2.1 Tutorial: `conv`

The MATLAB function `conv` computes the convolution sum

$$y[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m], \tag{2.3}$$

assuming that $x[n]$ and $h[n]$ are finite-length sequences. If $x[n]$ is nonzero only on the interval $n_x \le n \le n_x + N_x - 1$ and $h[n]$ is nonzero only on the interval $n_h \le n \le n_h + N_h - 1$, then $y[n]$ can be nonzero only on the interval

$$(n_x + n_h) \le n \le (n_x + n_h) + N_x + N_h - 2\,, \tag{2.4}$$

meaning that `conv` need only compute $y[n]$ for the $N_x + N_h - 1$ samples on this interval. If `x` is an N_x-dimensional vector containing $x[n]$ on the interval $n_x \le n \le n_x + N_x - 1$ and `h` is an N_h-dimensional vector containing $h[n]$ on the interval $n_h \le n \le n_h + N_h - 1$, then `y=conv(h,x)` returns in `y` the $N_x + N_h - 1$ samples of $y[n]$ on the interval in Eq. (2.4). However, `conv` does not return the indices of the samples of $y[n]$ stored in `y`, which makes sense because the intervals of `x` and `h` are not input to `conv`. Instead, you are responsible for keeping track of these indices, and will be shown how to do this in this tutorial.

✓ (a). Consider the finite-length signal

$$x[n] = \begin{cases} 1\,, & 0 \le n \le 5\,, \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{2.5}$$

Analytically determine $y[n] = x[n] * x[n]$.

✓ (b). Compute the nonzero samples of $y[n] = x[n] * x[n]$ using `conv`, and store these samples in the vector `y`. Your first step should be to define the vector `x` to contain the samples of $x[n]$ on the interval $0 \le n \le 5$. Also construct an index vector `ny`, where `ny(i)` contains the index of the sample of $y[n]$ stored in the i-th element of `y`, i.e., `y(i)` $= y[$`ny(i)`$]$. For example, `ny(1)` should contain $n_x + n_x$, where n_x is the first nonzero index of $x[n]$. Plot your results using `stem(ny,y)`, and make sure that your plot agrees with the signal determined in Part (a). As a check, your plot should also agree with Figure 2.1.

✓ (c). Consider the finite-length signal

$$h[n] = \begin{cases} n\,, & 0 \le n \le 5\,, \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{2.6}$$

Analytically compute $y[n] = x[n] * h[n]$. Next, compute `y` using `conv`, where your first step should be to define the vector `h` to contain $h[n]$ on the interval $0 \le n \le 5$. Again construct a vector `ny` which contains the interval of n for which `y` contains $y[n]$. Plot your results using `stem(ny,y)`. As a check, your plot should agree with Figure 2.2 and your analytical derivation.

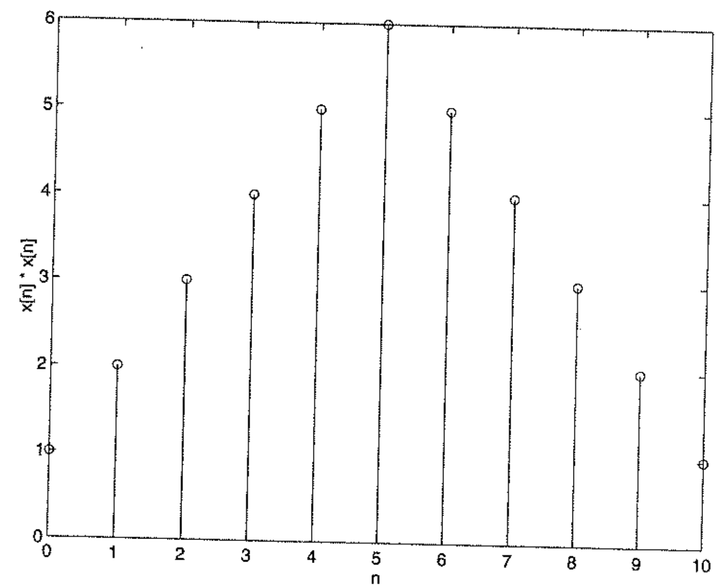

Figure 2.1. Plot of the signal $y[n] = x[n] * x[n]$.

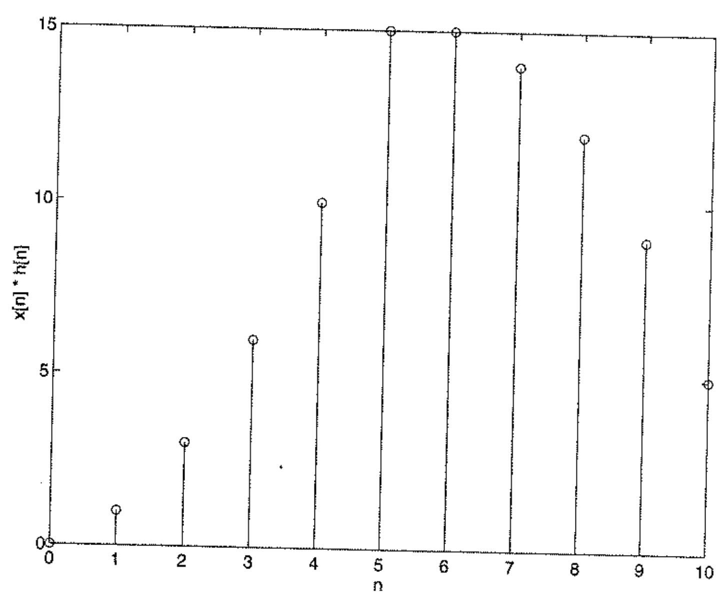

Figure 2.2. Plot of the signal $y[n] = x[n] * h[n]$.

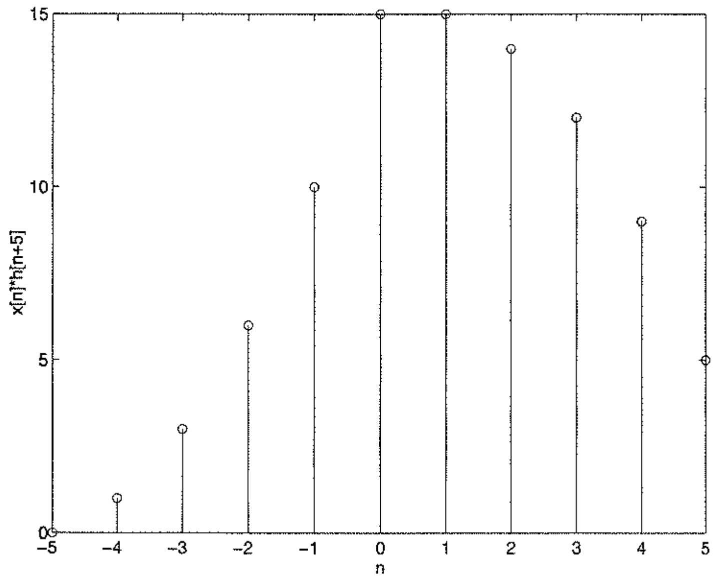

Figure 2.3. Plot of the signal $y_2[n] = x[n] * h[n+5]$.

For the example just considered—implementing $y[n] = x[n] * h[n]$ using `conv`—the signal $h[n]$ can be viewed as the impulse response of a linear time-invariant system for which $x[n]$ is the system input and $y[n]$ is the system output. Because $h[n]$ is zero for $n < 0$, this system is causal. However, `conv` can also be used to implement LTI systems which are noncausal. You must still be careful to keep track of the indices of $x[n]$, $h[n]$, and $y[n]$. For example, consider the system with impulse response $h[n+5]$, where $h[n]$ is defined in Eq. (2.6).

(d). How does $y_2[n] = x[n] * h[n+5]$ compare to the signal $y[n]$ derived in Part (c)?

(e). Use `conv` to compute the nonzero samples of $y_2[n]$, and store these samples in the vector y2. If done correctly, this vector should be identical to the vector y computed in Part (c). The only difference is that the indices of the values stored in y2 have changed. Determine this set of indices, and store them in the vector `ny2`. Plot $y_2[n]$ using `stem(ny2,y2)`. Your plot should agree with Figure 2.3.

2.2 Tutorial: `filter`

The `filter` command computes the output of a causal, LTI system for a given input when the system is specified by a linear constant-coefficient difference equation. Specifically, consider an LTI system satisfying the difference equation

$$\sum_{k=0}^{K} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m], \tag{2.7}$$

where $x[n]$ is the system input and $y[n]$ is the system output. If `x` is a MATLAB vector containing the input $x[n]$ on the interval $n_x \le n \le n_x + N_x - 1$ and the vectors `a` and `b` contain the coefficients a_k and b_m, then `y=filter(b,a,x)` returns the output of the causal LTI system satisfying

$$\sum_{k=0}^{K} \mathtt{a}(k+1)\mathtt{y}(n-k) = \sum_{m=0}^{M} \mathtt{b}(m+1)\mathtt{x}(n-m)\,. \tag{2.8}$$

Note that $\mathtt{a}(k+1) = a_k$ and $\mathtt{b}(m+1) = b_m$, since MATLAB requires that all vector indices begin at one. For example, to specify the system described by the difference equation $y[n] + 2y[n-1] = x[n] - 3x[n-1]$, you would define these vectors as `a=[1 2]` and `b=[1 -3]`.

The output vector `y` returned by `filter` contains samples of $y[n]$ on the same interval as the samples in `x`, i.e., $n_x \le n \le n_x + N_x - 1$, so that both `x` and `y` contain N_x samples. Note, however, that `filter` needs $x[n]$ for $n_x - M \le n \le n_x - 1$ and $y[n]$ for $n_x - K \le n \le n_x - 1$ in order to compute the first output value $y[n_x]$. The function `filter` assumes that these samples are equal to zero.

(a). Define coefficient vectors `a1` and `b1` to describe the causal LTI system specified by $y[n] = 0.5x[n] + x[n-1] + 2x[n-2]$.

(b). Define coefficient vectors `a2` and `b2` to describe the causal LTI system specified by $y[n] = 0.8y[n-1] + 2x[n]$.

(c). Define coefficient vectors `a3` and `b3` to describe the causal LTI system specified by $y[n] - 0.8y[n-1] = 2x[n-1]$.

(d). For each of these three systems, use `filter` to compute the response $y[n]$ on the interval $1 \le n \le 4$ to the input signal $x[n] = n\,u[n]$. You should begin by defining the vector `x=[1 2 3 4]`, which contains $x[n]$ on the interval $1 \le n \le 4$. The result of using `filter` for each system is shown below:

```
>> x = [1 2 3 4];
>> y1 = filter(b1,a1,x)
y1 =
    0.5000    2.0000    5.5000    9.0000
>> y2 = filter(b2,a2,x)
y2 =
    2.0000    5.6000   10.4800   16.3840
>> y3 = filter(b3,a3,x)
y3 =
         0    2.0000    5.6000   10.4800
```

From `y1(1)=0.5`, you can see that `filter` has set $x[0]$ and $x[-1]$ equal to zero, since both of these samples are needed to determine $y_1[1]$.

Basic Problems

(a). Many of the problems in this exercise will use the following three signals:

$$x_1[n] = \begin{cases} 1\,, & 0 \le n \le 4\,, \\ 0\,, & \text{otherwise}\,, \end{cases}$$

$$h_1[n] = \begin{cases} 1\,, & n = 0\,, \\ -1\,, & n = 1\,, \\ 3\,, & n = 2\,, \\ 1\,, & n = 4\,, \\ 0\,, & \text{otherwise}\,, \end{cases}$$

$$h_2[n] = \begin{cases} 2\,, & n = 1\,, \\ 5\,, & n = 2\,, \\ 4\,, & n = 3\,, \\ -1\,, & n = 4\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

Define the MATLAB vector `x1` to represent $x_1[n]$ on the interval $0 \le n \le 9$, and the vectors `h1` and `h2` to represent $h_1[n]$ and $h_2[n]$ for $0 \le n \le 4$. Also define `nx1` and `nh1` to be appropriate index vectors for these signals. Make appropriately labeled plots of all the signals using `stem`.

(b). The commutative property states that the result of a convolution is the same regardless of the order of the operands. This implies that the output of an LTI system with impulse response $h[n]$ and input $x[n]$ will be the same as the output of an LTI system with impulse response $x[n]$ and input $h[n]$. Use `conv` with `h1` and `x1` to verify this property. Is the output of `conv` the same regardless of the order of the input arguments?

(c). Convolution is also distributive. This means that

$$x[n] * (h_1[n] + h_2[n]) = x[n] * h_1[n] + x[n] * h_2[n]\,.$$

This implies that the output of two LTI systems connected in parallel is the same as one system whose impulse response is the sum of the impulse responses of the parallel systems. Figure 2.8 illustrates this property.

Verify the distributive property using `x1`, `h1` and `h2`. Compute the sum of the outputs of LTI systems with impulse responses $h_1[n]$ and $h_2[n]$ when $x_1[n]$ is the input. Compare this with the output of the LTI system whose impulse response is $h_1[n] + h_2[n]$ when the input is $x_1[n]$. Do these two methods of computing the output give the same result?

(d). Convolution also possesses the associative property, i.e.,

$$(x[n] * h_1[n]) * h_2[n] = x[n] * (h_1[n] * h_2[n])\,.$$

the interval $N_3 \le n \le N_4$. The output of the system $y[n] = x[n] * h[n]$ will also be finite-length, with its nonzero samples in the range $N_5 \le n \le N_6$. Find expressions for N_5 and N_6 in terms of N_1 through N_4.

(c). Let $x[n]$ be the finite-length signal

$$x[n] = \begin{cases} 1, & n = 0, \\ 5, & n = 1, \\ 2, & n = 2, \\ 4, & n = 3, \\ -2, & n = 4, \\ 2, & n = 5, \end{cases}$$

and let $h[n]$ be the impulse response of a noncausal system given by

$$h[n] = \begin{cases} 1 - (|n|/3), & |n| \le 3, \\ 0, & \text{otherwise}. \end{cases}$$

Define MATLAB vectors `x` and `h` to represent these signals, along with accompanying index vectors `nx` and `nh`. Make appropriately labeled plots of both signals with `stem`.

(d). Compute the output of the LTI system $y[n] = x[n] * h[n]$ using `conv` and the vectors you defined in the previous part. Define the vector `y` to represent this output vector, and define an index vector `ny`. You may find the expressions you derived in Part (b) useful in determining the indices in `ny`. Make an appropriately labeled plot of $y[n]$ using `stem`.

Intermediate Problems

One possible use for noncausal LTI systems with finite-length impulse responses is to interpolate between the samples of a discrete-time signal. In Chapter 7, you will consider this interpolation problem in more detail. In that context, you will see that there are a number of criteria for which the LTI system with the impulse response $h[n]$ defined in Part (c) may not be the ideal interpolation system. However, in some situations $h[n]$ provides an acceptable form of interpolation.

(e). Consider a new input signal

$$x_u[n] = \begin{cases} x[n/3], & n = 3k \text{ and } k \text{ integer}, \\ 0, & \text{otherwise}, \end{cases}$$

where $x[n]$ is the signal defined above in Part (c).

The process of inserting zeros between the samples of a signal like this is commonly referred to as expansion. Chapter 7 examines this operation in more detail, but for now you do not need to concern yourself with understanding in-depth what expansion does to the signal. Define MATLAB vectors `xu` and `nxu` to represent the expanded signal and its time indices.