BIOL 708L/NACS 728B Lab #1
Fall 2011

Signals & Systems in MATLAB

The goals of this lab are 1) to see how signals are represented and displayed in MATLAB, and 2) to try running three different signals through systems.

1) *Creating and displaying signals, e.g.* $x_1[n]$ *defined below,* $x_2[n]$ *defined below,* and $\delta[n]$, *known as the "unit impulse", i.e. the signal that is zero everywhere except at the origin (*$n = 0$*), where it is one.*
   a) Create a new signal, called `x1`, that has 6 zeros followed by 3 ones followed by 6 zeros. This is a signal that starts with no activity (i.e. zero), turns on (i.e. one), stays on, and then turns off again. Verify that your `x1` has the correct number of zeros and ones. What is the length of `x1` (using the Matlab function `length`)? What is the size of `x1` (using the Matlab function `size`)?
   b) We will use the convention that one-dimensional signals should be columns not rows (e.g. like typical Excel data). If your `x1` is a row, turn into a column (how?). Now what are its length & size?
   c) Next we graphically display `x1`. First open a new figure window, using the `figure` command. Then display `x1` using the function `stem`, e.g. `stem(x1)`. `stem` is a good way to plot a discrete signal.
   d) Change the limits on the y-axis, so that the signal can better be appreciated in perspective, to run from -2 to 2 instead of the default (Matlab's guess) of 0 to 1. Use the `ylim` function: `ylim([-2 2])`
   e) Now display `x1` a different way. Open another figure window using `figure`. Then display `x1` using the function `bar`, e.g. `bar(x1)`. `bar` is also a good way of plotting a discrete signal. Again change the y-axis limits to run from -2 to 2.
   f) In general, you should feel free to use either `bar` or `stem` to display discrete signals. Many naïve Matlab users use `plot` instead, but *you* should not use `plot` for discrete signals: it gives the illusion that the data is continuous. Only use `plot` if you specifically want that illusion. For example, try using `plot` to display `x1` in a new figure window. From that figure try to count how many zeros and ones there are—you should see that it's much harder than with `stem` or `bar` displays.
   g) Create another signal called `x2` which is a zero followed by a one followed by a zero followed by a one, *etc.*, that has the same length and size as `x1` above (and in particular, make it a column vector). Verify that its length and size are the same as `x1`. Open a new figure window and display it using `stem` or `bar` (whichever you prefer), again changing the y-axis limits to run from -2 to 2.
   h) Create a signal called `impulse` which has 7 zeros followed by 1 one followed by 7 zeros. This is a finite length version of the impulse $\delta[n]$. Make it a column vector. What is its length and size? Open a new figure window and display it using `stem` or `bar` (whichever you prefer), again changing the y-axis limits to run from -2 to 2.

2) *Taking the two-point moving average of signals*

a) The two-point moving average system is given by $y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1]$. We can construct $y[n]$ by taking half of the original signal, and adding to it half of a delayed (shifted) version of the original. It is not hard to shift a signal in Matlab, but first we have to decide what it means to shift a signal which has only a limited number samples. We should certainly move the leftmost value one sample to the right. And the next-to-leftmost value should also move one sample to its right, etc.. But there are still two details: 1) what goes in the leftmost slot of the shifted signal (since there isn't an original value to its left to be shifted over) and 2) what do we do with the rightmost value of the original signal (which doesn't have a slot to its right to move into)?
We will use the simplest possible solution to these issues: in the leftmost slot we will just put zero (what else would we put?), and the last value of the original signal we will just ignore (what else would we do with it?). In summeary, we implement the shifted signal as a new signal whose first value is zero, and all the rest of values have each of the values of the original signal except the last one. In Matlab we can write:

```
x1shift1 = [0;x1(1:end-1)];
```

to get a version of $x_1[n]$ delayed by 1 sample. Explain how this Matlab code implements the above strategy.

b) To compute $y_1[n]$, the two-point moving average of $x_1[n]$, add half of $x_1[n]$ to half of its delayed version.

```
y1 = 1/2*x1 + 1/2*x1shift1;
```

Open up a new figure and display `y1` using `stem` or `bar`, again changing the y-axis limits to run from -2 to 2. Compare it to your earlier display of `x1`. Does your answer look like a smoothed (less sudden) version of `x1`?

c) Similarly, calculate `y2`, the output of the two-point moving average system when the input is `x2`. Open up a new figure and display `y2` using `stem` or `bar`, again changing the y-axis limits to run from -2 to 2. Compare it to your earlier display of `x2`. Does your answer look like the two-point moving average of `x2`? In this case the filter has a very different effect, visually. Is this a smoothed (less sudden) version of `x2`? If someone asked you to smooth `x2`, would you do it differently?

d) Calculate `yimpulse`, the output of the two-point moving average system when the input is `impulse`. Open up a new figure and display `yimpulse` using `stem` or `bar`, again changing the y-axis limits to run from -2 to 2. Compare it to your earlier display of `impulse`. Is this a smoothed version of `impulse`?

3) *Taking the (two point) difference of signals*

Repeat question 2 for the change detector system defined by the Diff operator, i.e.
$y[n] = Diff\{x[n]\} = x[n] - x[n-1]$.

4) *Displaying signals with the correct index*

a) Implied above (but not stated) is that all the input signals above are centered on $n = 0$. The graphical plots of our signals do not reflect this. Create a sequence of the correct indices: `n=[-7:7].'` and verify that this has the same `length` and `size` as all the signals above, e.g. `x1`.

b) To correctly display a signal with the correct indices, use `stem` or `bar` with two inputs instead of one, and use `n` as the first input. For example, try `figure; bar(n, x1)`. Is `x1` centered correctly *with respect to n = 0*? Try it also with `y1`, `impulse`, and `yimpulse` from question 3. This method of displaying the impulse, and the system's response to the impulse, will allow us to visualize the "impulse response" of a system.

From now on, always define `n` to accompany your signal, and always use `n` with `bar` or `stem`!