Nicholas DiGregorio, 1220871392

Introduction

This is the first project for the dsp course. This project is a how to guide that helps the user generate some basic code for signal processing purposes. This is useful project to get familiar with matlab and to get in the coding mindset.

Ex 1-1

I got matlab 2024 running on my computer. I am able to run the help command from the command window to see how different functions work. Not much to show here.

Ex 1-2

Below is what happens when you run the version command and the computer command. It shows the version of matlab you are using and the type of computer you are using 32vs64 bit.

```
>> version

ans =

    '24.1.0.2578822 (R2024a) Update 2'
```

```
>> computer

ans =

    'PCWIN64'
```

Ex 1-3

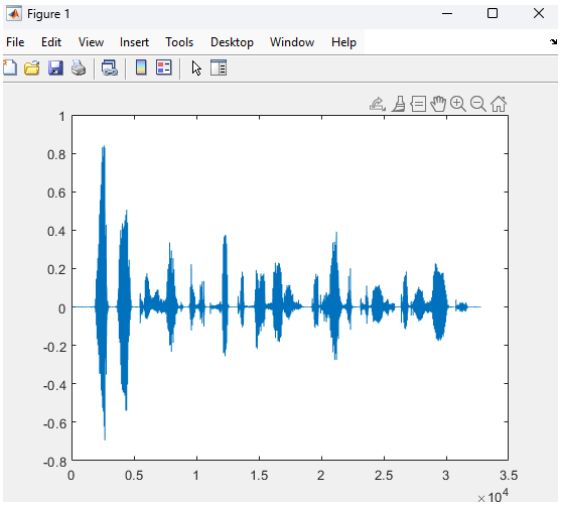
When running the command: ex13('cleanspeech.wav', 0) all the values from the return vector. This is because the semi colon terminates the print. If you want to print something to the command prompt all you need to do is leave the semicolon off.

Now when running the command: [s, fs, bits] = ex13('cleanspeech.wav', 0);. The script runs but nothing is printed to the screen and no sound plays still because the value for the play state we are passing in is still 0.

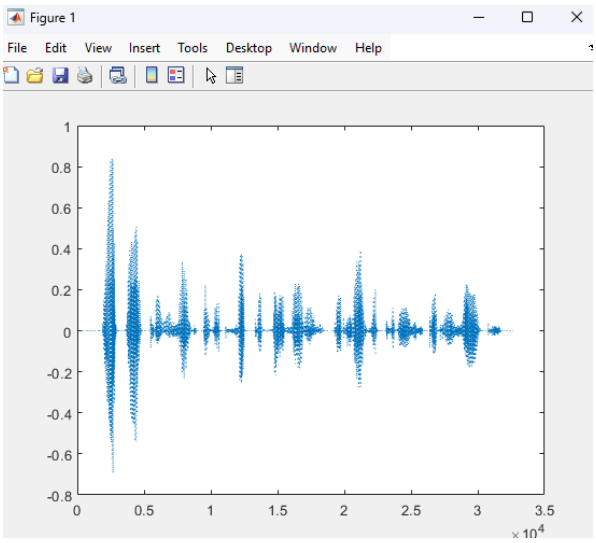
The file has 16 bits per sample and a sample rate of 8000 Hz. As shown from the screen shot below.

```
bits =

    16

>> fs

fs =

        8000
```

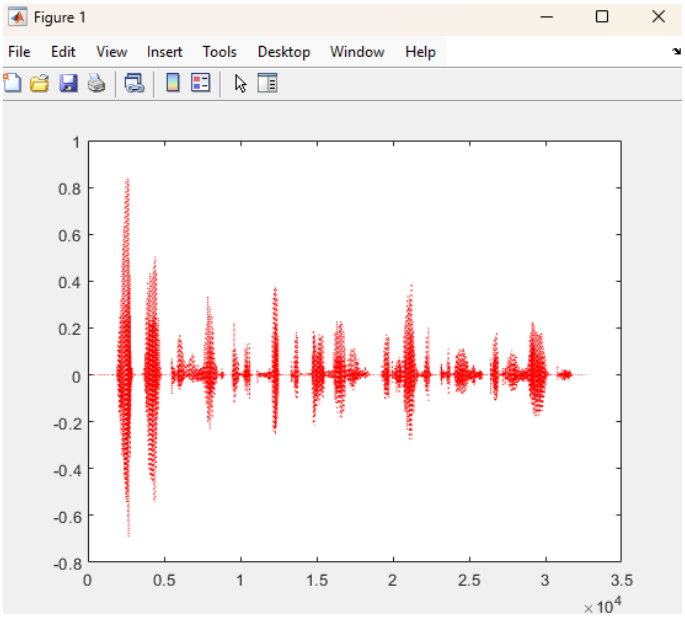
Running the plot(s) function plots the sound wave as shown below.



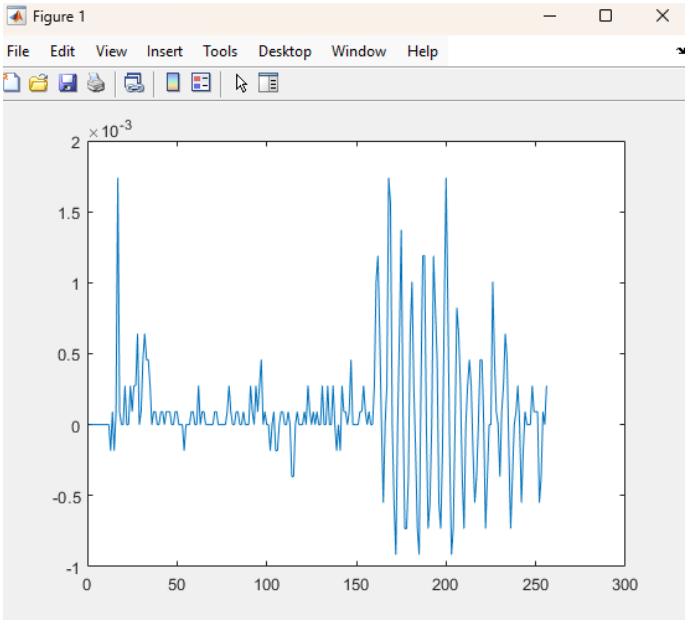
Adding the ':' making the command plot(s, ':') changes the line to a dotted line as seen below



Adding the r, making the command plot(s, ‘r:’) changes the line to dotted and the color of the line to red as shown below.

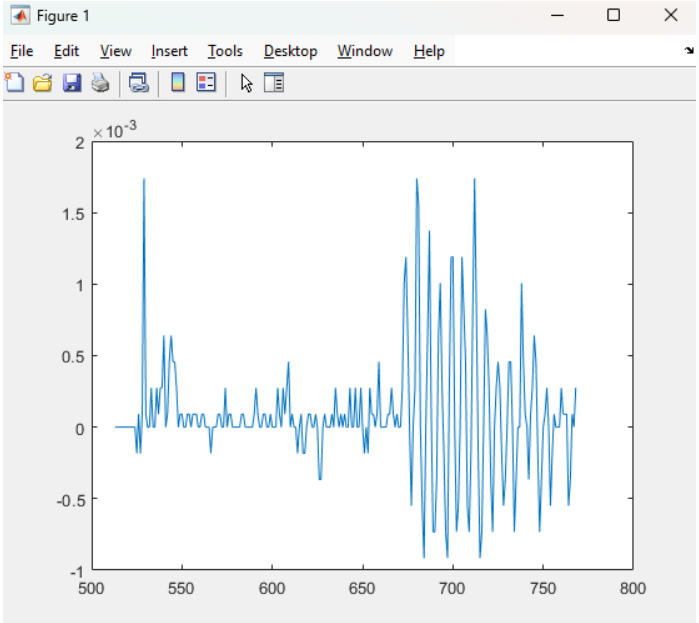


Running the plot(s(1:256)) command causes matlab to only plot 256 samples as shown below.

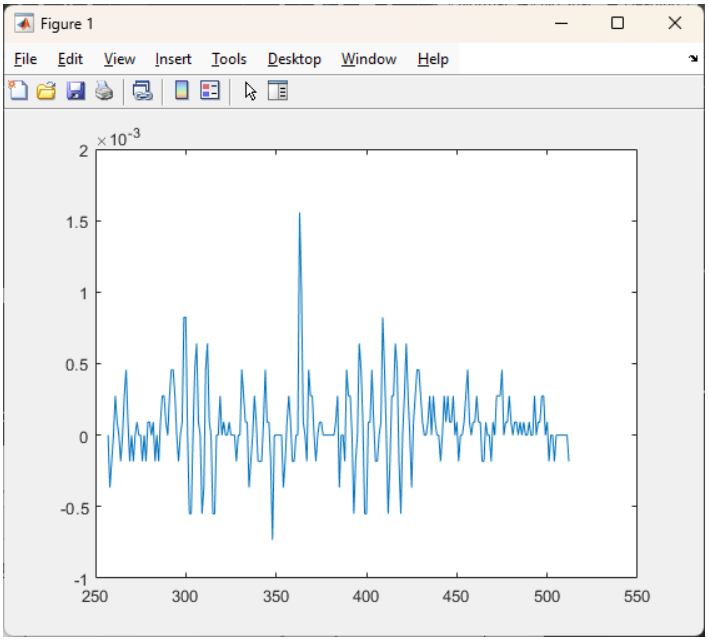


Using j=1:256; followed with a plot(s(j)); produces the same exact plot as prior because we are essentially abstracting the vector into a variable.

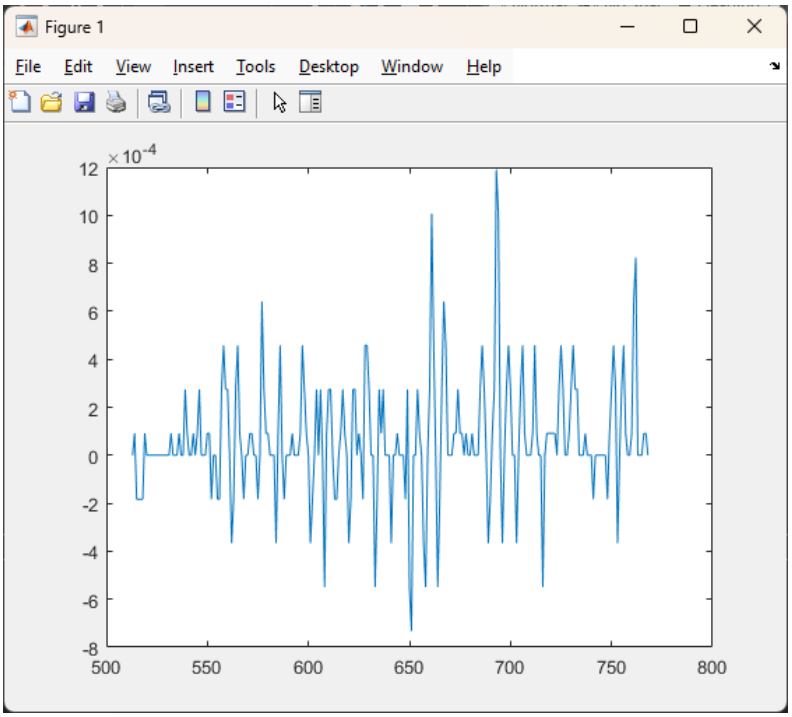
Running the plot(j+512,s(j)); command now shifts the x indices to the right as shown below. This happens because we are adding 512 to each element in the j vector.



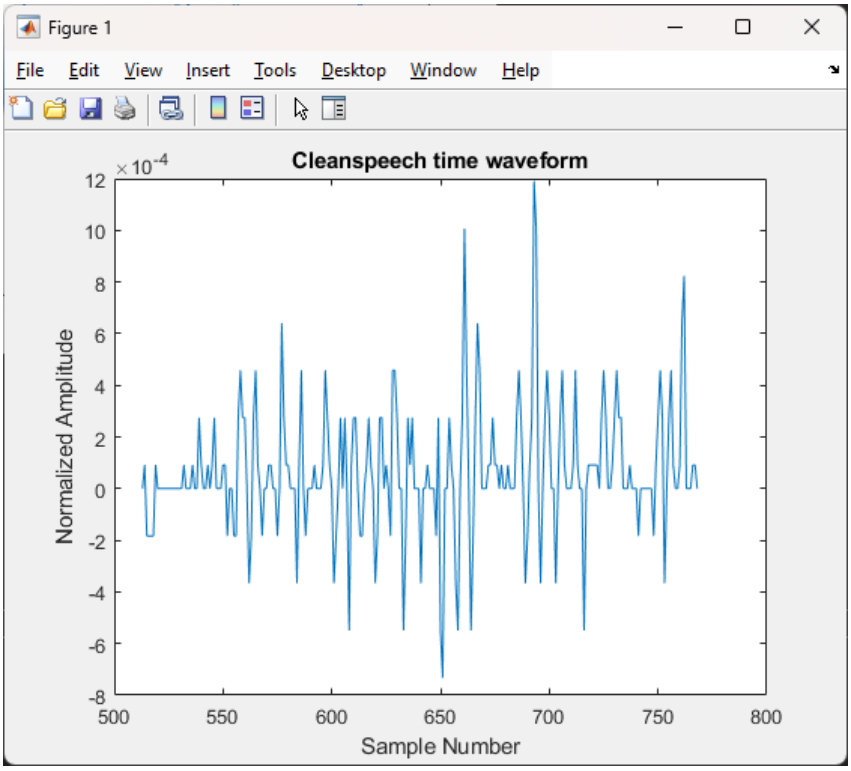
This time when we add N to the j vector and plot we are now plotting the next frame of the wave. We can use this to iterate through the data now.



Now running those commands again we get the next frame of data. We could throw this into a loop and plot our data frame by frame.



Now we added a title, xlabel, and a ylabel to the plot to make our plot more descritptive



Nothing happens when I run this command: [s,fs,bits]=ex13('cleanspeech',0); But when I run this command [s,fs,bits]=ex13('cleanspeech',1); I hear the words: "That doctrine has been accepted by many but has it produced good results?". This is because we set the playstate to make the conditional run.

When I run the same command but with the noisyspeech.wav file, You can vaguely hear the guy talking but you wouldn't really be ablet to tell what he said.

Ex 1-4

Per the assignment I added these lines of code:

```
25    l1 = length(s1);
26    l2 = length(s2);
27    M  = min(l1, l2);
28    K  = fix(M/N);
```

Here M is assigned the value of the length of the shorter file. Now dividing by N and assigning to K means that K is the number of frames in the audio files. The fix command will delete any remainder in the division, or in other words its rounding to the nearest integer but always towards zero. This implies that if there is a remainder of samples, that don't fit into a frame, we will throw them away.

The next thing we do is setting up a for loop that plots each frame and then iterates to the next frame and plots it until we have plotted all the frames.

To characterize the 2 different files I would say the clean speech file has a lot more zero in it than the noise speech file. It only has an amplitude when the person is actually talking. When doing a quick calculation of the mean and standard deviation we can see that the mean of the cleanspeech file is on much closer to and it deviates from 0 much less then the noisy speech file. That makes sense because the noise is constant throughout the noisyspeech file. Values shown below:

```
mean_s1 =

  -3.3162e-05


std_s1 =

    0.0484


mean_s2 =

   -0.0016


std_s2 =

    0.1607
```

Ex 1-5

Here I learned that Matlab is optimized for vector math. So rather than doing a multiply and an accumulate like some lower level languages one can take the transpose of the vector multiplied by the vector itself to achieve a lot of efficiency.

```
e1 = (s1'*s1)/N;
e2 = (s2'*s2)/N;
```

I added these lines of code to add these values to the plot.

```
msg = sprintf('Average power %d', e1);
text(n(1)+500.0, 0.4, msg);
```

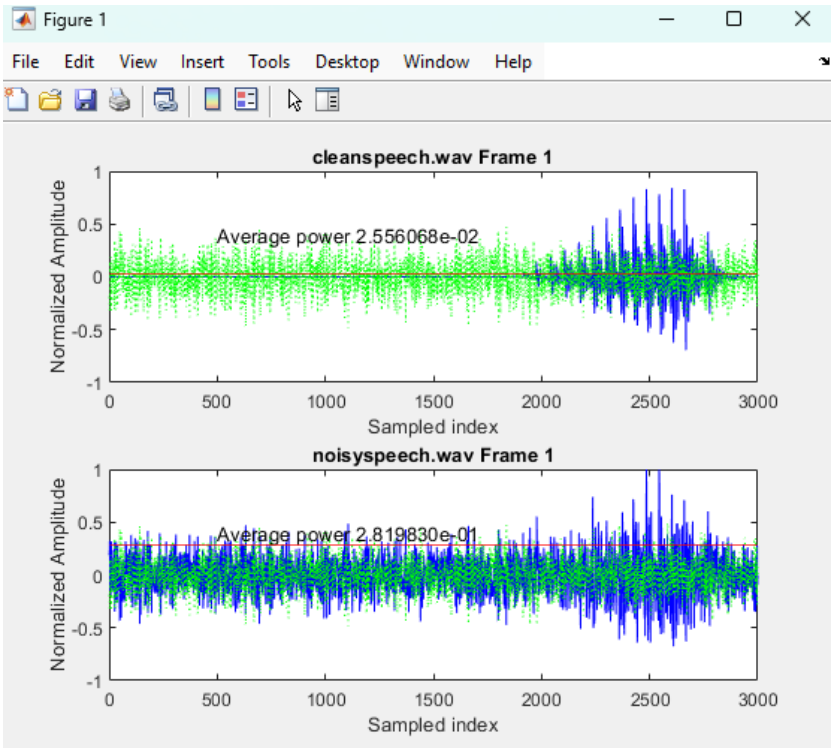
```
msg = sprintf('Average power %d', e2);
text(n(1)+500.0, 0.4, msg);
```

I also added a couple lines of code to show the value on the chart as a line in red.

```
ax = gca;
plot(ax, n, s1(n), 'b', n, e(n), 'g:');
yline(ax, e1,'r-')
```

```
ax = gca;
plot(ax, n, s2(n), 'b', n, e(n), 'g:');
yline(ax, e2, 'r-')
```

You can see the average power displayed in the window now as well as visually with the line. This shows that a noisy speech file consumes more power then the clean speech file and that makes a lot of sense when you see how much more active that signal is in comparison.



Below is my attempt at playing around with vector math. I tried to see if there was a difference but it wasn't very noticeable. I think maybe with when this document created compared to the power of my gaming PC, it just doesn't make that much of a difference in todays age. But I will definitely take the hint and try to do vector math when possible.

```
%%%%%%%%% This file is just to experirment with multiplying
N = 100000000;
v = randn(N, 1);

result1 = v'*v

mult = zeros(N, 1);
for i = 1:N
    mult(i) = v(i)^2;
end
result2 = sum(mult)
```

Ex 1-6

I already had the numerator calculated and the difference of the signal to noise calculated so all I had to do was calculate the square of the difference divided by the number of samples and to take the log10 of the numerator/denominator and multiple by 10.

```
e  = s1 - s2;
e1 = (s1'*s1)/N;
denom = (e'*e)/N;
e2 = (s2'*s2)/N;

SNR = 10*log10(e1/denom)
```

```
SNR =

  -10.0367
```

Ex 1-7

I started this exercise by calulating the difference and the summation of the clean and noisy speech. If you think about it intuitively the difference between them is going to take out the person talking and all that will be left is just the noise. If we add them together the person talking is going to sound louder then just the noisyspeech file where you can barely hear him.

```
diff  = s1 - s2;
f1_name = 'grabled1.wav';
audiowrite(f1_name, diff, fs1)
sum   = s1 + s2;
f2_name = 'grabled2.wav';
audiowrite(f2_name, sum, fs2)
```

Below is the code that I used to playback those signals. Indeed, we hear the only noise in the difference sound and we hear both the noise and the person talking but the person talking is amplified.

```
[s3,fs3]=audioread(f1_name); %difference signal
sound(s3, fs3)
%Pause between sound bytes, waiting for keypress
pause

[s4,fs4]=audioread(f2_name); %summation signal
sound(s4, fs4)
```

Conclusion

In conclusion, got much more familiar with Matlab and the functions and tools I expect to use in this class. Coming off of a python class this past spring, Matlab is easy to pick back up. There is a lot of crossover between the 2 languages but most things are a little easier in matlab. Things like plotting data, printing to the command prompt, and element by element processing are just a little easier in matlab. I look forward to getting to use this tool further in this class. Below is my code that I used to complete the exercises. The name of the function corresponds to the exercise and is separated by page breaks.

```
function [s,fs, bits] = ex13(infile, playstate)
%infile - .wavinput file
% playstate – Switch playback on/off
%
% s – signal loaded from infile
% fs – sample rate
% bits – bits per sample
%
% Function loads infile, displays entire
% record, then optionally plays back the
% sound depending upon state of playstate


%%Load Infile
[s,fs]=audioread(infile);
info = audioinfo(infile);
bits = info.BitsPerSample;

%%Display infile
plot(s, 'r:')

%%Playback infile if playstate is set
if playstate == 1
    sound(s, fs);
end
```

```
function [s,fs,bits] = ex14(infile1, infile2, N)

% [s,fs,bits]=ex14(infile1,infile2)
%
% infile1, infile2 - .WAV input files
% N – frame size (in samples)
%
% s – signals loaded from infile1 and infile2
% fs – sample rates
% bits – bits per sample in each file
%
% Function loads infile1 and infile2, then displays
% records frame-by-frame.

[s1,fs1]=audioread(infile1);
info1 = audioinfo(infile1);
bits1 = info1.BitsPerSample;

[s2,fs2]=audioread(infile2);
info2 = audioinfo(infile2);
bits2 = info2.BitsPerSample;

l1 = length(s1);
l2 = length(s2);
M  = min(l1, l2);
K  = fix(M/N);
e  = s1 - s2;
mean_s1 = mean(s1)
std_s1 = std(s1)
mean_s2 = mean(s2)
std_s2 = std(s2)

for k = 1:K
        %Compute indices for current frame
        n = (1:N)+(N*(k-1));

        %signal 1
        subplot(211);
        plot(n, s1(n), 'b', n, e(n), 'g:');
        msg = sprintf('%s Frame %d', infile1, k);
        title(msg);
        ylabel ('Normalized Amplitude');
        xlabel ('Sampled index');

        %signal 2
        subplot(212);
        plot(n, s2(n), 'b', n, e(n), 'g:');
        msg = sprintf('%s Frame %d', infile2, k);
        title(msg);
        ylabel ('Normalized Amplitude');
        xlabel ('Sampled index');

        %Pause between frames, waiting for keypress
        pause
```

```
end

s = [s1, s2];
fs = [fs1, fs2];
bits = [bits1, bits2];
```

```
function [s, fs, bits] = ex15(infile1, infile2, N)

% [s,fs,bits]=ex14(infile1,infile2)
%
% infile1, infile2 - .WAV input files
% N – frame size (in samples)
%
% s – signals loaded from infile1 and infile2
% fs – sample rates
% bits – bits per sample in each file
%
% Function loads infile1 and infile2, then displays
% records frame-by-frame. Computes average energy
% per sample in each file.

%%
%%Load in the files

[s1,fs1]=audioread(infile1);
info1 = audioinfo(infile1);
bits1 = info1.BitsPerSample;

[s2,fs2]=audioread(infile2);
info2 = audioinfo(infile2);
bits2 = info2.BitsPerSample;

l1 = length(s1);
l2 = length(s2);
M  = min(l1, l2);
K  = fix(M/N);
e  = s1 - s2;
e1 = (s1'*s1)/N;
e2 = (s2'*s2)/N;

for k = 1:K
        %Compute indices for current frame
        n = (1:N)+(N*(k-1));

        %signal 1
        subplot(211);
        ax = gca;
        plot(ax, n, s1(n), 'b', n, e(n), 'g:');
        yline(ax, e1,'r-')
        msg = sprintf('%s Frame %d', infile1, k);
        title(msg);
        msg = sprintf('Average power %d', e1);
        text(n(1)+500.0, 0.4, msg);
        ylabel ('Normalized Amplitude');
        xlabel ('Sampled index');

        %signal 2
        subplot(212);
        ax = gca;
        plot(ax, n, s2(n), 'b', n, e(n), 'g:');
        yline(ax, e2, 'r-')
```

```
        msg = sprintf('%s Frame %d', infile2, k);
        title(msg);
        msg = sprintf('Average power %d', e2);
        text(n(1)+500.0, 0.4, msg);
        ylabel ('Normalized Amplitude');
        xlabel ('Sampled index');

        %Pause between frames, waiting for keypress
        pause

end

s = [s1, s2];
fs = [fs1, fs2];
bits = [bits1, bits2];




%%%%%%%%% This code is just to experiment with multiplying vectors vs scalar
N = 100000000;
v = randn(N, 1);

result1 = v'*v

mult = zeros(N, 1);
for i = 1:N
    mult(i) = v(i)^2;
end
result2 = sum(mult)
```

```
function [s, fs, bits] = ex16(infile1, infile2, N)

% [s,fs,bits]=ex14(infile1,infile2)
%
% infile1, infile2 - .WAV input files
% N – frame size (in samples)
%
% s – signals loaded from infile1 and infile2
% fs – sample rates
% bits – bits per sample in each file
%
% Function loads infile1 and infile2, then displays
% records frame-by-frame. Computes average energy
% per sample in each file.

%%
%%Load in the files

[s1,fs1]=audioread(infile1);
info1 = audioinfo(infile1);
bits1 = info1.BitsPerSample;

[s2,fs2]=audioread(infile2);
info2 = audioinfo(infile2);
bits2 = info2.BitsPerSample;

l1 = length(s1);
l2 = length(s2);
M  = min(l1, l2);
K  = fix(M/N);
e  = s1 - s2;
e1 = (s1'*s1)/N;
denom = (e'*e)/N;
e2 = (s2'*s2)/N;

SNR = 10*log10(e1/denom)

for k = 1:K
        %Compute indices for current frame
        n = (1:N)+(N*(k-1));

        %signal 1
        subplot(211);
        ax = gca;
        plot(ax, n, s1(n), 'b', n, e(n), 'g:');
        yline(ax, e1,'r-')
        msg = sprintf('%s Frame %d', infile1, k);
        title(msg);
        msg = sprintf('Average power %d', e1);
        text(n(1)+500.0, 0.4, msg);
        ylabel ('Normalized Amplitude');
        xlabel ('Sampled index');

        %signal 2
        subplot(212);
```

```
        ax = gca;
        plot(ax, n, s2(n), 'b', n, e(n), 'g:');
        yline(ax, e2, 'r-')
        msg = sprintf('%s Frame %d', infile2, k);
        title(msg);
        msg = sprintf('Average power %d', e2);
        text(n(1)+500.0, 0.4, msg);
        ylabel ('Normalized Amplitude');
        xlabel ('Sampled index');

        %Pause between frames, waiting for keypress
        pause

end

s = [s1, s2];
fs = [fs1, fs2];
bits = [bits1, bits2];
```

```
function [s, fs, bits] = ex17(infile1, infile2)


% [s,fs,bits]=ex14(infile1,infile2)
%
% infile1, infile2 - .WAV input files
%
% s – signals loaded from infile1 and infile2
% fs – sample rates
% bits – bits per sample in each file
%
% Function loads infile1 and infile2, then
% computes the difference between the signals
% and saves as .wav, then it computes the
% summation and saves that as a .wav and then
% plays both difference signals and summation
% signals back to the user.

%%
%%Load in the files

[s1,fs1]=audioread(infile1);
info1 = audioinfo(infile1);
bits1 = info1.BitsPerSample;

[s2,fs2]=audioread(infile2);
info2 = audioinfo(infile2);
bits2 = info2.BitsPerSample;

diff  = s1 - s2;
f1_name = 'grabled1.wav';
audiowrite(f1_name, diff, fs1)
sum   = s1 + s2;
f2_name = 'grabled2.wav';
audiowrite(f2_name, sum, fs2)

[s3,fs3]=audioread(f1_name); %difference signal
sound(s3, fs3)
%Pause between sound bytes, waiting for keypress
pause

[s4,fs4]=audioread(f2_name); %summation signal
sound(s4, fs4)
s = [s1, s2];
fs = [fs1, fs2];
bits = [bits1, bits2];
```