Discrete Time Signals & Matlab

A discrete-time signal x is a bi-infinite sequence, $\{x_k\}_{k=-\infty}^{\infty}$. The variable k is an integer and is called the *discrete time*. An equivalent way to think about x is that it is a function that assigns to k some real (or complex) number x_k.

The graph of x_k vs. k is called a time series. MATLAB provides several ways of plotting time series, or discrete data. The simplest is the stem plot. We let the discrete signal be

$$x = (\ \cdots \quad 0 \quad -1 \quad 2 \quad 3 \quad -2 \quad 0 \quad 1 \quad 0 \quad \cdots\ ), \tag{1}$$

where the first non-zero entry corresponds to $k = 0$ and the last to $k = 5$. For values of k larger than 5 or less than 0, $x_k = 0$. To plot x_k for $k = -2$ to $k = 7$, which will include some zeros, we use these commands. (See Figure 1.)

```
x=[0 0 -1 2 3 -2 0 1 0 0];
dtx= -2:7; (discrete time for x)
stem(dtx,x)
```

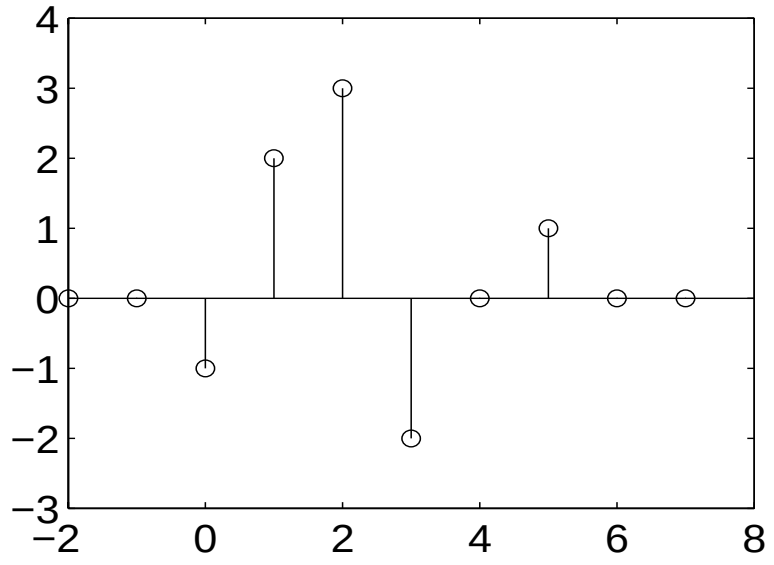

Figure 1: A stem plot of x_k vs. k

The convolution of two discrete-time signals x and y is $x * y$, which is defined by

$$(x * y)_n := \sum_{k=-\infty}^{\infty} x_{n-k} y_k. \tag{2}$$

As is the case with the continuous-time convolution, $x * y = y * x$. The convolution is of interest in discrete-time signal processing because of its connection with linear, time-invariant filters. If H is such a filter, than there is a sequence $\{h_k\}_{k=-\infty}^{\infty}$ such that $H[x] = h * x$; h is called the *impulse response* (IR) of the filter H. When the IR h has only a finite number of non-zero h_k's, we say that H has *finite impulse response* (FIR). Otherwise, it has *infinite impulse response* (IIR).

In practical situations, we will work only with finite sequences, but these sequences may be arbitrarily large. To compute convolutions of such sequences, we need to discuss indexing of sequences. For a finite sequence x we will let s_x be the starting value of k considered. We thus assume that $x_k = 0$ if $k < s_x$. In addition, we let ℓ_x be the last value of k we consider; again, we assume that $x_k = 0$ if $k > \ell_x$. Also, we will let n_x be the length of the stretch between the starting and last x_k's, so $n_x = \ell_x - s_x + 1$. For example, if we only wish to work with nonzero x_k's in the sequence x defined previously in (1), then $s_x = 0$, $\ell_x = 5$, and $n_x = 6$.

Our goal is to use MATLAB's function `conv` to compute $x * y$ when x and y are finite. To do this, we first need to look at what the indexing of the convolution $x * y$ is. That is, we want to know what s_{x*y}, ℓ_{x*y}, and n_{x*y} are, given the corresponding quantities for finite x and y. When y is finite, equation (2) has the form

$$(x * y)_n := \sum_{k=s_y}^{\ell_y} x_{n-k} y_k = \sum_{k=0}^{\ell_y - s_y} x_{n-k-s_y} y_{k+s_y}, \tag{3}$$

where the second equality follows from a change of index. Since $0 \le k \le \ell_y - s_y$, the index $n - k - s_y$ satisfies

$$n - s_y \ge n - k - s_y \ge n - \ell_y$$

If $n - \ell_y > \ell_x$, all the $x_{n-k-s_y} = 0$, and $(x * y)_n = 0$. Similarly, if $n - s_y < s_x$, we have $(x * y)_n = 0$. In addition, by direct substitution we see that $(x * y)_{s_y+s_x} = x_{s_x} y_{s_y}$ and $(x * y)_{\ell_y+\ell_x} = x_{\ell_x} y_{\ell_y}$. It follows that the starting index for $x * y$ is $s_{x*y} = s_x + s_y$, that the last index is $\ell_{x*y} = \ell_x + \ell_y$, and that

the number of terms we need to compute is $n_{x*y} = \ell_x + \ell_y - s_x - s_y + 1 = n_x - 1 + n_y - 1 + 1 = n_x + n_y - 1$. We list these below.

$$\left.\begin{array}{rcl} s_{x*y} & = & s_x + s_y \\ \ell_{x*y} & = & \ell_x + \ell_y \\ n_{x*y} & = & n_x + n_y - 1 \end{array}\right\} \tag{4}$$

MATLAB stores a finite sequence in a row or column vector, depending on the user's choice. Such vectors are always indexed starting with 1 and going up to the number of entries in the vector. Let's look at our earlier example, where we let `x=[0 0 -1 2 3 -2 0 1 0 0]`. To access the fourth entry, we type in `x(4)`. When we hit the `ENTER` key, we would get `ans = 2`. Now, as a sequence, x is a function the discrete time, k. In the entries given above, k runs from $k = -2$ through $k = 7$. For plotting purposes and other reasons, we define a discrete time vector to carry this information, `dtx=-2:7`. For example, to find the discrete time that corresponds to the fourth entry in `x`, we type in `dtx(4)` and hit `ENTER` to get `ans=1`.

Let's use this to finding $x * y$. Again, take `x=[0 0 -1 2 3 -2 0 1 0 0]` and `dtx=-2:7`. Also, let `y=[1 -1 2 4]` and `dty=8:11`. To find the convolution of these two and plot the result, we use these commands:

```
x=[0 0 -1 2 3 -2 0 1 0 0];
dtx= -2:7;
y=[1 -1 2 4];
dty=8:11;
z=conv(x,y);
dtz=6:18;
stem(dtz,z)
```

The result of this is that `z=[0 0 -1 3 -1 -5 16 9 -9 2 4 0 0]`. Apart from these entries, which are for discrete times $k = 6$ through $k = 18$, the other entries in $z = x * y$ are all 0. So, for example, $z_{-20} = 0$ and $z_5 = 0$. To include more entries on the plot, say for $k = -5$ to $k = 27$, one needs to pad the row vector `z` with zeros. There are many ways to do this. Here is one of them.

```
dtz=-2:27;
stem(dtz,[zeros(size(-2:5)) z zeros(size(19:27))])
```

Exercises. These exercises require MATLAB. A basic introduction may be found online at `http://www.mathworks.com/products/education`. Click on the MATLAB and SIMULINK Tutorials to get started.

1. Do each of the sets of commands listed in the discussion. Print the resulting plots.

2. Find the convolution of the x and y. Here, $x_k = 0$ for $k > 3$ and $k < -4$, and $x_k = 1$ for $-4 \leq k \leq 3$. For y_k, assume that $y_k = 0$ for $k > -2$ and for $k < -8$. When $-8 \leq k \leq -2$, $y_k = k + 2$. Find $x * y$ and determine the discrete time index. Plot the result with `stem`, again using the correct time index. Put a title on your plot by using the command `title('Convolution of x*y')`. Print the result.

3. Take `t=linspace(0,2*pi,20), x=sin(t)`. Do the plots `stem(t,x)`, `stem(t,x,':r','fill')`, and `stem(t,x,'-.sb','fill')`. Put them all in one plot with the commands below and print the result.

```
subplot(1,3,1), stem(t,x)
title('Default Stem Plot')
subplot(1,3,2), stem(t,x,':r','fill')
title('Filled Stem Plot')
subplot(1,3,3), stem(t,x,'sk')
title('Square Marker Stem Plot')
```

4. This exercise illustrates the use of another plotting tool, `stairs`. Start with the following commands.

```
t=linspace(-pi,pi,20); x=sin(t);
stairs(t,x)
```

   Next, change the plot by using a "dash-dot" line instead of a solid one. We will also change the color to red: `stairs(t,x,'-.r')`. We will now combine `stairs`, `stem` and `plot`. Title the plot and print the result.

```
t=linspace(-pi,pi,20); x=sin(t);
tt=linspace(-pi,pi,600); xx=sin(tt);
stairs(t,x,'-.r'), hold on
stem(t,x,':sb','fill') (Dotted stems & filled circles)
plot(tt,xx,'k'), hold off
```

5. The `stairs` plots are well suited to doing plots involving the Haar scaling function and wavelet. Recall that the Haar scaling function is defined by

$$\varphi(x) := \begin{cases} 1, & \text{if } 0 \le t < 1, \\ 0, & \text{otherwise.} \end{cases}$$

Use `stairs` to plot $f(x) = 3\varphi(x+1) - 2\varphi(x) + \varphi(x-1) + \varphi(x-2)$ on the interval $-3 \le x \le 4$. On the same interval, plot $g(x) = \varphi(2x+3) - \varphi(2x+2) + 2\varphi(2x) - \varphi(2x-3)$. For g, use a dash-dot pattern (-.) and make the color red. (Hint: to plot two functions, you will need to use `hold on`.)

6. This problem pertains to the Z-transform. Let h be a finite discrete-time signal. If we let $z = e^{i\omega}$, then the Z-transform is

$$\hat{h}(\omega) = \sum_{k=s_h}^{\ell_h} h_k z^{-k}.$$

We want to illustrate how to numerically compute and plot $\hat{h}$. For simplicity, we will work with an h for which $s_h = -n$, $\ell = n$, and $h_k = 1/n$ for $-n \le k \le n$. We will do the following.

```
w=linspace(-pi,pi,600); z=exp(i*w);  (Initialize z.)
n=2; h=ones(1,2*n+1)/(2*n+1);
H=z.^n.*polyval(h,1./z);  (Compute the Z-transform.)
norm(imag(H))  (This should be small; if not, H is wrong.)
plot(w,real(H))
```

In addition to $n = 2$, do this for $n = 4$, and $n = 5$. Title, and then print, each plot.