**Tune-Up Tuesday #4 Deconvolution for October 10, 2024**

Before we talk about deconvolution, let’s define convolution. Then, we’ll derive a deconvolution algorithm and apply it to two examples. You could have worked either example for the Tune-Up.

**Convolution**. For a finite impulse response filter with filter coefficients , the output signal for an input signal is computed according to

If we input the discrete-time impulse signal , which has value of 1 at and 0 otherwise, the output is called the *impulse response* (response is synonymous with output):

Hence,

Otherwise, . Because ,

This computation is known as convolution of and :

Given input signal and impulse response , we can compute output signal . If is also finite in length, then the length of will be the length of plus the length of minus 1.

**Deconvolution.** Whereas convolution computes the output signal from input signal and an impulse response of a FIR filter, deconvolution seeks to find impulse response given input signal and output signal . We can choose the input signal , also known as a test signal, and observe the output signal.

**Practical scenario**. We would start the test signal and the observation at a particular point in time, which we’ll say is at without loss of generality. Further, we will assume that ; i.e., is a causal signal.

**Deconvolution Algorithm.** We’ll work backwards in the time-domain to compute the FIR filter coefficients. We derive a time-domain deconvolution algorithm by first evaluating the output at :

As mentioned above, we’ll assume is a causal signal; i.e., *.* Since we know *x*[*n*] and *y*[*n*], we have one equation and one unknown at *n* = 0:

and we can compute

For this calculation to be valid, the first value of the test signal, *x*[0], cannot be zero.

Second output: , and therefore, .

Third output: and

In general, for the *N*th output, .

The MATLAB script [utdeconvolve.m](http://users.ece.utexas.edu/~bevans/courses/signals/tuneups/fall2024/utdeconvolve.m) implements this algorithm.

**Example #1**. *Problem 4.3(b).* In this problem, we’re given

* causal input signal *x*[*n*] with non-zero values [ 1 2 3 4 5 ]
* causal output signal *y*[*n*] with non-zero values [ 1 1 1 1 1 -5 ]

We can compute the FIR filter coefficients using the above deconvolution algorithm:

The values of for are zero. The MATLAB script [utdeconvolve.m](http://users.ece.utexas.edu/~bevans/courses/signals/tuneups/fall2024/utdeconvolve.m) will give the same answer for . We can validate the answer by convolving and . We can use the Matlab command conv to do this:

y = conv( [1 -1], [ 1 2 3 4 5 ] )

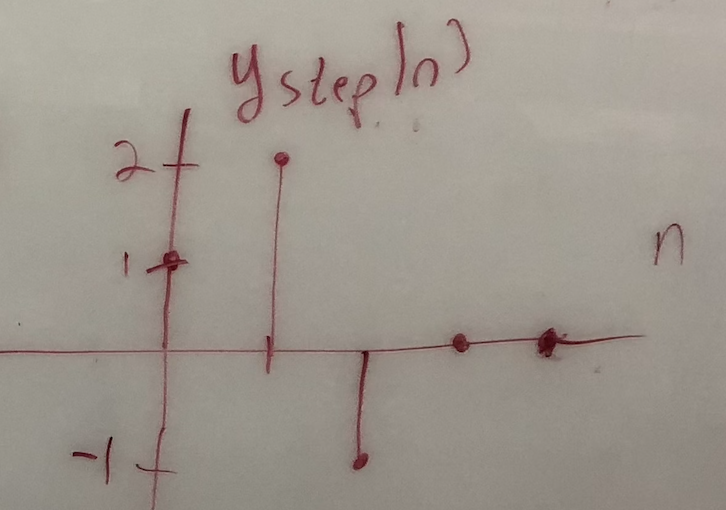
y =

1 1 1 1 1 -5

When convolving two finite-length signals and , the result has finite length. The length of is the length of plus the number of filter coefficients minus 1. Since the length of is 6 and the length of is 5, there are 2 filter coefficients.

**A white board with writing on it

Description automatically generatedExample #2**. Step response. A step response is the response of a system to a step function. A step function *u*[*n*] is a function that turns “on” at the origin and stays on, as plotted on the right. Mathematically,



When the input is , the step response for an FIR filter gives the output plotted on the right:

We can compute the FIR filter coefficients using the above deconvolution algorithm:

The values of for are zero. We can validate the answer by convolving and . We can use the Matlab command filter to do this:

y = filter( [1 1 -3 1], 1, [ 1 1 1 1 1 1 1 1 ] )

y =

1 2 -1 0 0 0 0 0

We want to use the filter function because the input signal is infinite in length and we’ll have to truncate it to be of finite length in Matlab. Convolving two finite length signals, e.g. using the conv function, will have trailing effects that would not be present for when the input signal is infinite in length. The filter function also performs convolution but only computes as many output samples as there are input samples.

We can use [utdeconvolve.m](http://users.ece.utexas.edu/~bevans/courses/signals/tuneups/fall2024/utdeconvolve.m) to compute the filter coefficients in vector b.

|  |  |  |
| --- | --- | --- |
| **x** | **y** | **b** |
| [ 1 1 1 ] | [ 1 2 -1 ] | [ 1 1 -3 ] |
| [ 1 1 1 1 ] | [ 1 2 -1 0 ] | [ 1 1 -3 1] |
| [ 1 1 1 1 1 ] | [ 1 2 -1 0 0 ] | [ 1 1 -3 1 0 ] |
| [ 1 1 1 1 1 1 ] | [ 1 2 -1 0 0 0 ] | [ 1 1 -3 1 0 0 ] |