Lab 3: Low-Pass Filtering and High-Pass Filtering in Time Domain and Frequency Domain

Professor Lang Tong

October 27–30, 2014

1 Introduction

- In this lab you will see two filters, one is the 2 -point running average filter, and the other one is the difference filter.
- The 2-point running average filter is a low-pass filter, since the low frequency components of the input signal are preserved, while the high frequency components are attenuated. To see this effect, you will work in the time domain (remember convolution?) to observe the output of several different input signals with different frequencies. You will see that the higher the frequency of the input signal, the larger the attenuation in the output, hence called a "low-pass filter". Then you will work under frequency domain, generate the frequency response plot, and you will be able to tell from the plot directly that the 2-point running average filter is a low-pass filter, without having to try all different kinds of input signals.
- You will apply the filters you learned to some images. Images can also be considered as signals. You will see (instead of the hear) the results of filtering this time.

2 Pre-lab

1. Given a complex sinusoidal signal $x[n] = A\exp(j\Omega_0 n) = A(\cos(\Omega_0 n) + j\sin(\Omega_0 n))$ as input, verify that the output of the system $y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1]$ is given by $y[n] = H(\Omega_0)x[n]$, where $H(\Omega) = \frac{1}{2}(1+\exp(-j\Omega))$ is called the *frequency response* of the system $y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1]$. Note that $H(\Omega)$ is complex.
2. Following the previous problem, calculate the output signal for the following input signals:
   (a) $A\exp(j0n)$
   (b) $A\exp\left(j\frac{\pi}{3}n\right)$, and

(c) $A\exp(j\pi n)$.

What is the maximum amplitude of each of the output signals? What is the frequency of each of the input signals?

3. A low-pass filter is a filter that blocks high frequencies from an input signal (it passes only low frequencies, with some designed cut-off frequency between the blocked and allowed frequencies), and a high-pass filter blocks low frequencies from an input signal (it passes only higher frequencies, also at some designed cut-off frequency). Now, does the system $y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1]$ behave like a low-pass filter or a high-pass filter?

4. This problem is to teach you some basics of manipulating images in MATLAB. An image can be represented as a function I(m,n) of two variables *vertical* (`m`) and *horizontal* (`n`) coordinates (notice the order). A common convention is to set the upper left corner as origin (0,0). For color R-G-B images, the function I(m,n) is a vector function: I(m,n) = [R(m,n), G(m,n), B(m,n)], where the three components specify the intensity of the Red, Green, and Blue color component. In this lab, we will only deal with monochromatic images. You can think of the intensities of this single component as the average of the three color components, although this does not have to be strictly the case. Let's open a monochromatic image and display it:

```
>> I = imread('cameraman.tif');
>> imshow(I)
```

You should see a cameraman in a new window. This image file comes with MATLAB, so you don't have to download it from the Blackboard. There are also other images that come pre-loaded with MATLAB: moon.tif, tire.tif, trees.tif, among others. You can try your code later in the lab with these images as well.

(a) What is the size of the image?

(b) What are the maximum and minimum values of this image? What are the coordinates of these extremes? You might need the commands `max`, `min`, `find`. Finding the coordinates of the extremes might require more trial and error, and is an optional problem.

(c) Use MATLAB commands to modify and display the array I so that a black vertical line is shown at the 15^{th} column from the right.

(d) How many gray levels are displayed by the function `imshow`? Read the help documentation. You can verify your answer to the previous question by typing

```
>> m=colormap
```

and examining the contents of `m`. Read `help colormap` and briefly (no more than 30 words) explain what the values in matrix `m` represent.

Monochrome images are also called grayscale images, as they are often (but do not necessarily have to be) displayed in black and white and shades of gray, as you now see with the cameraman. We emphasize the word "displayed", since you have the freedom to decide in what color to display the intensities. In other words, you can define the mapping from intensity to color by yourself.

(e) Modify the how cameraman (or moon, tire, trees) is displayed by using `colormap` (without modifying the values in image `I`) so that the lowest intensity is displayed in red, and the highest in blue, and no other colors.

Now you should understand that a grayscale image does not necessarily have to be in white and black and gray. MATLAB has lots of predefined colormaps, but the one we will be using from now on is `colormap(gray(256))`, which is the one the cameraman was displayed at default when using a 24-bit monitor display. MATLAB uses `colormap(gray(64))` if it detects your computer has a lower display setting.

3 In-lab

1. The 2-point running average filter is defined as $y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1]$. It takes 2 consecutive points of signal x and stores the average value in y. In the Pre-lab you learned that $H(\Omega) = \frac{1}{2}(1 + \exp(-j\Omega))$ is the frequency response of the 2-point running average filter $y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1]$. Go back to the Pre-lab and review it before you proceed. Plot the magnitude for $H(\Omega) = \frac{1}{2}(1 + \exp(-j\Omega))$. Use the following MATLAB code:

```
b=[0.5, 0.5];
w=-pi:(0.01*pi):pi;
H=freqz(b, 1, w);
plot(w, abs(H));
```

The "`b`" in the command `freqz` corresponds to the two "0.5" coefficients in $y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1]$. This filter can be called a low-pass filter for the following reason: From the plot that you just generated, you can see that this filter fully keeps the DC components of an input signal. The frequency components of the input signal are attenuated more and more as the frequency gets closer to $\pm\pi$.

TA CHECK: Show your plot. In this problem you saw from the frequency response that the filter behaves like a low-pass filter, while in the pre-lab, you were asked to guess its behavior from

three input signals with different frequencies. Which way do you think is easier or better to determine the frequency response of a filter?

2. Now consider the difference filter $y[n] = x[n] - x[n-1]$.

   (a) Use MATLAB to plot the magnitude of the frequency response of the difference filter.

   (b) Is this filter a lowpass or highpass filter? Is there a frequency that this filter completely cuts off?

   (c) If you input the signal $x[n] = 2.3$ for $n = 1$ to 100 to the difference filter, what do you get as output? What happens to the DC components of the input signal? Ignore the end effects (first and last sample from output).

   (d) Use MATLAB to generate a signal of the form $x[n] = \cos(0.01\pi n)$. Use MATLAB convolution (`conv`) to generate the output (see sample code below). Plot the input and output signals on the same figure (use the command `hold on`), and label the axes clearly.

```
n=0:1000;
x=cos(0.01*pi*n);
plot(x)
h=[1 -1];
y=conv(x,h);
hold on
plot(y,'r')
```

   (e) From the plot, estimate the amplitude and frequency of the output signal.

   (f) Is the frequency of the output signal the same as the input signal?

   (g) Use the frequency response plot you got in (a) to estimate the amplitude of the output signal. Does it match with the result in (e)?

   TA CHECK: Show your plot and be prepared to answer all the questions from the above parts.

3. The 7-point running average filter is defined as $y[n] = \frac{1}{7}x[n] + \frac{1}{7}x[n-1] + \ldots + \frac{1}{7}x[n-6]$. It takes 7 consecutive points of signal x, and stores the average value in y.

   (a) Consider each row of an image as a one-dimensional signal. Apply the 7-point running average filter to each row of the cameraman image (see sample code in Part b below). Save the resulting image as `newI`. If MATLAB complains something about `uint8`, here is a hint: use the `whos` command to see the type of the variables currently in memory. Some functions require specific datatypes, for example, `double`, `uint8`. If `z` has type `uint8`, you can simply type `z=double(z)` to convert it to double precision.

(b) Display the resulting image. Although MATLAB says `imshow` supports `uint8` and `double` (read `help imshow`), you should do some type conversions until you see the cameraman. Is the image more blurred or sharpened? Also, after the convolution, the image is no longer the same size. Why is this? Here is some code:

```
nn=0:100;
xx=square( 0.05 * pi * nn );
bb=[1/7 1/7 1/7 1/7 1/7 1/7 1/7];
yy=conv(bb,xx);
plot(nn,xx)
hold
plot(0:length(yy)-1,yy,'r')
I = imread('cameraman.tif');
imshow(I)
bb=[1/7 1/7 1/7 1/7 1/7 1/7 1/7];
for i=1:256
yy2(i,:) = conv(bb,double(I(i,:)));
end
imshow(uint8(yy2))
```

(c) Above, we have filtered each row, but filtering can also be done on the columns. Modify your code to filter each column and plot the result. What is different?

(d) The MATLAB command `imfilter` is very useful for filtering images (read `help imfilter`). Modify your code above using the function `imfilter` and comment on what happens.

(e) Repeat the above with a 31-point running average filter. Compare it with the 7-point filter and describe the difference.

(f) Plot the magnitude of the frequency response of the 7-point and 31-point filters on one plot using different colors. Use this plot to help explain your result from (a) and (e).

(g) Let $y[n] = -x[n]+3x[n-1]-x[n-2]$. Apply this filter to cameraman. Compare it with the 7-point running average filter and describe the difference.

TA CHECK: Show your plots and be prepared to answer all the questions from the above parts.