Lab 1 - Matlab Exercises

Colt Thomas

June 27, 2021

Introduction

The purpose of this lab is to establish the fundamental skills in MATLAB that will be used in the remainder of the class. The lab is broken into three parts:

1. Plotting Basic Signals
2. Time Shifting and Scaling
3. Signal Spectral Analysis

The conclusion will walk through the details of work performed, and will include figures. Essential tools will also be emphasized. Code used for this lab is in the Appendix.

Conclusion

Part 1 - Plotting Signals

The objective of this part was to plot three basic signals that are commonly used in signal processing:

1. Sine Wave (1 Hz)

$$x(t) = sin(2\pi t)$$

2. Unit Rectangle Function

$$\Pi(t) = \begin{cases} 1 & |x| \leq \frac{1}{2} \\ 0 & |x| > \frac{1}{2} \end{cases}$$

3. Unit Triangle Function

$$\Delta(t) = \begin{cases} 1 - 2|x| & |x| < \frac{1}{2} \\ 0 & |x| > \frac{1}{2} \end{cases}$$

Figure shows the three signals in the top row before the shifting and scaling modifications are made. Lines 13-21 in the Matlab code are where the signals are defined, with the rectangle and triangle functions being defined from scratch.

Part 2 - Signal Operations

The signals in part 1 were time shifted (by 0.5s) and time compressed (by 3) for the second portion of the lab. These were included in figure as well for comparison to the original signals. Notice that the time shifting moves the signal right on the time axis, while the compression scales inwards toward $t = 0$. These are the results I expected, and it should be noted that these operations are easily done by multiplying/adding/subtracting the time vector in Matlab.

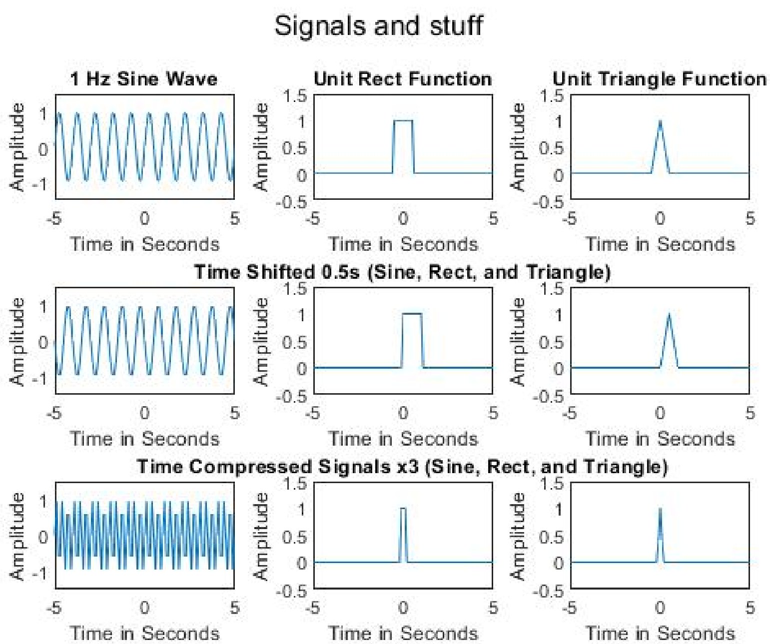

Figure 1: The first row are the original signals, with the subsequent rows illustrating shifted and time compressed variations.

Part 3 - Signal Spectra

The fft and fftshift commands are useful skills for analyzing signals in the frequency domain. Note that when using the fft command, it actually performs the discrete fourier transform on the vector passed in. Plotting the spectrum is done in a similar fashion to the time domain signal, except that you replace your time vector with a frequency vector. I did this on line 114 in the code by creating f with a length of the spectrum vector, and had to multiply by my "sampling rate" to scale the unit spectrum to the proper frequency range.

Part A required a spectrum plot of the signals from figure ; I noticed that the time shifting didn't effect the signal spectrum while the compression did. This is because a time shift only effects the phase of the signal, but not the frequency. Compression scales the frequency to a larger value, so the bandwidth stretches out by a factor of 3 while also reducing the magnitude.

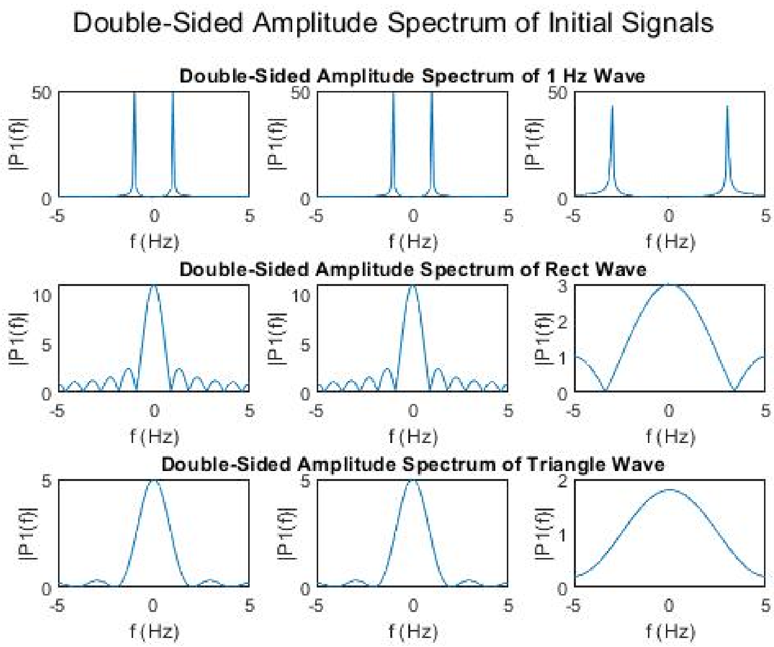

Figure 2: Spectrum of the same signals represented in Figure 1. Notice how the spectrum changes by operation: The first column is the original signal, the second is the time delay, while the third is time compressed.

Part B applied the fft to a real-world signal imported into Matlab using the audioread command. The audio is just a voice singing a constant pitch for 9 seconds. In figure you can see the spectrum of the voice. I expected to see a peak at whatever pitch the singer was playing, but to my surprise there was a harmonic being picked up! The pitch was at 395 Hz (a slightly sharp G in music) with a harmonic at the next octave of 790Hz. The spectrum is a sum of all frequencies for the duration, so you can see there is some variation from not holding a perfect tone. This is more of a fun fact but bands and orchestras are taught that being in tune makes the group sound louder. The spectrum shows this since exact frequency components add together in magnitude.

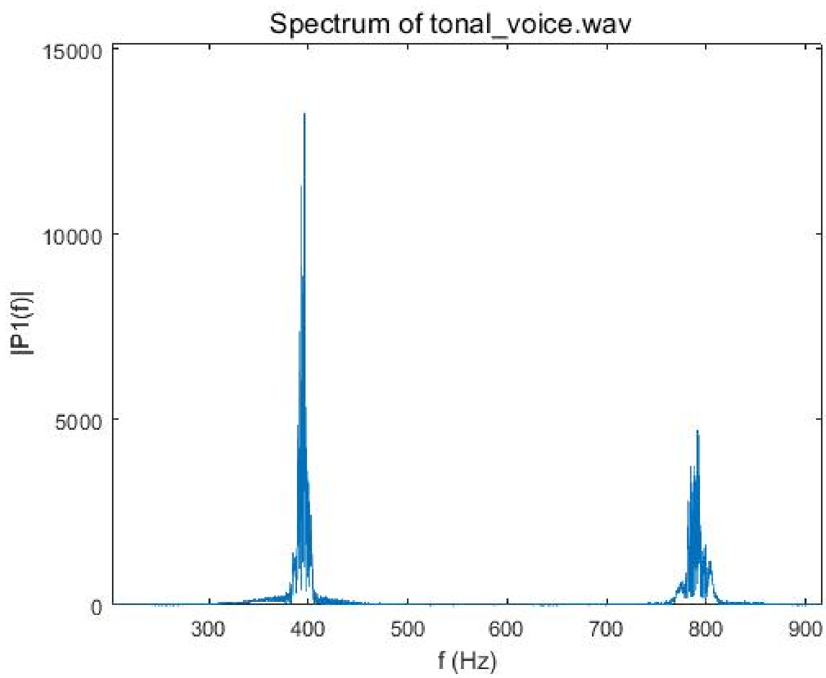

Figure 3: Spectrum of the tonal voice of 400Hz, with a harmonic at around 800Hz.

Part C applied spectrum analysis to a .wav file containing keyboard sounds. The keystrokes are impulsive sounds at an identical frequency, with some slight background noise. Notice that there are sharp spikes below 1 KHz, indicating the keystrokes. There is also some noticeable noise up to about 7.5KHz, which could just be the ambient noise from the room itself.

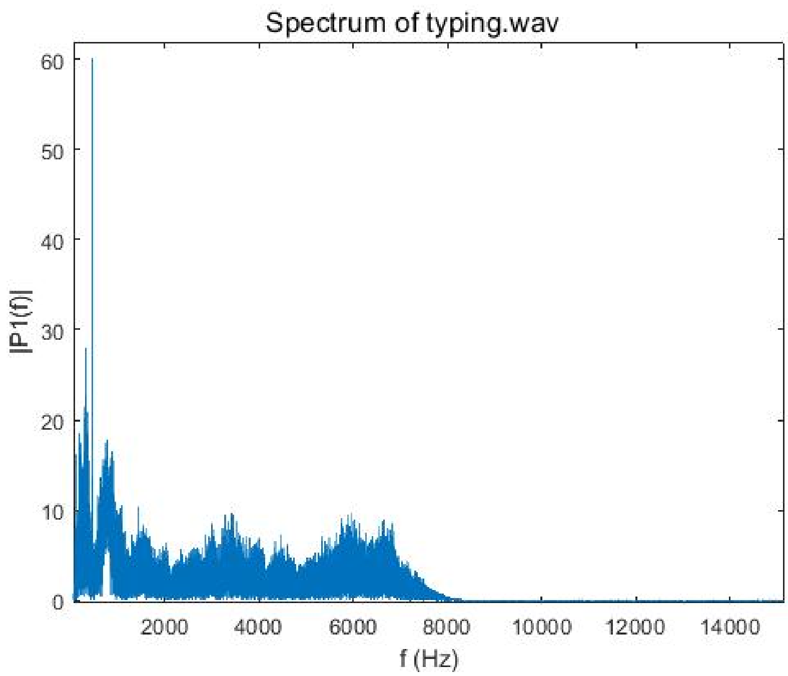

Figure 4: Audio sample spectrum of a keyboard with background noise.

Matlab Code

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%
% Lab 1 - MATLAB Basics
%%%%%%%%%%%%%%%%%%%%%%%%%

%% Exercise 1 - Plotting Signals

Fs = 10;              % Sampling frequency
T = 1/Fs;               % Sampling period

t = (-5:T:5); % time interval -5 sec to 5 sec


% 1 Hz sine wave
f = 1; % 1 hz frequency
signal_1 = sin(2*pi*f*t);

% unit rectangle function (see pg 99 in the book)
signal_2 = (sign(t+0.5)-sign(t-0.5) > 0);

% unit triangle function (see pg 100 in the book)
signal_3 = (1-2*abs(t)).*(t>=-1/2).*(t<1/2);

% Plot all the signals on subplots so you see them in 1
    figure
figure(1);
subplot(3,3,1); % total rows, total columns, nth plot
plot(t,signal_1); % time first for x axis, signal for y
    axis
title('1 Hz Sine Wave');
xlabel('Time in Seconds');
ylabel('Amplitude');
ylim([-1.5,1.5]);

subplot(3,3,2);
plot(t,signal_2);
title('Unit Rect Function');
xlabel('Time in Seconds');
ylabel('Amplitude');
ylim([-0.5,1.5]);

subplot(3,3,3);
plot(t,signal_3);
title('Unit Triangle Function');
```

```
xlabel('Time in Seconds');
ylabel('Amplitude');
ylim([-0.5,1.5]);
%% Exercise 2 - Signal Operations

% Plot the signals above with a time shift of 0.5 seconds delay ( just
% subtract 0.5 from t)
signal_1_shifted = sin(2*pi*f*(t-0.5));
signal_2_shifted = (sign((t-0.5)+0.5)-sign((t-0.5)-0.5) > 0);
signal_3_shifted = (1-2*abs(t-0.5)).*((t-0.5)>=-1/2).*((t-0.5)<1/2);

% plot the results on the same subplot
subplot(3,3,4);
plot(t,signal_1_shifted)
xlabel('Time in Seconds');
ylabel('Amplitude');
ylim([-1.5,1.5]);

subplot(3,3,5);
plot(t,signal_2_shifted)
title('Time Shifted 0.5s (Sine, Rect, and Triangle)');
xlabel('Time in Seconds');
ylabel('Amplitude');
ylim([-0.5,1.5]);

subplot(3,3,6);
plot(t,signal_3_shifted)
xlabel('Time in Seconds');
ylabel('Amplitude');
ylim([-0.5,1.5]);

% Plot the signals above with a time scaling compression of 3 (multiply the
% time vector by 3 to compress)
signal_1_compress = sin(2*pi*f*(3*t));
signal_2_compress = (sign((3*t)+0.5)-sign((3*t)-0.5) > 0);
signal_3_compress = (1-2*abs(3*t)).*((3*t)>=-1/2).*((3*t)<1/2);

% plot the results on the same subplot
subplot(3,3,7);
plot(t,signal_1_compress)
```

```matlab
xlabel('Time in Seconds');
ylabel('Amplitude');
ylim([-1.5,1.5]);

subplot(3,3,8);
plot(t,signal_2_compress)
title('Time Compressed Signals x3 (Sine, Rect, and Triangle)');
xlabel('Time in Seconds');
ylabel('Amplitude');
ylim([-0.5,1.5]);

subplot(3,3,9);
plot(t,signal_3_compress)
xlabel('Time in Seconds');
ylabel('Amplitude');
ylim([-0.5,1.5]);

sgtitle('Signals and stuff') % put a title for the entire subplot

%% Exercise 3 - Signal Spectra

% Part A - Plot a spectra of the signals above (see fft documentation for an
% example)

% -- 1 Hz sine wave --
% Note: fftshift makes the spectrum more readable by centering the data to
% zero.
signal_1_spectrum = fft(signal_1);
signal_1_spectrum = fftshift(signal_1_spectrum);
signal_1_shifted_spectrum = fft(signal_1_shifted);
signal_1_shifted_spectrum = fftshift(signal_1_shifted_spectrum);
signal_1_compress_spectrum = fft(signal_1_compress);
signal_1_compress_spectrum = fftshift(signal_1_compress_spectrum);

% These parameters scale the spectrum to Hz (applies to all 3 signals)
L = length(signal_1_spectrum)-1;
f = Fs*((-L/2):(L/2))/L; % multiplying by Fs converts from a unit spectrum

```

```
% Plot the original signal
figure(2)
subplot(3,3,1);
plot(f,abs(signal_1_spectrum));
xlabel('f (Hz)')
ylabel('|P1(f)|')


% A time shift of t_0 causes a linear phase shift, but
    doesn't effect freq
subplot(3,3,2);
plot(f,abs(signal_1_shifted_spectrum));
xlabel('f (Hz)')
ylabel('|P1(f)|')
title('Double-Sided Amplitude Spectrum of 1 Hz Wave')

% compressing in time stretches out in the frequency, and
     vice versa
subplot(3,3,3);
plot(f,abs(signal_1_compress_spectrum));
xlabel('f (Hz)')
ylabel('|P1(f)|')

% -- Unit Rectangle Wave --
signal_2_spectrum = fft(signal_2);
signal_2_spectrum = fftshift(signal_2_spectrum);
signal_2_shifted_spectrum = fft(signal_2_shifted);
signal_2_shifted_spectrum = fftshift(
    signal_2_shifted_spectrum);
signal_2_compress_spectrum = fft(signal_2_compress);
signal_2_compress_spectrum = fftshift(
    signal_2_compress_spectrum);

% Plot the original signal
subplot(3,3,4);
plot(f,abs(signal_2_spectrum));
xlabel('f (Hz)')
ylabel('|P1(f)|')

% A time shift of t_0 causes a linear phase shift, but
    doesn't effect freq
subplot(3,3,5);
plot(f,abs(signal_2_shifted_spectrum));
title('Double-Sided Amplitude Spectrum of Rect Wave')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

```
% compressing in time stretches out in the frequency, and
    vice versa
subplot(3,3,6);
plot(f,abs(signal_2_compress_spectrum));
xlabel('f (Hz)')
ylabel('|P1(f)|')

% -- Unit Triangle Wave --
signal_3_spectrum = fft(signal_3);
signal_3_spectrum = fftshift(signal_3_spectrum);
signal_3_shifted_spectrum = fft(signal_3_shifted);
signal_3_shifted_spectrum = fftshift(
    signal_3_shifted_spectrum);
signal_3_compress_spectrum = fft(signal_3_compress);
signal_3_compress_spectrum = fftshift(
    signal_3_compress_spectrum);

% Plot the original signal
subplot(3,3,7);
plot(f,abs(signal_3_spectrum));
xlabel('f (Hz)')
ylabel('|P1(f)|')

% A time shift of t_0 causes a linear phase shift, but
    doesn't effect freq
subplot(3,3,8);
plot(f,abs(signal_3_shifted_spectrum));
title('Double-Sided Amplitude Spectrum of Triangle Wave')
xlabel('f (Hz)')
ylabel('|P1(f)|')

% compressing in time stretches out in the frequency, and
    vice versa
subplot(3,3,9);
plot(f,abs(signal_3_compress_spectrum));
xlabel('f (Hz)')
ylabel('|P1(f)|')


sgtitle('Double-Sided Amplitude Spectrum of Initial
    Signals')

%% Part b - Spectrum of tonal voice

```

```
[y,Fs] = audioread('tonal_voice.wav');

% note that y has a left and right channel
audio_spectrum = fft(y(:,1));

audio_spectrum = fftshift(audio_spectrum);
L = length(audio_spectrum)-1;
f = Fs*((-L/2):(L/2))/L;
figure(3)
plot(f,abs(audio_spectrum));
sgtitle('Spectrum of tonal\_voice.wav')
xlabel('f (Hz)')
ylabel('|P1(f)|')
%% Part c - Spectrum of another wav file

[y,Fs] = audioread('typing.wav');

% note that y has a left and right channel
audio_spectrum = fft(y(:,1));

audio_spectrum = fftshift(audio_spectrum);
L = length(audio_spectrum)-1;
f = Fs*((-L/2):(L/2))/L;
figure(4)
plot(f,abs(audio_spectrum));
sgtitle('Spectrum of typing.wav')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```