Louis Rosenblum

EEE-509

05/24/2020

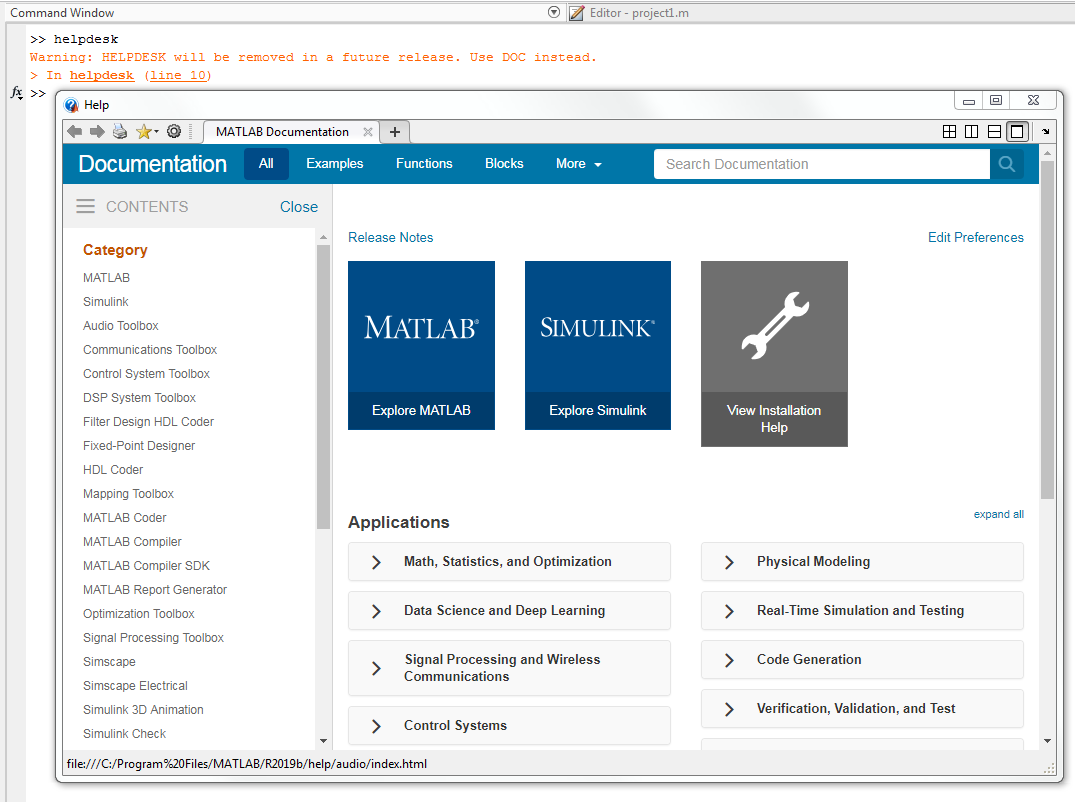
**MATLAB Project 1 Report**

Introduction:

This project served as an intro to the MATLAB command line and script editor. It introduced plotting, vectorization, and reading/writing signals from/to audio files. Having used MATLAB in several of my undergraduate courses these exercises served mostly as a review.

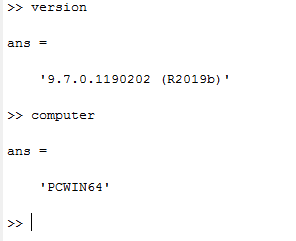
Report:

Section 1-1)



Section 1-2)

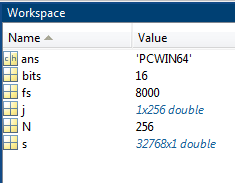
The ‘version’ command reports the version of MATLAB that is currently running. The ‘computer’ command indicates what type of system MATLAB is currently running on.



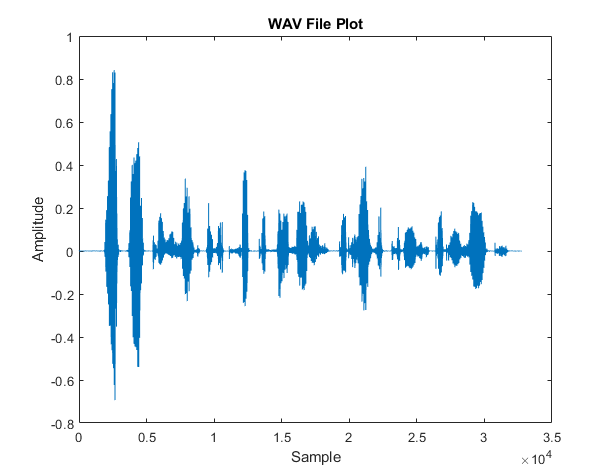
Section 1-3)

d) Running the ex13 function loaded a .wav file from the hard disk and created a corresponding vector in MATLAB. This vector was then played aloud using the soundsc( ) function.

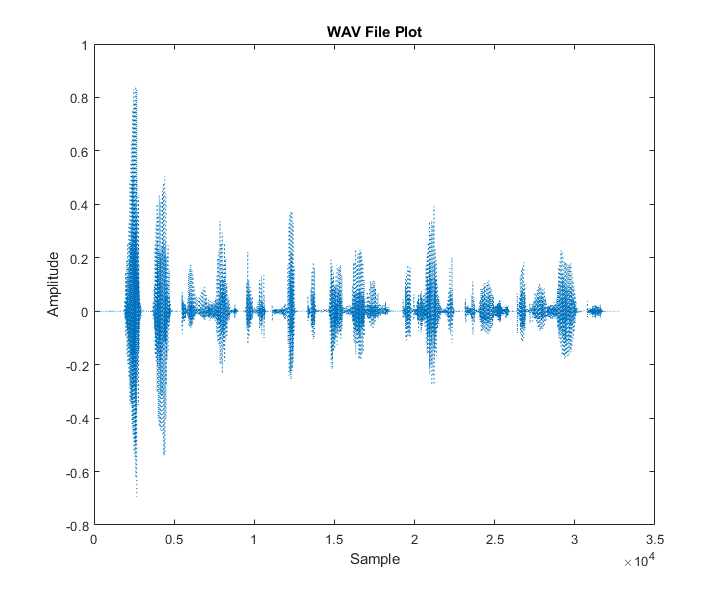
e) The audio file ‘cleanspeech.wav’ has 16 bits per sample and a sampling rate of 8 kHz.



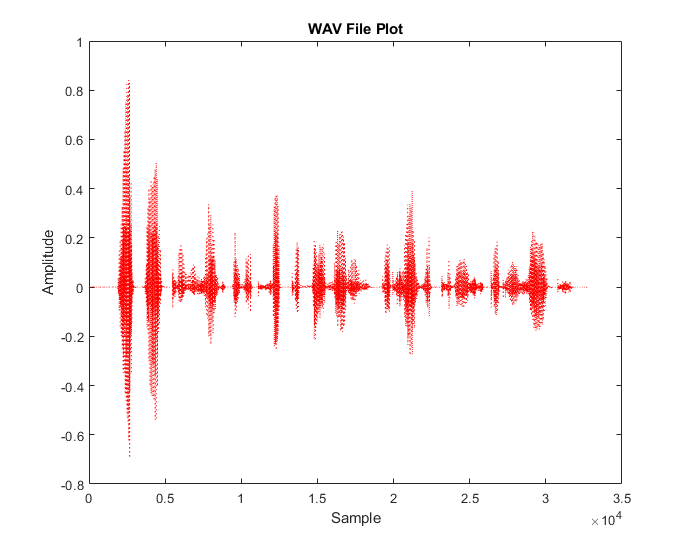
f) Running the plot command produces an amplitude versus sample plot of the signal.



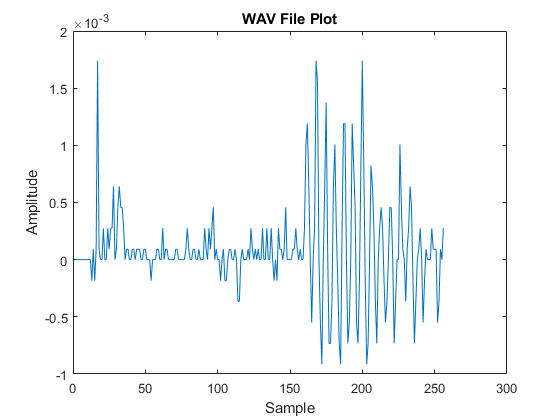
Running the plot function with an additional argument ‘:’ produced a discrete plot where the individual data points are not automatically connected by a continuous curve.

****

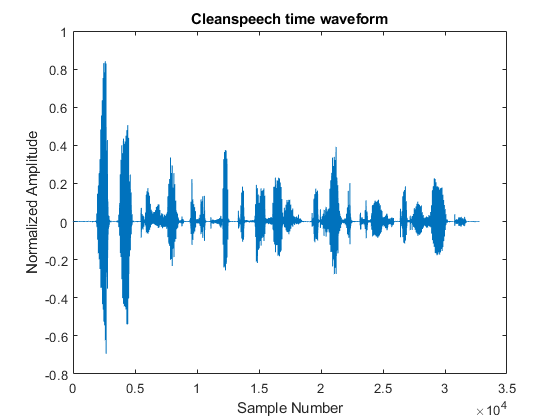
Running the plot command again with an ‘r’ argument plots the signal using the color red.

****

g) Running the plot command again and indexing the signal from (1:256) produced a plot of the first 256 data points of the signal.

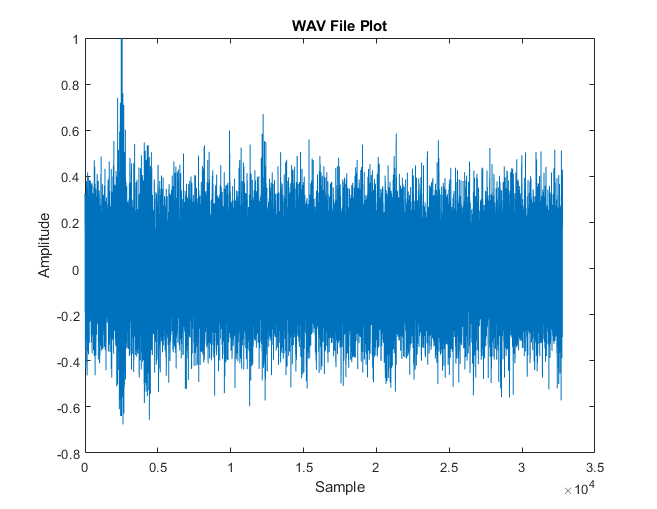


Adding axis labels and a chart title created the following plot.



h) Running the ex13 command with a ‘0’ in the boolean argument does not produce a sound. However running it with a ‘1’ in the argument will trigger the soundsc( ) function that is part of the logical statement.

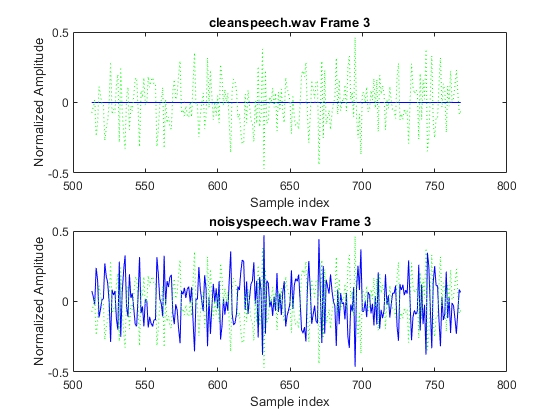
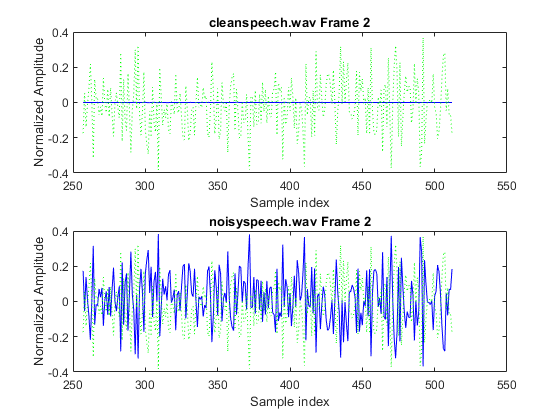
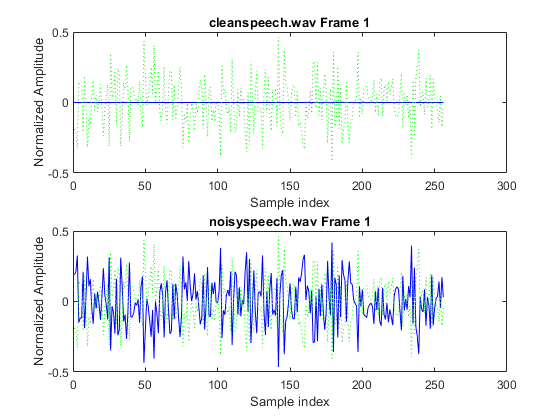
Executing the function again with the noise speech file instead produces this plot and a much more distorted audio playback.



Section 1-4)

In the function ex14 K represents the frame number, a certain set of samples that together make up the entire signal. The ‘fix’ command is rounding an input value towards zero.

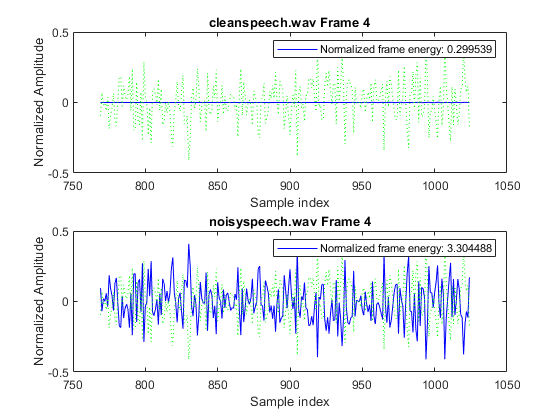
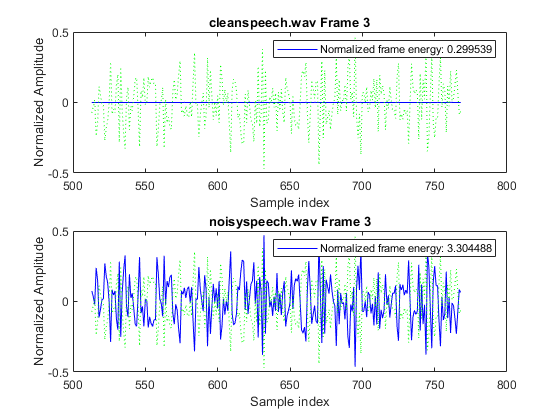
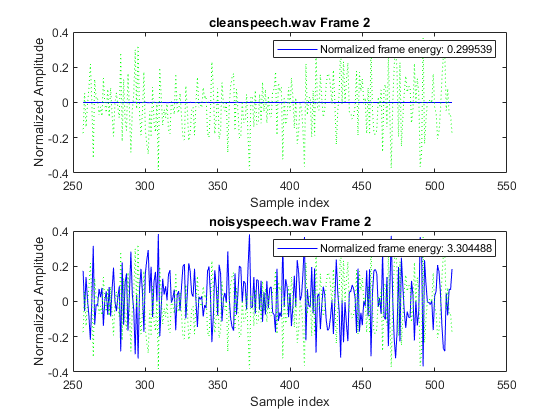
The first three frames of the ex14 function output for signals ‘cleanspeech.wav’ and ‘noisyspeech.wav’ look as follows.



In the clean speech file there is much less noise included with the signal. This can be seen in the blue curve representing the normalized amplitude of the signal itself. The difference between the two signals is far greater than the amplitude of the clean speech signal, meaning the noisy speech file includes a significant amount of distortion.

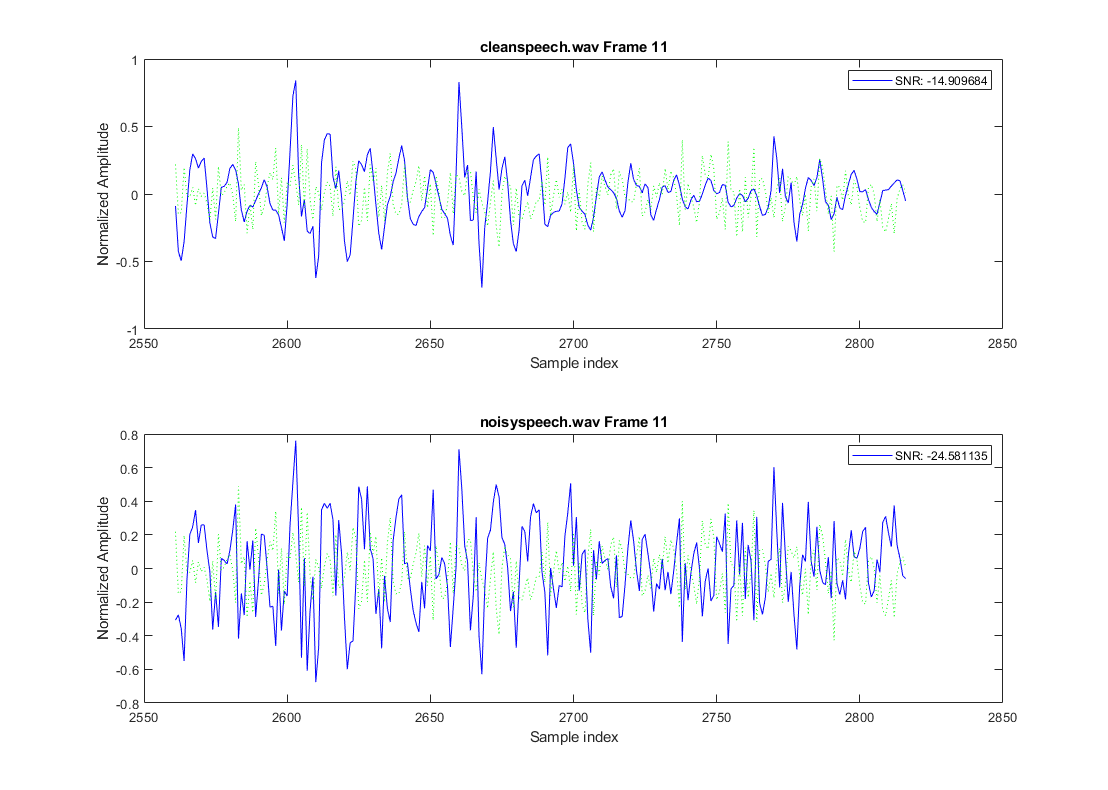
Section 1-5)

Show below are frames 2-4 of the output of function ex15, displaying the average energy per sample in the legend.



Section 1-6)

This section involved computing the signal to noise ratio of each signal and displaying them alongside the plot. The global SNR for the noisy speech audio file was -24.58.

****

Section 1-7)

The summation signal sounds very close to the original ‘cleanspeech.wav’ audio file. However the difference signal sounds extremely distorted and heavily impacted by noise, much like the ‘noisyspeech.wav’ file.

Conclusion:

MATLAB is a powerful tool for many purposes, including digital signal processing. The vectorization, plotting, playback, and I/O features make it extremely useful both for manipulating and presenting data. I look forward to using it this term and throughout my career.

**Appendix**



|  |
| --- |
| MATLAB Code:  % Louis Rosenblum % EEE 509 - ASU % 5/21/2020  %% Initialization clear all close all  cd 'C:\Users\Louis\Desktop\DSP\Project 1' %% Section 1-1: Help help  %% Section 1-2: Info version computer %% Section 1-3: Load, Display, and Playback Speech Files  % (See function definitions at bottom of script)  [s,fs,bits] = ex13("cleanspeech.wav",1);  figure() plot(s) title("WAV File Plot"); xlabel("Sample"); ylabel("Amplitude");  figure() plot(s,':') title("WAV File Plot"); xlabel("Sample"); ylabel("Amplitude");  figure() plot(s,'r:') title("WAV File Plot"); xlabel("Sample"); ylabel("Amplitude");  figure() plot(s(1:256)) title("WAV File Plot"); xlabel("Sample"); ylabel("Amplitude");  figure() j=1:256; plot(s(j)); title("WAV File Plot"); xlabel("Sample"); ylabel("Amplitude");  figure() plot(j+512,s(j)); title("WAV File Plot"); xlabel("Sample"); ylabel("Amplitude");  figure() N=256; j=j+N; plot(j,s(j)) title("WAV File Plot"); xlabel("Sample"); ylabel("Amplitude");  figure() plot(s) title('Cleanspeech time waveform'); xlabel('Sample Number'); ylabel('Normalized Amplitude');  % Test false case [s,fs,bits] = ex13("cleanspeech.wav",0);  [s,fs,bits] = ex13("noisyspeech.wav",1);  %% Section 1-4: Create m-file to Load and Display Two Speech Files Frame-by-Frame % (See function definitions at bottom of script)  close all  % Run function ex14 %[s,fs,bits] = ex14("cleanspeech.wav","noisyspeech.wav",256);  %% Section 1-5: Incorporate Frame Energy Computation % (See function definitions at bottom of script)  [s,fs,bits] = ex15("cleanspeech.wav","noisyspeech.wav",256);  %% Section 1-6: Incorporate SNR Calculation  [s,fs,bits] = ex16("cleanspeech.wav","noisyspeech.wav",256);  %% Section 1-7: Save and Playback Features  [s,fs,bits] = ex17("cleanspeech.wav","noisyspeech.wav");  [s,fs,bits] = ex17("garbled1.wav","garbled2.wav"); %% Function definitions **function** [s,fs,bits] = **ex17**( infile1, infile2) % [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); [s2,fs2] = audioread(infile2);  diff = s1 - s2; mag = s1 + s2;  audiowrite('C:\Users\Louis\Desktop\DSP\Project 1\garbled1.wav',diff,fs1); audiowrite('C:\Users\Louis\Desktop\DSP\Project 1\garbled2.wav',mag,fs1);  info1 = audioinfo(infile1); info2 = audioinfo(infile2); s = [s1 s2]; fs = [fs1 fs2]; bits = [info1.BitsPerSample info2.BitsPerSample];  soundsc(diff,fs1); soundsc(mag,fs1); **end**   **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.  [s1,fs1] = audioread(infile1); [s2,fs2] = audioread(infile2);  length1 = length(s1); length2 = length(s2); M = min(length1,length2); K = fix(M/N);  e = s1-s2;  i = 0:N-1;  **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('Sample index');    E1 = snr(s1);  legend( sprintf('SNR: %f', E1));  % 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('Sample index');    E2 = snr(s2);  legend( sprintf('SNR: %f', E2));    % Pause between frames, waiting for keypress  pause **end**  info1 = audioinfo(infile1); info2 = audioinfo(infile2);  s = [s1 s2]; fs = [fs1 fs2]; bits = [info1.BitsPerSample info2.BitsPerSample]; **end**  **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.  [s1,fs1] = audioread(infile1); [s2,fs2] = audioread(infile2);  length1 = length(s1); length2 = length(s2); M = min(length1,length2); K = fix(M/N);  e = s1-s2;  i = 0:N-1;  **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('Sample index');    E1 = 1/N \* sum((s1' \* s1));  legend( sprintf('Normalized frame energy: %f', E1));  % 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('Sample index');    E2 = 1/N \* sum((s2' \* s2));  legend( sprintf('Normalized frame energy: %f', E2));    % Pause between frames, waiting for keypress  pause **end**  info1 = audioinfo(infile1); info2 = audioinfo(infile2);  s = [s1 s2]; fs = [fs1 fs2]; bits = [info1.BitsPerSample info2.BitsPerSample]; **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); [s2,fs2] = audioread(infile2);  length1 = length(s1); length2 = length(s2); M = min(length1,length2); K = fix(M/N);  e = s1-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('Sample 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('Sample index');  % Pause between frames, waiting for keypress  pause **end**  info1 = audioinfo(infile1); info2 = audioinfo(infile2);  s = [s1 s2]; fs = [fs1 fs2]; bits = [info1.BitsPerSample info2.BitsPerSample]; **end**    **function** [s,fs,bits] = **ex13**(infile, playstate) % ex13(infile,playstate) % % infile - .WAV input 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  [y,Fs] = audioread(infile);  t = 0:1/Fs:(length(y)-1)/Fs;  plot(y) title("WAV File Plot"); xlabel("Sample"); ylabel("Amplitude");  **if**(playstate)  soundsc(y); **end**  info = audioinfo(infile);   s = y; fs = Fs; bits = info.BitsPerSample;  **end** |