that is used to count steps. In order to get an accurate step count, we will likely need to do some sort of filtering operation. The way this is done in practice is by convolution:

$$y[n] = \sum_{k=0}^{N-1} b[k]\, x[n-k] - \sum_{m=1}^{M-1} a[m]\, y[n-m] \tag{1}$$

where $x[n]$ is the input sequence (stuff we measure, e.g. accelerometer readings), and $y[n]$ is the output of our digital system (e.g. filtered acceleration signal). We assume the weights $b[k]$ and $a[m]$ to be constants. The first summation term just computes some weighted average of using the most recent N terms of the input sequence. The second summation is the ***recursive*** term that uses the most recent M previous outputs of the filtered sequence in computing the new filtered output.

Note that the filter command in matlab looks like this:

y = filter(b,a,x)

You might have previously asked yourself "What the heck are b and a anyway?!" Now you can already start to see where they come from. Just glance back up at Eqn 1.

We'll work with non-recursive systems today, to simplify things a bit. Thus, we will set all $a[m]$ terms in Eqn 1 equal to 0, so we just have our output sequence $y[n]$ as a weighted sum of the input sequence $x[n]$:

$$y[n] = \sum_{k=0}^{N-1} b[k]\, x[n-k] = b * x = x * b \tag{2}$$

where $*$ denotes the convolution operator

Notice that convolution just takes two sequences of numbers b and x and produces another sequence of numbers y. The magic is in setting the coefficient $b[k]$ to do some desired operation. Speaking of, time to look at some familiar and very practical examples below!

2 Quick bit of practice

For each of the examples below using various coefficients b do the following

1. Make a stem-plot of the coefficients $b[k]$.
2. BEFORE you start calculating like crazy, try to get some intuition for what math operation each of these perform. Pick a few spots to convolve coefficient $b[k]$ with the input sequence $x[n]$ that are different (e.g. flat regions vs. quick changing regions of $x[n]$).
3. Compute the output sequence $y[n]$ and make a stem plot thereof. Do at least a few of the computations BY HAND first, again to gain some intuition. Then you can deploy matlab as necessary. Functions that will likely be helpful include: stem, conv, subplot.
4. Compare and contrast the input and output sequences. It may be very helpful to plot them side by side. What features were preserved? Which were filtered out? Therefore, describe in words and math terms what each type of filter is encoded in the coefficients b.

The input sequence is given in order: $x[0], x[1], x[2], \ldots$. In other words, the most recent samples acquired are at the end of this sequence:

$$x[n] = \{0, 0, 0, 1, 2, 3, 4, 5, 4, 3, 2, 1, 0, 2, 2, 2, 2, -2, -20, 0, 20, 5, 0, 1, -1, 0, 0, 0\}$$

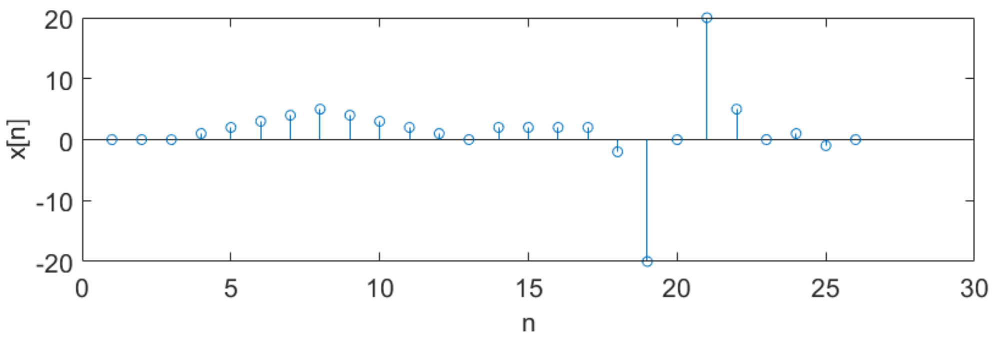

Figure 1: Stem plot of input sequence $x[n]$

There are 5 examples to work with here for coefficients b:

b[0]	b[1]	b[2]	b[3]	b[4]
1/5	1/5	1/5	1/5	1/5

Table 1: ex1

b[0]	b[1]
1	-1

Table 2: ex2

b[0]	b[1]	b[2]	b[3]	b[4]	b[5]
0.6444	0.4103	0	-0.1368	0	0.0821

Table 3: ex3

b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]
-0.5	-0.5	-0.5	3	-0.5	-0.5	-0.5

Table 4: ex4

b[0]	b[1]	b[2]	b[3]	b[4]	b[5]
1/5	1/3	1	-1	-1/3	-1/5

Table 5: ex5