MATLAB for DSP Laboratory Exercise 1

Getting Started in MATLAB

Andreas Spanias

Lab Exercise 1: Getting Started in MATLAB

Introduction

This lab introduces some fundamental concepts in MATLAB computing through a simple step-by-step programming example. The objectives are for the student to

- Gain some familiarity with the MATLAB environment and programming practices
- Learn how to create and debug an m-file
- Construct a working m-file (or set of m-files) which processes canned speech data

The exercise proceeds in a step-by-step fashion and assumes no prior working knowledge of MATLAB. The program created by the student during this exercise will load and display two speech files, frame-by-frame. In addition, the program will compute and display signal-to-noise ratios for the two files, as well as compute and display frame energies. The results of this exercise will be used in a later exercise on speech enhancement. A sample solution is provided at the end of the exercise.

Exercise 1-1: Run MATLAB and the Help System

a) Double click the MATLAB shortcut on the WIN95 desktop to enter the MATLAB environment.

b) Double click the NETSCAPE shortcut on the WIN95 desktop to start NETSCAPE as the help system host (as discussed during the lecture).

c) Click the question mark icon in the MATLAB command window to bring up the initial help window, then click the HELP DESK button in the help window to start the hypertext (.HTML) help system in Netscape. Alternatively, you may type “helpdesk” at the MATLAB command prompt to invoke the help desk in the Netscape command window. You now have the MATLAB hypertext help resource at your disposal in the familiar browser environment. Take a few minutes to browse some of the help windows and get a feel for how the system works.

d) NOTE: Throughout the remainder of these exercises, learn to rely upon the help system to supply details on how to use the various built-in MATLAB functions and programming constructs. The early exercises, below, will give many of the necessary programming details. Later, exercises, however, will expect you to figure out the implementation details on your own. The help system will serve as an invaluable resource. Even though the details are provided, you should still refer to the help entries for the commands described in exercises 1-3 through 1-7 to gain some familiarity with usage of the help system in an actual programming assignment.

Exercise 1-2: Start the MATLAB Editor

a) Go to the File menu, then select New | m-file to invoke the MATLAB m-file editor / debugger. Alternatively, you may type "edit" at the >> command prompt. You are now ready to begin work on an m-file. Throughout the remainder of this document, command prompt entries we expect you to type will be shown in the `courier` font with a leading `>>`, as follows:

```
>> version
```

Text we expect you to type in the editor / debugger will be shown without the leading prompt, as follows:

```
computer
```

What does the version command do? What does the computer command do?

Exercise 1-3: Create m-file to Load, Display, and Playback Speech Files

The purpose of this exercise is to begin experimenting with MATLAB m-files. You will create an m-file that loads and displays a speech file. You will also gain some experience with importing data into MATLAB, plotting routines, conditional branch control flow, and sound routines. This exercise and exercise 1-4 are both given in a step-by-step manner. Later exercises will require you to work out most of the details on your own.

a) The following steps occur in the m-file editor / debugger. Type a function header on the first line as follows:

```
function  [s,fs,bits] = ex13( infile, playstate )
```

b) You have now defined a function in a MATLAB m-file. The next step is to define the help text using the comment delimiter, %. On the next lines type the following:

```
% 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
```

c) You have now defined the help text for the function ex13.m. You are ready to begin entering the actual executable portion of the program. The steps our program should follow are

i) Load infile
ii) Display infile
iii) Playback infile if playstate is set

These steps are accomplished using the four simple commands given in Table 1.

wavread	plot	if	sound

Table 1. Built-in Functions Used in Exercise 1-3.

Each of these MATLAB programming constructs has detailed help available in the Netscape help browser or on the MATLAB command line using the `help` command. You should refer to the help screens if unsure of how to use the commands.

d) Given the program outline above, your task is to implement steps i-iii: First, locate the hypertext help for the command wavread. MATLAB uses this command to load a .WAV file into memory. It will optionally return important information about the .WAV file such as sample rate and bits per sample. After your comments, enter the following text:

```
[s,fs,bits]=wavread(infile);
```

Save the program as ex13.m. Your program is now ready to load the .WAV file specified by the variable infile. Let's test your program to see how you are doing so far.

Go back to the command prompt in MATLAB (>>) and type:

```
>> ex13('cleanspeech',0)
```

What happened? Why?

e) The large amount of rapidly scrolling text you saw was the return vector from your function displayed in the command window. This occurred because you didn't terminate the call to ex13 with a semicolon. Now type the following, again in the command window:

```
>> [s,fs,bits]=ex13('cleanspeech',0);
```

This time, you terminated with a semicolon so the results are not displayed in the command window. To examine the values returned by your m-file, type:

```
>> bits
```

and then,

```
>> fs
```

How many bits per sample in 'cleanspeech' ? What is the sample rate?

f) At this point, you have completed step i of the program outline. It is now time to display the data. On the next line of your m-file, add the command:

```
plot(s)
```

Save the file ex13.m and run your program again from the command line. What happened?

Now do some experimentation with the plot command. It is a very flexible command that allows many different types of plotting. Browse some of the hypertext help on plot() and try to generate plots with different line styles. *REMEMBER TO SAVE THE FILE AFTER EACH CHANGE.* Try, for example,

```
plot(s,':')
```

Save the m-file, then run. What happened? Next try

```
plot(s,'r:')
```

Now what happened? Why?

Also try experimenting with plot commands on the command line. Run your program as before, then plot your data in s from the command line directly instead of from the m-file. You should now understand the relationship between the command line interpreter and the interpreted m-file.

g) Now is a good time to experiment a bit with the indexing features of MATLAB. You will need to understand these features to manipulate vectors, matrices, and higher-dimensional data structures. Let's say you only care about the first 256 samples of the speech file. Try the following:

```
>> plot(s(1:256))
```

What happened?

Now use a vector to do the indexing.

```
>> j=1:256;
>> plot(s(j));
```

Was the result the same? Next, try this variation:

```
>> j=1:256;
>> plot(j+512,s(j));
```

What changed? Why?

Now, let's examine the data manually frame-by-frame. Again, use a variable to do the indexing. Let's suppose a frame size of 256 is required:

```
>> N=256;
>> j=j+N;
>> plot(j,s(j));
```

What are we doing differently with the plot statement? If you don't understand, read the hyper-text help. To sequence manually through the frames, repeat the following steps at the command line:

```
>> j=j+N;
>> plot(j,s(j));
```

NOTE: You can use the up arrow key to repeat prior commands.

Also, try some of the annotation features for plotting. Type the following:

```
>> title('Cleanspeech time waveform');
>> xlabel('Sample Number');
>> ylabel('Normalized Amplitude');
```

h) At this point, you have completed steps i) and ii) of our program outline. It is now time to conditionally play back the sound data. On the next line of your m-file, add the commands:

```
if playstate == 1
     sound(s,fs);
end
```

Test your program again using the command line:

```
>> [s,fs,bits]=ex13('cleanspeech',0);
```

Nothing new should have happened. You set the logical variable playstate to 0 so the conditional branch of the if statement was skipped. Now try:

```
>> [s,fs,bits]=ex13('cleanspeech',1);
```

This time, your program should have played back the cleanspeech file. What does it say?

Next, use your program to process the file "noisyspeech". How is this file different from "cleanspeech" ?

At this point, you have completed your first MATLAB programming assignment.

Make sure your ex13.m file is saved, then close it in preparation for the next exerciese.

Exercise 1-4: Create m-file to Load and Display Two Speech Files Frame-by-Frame.

The purpose of this exercise is to build on the knowledge you gained during exercise 1-3 in order to create an m-file which will load and display two speech files frame-by-frame. This program will be used later during an exercise on speech enhancement. In this exercise, you will be introduced to MATLAB looping constructs. You will also gain some further experience with MATLAB indexing and plotting facilities.

a) The following steps occur in the m-file editor/debugger. Open a new file in the editor/debugger. Type a function header on the first line as follows:

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

b) As in exercise 1-3, define the help text using the comment delimiter, %. On the next lines type the following:

```
% [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.
```

c) You have now defined the help text for the function ex14.m. You are ready to begin entering the actual executable portion of the program. The outline for your program is:

i) Load infile1
ii) Load infile 2
iii) Determine the number of frames in the shorter of the two files
iv) For each frame, display contents of each file in a subplot window (solid, blue)
v) For each frame, display difference signal in both windows (dashed line, green)
vi) Pause after each frame and wait for keypress to continue

These steps are accomplished using vector indexing, some simple computations, and the built-in functions shown in Table 2.

wavread	length	plot	subplot
for	pause	min	sprintf

xlabel	ylabel	title	fix

Table 2. Built-in Functions Used in Exercise 1-4.

Each of these MATLAB programming constructs has detailed help available in the Netscape help browser. You should refer to the help screens if still unsure of how to use the commands after reading the information in this handout.

d) Given the program outline above, your task is to implement steps i-vi: After your comments, enter the following text (Step i):

```
[s1,fs1,bits1]=wavread(infile1);
[s2,fs2,bits2]=wavread(infile2);
```

Save the program as ex14.m. Your program is now ready to load the .WAV files specified by the variables infile1 and infile2.

e) Once the speech records are in memory, we next wish to determine the number of frames in the shorter of the two files (Step ii). Add the following commands to p14.m after the file loading section. NOTE: Feel free to insert comments using the % as you go.

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

What does K represent? What is the fix command doing here (check the help text). What does this imply about files that do not have a number of samples evenly divisible by the frame size?

Also in this step, compute a difference signal:

```
e=s1-s2;
```

You probably already realize that K represents the number of frames to be displayed and processed. Next, establish a loop using the for command which will be used to sequence through the frames of the two files. Compute for each frame the samples to be displayed, then plot the two signals, each in its own window. Several new commands are introduced here. You should try to use the help system to understand what the commands in this loop are doing if they look unfamiliar. All were discussed during the lecture.

```
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
```

How would you characterize the differences between the files cleanspeech and noisyspeech? What does the difference signal look like? What does it sound like? Can you characterize the difference signal using some other built-in functions? (use help to browse the statistical functions such as `mean()` and `std()`)

At this point, you have completed the second MATLAB programming assignment. This section has introduced looping, formatted printing, subplots, and the pause command. You now have a program to display two speech files frame by frame. The plots also include a difference signal. This type of processing will be used in the later exercises.

The next three exercises will add some features to this program. In these sections, you will be expected to figure out the details on your own.

Exercise 1-5: Incorporate Frame Energy Computation

The purpose of this exercise is to add functionality to the program you developed during exercise 1-4. The goal is to add a frame energy calculation. In the program editor and the ex14.m window, do a Save As and save the file as ex15.m so that you can start with the program already developed. This will allow you to prevent ruining the program you already have working when making changes.

a) In the new file p15.m, modify the function header on the first line to match the filename as follows:

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

b) Keep the same help text from exercise 1-4 and add a comment about computing frame energy.

```
% [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.
```

c) You have now defined the help text for the function ex15.m. You are ready to begin modifying the existing program. The new outline for your program is:

i) Load infile1
ii) Load infile 2
iii) Determine the number of frames in the shorter of the two files
iv) For each frame, display contents of each file in a subplot window (solid, blue)
v) For each frame, display difference signal in both windows (dashed line, green)
vi) For each frame, display an average energy per sample in both windows
vii) Pause after each frame and wait for keypress to continue

Only step vi) is new in the program outline. It does not require the use of any additional built-in functions.

d) Modify the existing MATLAB code to compute normalized frame energy (average energy per sample) for each frame and for each file. Normalized frame energy, E_m, isgiven by

$$E_m = \frac{1}{N}\sum_{i=0}^{N-1} x^2(mN+i) \tag{1}$$

where $x(n)$ are the time-domain speech samples, N is the frame size, and m is the frame number.

NOTE: This is a good example of a situation in which you should use vector processing instead of the usual scalar processing you might use when programming in another language such as 'C'. The summation in Eq. (1) could be implemented using either a for loop in MATLAB or a sum() call in MATLAB. Both are inefficient choices. The summation in Eq. (1) is nothing more than a vector inner product, or using vector notation for the Nx1 vector x(mN+(0:N-1)),

$$E_m = \frac{1}{N}\langle \mathbf{x}, \mathbf{x} \rangle = \frac{1}{N}\mathbf{x}^T\mathbf{x} \tag{2}$$

In MATLAB, it is preferable to use a vector multiplication in this circumstance, or something like this:

```
E(m)=(x'*x)/N;
```

rather than a for loop or sum() call. MATLAB is highly optimized for vector operations rather than scalar operations in iterative loops such as for or while loops. You should keep this in mind at all times when programming in MATLAB, particularly if the algorithms you are implementing are of even modest complexity. You can verify this experimentally by creating rather large ran-

dom vectors and computing the energies using different m-files. Try vectors of 20,000 to 100,000 samples. Generate the test vector using the command:

```
>>v=randn(100000,1);
```

Then compare the execution times on the command line for energy calculations using the sum() command, a for loop, and the vectorized version. The difference will become more dramatic as the vector size increases. You can also make the example more dramatic by adding more operations inside of the loop, for example, adding or subtracting constants. These operations should always be vectorized whenever possible. Try some simple experiments to convince yourself of this.

e) Modify your program to compute the average energy per sample and display it in a text message in the subplot window. This can be done using the sprintf () and text() commands. Use the help facility to determine the proper usages. You could also display the results on the command line using the fprintf or display commands.

Exercise 1-6: Incorporate an SNR Calculation

a) Modify the existing MATLAB code to compute the global signal-to-noise ratio (SNR) between cleanspeech.wav and noisyspeech.wav. For the purposes of this exercise, SNR is defined as follows. The SNR is the ratio of the signal variance to the noise variance. For an N x 1 signal vector x, and an N x 1 noise corrupted version of the same vector, $\hat{x}$, the SNR is defined as

$$SNR = 10\log 10\left(\frac{\frac{1}{N}\sum_{i=1}^{N} x^2(i)}{\frac{1}{N}\sum_{i=1}^{N}\left(x(i)-\hat{x}(i)\right)^2}\right) \tag{3}$$

where $x(n)$ are the samples, N is the vector dimension, and the logarithmic transformation means that the SNR is expressed in terms of decibels (dB).

b) Modify the program to display the SNR in the plot window or on the command line. What is the SNR of the noisyspeech file?

This portion of the program will be used in a later exercise on speech enhancement.

Exercise 1-7: Add Save and Playback Features

a) Modify the existing program to generate and save two new signals. First, compute the difference signal and save it as a .WAV file.

b) Next, compute a summation signal and save it as a .WAV file.

c) Process the signals “garbled1.wav” and “garbled2.wav”. Playback the summation and difference signals. What do they contain?

d) How are these signals related to the originals?

Summary and Conclusions for Exercises 1-1 to 1-7

This exercise has introduced a variety of MATLAB built-in functions (Tables 1 and 2) and given you exposure to several fundamental and quite important MATLAB programming concepts, including:

- m-file editing and debugging
- Data input/output, both file-based and console-based
- Vector indexing
- Conditional branching
- Iterative processing – for loops
- Vectorized versus iterative scalar operations
- Basic data visualization and plotting

Project Deliverables

Prepare a report where all the results and software are included:

- Please give a small introduction in the report.
- Please itemize sections in the report in the same order as in the exercise. Include all graphs, all computations, etc. Attach in an appendix all m code you wrote.
- Please write a conclusions section detailing what have you learned from the exercise
- Be concise (email your report as a doc file)