Solution Set for Homework #4

By Prof. Brian Evans and Mr. Firas Tabbara

October 12, 2024

PROBLEM 1: SAMPLING AND ALIASING

Prologue: The purpose of this problem is to gain more practice with sampling continuous-time signals to produce discrete-time signals and analyzing aliasing that will occur in practice due to sampling.

- The Sampling Theorem says when sampling a continuous-time signal with a maximum frequency of $f_{\max}$, the sampling rate f_s must be chosen to be greater than 2 $f_{\max}$ to be able to reconstruct the continuous-time signal from its samples.
- Using the relationship $f_s > 2\ f_{\max}$, we can divide both sides by 2 to obtain $f_{\max} < \frac{1}{2} f_s$. Sampling captures non-negative continuous-time frequencies up to, but not including, $\frac{1}{2} f_s$. If we include negative frequencies, the frequencies captured during sampling are $-\frac{1}{2} f_s < f < \frac{1}{2} f_s$.
- In practice, a sampler is an analog circuit that closes and opens at the sampling rate. The sampling rate is limited by the device's switching speed and power needed. A CMOS transistor switching speed is 8 GHz in an older 90nm CMOS process and increases as CMOS process size decreases. The switching speed for GaAs semiconductors can exceed 200 GHz. Power consumption is proportional to the sampling rate, or sometimes its square. See Prof. Murmann's 10-minute video on this.
- Thermal noise due to random motion of electrons that contains all frequencies of roughly equal power up to about 10^{15} Hz and the frequency content gradually decays in strength beyond that point.

Problem: Refer to Fig. 4-26 for the system with ideal C-to-D and D-to-C converters.

Part (a) Suppose that the discrete-time signal is

$$x[n] = 10\cos\left(0.13\pi n + \frac{\pi}{13}\right)$$

If the sampling rate is $f_s = 1000$ samples/sec, determine two different continuous-time signals $x(t) = x_1(t)$ and $x(t) = x_2(t)$ that could have been inputs to the above system; i.e., find $x_1(t)$ and $x_2(t)$ such that $x[n] = x_1(t) = x_2(t)$ if $T_s = 0.001$ sec.

Solution for part (a): From sampling a continuous-time signal at frequency f_0,

$$x[n] = x_1(n\,T_s) = x_1\left(\frac{n}{f_s}\right) = 10\cos\left(2\pi\frac{f_0}{f_s}n + \frac{\pi}{13}\right)$$

where

$$\widehat{\omega}_0 = 2\pi\frac{f_0}{f_s} = 0.13\pi$$

We can solve for

$$f_0 = \frac{\widehat{\omega}_0}{2\pi} f_s = 0.065 * 1000\ \text{Hz} = 65\ \text{Hz}$$

Therefore, the

$$x_1(t) = 10\cos\left(2\pi f_0 t + \frac{\pi}{13}\right) = 10\cos\left(2\pi(65)t + \frac{\pi}{13}\right) = 10\cos\left(130\pi t + \frac{\pi}{13}\right)$$

For the second continuous-time signal, we know from lecture slides 5-11 and 5-12, which come from Section 4-1.2 in *Signal Processing First*, that there is an infinite number of continuous time frequencies of the form $f_0 + kf_s$ and also of the form $-f_0 + kf_s$, where *k* is an integer, that will look like frequency f_0 when sampled at sampling rate f_s.

Another way to think about this is that the discrete-time frequency domain is periodic with period 2π. The discrete-time frequency of 2π corresponds to a continuous-time frequency of f_s:

$$\widehat{\omega}_s = 2\pi\frac{f_s}{f_s} = 2\pi$$

With respect to sampling at sampling rate f_s, a shift in continuous-time frequency of f_s is the same as a shift in the discrete-time frequency domain of 2π. That is,

$$x[n] = 10\cos\left(0.13\pi n + \frac{\pi}{13}\right) = 10\cos\left(0.13\pi n + \frac{\pi}{13} + 2\pi n\right) = 10\cos\left(2.13\pi n + \frac{\pi}{13}\right)$$

where

$$\widehat{\omega}_1 = 2\pi\frac{f_1}{f_s} = 2.13\pi$$

and

$$f_1 = \frac{\widehat{\omega}_1}{2\pi}f_s = 1.065 * 1000\text{ Hz} = 1065\text{ Hz}$$

We can write $f_1 = f_0 + f_s = 65\text{ Hz} + 1000\text{ Hz} = 1065$

Part (b) If the input $x(t)$ is given by the two-sided spectrum representation in Fig. P-4.12, determine a simple formula for $y(t)$ when $f_s = 700$ samples/sec (for both C-to-D and D-to-C converters).

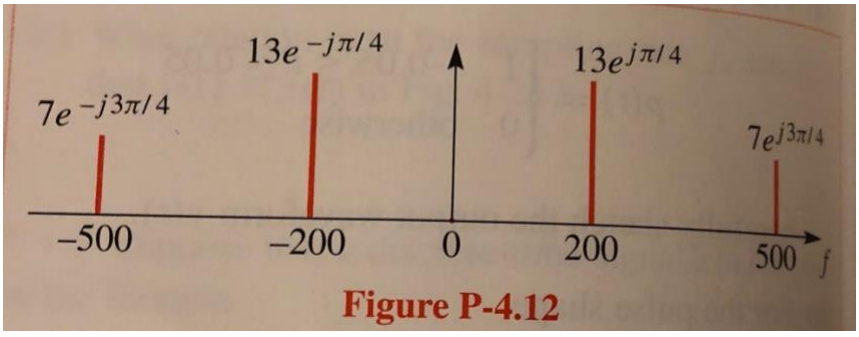


Figure P-4.12

Solution for part (b): Using Figure P-4.12, we can interpret the sinusoidal signals corresponding to the spectral lines by recalling from inverse Euler's formula that

$$A\cos(2\pi f_0 t + \theta) = \frac{A}{2}e^{-j\theta}e^{-j2\pi f_0 t} + \frac{A}{2}e^{j\theta}e^{j2\pi f_0 t}$$

We have a cosine term at 200 Hz with amplitude 26 and phase shift π/4 and a cosine term at 500 Hz with amplitude 14 and phase shift 3π/4:

$$y(t) = 26\cos\left(2\pi(200)t + \frac{\pi}{4}\right) + 14\cos\left(2\pi(500)t + \frac{3\pi}{4}\right)$$

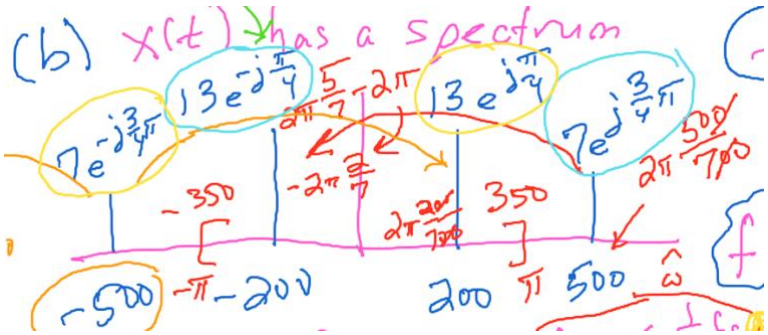
The discrete-time signal obtained by sampling at f_s = 700 samples/sec is:

$$y[n] = 26\cos\left(2\pi\frac{200\text{ Hz}}{700\text{ Hz}}n + \frac{\pi}{4}\right) + 14\cos\left(2\pi\frac{500\text{ Hz}}{700\text{ Hz}}n + \frac{3\pi}{4}\right)$$

For f_1 = 200 Hz, sampling at f_s = 700 Hz does not cause aliasing. However, for f_2 = 500 Hz, sampling at f_s = 700 Hz causes aliasing (folding) because $f_s < 2\,f_2$. After sampling, f_2 = 500 Hz appears the same as -200 Hz. We can show this by simplifying

$$y[n] = 26\cos\left(2\pi\frac{2}{7}n + \frac{\pi}{4}\right) + 14\cos\left(2\pi\frac{5}{7}n + \frac{3\pi}{4}\right)$$
$$y[n] = 26\cos\left(2\pi\frac{2}{7}n + \frac{\pi}{4}\right) + 14\cos\left(2\pi\frac{5}{7}n - 2\pi n + \frac{3\pi}{4}\right)$$
$$y[n] = 26\cos\left(2\pi\frac{2}{7}n + \frac{\pi}{4}\right) + 14\cos\left(-2\pi\frac{2}{7}n + \frac{3\pi}{4}\right)$$
$$y[n] = 26\cos\left(2\pi\frac{2}{7}n + \frac{\pi}{4}\right) + 14\cos\left(2\pi\frac{2}{7}n - \frac{3\pi}{4}\right)$$

Here's another way to see what's going on visually. Sampling causes replicas of each spectral line in $x(t)$ at offsets of multiples of 2π in discrete-time frequencies (in red on the frequency axis) or equivalently f_s in continuous-time frequencies (in blue on the frequency axis):



Using a trigonometric formula from Table 1 of "Sum of Two Sinusoids" by Richard G. Lyons:

$$y[n] = 12\cos\left(2\pi\frac{2}{7}n + \frac{\pi}{4}\right)$$

Therefore, the output of the ideal D-to-C converter is

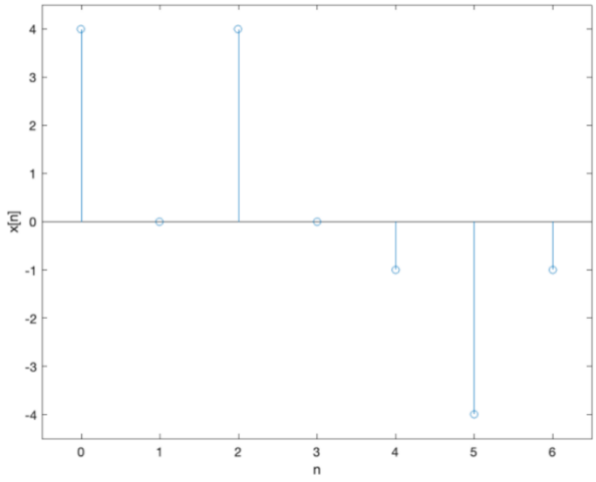
$$y(t) = 12\cos\left(2\pi(200)t + \frac{\pi}{4}\right)$$

PROBLEM 2: FINITE IMPULSE RESPONSE (FIR) FILTER

Problem: A linear time-invariant system is described by the difference equation

$$y[n] = 2\,x[n] - 3\,x[n-1] + 2\,x[n-2]$$

Part (a): When the input to this system is $x[n]$, compute the values of $y[n]$, over the index range $0 \le n \le 10$.




```
x = [ 4 0 4 0 -1 -4 -1 ];
n = 0 : length(x) - 1;
stem(n, x);
xlim( [-0.5, 6.5] );
ylim( [-4.5, 4.5] );
xlabel( 'n' );
ylabel( 'x[n]' );
```

$$x[n] = \begin{cases} 4 & n = 0 \\ 0 & n = 1 \\ 4 & n = 2 \\ 0 & n = 3 \\ -1 & n = 4 \\ -4 & n = 5 \\ -1 & n = 6 \end{cases}$$

Solution for Part (a)

$x[n] = 0$ for $n < 0$ and $n > 6$ therefore: $x[n] = 0$ for $n < 0$ or $n > 6$

y[0] = 2x[0] - 3x[-1] + 2x[-2] = 2*4 - 3*0 + 2*0 = 8
y[1] = 2x[1] - 3x[0] + 2x[-1] = 2*0 - 3*4 + 2*0 = -12
y[2] = 2x[2] - 3x[1] + 2x[0] = 2*4 - 3*0 + 2*4 = 16
y[3] = 2x[3] - 3x[2] + 2x[1] = 2*0 - 3*4 + 2*0 = -12
y[4] = 2x[4] - 3x[3] + 2x[2] = 2*(-1) - 3*0 + 2*4 = 6
y[5] = 2x[5] - 3x[4] + 2x[3] = 2*(-4) - 3*(-1) + 2*0 = -5
y[6] = 2x[6] - 3x[5] + 2x[4] = 2*(-1) - 3*(-4) + 2*(-1) = 8
y[7] = 2x[7] - 3x[6] + 2x[5] = 2*(0) - 3*(-1) + 2*(-4) = -5
y[8] = 2x[8] - 3x[7] + 2x[6] = 2*(0) - 3*(0) + 2*(-1) = -2
y[9] = 2x[9] - 3x[8] + 2x[7] = 2*(0) - 3*(0) + 2*(0) = 0
y[10] = 2x[10] - 3x[9] + 2x[8] = 2*(0) - 3*(0) + 2*(0) = 0

We can verify the solution using MATLAB:

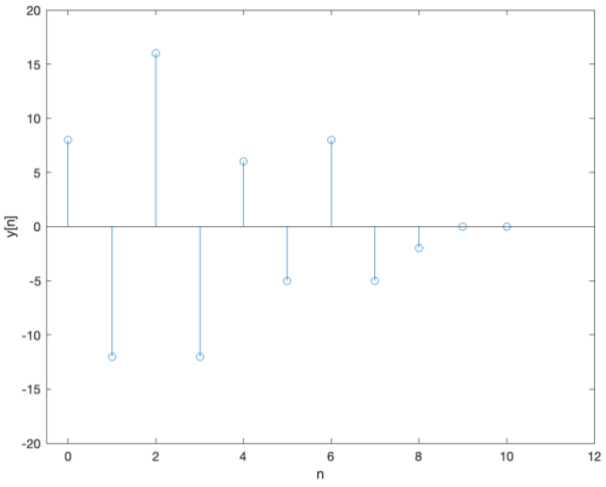
```
x = [ 4 0 4 0 -1 -4 -1 ];
b = [ 2 -3 2 ];
y = conv(x, b);
```

And y is [8 -12 16 -12 6 -5 8 -5 -2].

Part (b) For the previous part, plot $x[n]$ and $y[n]$.

Solution for part (b). The "UT" input signal $x[n]$ was plotted as part of the revised problem.

```
y = [ 8 -12 16 -12 6 -5 8 -5 -2 0 0];
n = 0 : length(y) - 1;
stem(n, y);
xlim( [-0.5, 12] );
ylim( [-20, 20] );
xlabel( 'n' );
ylabel( 'y[n]' );
```



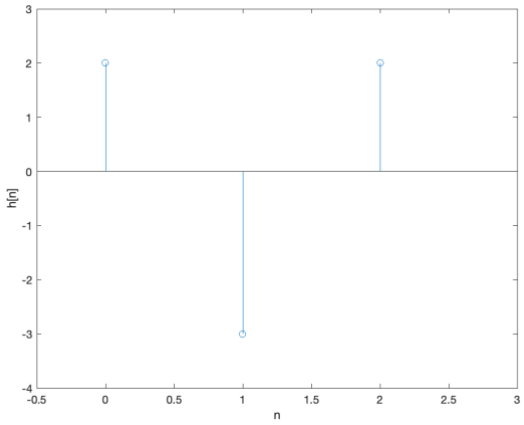
Part (c) Determine the response of this system to a unit impulse input; i.e., find the output $y[n] = h[n]$ when the input is $x[n] = \delta[n]$. Plot $h[n]$ as a function of n.

Solution for part (c): Let $x[n] = \delta[n]$.

Therefore, the output is defined as follows:

$$h[n] = 2\,\delta[n] - 3\,\delta[n-1] + 2\,\delta[n-2]$$

```
h = [ 2 -3 2 ];
n = 0 : length(h) - 1;
stem(n, h);
xlim( [-0.5, 3] );
ylim( [-4, 3] );
xlabel( 'n' );
ylabel( 'h[n]' );
```



Epilogue: FIR filter coefficients are equal to the FIR filter impulse response. FIR filter output is the convolution of the input signal and the FIR filter impulse response (filter coefficients).

PROBLEM 3: SYSTEM IDENTIFICATION

Prologue: A common problem that arises in audio and other systems is characterizing an unknown system. Consider an audio system with a speaker transmitting sound in a concert hall that is recorded by a microphone, and we'd like to characterize the acoustic response in the concert hall.

If we model an unknown system as a linear time-invariant finite impulse response (FIR) filter, we can try to infer its FIR filter coefficients b_k (i.e. the impulse response of the FIR filter). In practice, the unknown system will likely not be linear and time-invariant, but sometimes, a linear time-invariant system model captures useful information about the unknown system for the application at hand.

For the concert hall acoustics example, if we know the FIR filter coefficients for the linear time-invariant model of the concert hall acoustics, then we can mimic the effect of the concert hall by using its FIR coefficients to filter a music track and the output will sound as if it was played in the concert hall. Several audio systems with a "concert hall" effect will list several different concert halls.

Problem:
Here is the input-output relationship for a linear time-invariant FIR filter with *N* coefficients with input *x*[*n*] and output *y*[*n*]:

$$y[n] = b_0\,x[n] + b_1\,x[n-1] + b_2\,x[n-2] + \cdots + b_{N-1}\,x[n-(N-1)]$$

In advance, we don't know the value of *N*.

If we could input a known test signal for *x*[*n*] and observe *y*[*n*], we can attempt to compute the FIR coefficients b_k by *deconvolution*. In the test setup, we would assume that a laptop is sending the test signal to the speakers and the same laptop is receiving the output from the microphone. We'll assume the test will begin at time *n* = 0 and perform deconvolution in the time domain.

The first output value (i.e. when $n = 0$) is

$$y[0] = b_0\, x[0] + b_1\, x[-1] + b_2\, x[-2] + \cdots + b_{N-1}\, x[-(N-1)]$$

For linear time-invariant systems, it is a necessary (but not sufficient) condition for the system to be "at rest", which means that all initial conditions $x[-1], x[-2], \ldots, x[-(N-1)]$ must be zero.

$$y[0] = b_0\, x[0] + b_1\, x[-1] + b_2\, x[-2] + \cdots + b_{N-1}\, x[-(N-1)]$$

Since we know $x[n]$ and $y[n]$ in our test setup, we have one equation and one unknown at $n = 0$:

$$y[0] = b_0\, x[0]$$

and we can compute

$$b_0 = \frac{y[0]}{x[0]}$$

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

Part (a) Develop an algorithm to compute the remaining values of b_k assuming you know N.

Solution for part (a)

The first output value is: $y[0] = b_0\, x[0]$

Therefore, $b_0 = \frac{y[0]}{x[0]}$

The second output value is: $y[1] = b_0\, x[1] + b_1\, x[0]$

Therefore, $b_1 = \frac{y[1] - b_0 x[1]}{x[0]}$

The third output value is: $y[2] = b_0\, x[2] + b_1\, x[1] + b_2\, x[0]$

Therefore, $b_2 = \frac{y[2] - b_0 x[2] - b_1 x[1]}{x[0]}$

The fourth output value is: $y[3] = b_0\, x[3] + b_1\, x[2] + b_2\, x[1] + b_2\, x[0]$

Therefore, $b_3 = \frac{y[3] - b_0 x[3] - b_1 x[2] - b_2 x[1]}{x[0]}$

$$b_N = \frac{y[N] - \sum_{i=0}^{N-1} b_i x[N-i]}{x[0]}$$

Part (b) By hand, compute the values of b_k given and

i. input signal $x[n]$ with non-zero values [1 2 3 4 5]
ii. output signal $y[n]$ with non-zero values [1 1 1 1 1 -5]

Use the MATLAB command conv to convolve $x[n]$ and b_k to make sure the result is $y[n]$.

Solution for part (b)

x[n] = [1 2 3 4 5]
y[n] = [1 1 1 1 1 -5]

$$b_0 = \frac{\mathrm{y}[0]}{x[0]} = \frac{1}{1} = 1$$
$$b_1 = \frac{\mathrm{y}[1] - b_0 x[1]}{x[0]} = \frac{1 - 1 * 2}{1} = -1$$
$$b_2 = \frac{\mathrm{y}[2] - b_0 x[2] - b_1 x[1]}{x[0]} = \frac{1 - 1 * 3 + 1 * 2}{1} = 0$$
$$b_3 = \frac{\mathrm{y}[3] - b_0 x[3] - b_1 x[2] - b_2 x[1]}{x[0]} = \frac{1 - 1 * 4 + 1 * 3 - 0 * 2}{1} = 0$$
$$b_4 = \frac{y[4] - b_0 x[4] - b_1 x[3] - b_2 x[2] - b_3 x[1]}{x[0]} = \frac{1 - 1 * 5 + 1 * 4}{1} = 0$$
$$b_5 = \frac{y[5] - b_0 x[5] - b_1 x[4] - b_2 x[3] - b_3 x[2] - b_4 x[1]}{x[0]} = \frac{-5 - 1 * 0 + 1 * 5}{1} = 0$$

```
x = [ 1 2 3 4 5 ];
b = [ 1 -1 ];
y = conv(x, b);
```

We can confirm that the result of `conv(x, b)` is $y[n]$.

Additional Insight: When convolving two finite-length signals $x[n]$ and b_n, the result $y[n]$ has finite length. The length of $y[n]$ is the length of $x[n]$ plus the number of filter coefficients minus 1. Since the length of $y[n]$ is 6 and the length of $x[n]$ is 5, there are 2 filter coefficients.

Part (c) Write a MATLAB program for your algorithm in (a) and apply it to the signals in part (b) to compute b_k. In your algorithm, stop computing values of b_k when $|b_k - b_{k-1}| \leq 10^{-7}|b_k|$. That way, we don't need to know the value of N in advance. I have the stopping criterion as $|b_k - b_{k-1}| \leq 10^{-7}|b_k|$ instead of $\left|\frac{b_k - b_{k-1}}{b_k}\right| \leq 10^{-7}$to avoid a possible division by zero error.

Solution for part (c) First, we'll provide the code when we know N.

```
%% This is code was worked out today (10-08-2021) in
%% Friday 2-3pm office hours by Prof. Evans.
%% It's a way to validate your answer to 4.4(b) from
%% manual calculations.
%% The code also provides a partial answer for 4.4(c).
%% Keep in mind that the first element in a vector
%% in MATLAB has index 1 and not 0.

x = [ 1 2 3 4 5 ];
y = [ 1 1 1 1 1 -5 ];

% y = conv(x, b) where b is the vector of coefficients
% Length of y = length of x + Number of Coeffs - 1

Nmax = length(y) - length(x) + 1;
b = zeros(1, Nmax);
b(1) = y(1) / x(1);
```

```
% b(2) = ( y(2) - b(1)*x(2) ) / x(1);
% b(3) = ( y(3) - b(1)*x(3) - b(2)*x(2) ) / x(1);
for k = 2:Nmax
    numer = y(k);
    n = k;
    for m = 1:(k-1)
        numer = numer - b(m) * x(n);
        n = n - 1;
    end
    b(k) = numer / x(1);
end
```

We can add the additional test to stop the algorithm by placing the following code right after the line `b(k) = numer / x(1);`

```
    if abs(b(k) - b(k-1)) <= (1e-7)*abs(b(k))
        break
```

Comment: This particular test can cause the algorithm to stop too early in certain cases. If the filter coefficients were for an averaging filter, for example, [¼ ¼ ¼ ¼] the test would stop the algorithm after computing the second coefficient. A better test would have been $|b_k| \leq 10^{-7}$.

Epilogue: Returning to the room acoustics example, the input test signal could be a chirp signal that sweeps all audible frequencies.

We can write the deconvolution form as a system of linear equations in the unknown filter coefficients. Let's start with the equation for the first three values of n:

$y[0] = b_0\, x[0] = x[0]\, b_0$
$y[1] = b_0\, x[1] + b_1\, x[0] = x[1]\, b_0 + x[0]\, b_1$
$y[2] = b_0\, x[2] + b_1\, x[1] + b_2\, x[0] = x[2]\, b_0 + x[1]\, b_1 + x[0]\, b_2$

We can create vector $\vec{y} = [\, y[0]\ \ y[1]\ \ y[2]\ \ \ldots\ \ y[N-1]\,]^T$ and $\vec{b} = [\ b_0\ \ b_1\ \ b_2\ \ \ldots\ \ b_{N-1}\,]^T$ and an $N\ x\ N$ lower triangular matrix

$$\mathbf{A} = \begin{bmatrix} x[0] & 0 & 0 & 0 & \cdots & 0 \\ x[1] & x[0] & 0 & 0 & \cdots & 0 \\ x[2] & x[1] & x[0] & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x[N-1] & x[N-2] & x[N-3] & x[N-4] & \cdots & x[0] \end{bmatrix}$$

to form the linear system of equations

$$\vec{y} = \mathbf{A}\,\vec{b}$$

The algorithm in part (b) to solve for the filter coefficients $\vec{b}$ solved the linear system of equations one filter coefficient at a time using backsubstitution. The Matlab command to solve the linear system of equations for $\vec{b}$ is

```
b = A \ y
```