ECE 320 SIGNALS & SYSTEMS II

Matlab Project 1: Sample Rate Conversion

Spring 2017

Issued: Tuesday, March 7, 2017 **Due:** Friday, March 31, 2017 at 11:59pm

---

This is an *individual* project. Each student will turn in their own solutions.

The purpose of this project is to explore upsampling and downsampling of discrete-time signals. You will compare analytical calculations with Matlab simulation results. In the final part, you will demonstrate what you've learned by changing the sample rate of an audio signal. This project is adapted from a similar one in *Computer Explorations in Signals and Systems Using Matlab, Second Edition* by Buck, Daniel, and Singer.

1 Preliminary Research **20 points**

The purpose of this preliminary research project is to learn about upsampling and downsampling, which you will use in the remainder of the project. For this part you will prepare and submit a short report. The report should answer the following questions:

- What is downsampling and/or decimation? How does it work?
- What is upsampling and/or interpolation? How does it work?
- What types of applications use up/down sampling?
- Search the IEEE Xplore database for references on upsampling/interpolation and downsampling/decimation. Provide brief summaries of two papers that discuss these topics.
- In addition to the technical information, your report should include a discussion of how you located the references you used. Which sources were most helpful to you? Why? Are there any other types of sources you could potentially consult?

Your report should be typed. It must be 2-4 pages. See the IEEE style guide for information on citations: `http://www.ieee.org/documents/ieeecitationref.pdf` Important reminders:

- The 4-page limit includes **everything**: references, title and author information, etc.
- Your references must be cited in the technical discussion, not just at the end of the paper.
- Your preliminary research project should be submitted in a file of its own, i.e., separate from the rest of the project.

2 Analytical Calculations **10 points**

Consider the infinite-length discrete-time signal $x[n]$ defined below:

$$x[n] = \left(\frac{\sin(A\pi(n-D))}{A\pi(n-D)} \right)^2$$

(a) Provide a rough sketch of this discrete-time signal. This signal is symmetric about its maximum value. Determine the maximum value and the point of symmetry. Also determine the locations of the zero crossings of the signal, and indicate those on your sketch.

(b) Determine an analytical expression for the DT Fourier Transform (DTFT) $X(e^{j\Omega})$ of $x[n]$. Show your work. You may find it helpful to first find the DTFT of the signal $s[n] = \frac{\sin(\Omega_c n)}{\pi n}$.

(c) Sketch the DTFT magnitude $|X(e^{j\Omega})|$.

3 Upsampling **20 points**

In this part you will consider finite segments of two signals:

$$x_1[n] = \left(\frac{\sin(0.4\pi(n-64))}{0.4\pi(n-64)}\right)^2 \qquad 0 \le n \le 127,$$

$$x_2[n] = \left(\frac{\sin(0.2\pi(n-64))}{0.2\pi(n-64)}\right)^2 \qquad 0 \le n \le 127.$$

(a) Using Matlab's `sinc` command, define vectors containing the 127 samples of $x_1[n]$ and $x_2[n]$. Plot these signals using the `stem` command. Verify that the signals are symmetric about the correct point, and have a peak value and zero crossings equal to those predicted by your analytical work in the previous section.
Important note about `sinc` command in Matlab:
Type `help sinc` to read about the sinc command. Note that Matlab defines sinc as $\frac{\sin(\pi x)}{\pi x}$. You pass it the argument x and it multiplies x by π and then computes $\frac{\sin(\pi x)}{\pi x}$. Thus, you do not need to include the factor of π in what you send to it. In other words, I could define the vector of samples for the signal $x_1[n]$ with the following commands:

```
n=0:127;
x1=sinc(.4*(n-64)).^2;
```

(b) Use the `fft` command to compute 2048 samples of the DTFT of $x_1[n]$ and $x_2[n]$. Plot the DTFT magnitudes $|X_1(e^{j\Omega})|$ and $X_2(e^{j\Omega})|$. How do these plots compare with the analytical prediction you obtained in Section 1? Since the vectors `x1` and `x2` contain only a truncated portion of the signals $x_1[n]$ and $x_2[n]$, we would not expect the results to be identical, but they should be close.

See the end of the project description for hints about plotting the output of the `fft` command.

(c) The first step in upsampling is to expand the signal. The expanded signal $x_e[n]$ is defined as follows

$$x_e[n] = \begin{cases} x[n/L] & n = 0, \pm L, \pm 2L, \pm 3L, \ldots \\ 0 & \text{otherwise.} \end{cases}$$

Expanding a signal by a factor of L consists of inserting $L-1$ zeros between the samples. In Matlab a vector `x` can be expanded using the following commands:

```
xe=zeros(1,L*length(x));
xe(1:L:length(xe))=x;
```

Define the signals `xe1` and `xe2` by expanding the signals `x1` and `x2` by a factor of $L = 3$. Based on your knowledge of the expansion operation, sketch what the magnitude of the DTFT of these two expanded signals should look like. Confirm your results by plotting the magnitude of the 2048-pt FFT of the signals `xe1` and `xe2`.

(d) After expansion, the second step in the upsampling process is to lowpass filter the expanded signals. The cutoff frequency of the lowpass filter is $\Omega_c = \frac{\pi}{L}$ and the gain is L. Design a 65-point finite impulse response (FIR) lowpass filter to complete the upsampling by a factor of 3 of the signals `x1` and `x2`. The *Hints* section below provides directions for designing this filter. You may use those directions or you may use your own code to design the filter. Include plots of both the impulse response and frequency response magnitude of your filter in your report.

(e) Use the `filter` command to filter $x_e[n]$ and obtain the interpolated $x_i[n]$ for each of your signals, ***i.e.***,

```
xi=filter(hlpf,1,xe);
```

where `hlpf` contains the impulse response of your FIR filter. Use `fft` to compute the Fourier transform of the interpolated signals `xi1` and `xi2`. Does the magnitude of these transforms look like it should if the continuous-time signals had been sampled 3 times faster?

4 Downsampling **20 points**

In this part you will consider downsampling the signals $x_1[n]$ and $x_2[n]$ defined above.

(a) Suppose that you want to downsample these signals by a factor of $M = 2$. Based on your analytical results, which of these signals can be downsampled by 2 without aliasing? Sketch the DTFT of each of the downsampled signals. (This sketch should be based on your analytical calculations, not Matlab results.) If there is aliasing, indicate which frequencies are affected and which are not.

(b) It is straightforward to downsample a signal in Matlab. The following command downsamples the signal in the vector `x` by a factor of M:

```
xd=x(1:M:length(x));
```

Downsample the signals `x1` and `x2` and store the results in the vectors `xd1` and `xd2`, respectively. Compute the 2048-point transforms of `xd1` and `xd2` and plot their magnitudes. Do your plots agree with your sketches from the previous part?

(c) You should have found that one of the signals was aliased when you downsampled it. To avoid aliasing, a lowpass filter is typically implemented prior to downsampling. This filter, known as an anti-aliasing filter, has a gain of 1 and a cutoff of $\Omega_c = \frac{\pi}{M}$. Design an anti-aliasing filter for the $M = 2$ case, and implement it prior to downsampling the signal that was aliased. Use the `fft` to compute the transform and plot the magnitude of the result.

5 Changing the sample rate of an audio signal **20 points**

In this part you will use what you've learned about upsampling and downsampling to change the sample rate of an audio signal. The file `orig11k` contains a recording of Homer Simpson that has been sampled at 11025 Hz. Your job is to change the sample rate to 8820 Hz. This can be done by first interpolating the signal to a higher rate and then downsampling.

(a) Determine the integer factors you have to upsample and downsample by in order to implement the sample rate change from 11025 Hz to 8820 Hz.

(b) Does it matter whether you upsample or downsample first? Why?

(c) How many filters are required to implement this sample rate change?

(d) Implement the sample rate change in Matlab and store the results in a variable called `x8k`. Play your results. Verify that the signal is still intelligible by playing both the original signal and the new signal using the `soundsc` command. (Remember to tell `soundsc` the sample rate for each signal.)

(e) In addition to including the code for this part in your report, you should also upload a *single* script file containing the code to implement the sample rate change to Blackboard. Note that this script should *not* contain the code for any other part. You can assume that Prof. Wage has the sound file `orig11k.mat` and will store it in the same directory as your script file. Your script file should be called `firstlastname.m` where `firstlastname` is replaced with your first and last name (combined together as one word).

6 Report **10 points**

Prepare a report according to the Laboratory Project Report Guidelines posted on the ECE 320 Blackboard site.

7 Hints

- You can define a vector of frequencies for plotting the output of 2048-point FFT as follows:

```
k=0:2047;
wk=2*pi*k/2048;
```

- When plotting DT frequency responses, it is often helpful to normalize the frequency axis by π to make it easier to read the graph. Using the `wk` vector defined above, the following code computes samples of the DTFT and plots its magnitude versus a normalized frequency axis.

```
Xw=fft(x,2048);
plot(wk/pi,abs(Xw));
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|X(e^{j\Omega})|')
```

- FIR filter design:
  The impulse response of the ideal lowpass filter (LPF) is

$$h_{\text{lpf}}[n] = \frac{\sin(\Omega_c n)}{\pi n}.$$

  This impulse response is infinite length in both directions, and the filter cannot be implemented in Matlab. A simple way to design a lowpass filter that approximates the ideal LPF is to truncate the ideal impulse response. Since a sharp truncation of a signal can introduce high frequency components in the frequency response, we often multiply the truncated signal by a smooth window. The following Matlab code defines a P-point segment of the ideal filter with cutoff frequency $\Omega_c = \alpha\pi$. Note that P is an odd integer. The impulse response is multiplied by a Hamming window function.

```
alpha=0.5;
P=65;
hlpf=alpha*sinc(alpha*(-(P-1)/2:(P-1)/2)).*(hamming(P)');
```