Carnegie Mellon DSP Electrical & Computer ENGINEERING

Digital Signal Processing (18-491/18-691)
Spring Semester, 2024

Problem Set 2

Issued: 2/1/24

Due: 2/8/24 at midnight via Gradescope

Note: You will be submitting your homework using Gradescope. Write your homework on paper, scan your results, and upload them to the gradescope site.

Reading: During the past week we began our discussion of z-transforms, following the Secs. 3.0-3.2 in the text. During the coming week we will complete our discussion of z-transforms and their properties, including difference equations and the development of the magnitude and phase of the DTFT from the locations of poles and zeros in the z-plane. Be sure to look over the class notes on DTFTs and delta functions that are posted on the Web. Please read the notes on delta functions that are posted, and look over the summary class notes on DTFTs as well.

Problem 2.1: Compute the discrete-time Fourier transforms for each of the following signals. You should obtain analytical expressions for your results (i.e. no infinite sums).

(a) $x[n] = 3(5)^{-|n-2|}$, for all n (Break up the function and solve using superposition.)

(b) $x[n] = \alpha^n \cos(\omega_0 n + \phi)u[n], |\alpha| < 1$

(c) $x[n] = 7$ for all n

Hint: This pair can be solved directly using the inverse-transform relation.

(d) $x[n] = A\cos(\omega_0 n + \phi)$ for all n

Hint: You can obtain this answer easily using your result from part (c), the complex exponential definition of the cosine, and the multiplication property.

(e) $x[n] = A\sin(\omega_0 n + \phi)(u[n] - u[n-9])$

Hint: Obtain the solution for the finite-duration pulse first (which should be of the form $\sin(Nx)/\sin(x)$ plus a linear phase term) and then apply the effect of the multiplication by the sine using the multiplication property.

Problem 2.2: In Problem 1.4(c) you obtained the convolution of the function $x[n] = (1/3)^{n-1}u[n-1]$ with $h[n] = (1/3)^{-n+1}u[-n+1]$.

(a) Using the Fourier transform properties, obtain closed-form expressions for $X(e^{j\omega})$ and $H(e^{j\omega})$, the DTFTs of $x[n]$ and $h[n]$, respectively.

Note: Tou may wish to make use of the relationship discussed last week

$$x[-n] \Leftrightarrow X(e^{-j\omega})$$

Table 2.2 on OSYP page 58 has a complete list of useful DTFT properties. While we discussed most of these properties on Wednesday, we will discuss a number of the remaining properties on Monday.

(b) Again using Fourier transform properties, obtain $Y(e^{j\omega})$, the DTFT of $y[n]$, the result of the convolution of $x[n]$ and $h[n]$ that you obtained in solving Problem 1.4(c).

(c) Show that your expression for $Y(e^{j\omega})$ from part (b) is equal to the product of your expressions for $X(e^{j\omega})$ and $H(e^{j\omega})$ obtained in part (a).

Problem 2.3: In class on January 24 we discussed the computation of the inverse DTFT of the ideal lowpass filter, specifically developing the transform pair

$$h_{LP}[n] = \frac{\sin(\omega_c n)}{\pi n} \Leftrightarrow H_{LP}(e^{j\omega}) = \begin{cases} 1, & |\omega| \le \omega_c \\ 0, & \omega_c < |\omega| \le \pi \end{cases}$$

While this relationship is easily obtained by computing the inverse DTFT of $H_{LP}(e^{j\omega})$ in the usual fashion; it is not at all easy to obtain going from the time doman to the frequency domain. On the other hand, going from the frequency domain to the time domain using the IDTFT is quite straightforward.

(a) Using DTFT properties, obtain $h_{HP}[n]$, the unit sample response of an ideal highpass filter with frequency response

$$H_{HP}(e^{j\omega}) = \begin{cases} 0, & |\omega| \le \omega_c \\ 1, & \omega_c < |\omega| \le \pi \end{cases}$$

(b) Now let us consider the unit sample response $h_{BP}[n]$ of the ideal bandpass filter described by the DTFT

$$H_{BP}(e^{j\omega}) = \begin{cases} 0, & 0 \le |\omega| \le \omega_1 \\ 1, & \omega_1 < |\omega| < \omega_2 \\ 0, & \omega_2 \le |\omega| \le \pi \end{cases}$$

1. Obtain $h_{BP}[n]$ by subtracting the unit sample responses of two lowpass filters with different cutoff frequencies, using the properties of DTFTs. Be sure to specify the cutoff frequencies of the filters.

2. Obtain $h_{BP}[n]$ by multiplying the unit sample response of a lowpass filter by a cosine. Be sure to specify the cutoff frequency of the lowpass filter and the amplitude and frequency of the cosine.

3. Show that the mathematical expressions you obtained for $h_{BP}[n]$ using the two methods above are mathematically equivalent. You can do this most easily by converting the sines and cosines into their representations as sums and differences of complex exponentials and using arithmetic operations to show that the two representations are equivalent.

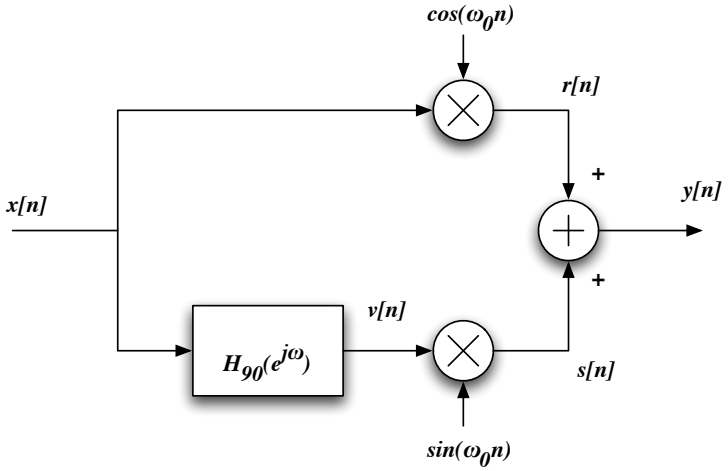

Figure 2.4a. Phase-shift single-sideband modulator.

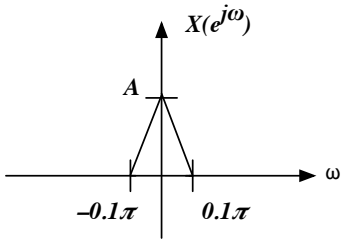

Figure 2.4b. Spectrum of input.

Problem 2.4: Figure 2.4a above is a block diagram of a simple phase-shift single-sideband modulator. In many communications applications single side band (SSB) modulation is preferred because the modulated signal can be transmitted with only half the bandwidth of the original double sideband (DSB) signal without losing any information.

The LSI system labelled $H_{90}(e^{j\omega})$ is an *ideal phase shifter*. It has the response

$$H_{90}(e^{j\omega}) = \begin{cases} j, & -\pi < \omega < 0 \\ -j, & 0 < \omega < \pi \end{cases}$$

and (as usual) $H_{90}(e^{j\omega})$ is periodic in ω with period 2π.

Figure 2.4b depicts the spectrum of the input, the DTFT $X(e^{j\omega})$. The rest of the system is defined by the relationships

$$s[n] = v[n]\sin(\omega_0 n)$$

$$r[n] = x[n]\cos(\omega_0 n)$$

$$y[n] = r[n] + s[n]$$

where $v[n]$ is the output of the phase shifter.

(a) Given the spectrum of the input $X(e^{j\omega})$ shown in Figure 2.4b and a value of 0.4π for ω_0 for, sketch and dimension the following spectra:

(1) $V(e^{j\omega})$ the DTFT of $v[n]$

(2) $R(e^{j\omega})$ the DTFT of $r[n]$

(3) $S(e^{j\omega})$ the DTFT of $s[n]$

(4) $Y(e^{j\omega})$ the DTFT of $y[n]$

(b) Sketch the block diagram of a system that contains ideal multipliers, adders and ideal filters that can be used to recover the input function $x[n]$ from $y[n]$.

Problem 2.5: In this problem we review a subtle property of continuous-time delta functions that is important to understand in order to understand the continuous time to discrete time conversion of periodic time functions.

Recall that we define delta functions in continuous time implicitly, that is by what they do when they are part of an integral, rather than explicitly in terms of how they transform their inputs. Specifically,

$$\int_{-\infty}^{\infty} \delta(t-a)\phi(t)dt = \phi(a)$$

As I frequently say in class, we evaluate these integrals by asking the following three questions:

1. What is the variable being integrated?
2. What is the value of that variable that causes the argument of the delta function to equal zero?
3. What is the value of the remaining part of the integrand (excepting the delta function) for that value of the variable?

The answer to the third question, of course, is the value of the integral.

(a) Apply the rules above to determine the value of the integral

$$\int_{-\infty}^{\infty} e^{-5t}\delta(t-2)dt$$

(b) One corollary of the original definition of the delta function is the familiar integral

$$\int_{-\infty}^{\infty} \delta(t)dt = 1$$

But what does that say about the solution to the slightly different integral below?

$$\int_{-\infty}^{\infty} \delta(3t)dt = 1$$

In truth, the integral of $\delta(3t)$ is not yet defined. Using an appropriate change of variables, obtain the value of k that correctly solves the expression

$$\int_{-\infty}^{\infty} \delta(at)dt = k\int_{-\infty}^{\infty} \delta(t)dt$$

(c) Use your result in part (b) to obtain the solution for the integral

$$\int_{-\infty}^{\infty} e^{-2t}\delta(5t-20)dt$$

MATLAB Problems

Reading: We will be using MATLAB for verifying homework results, and for use as a design tool throughout the course. In working the MATLAB problems, turn in a printout of your results, a copy of the MATLAB code you developed to work the problem, as well as any additional comments you'd like to add.

If you are not already familiar with MATLAB , we warmly recommend that you look over the tutorial `MatlabTutorial` written by our own Carlos Taveras (based on an earlier tutorial by John Shi). It is posted on the course website. In addition, there are a number of tutorials on the Web

that may be found easily by Googling something like `MATLAB intro` or `MATLAB tutorial`.

In working the MATLAB problems, turn in a printout of your results, a copy of the MATLAB code you developed to work the problem, as well as any additional comments you'd like to add. Please use the MATLAB command `publish` to publish the MATLAB code.

For Python users: This semester for the first time users of Python will be permitted to turn in the computer assignments in Python. We remind you that not all computer assignments will be amenable to Python implemenations and the level of support for Python will be lower than for MATLAB. We thank Ashwin Pillay for providing support for the Python implementations.

The recommended general local Python configuration is

```
- Python Version: v3.10.x
- Libraries & Recommended Versions
   - SciPy: 1.10.0
   - NumPy: 1.24.1
   - Matplotlib: 3.6.3
```

Some general recommendations for Python users include:

- Store signals as numpy arrays
- Read the online documentations of the functions we use to understand the differences between the MATLAB and Python implementations (although they are mostly like-for-like, nuanced differences may exist for a few of these functions)
- You would be able to find Python equivalents for most MATLAB functions by searching for `Python equivalent for <MATLAB_function_name>` online. `Stack overflow` often helps as well.

Problem C2.1: In this problem you will develop a MATLAB routine that evaluates DTFTs and you will use it to check your results in Problem 2.1. Please also look over the documentation for the MATLAB routine `freqz`. This M-file will be used frequently when you are describing the frequency response of an LSI system.

You will be given a main file called `main_2_1.m` that you must complete.

(a) Write a short MATLAB script called `dtft_491.m` that calculates the DTFT of discrete-time functions. The function should begin with the descriptive preamble

```
function [X] = dtft_491(x,n,w)
  % Computes Discrete-time Fourier Transform
  % [X] = dtft_491(x,n,w)
  %
  % X = DTFT values computed at frequencies w
```

```
% x is a finite-duration sequence over n
% n is the vector of "time" values over which the computation is
% performed
% w is a vector of frequencies used in the output
```

(b) Use your function to compute and display the magnitude and phase of the DTFTs you developed in Problem 2.1. Note that these results will be inexact because the time functions in Problem 2.1 are generally infinite in duration, and because the frequency responses in some cases are impulsive. Nevertheless, try to explore values of the vectors `n` and `w` that provide reasonable approximations.

(c) Compare the results you get using the function you developed with the results provided by the built in MATLAB function `freqz`. Note that `freqz` uses normalized frequency, and returns the magnitude in decibels, at least using the default inputs. Look over the help file for `freqz` to find out how to store the results in a variable, which you then can plot in any fashion desired.

What you will upload for the Gradescope MATLAB submission:

- `main_2_1.m`
- `dtft_491.m`
- All of the plots
- Answers to all of the questions

For Python users:

Suggested Python equivalents of MATLAB functions:

- `freqz -> scipy.signal.freqz`

Starter code:

```
def dtft_491(x,n,w):
   # Computes Discrete-time Fourier Transform
   # [X] = dtft_491(x,n,w)
   #
   # X = DTFT values computed at frequencies w
   # x is a finite-duration sequence over n
   # n is the vector of "time" values over which the computation is performed
   # w is a vector of frequencies used in the output

   return NotImplemented
```

Problem C2.2:

We developed the unit sample response for an ideal lowpass filter in class on January 24, and you obtained the corresponding unit sample responses for ideal highpass and bandpass filters in Problem 2.3. A fundamental problem with all of these unit sample responses is that they are (in principle) nonzero for all n from $-\infty$ to ∞.

In this problem we will explore one easy way of using your answers for Problem 2.3 as the basis of a practical realizable digital filter. Specifically, we will truncate both tails of the ideal unit sample responses equally on both ends and then delaying the resulting finite-duration sample response enough to make it causal. (This approach is a special case of the window design method, which will be discussed in detail later in the semester.) For example, in the case of the lowpass filter, we would do the following:

1. $h_N[n] = \begin{cases} h_{LP}[n], & -N \le n \le N \\ 0, & \text{otherwise} \end{cases}$

2. $h[n] = h_N[n-N] = \begin{cases} \frac{\sin(\omega_c(n-N))}{\pi(n-N)}, & 0 \le n \le 2N \\ 0, & \text{otherwise} \end{cases}$

You will be given a main file called `main_2_2.m` that you must complete.

(a) Write a simple MATLAB function called `lowpass_491.m` that implements the lowpass filter $h[n]$ with ω_c and N as input arguments.

1. Display the frequency response of your filter using the command `freqz(h,1)`. As noted above, the magnitude is in decibels.

2. What is the effect of changing the values of the parameters ω_c and N? How would you decide how to select numerical values for these parameters?

(b) Now we will consider the impact of your lowpass filter on audio input. Read in a short clip of audio using the commands:

```
[x,fs] = audioread('PSO_B1short.wav');  % read in audio excerpt
x = mean(x,2); % convert it to mono
```

Use a value of $\omega_c = 0.35\pi$ and a value of your choosing for N. (Because the sampling rate `fs` is 16,000 Hz, this value of ω_c corresponds to an actual cutoff frequency of 2800 Hz, as we will discuss later in the semester.) Convolve the audio signal `x` with your filter response `h` using the MATLAB command `y = conv(x,h)`.

1. Listen to the first two seconds of the audio before and after filtering using the MATLAB commands

```
soundsc(x(1:2*fs),fs)
pause(2)
soundsc(y(1:2*fs),fs)
```

To what extent does it appear that the filter is performing as expected?

2. Compare the spectrograms of the first two seconds of the audio before and after filtering using the commands:

```
subplot(2,1,1),specgram(x(1:2*fs),[],fs)
subplot(2,1,2),specgram(y(1:2*fs),[],fs)
```

Note: MATLAB is phasing out the function `specgram` in favor of the far less intuitive function `spectrogram`. The following commands should produce the same result, at least for now:

- `specgram(x,[],fs)`
- `spectrogram(x,hamming(320),160,[],fs,'yaxis'); colormap jet; colorbar off;`

What to upload for your Gradescope MATLAB submission:

- `main_2_2.m`
- `lowpass_491.m`
- Answers to all the questions
- All of the plots

For Python users:

Suggested Python equivalents of MATLAB functions:

- `freqz -> scipy.signal.freqz`
- `[x,fs] = audioread(PSO_B1short.wav) -> fs,x = scipy.io.wavfile.read('PSO_B1short.wav')`
- `conv -> numpy.convolve` item `soundsc ->` (Alternative: saves the sound file as a .wav file) `scipy.io.wavfile.write('destination_file_name.wav',fs,x)`
- `specgram(x(1:2*fs),[],fs) -> matplotlib.pyplot.specgram(x[1:2*fs],Fs=fs)`

Starter Code:

```
fs,x = scipy.io.wavfile.read('PSO_B1short.wav') # read in audio excerpt
x = np.mean(x,axis=1) # convert it to mono
```