Introduction to Signals and Systems, Part IV: Lecture Summary

Last time we introduced *difference equations* as the mathematical representation of discrete-time systems. Moreover, we said that in this class, we will restrict ourselves mostly to the study of a class of difference equations known as *Linear, Time-Invariant (LTI)* systems. These systems, or difference equations, assume the following general form:

$$y[n] = \sum_{l=1}^{N} a_l y[n-l] + \sum_{k=0}^{M} b_k x[n-k] \tag{1}$$

In today's lecture, we look at some examples of LTI systems, specifically focusing on filtering, and introducing the important concept of *frequency response*. As we will see, designing an LTI filter with desirable properties requires the designer to select (1) the order of the filter (N, M), (2) the filter coefficients a_l and b_k, and (3) the sampling frequency.[1]

To start, let us restrict ourselves to non-recursive filters; that is filters for which the coefficients a_l are all equal to zero, so that:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] \tag{2}$$

In our first example, let us further restrict ourselves to the following simple filter equation:

$$y[n] = b_o x[n] + b_1 x[n-1] \tag{3}$$

Now, suppose that we wish to design a *low-pass filter*; that is, a filter that allows low frequencies in a signal through, while attenuating higher frequencies in a signal.[2] Given that we assume difference equation (3), we need to decide what the coefficients b_o and b_1 should be. After a little thought, consider the following candidate values:

$$b_0 = b_1 = 1/2 \tag{4}$$

To understand why the choice of filter coefficients in (4) might accomplish low-pass filtering of a signal, let us see how the filter in equation (3) affects the two signals $x_1[n]$ and $x_2[n]$ in Figure 1.

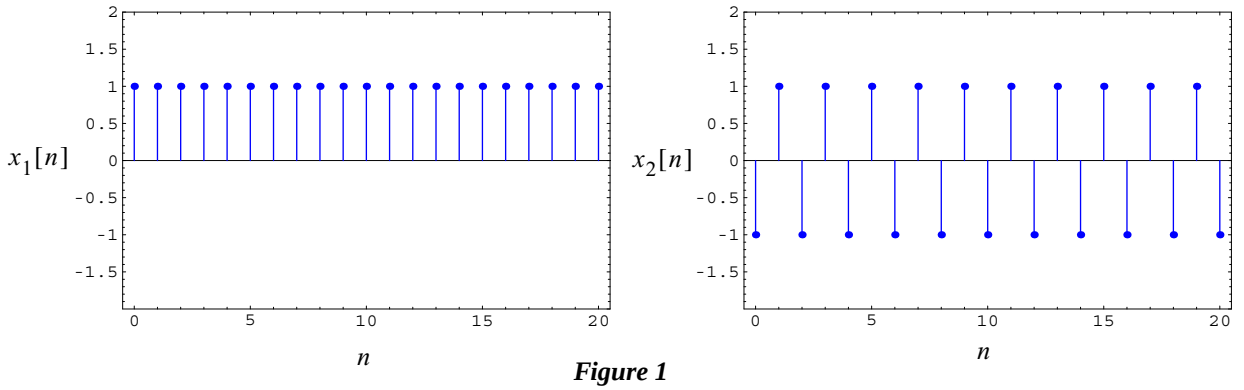


Figure 1

For signal $x_1[n]$, a constant (zero-frequency) signal over time index n, the resulting output $y_1[n]$ will be:

$$y_1[n] = 1/2 x_1[n] + 1/2 x_1[n-1] \tag{5}$$

1. *For today's lecture, we assume that the sampling frequency is already given; we will talk more about the implications of sampling later in this course.*
2. *An ideal low-pass filter allows frequencies in a signal below some cut-off frequency f_c through unchanged, while zeroing all frequencies in the signal above that same cut-off frequency.*

$$y_1[n] = 1/2 \cdot 1 + 1/2 \cdot 1 = 1,\ n \geq 1. \tag{6}$$

For signal $x_2[n]$, a fast-changing (i.e. high-frequency) signal over time index n, the resulting output $y_2[n]$ will be:

$$y_2[n] = 1/2 x_2[n] + 1/2 x_2[n-1] \tag{7}$$

$$y_2[n] = 1/2 \cdot (-1) + 1/2 \cdot 1 = 0,\ n \geq 1. \tag{8}$$

In Figure 2 we plot the input and output signals computed in (6) and (8). In other words, this choice of filter coefficients appears to allow low-frequency signals through unscathed, while completely eliminating the highest-frequency signal. That's all well and good, but what can we say about intermediate frequencies?

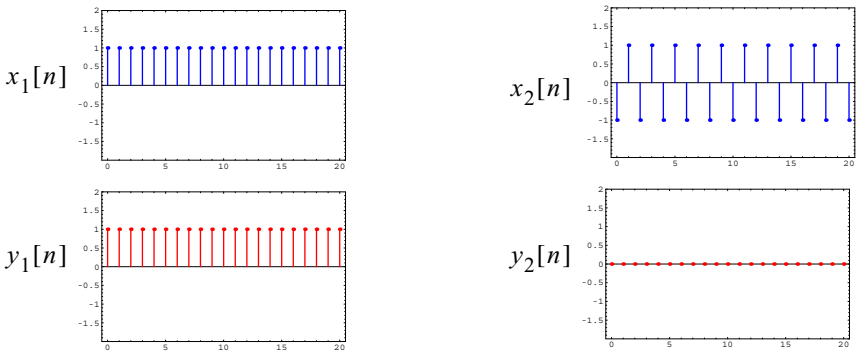


Figure 2

To answer this question, we can compute the *frequency response* of the system. The frequency response of an LTI system tells us to what extent different frequency components in a signal are affected as an input signal is modified by the LTI system (difference equation) to generate an output signal. While we will omit the details of how to compute a system's frequency response for now, in Figure 3 we plot the magnitude frequency response $|H(f)|$ [1] of the low pass filter in equations (3) [with filter coefficients from (4)].

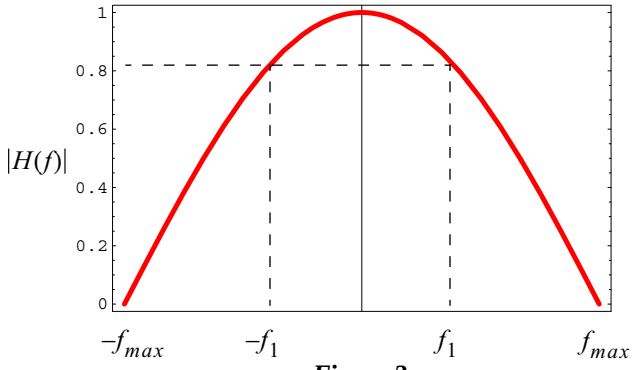


Figure 3

This plot confirms what we previously derived experimentally for the two sample signals in Figure 1. High-frequency components of a signal are completely eliminated, while low-frequency components are attenuated little, if at all. Intermediate frequencies are attenuated to varying degrees. For example, if a signal contains a frequency component at frequency f_1 with amplitude A, the output signal will contain that frequency component at an approximate magnitude of $0.8A$.

1. *$|H(f)|$ denotes the magnitude of the frequency response as a function of frequency. There is another aspect of the frequency response, namely, the phase frequency response $\angle H(f)$. We will cover this concept once we develop a mathematical treatment of LTI systems.*

We can generate higher-order low-pass filters by generalizing the difference equation in (3) to:

$$y[n] = \frac{1}{L}x[n] + \frac{1}{L}x[n-1] + \frac{1}{L}x[n-2] + \ldots + \frac{1}{L}x[n-L+1] \tag{9}$$

$$y[n] = \frac{1}{L}(x[n] + x[n-1] + x[n-2] + \ldots + x[n-L+1]) \tag{10}$$

The filter in equation (10) is known as a *running-average filter*. In Figure 4, we plot the filtered output signal $y[n]$, the magnitude frequency spectrums $|X(f)|$ and $|Y(f)|$, and the magnitude frequency response $|H(f)|$ for a sample input signal $x[n]$ and $L = 2$ (previous example), $L = 10$, $L = 30$ and $L = 100$. (See *Mathematica* notebook, section "Non-recursive difference equation example: simple low-pass filter" for these examples.) The input signal $x[n]$ is a discrete-time signal sampled at $f_s = 500$ Hz from the continuous-time signal,

$$x(t) = \sin(2\pi \cdot 4t) + noise \tag{11}$$

where, for each sample, the "noise" component of the signal consists of a uniformly distributed random number in the interval $[-1/2, 1/2]$. Note that as the size of the filter (i.e. L) is increased, higher frequencies become more and more suppressed, although not uniformly. In fact, a discrete number of frequencies are zeroed entirely; let us examine why this is so.

Consider the frequency response of the running-average filter for $L = 10$. Note that this filter zeroes frequency components at 50, 100, 150, 200 and 250Hz. In Figure 5 below, we plot a short, sampled 50Hz sine wave signal with the same sampling frequency as before. Note that for this signal, it doesn't matter which consecutive 10 samples we average, we will always get zero. That's why the frequency response for the 10-point running average filter is zero at 50Hz. The same argument applies to the other zeroed frequencies.

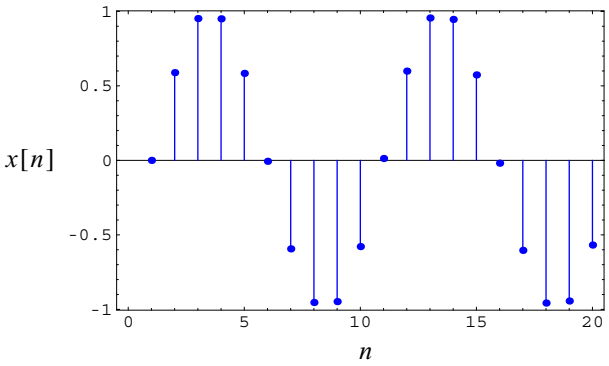


Figure 5

Now, let us switch gears and consider the task of *high-pass filtering*. In high-pass filtering, we want to do exactly the opposite of low-pass filtering; that is we want to attenuate low-frequency components of signals while passing through high-frequency components relatively unattenuated. We begin with the same basic difference equation as before:

$$y[n] = b_o x[n] + b_1 x[n-1] \tag{12}$$

and choose the following coefficient values:

$$b_0 = 1/2 \text{ and } b_1 = -1/2 \tag{13}$$

so that equation (12) becomes:

$$y[n] = 1/2x[n] - 1/2x[n-1]\,^{1} \tag{14}$$

1. Note that this is a discrete-time, first-order approximation of a derivative.

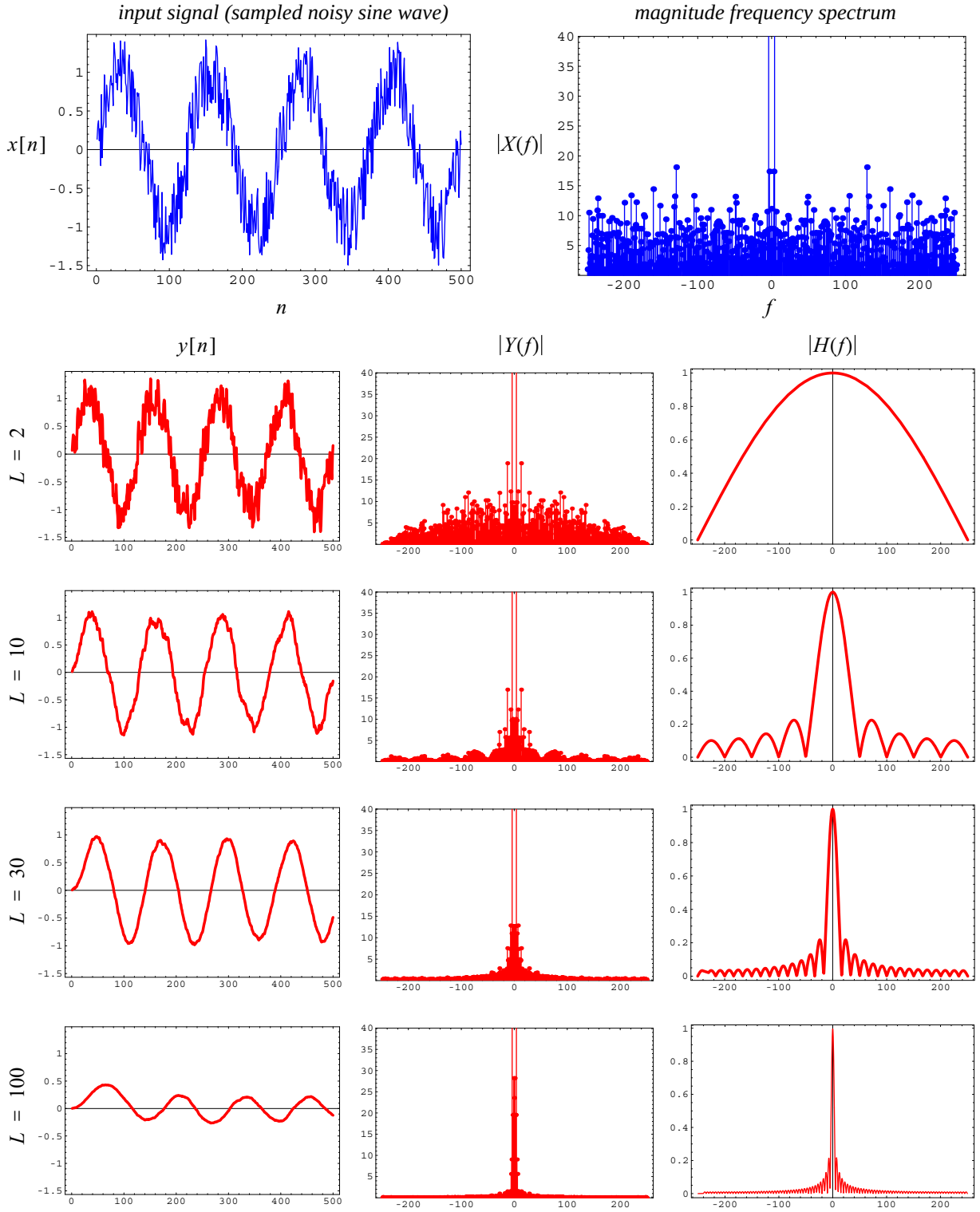


Figure 4: Low-pass filtering examples

Now, we examine how this filter affects the two signals in Figure 1. For signal $x_1[n]$, a constant (zero-frequency) signal over time index n, the resulting output $y_1[n]$ will be:

$$y_1[n] = 1/2 \cdot 1 - 1/2 \cdot 1 = 0,\ n \geq 1. \tag{15}$$

For signal $x_2[n]$, a fast-changing (i.e. high-frequency) signal over time index n, the resulting output $y_2[n]$ will be:

$$y_2[n] = 1/2 \cdot 1 - (1/2 \cdot -1) = 1,\ n \geq 1,\ n = \text{odd}. \tag{16}$$

$$y_2[n] = 1/2 \cdot (-1) - (1/2 \cdot 1) = -1,\ n \geq 2,\ n = \text{even}. \tag{17}$$

In Figure 6 we plot the input and output signals computed in (15) through (17). Note that now, the constant signal is zeroed, while the high-frequency signal passes through unscathed.

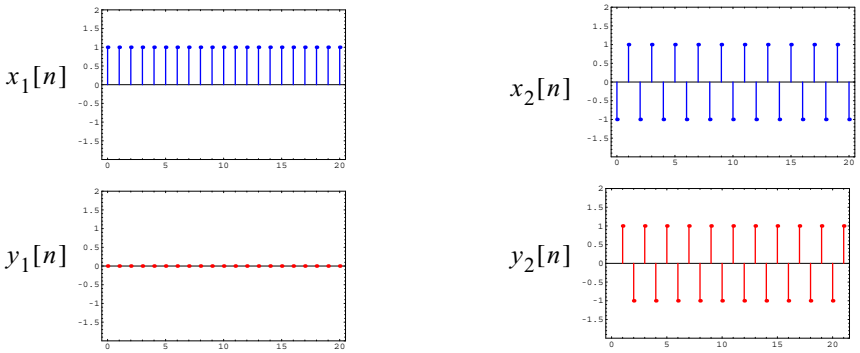


Figure 6

We can derive higher-order high-pass filters by deriving approximations of higher-order derivatives. For example, the second derivative can be approximated by:

$$x'[n] = 1/2x[n] - 1/2x[n-1] \tag{18}$$

$$\begin{aligned} y[n] &= 1/2x'[n] - 1/2x'[n-1] \\ &= 1/2(1/2x[n] - 1/2x[n-1]) - 1/2(1/2x[n-1] - 1/2x[n-2]) \\ &= (1/4)x[n] - (1/2)x[n-1] + (1/4)x[n-2] \end{aligned} \tag{19}$$

Similarly, the third and fourth derivative approximations are given, respectively, by:

$$y[n] = (1/8)x[n] - (3/8)x[n-1] + (3/8)x[n-2] - (1/8)x[n-3] \text{ and} \tag{20}$$

$$y[n] = (1/16)x[n] - (1/4)x[n-1] + (3/8)x[n-2] - (1/4)x[n-3] + (1/16)x[n-4] \tag{21}$$

In Figure 7, we plot the filtered output signal $y[n]$, the magnitude frequency spectrums $|X(f)|$ and $|Y(f)|$, and the magnitude frequency response $|H(f)|$ for a sample input signal $x[n]$ and the first, second, third and fourth-order high-pass filters in equations (14), (19), (20) and (21), respectively. (See *Mathematica* notebook, sections "Non-recursive difference equation example: simple high-pass filter #1, #2, #3 and #4" for these examples.) As before, the input signal $x[n]$ is a discrete-time signal sampled at $f_s = 500$ Hz from the continuous-time signal,

$$x(t) = \sin(2\pi \cdot 4t) + noise \tag{22}$$

where, for each sample, the "noise" component of the signal consists of a uniformly distributed random number in the interval $[-1/2, 1/2]$. Note that as we increase the order of the high-pass filter, low frequencies are attenuated more and more.

We can also observe low-pass and high-pass filtering visually. In images, low-pass filtering results in smoothing of an image, while high-pass filtering results in detection of edges (i.e. sharp transitions in intensity), as shown in Figure 8. The low-pass filtered image had a two-dimensional 20-point averaging filter applied to it, while the high-pass filtered image had a a second-order high-pass filter [equation (19)] applied to it. (See *Mathematica* notebook, section "Simple

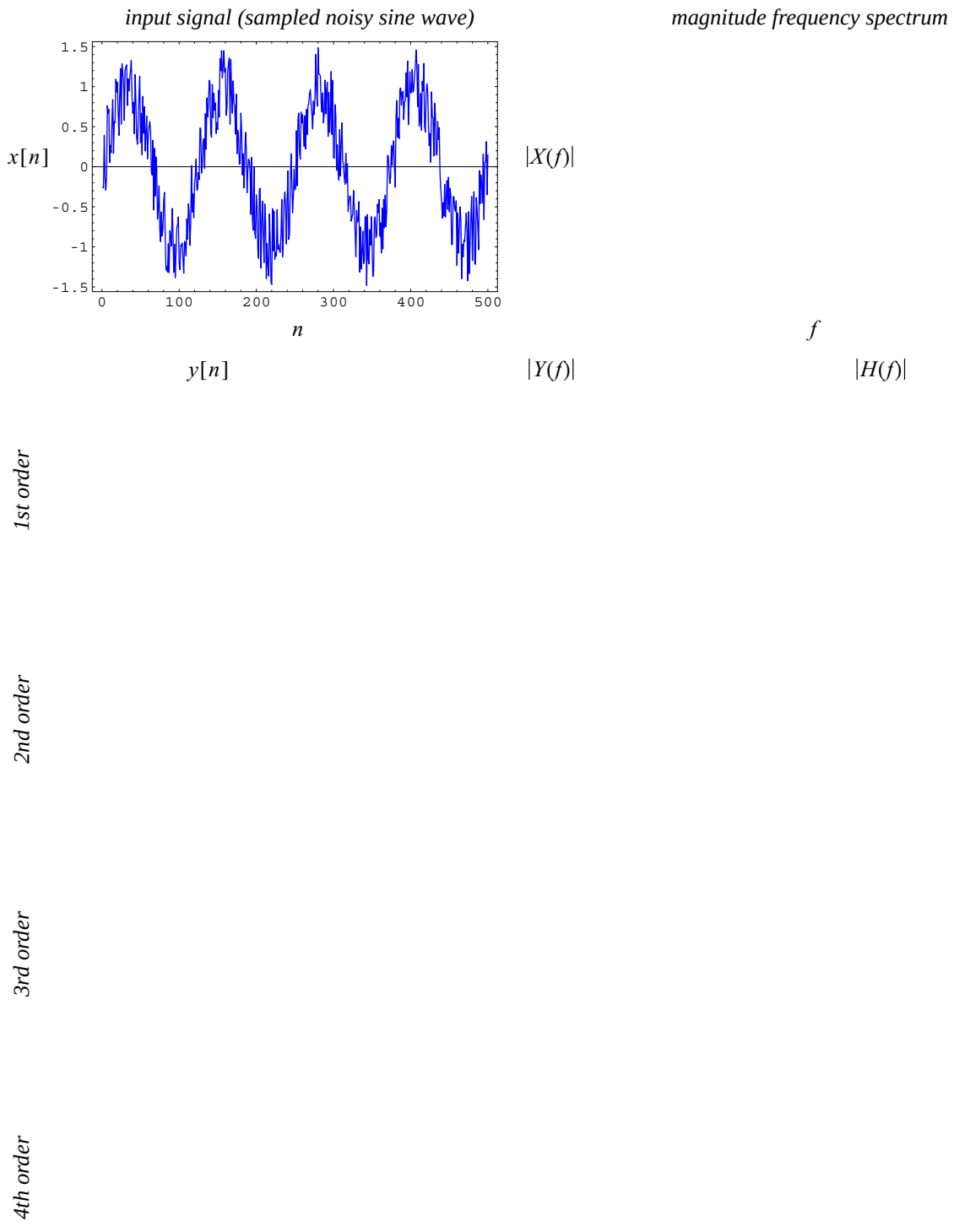
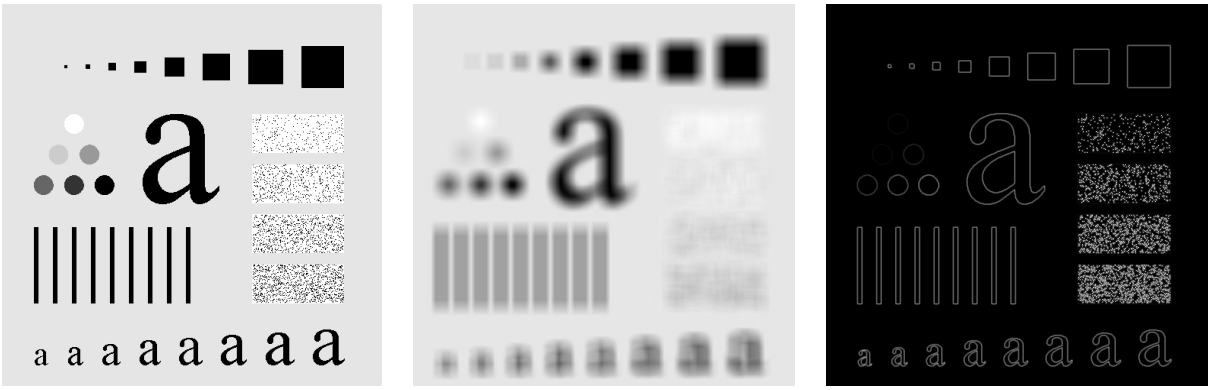


Figure 7: High-pass filtering examples



original image *low-pass filtered image* *high-pass filtered image*

Figure 8

non-recursive filters in two dimensions (image)" for this example, and section "Simple non-recursive filters on speech example" for low and high-pass filtering of a sample speech signal.)

At this point, you may be wondering if there are more systematic ways of designing filters, whether they be low-pass, high-pass or band-pass, than just intuitively guessing at filter coefficients. The answer is, of course, yes, and many filter design algorithms are easily implemented (or already available) in mathematical software packages like *Matlab* and *Mathematica*. In these filter design algorithms, typically the design engineer specifies an idealized desired frequency response, the desired filter order, and the algorithm returns a good approximation of the idealized frequency response given the desired filter order. Below we give an example of one 24-point band-pass filter that was derived using an advanced filter design algorithm. The difference equation for this filter is given by:

$$y[n] = \sum_{k=0}^{23} b_k x[n-k] \tag{23}$$

where the approximate filter coefficients are listed in the table below.

k	b_k	k	b_k	k	b_k	k	b_k
0	-0.0193	6	0.0264	12	0.2442	18	-0.0649
1	0.0099	7	-0.0126	13	-0.3331	19	0.0000
2	-0.0003	8	0.1188	14	0.0000	20	0.0276
3	0.0276	9	0.0000	15	0.1188	21	-0.0003
4	0.0000	10	-0.3331	16	-0.0126	22	0.0099
5	-0.0649	11	0.2442	17	0.0264	23	-0.0193

Figure 9 plots the magnitude frequency response for this filter (red line); an ideal band-pass filter might have the frequency response of the dashed blue line in Figure 9. Note that the 24-point filter approximates the characteristics of an ideal band-pass filter pretty well, with sharp cut-off frequencies f_1 and f_2 beyond which frequencies are almost entirely zeroed, while inside the band, frequencies pass through with virtually no magnitude change. In general, the higher the order of the filter, the better that filter will be able to approximate the frequency response of an ideal filter. (See *Mathematica* notebook, section "Non-recursive difference equation example: well-designed band-pass filter" for

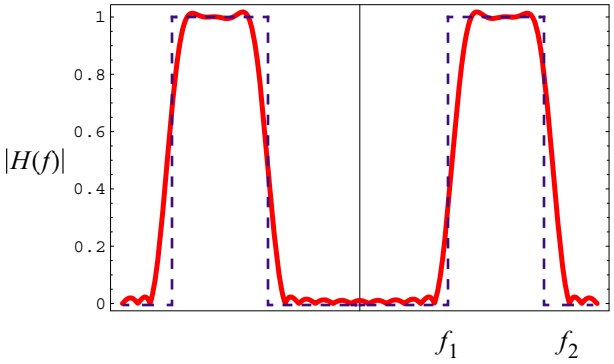


Figure 9

the filtered output corresponding to this filter and a noisy sine wave input.) Later in this course, we will look at how we can design filters like the one in Figure 9, from a user-specified desired frequency response.

So far we have only looked at examples of *non-recursive* difference equations; that is, difference equations where the output is only dependent on time-delayed values of the input signals $x[n]$, $x[n-1]$,..., $x[n-M+1]$. Next time, we will look at a few examples of *recursive* difference equations, and introduce the concept of *stability* for such systems.