GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

EE 2200 Fall 1998

Lab #1: Introduction to MATLAB

Date: week of 28 Sept 1998

Lab is held in College of Computing Building, room 309.

This is *the official* Lab#1 description; it is *different* from the one in Appendix C of the text.

The Warm-up section of each lab must be completed in Lab and the steps marked *Instructor Verification* must also be signed off **during the lab time**. One of the laboratory instructors must verify the appropriate steps by initialing on the **Instructor Verification** line. When you have completed a step that requires verification, simply raise your hand and demonstrate the step to the instructor.

It is only necessary to turn in Section 2 with explanations and Section 3 with answers to the review questions as this week's lab report. Staple the **Instructor Verification** sheet to your lab report to prove that the appropriate steps were witnessed by the instructor. *Forgeries are a violation of the honor code and will be referred to the Dean of Students for disciplinary action.*

The report will **due during the week of 5-Oct at the start of your lab**.

---

1 Warm-up

1.1 Overview

MATLAB will be used extensively in all the labs. The primary goal of this lab is to familiarize you with using MATLAB. Please read Appendix B: *Programming in* MATLAB for an overview. Here are three specific goals for this lab:

1. Learn basic MATLAB commands and syntax, especially the `help` system.
2. Write your own functions in MATLAB.
3. Learn a little about advanced programming techniques for MATLAB, i.e., vectorization.

1.2 Preliminaries

Before doing the remainder of the lab it will be necessary to set up your computer account by following the directions from the Laboratory Information on Web-CT. Ask a TA for help if you encounter problems.

    http://webct.ece.gatech.edu

1.3 Getting Started

Once your account is set up you can log in and then start MATLAB by typing `matlab` in a terminal window or by selecting MATLAB from a menu such as the `START` menu under Windows-95/NT.

The following steps will help to orient you to MATLAB.

(a) View the MATLAB introduction by typing `intro` at the MATLAB prompt. This short introduction will demonstrate some of the basics of using MATLAB.

(b) Explore the MATLAB help capability. Try the following (it is possible to force MATLAB to display only one screen-full of information at once by issuing the command `more on`):

```
help
help plot
help colon
help ops
help zeros
help ones
lookfor filter       %<--- keyword search
```

(c) Run the MATLAB help desk by typing `helpdesk`. The help desk is a hypertext interface to the MATLAB documentation. The preferences can be set to use Netscape or Internet Explorer as the browser.

(d) Run the MATLAB demos: type `demo` and explore a variety of basic MATLAB commands and plots.

(e) Use MATLAB as a calculator. Try the following:

```
pi*pi - 10
sin(pi/4)
ans ^ 2      %<--- "ans" holds the last result
```

(f) Do variable name assignment in MATLAB. Try the following:

```
x = sin( pi/5 );
cos( pi/5 )      %<--- assigned to what?
y = sqrt( 1 - x*x )
ans
```

(g) Complex numbers are natural in MATLAB. The basic operations are supported. Try the following:

```
z = 3 + 4i
conj(z)
abs(z)
angle(z)
real(z), imag(z)
exp( j*pi )
exp(j*[ pi/4, 0, -pi/4 ])
```

(h) Plotting is easy in MATLAB for both real and complex numbers. The basic plot command will plot a vector `y` versus a vector `x`. Try the following:

```
x = [-3 -1 0 1 3 ];
y = x.*x - 3*x;
plot( x, y )
z = x + y*sqrt(-1)
plot( z )       %<---- complex values can be plotted
```

Use `help arith` to learn how the operation `xx.*xx` works; compare to matrix multiply.

When unsure about a command, use `help`.

1.4 MATLAB Array Indexing

(a) Make sure that you understand the colon notation. In particular, explain what the following MATLAB code will produce

```
jkl = 2 :  4 :  17
jkl = 99 :  -1 :  88
ttt = 2 :  (1/9) :  4
tpi = pi * [ 2 :  (-1/9) :  0 ];
```

(b) Extracting and/or inserting numbers in a vector is very easy to do. Consider the following definition:

```
xx = [ ones(1,4), [2:3:17], zeros(1,3) ]
xx(4:6)
size(xx)
length(xx)
xx(2:2:length(xx))
```

Explain the result echoed from the last four lines of the above code.

(c) Observe the result of the following assignment:

```
xx(4:6) = pi*(1:3)
```

Now write a statement that will replace the even indexed elements (`xx(2)`, `xx(4)`, etc) with the constant -13. *Use a vector replacement, not a loop.*

Instructor Verification (separate page)

1.5 MATLAB Script Files

(a) Experiment with vectors in MATLAB. Think of the vector as a set of numbers. Try the following:

```
kset = 0:11;
kset
cos( pi*kset/4 )      %<---comment:  compute cosines
```

Explain how the last example computes the different values of cosine.
The text following the `%` is a comment; it may be omitted.
NOTE: the semicolon at the end of a statement will suppress the echo to the screen.

(b) (A taste of vectorization) Loops can be written in MATLAB, but they are NOT the most efficient way to get things done. It's better to **avoid loops** and use the vector notation instead, but here is a loop that computes values of the cosine function. (The index of `x()` must start at 1.) Rewrite this computation without using the loop (as in the previous part).

```
x = [ ];   %<--- initialize the x vector to a null
for k=-3:4
   x(k+4) = cos( k*pi/4 )
end
x
```

(c) Use the built-in MATLAB editor (on a Mac), or an external one such as `EMACS` or `notepad` (on Windows-95/NT or UNIX), to create a script file called `asdfg.m` containing the following lines:

```
tt = -2 :  0.05 :  3;
xx = cos( 2*pi*0.8*tt );
plot( tt, xx ), grid on     %<--- plot a sinusoid
title('TEST PLOT of a SINUSOID')
xlabel('TIME (sec)')
```

(d) Run your script from MATLAB. To run the file `asdfg` that you created previously, try

```
asdfg          %<---will run the commands in the file
type asdfg     %<---will type out the contents of
               %   asdfg.m to the screen
```

(e) Add three lines of code to your script, so that it will plot a phase-shifted cosine on top of the first cosine. Use the `hold` function to add a plot of

```
0.5*cos( 2*pi*0.8*tt - pi/2 )
```

to the plot created by the above statements. See `help hold` in MATLAB.

Instructor Verification (separate page)

1.6 MATLAB Sound

(a) Run the MATLAB sound demo by typing `xpsound` at the MATLAB prompt. If you are unable to hear the sounds in the MATLAB demo then ask an instructor for help.

When unsure about a command, use `help`.

(b) Now generate a tone (i.e., a sinusoid) in MATLAB and listen to it with the `sound` command. The frequency of your tone should be 1.7 kHz and the duration should be 0.7 sec. Use a sampling rate of 11025 samples/sec. The sampling frequency demands that the time-vector be defined as follows:

```
tt = 0:(1/fs):dur;
```

where `fs` is the desired sampling rate and `dur` is the desired duration (in seconds). Read the online `help` for `sound` and `soundsc` to get more information on using this command.

Instructor Verification (separate page)

1.7 Functions

The following exercises are to help you in writing functions in MATLAB (see Section B.5 in Appendix B). This capability will be valuable so that you can write modular code based on functions.

(a) Find the mistake(s) in the following function:

```
function  [xx,tt] = cosgen(f,dur)
%COSGEN  Function to generate a cosine wave
%  usage:
%        xx = cosgen(f,dur)
%          f = desired frequency
%        dur = duration of the waveform in seconds
%
tt = 0:1/(20*f):dur;  % gives 20 samples per period
cosgen = cos(2*pi*freq*tt);
```

(b) Explain the following lines of MATLAB code:

```
yy = ones(7,1) * rand(1,4);

xx = randn(1,3);
yy = xx(ones(6,1),:);
```

When unsure about a command, use `help`.

(c) Write a function that performs the same task as the following without using a `for` loop.

```
function Z = expand(x,ncol)
%EXPAND  Function to generate a matrix Z with identical columns equal
%        to an input vector x
%  usage:
%        Z = expand(x,ncol)
%        x = the input vector containing one column for Z
%     ncol = the number of columns needed in Z
%
x = x(:);  %--this turns the input vector x into a column vector
Z = zeros(length(x),ncol);
for i=1:ncol
   Z(:,i) = x;
end
```

Instructor Verification (separate page)

1.8 Vectorization of Logical Operations

(a) Explain the following lines of MATLAB code:

```
A = randn(6,3);
A = A .* (A>0);
A = A + 2*(A==0);
```

(b) Write a new function that performs the same task as the following function without using a `for` loop. Use the idea in part (a) and also consult Section B.7.3 on vector logicals in Appendix B: *Using* MATLAB. In addition, the MATLAB logical operators are summarized via `help relop.`

```
function  Z = replacez(A)
%REPLACEZ  replace the negative elements of a matrix with the number 77
%   usage:
%      Z = replacez(A)
%      A = input matrix whose negative elements are to be replaced
%
[M,N] = size(A);
for i=1:M
   for j=1:N
      if A(i,j) < 0
         Z(i,j) = 77;
      else
         Z(i,j) = A(i,j);
      end
   end
end
```

2 Laboratory: Manipulating Sinusoids with MATLAB

Now you're on your own. $\boxed{\textbf{Include a short report of this Section in your Lab report.}}$

Generate two 2.222 kilohertz sinusoids with arbitrary amplitude and phase.

$$x_1(t) = A_1 \cos(2\pi(2222)t + \phi_1) \qquad\qquad x_2(t) = A_2 \cos(2\pi(2222)t + \phi_2)$$

(a) Select the value of the amplitudes and phases at random. Let $A_1 = 29$ and use your age for A_2. For the phases, use the middle two digits of your SSN for ϕ_1 (in degrees), and take $\phi_2 = -45°$.
NOTE: when doing computations, make sure to convert degrees to radians!

(b) Make a plot of both signals over a range of t that will exhibit approximately 3 cycles. Make sure the plot starts at a negative time so that it will include $t = 0$. **AND make sure that you have at least 20 samples per period of the wave.**

(c) Verify (by hand) that the phase of the two signals $x_1(t)$ and $x_2(t)$ is correct at $t = 0$, and also verify that each one has the correct maximum amplitude.

(d) Use `subplot(3,1,1)` and `subplot(3,1,2)` to make a three-panel subplot that puts both of these plots on the same page. See `help subplot`.

(e) Create a third sinusoid as the sum: $x_3(t) = x_1(t) + x_2(t)$. In MATLAB this amounts to summing the vectors that hold the samples of each sinusoid. Make a plot of $x_3(t)$ over the same range of time as used in the last plot. Include this as the third panel in the plot by using `subplot(3,1,3)`.

(f) Put a title containing your name on your plot, and then print it. See `help title` and `help print`.

(g) Measure the magnitude and phase of $x_3(t)$ directly from the plot. Turn in this plot with sufficient annotation to show how the magnitude and phase were measured.

3 Lab Review Questions

In general, your lab write-up should indicate that you have acquired a better understanding of the topics treated by the laboratory assignment. Here are a few questions for you to answer in order to assess your understanding of this lab's objective: a working knowledge of the basics of MATLAB. If you do not know the answers to these questions go back to the lab and try to figure them out in MATLAB (remember the commands `help` and `lookfor`).

1. You saw how it easy it is for MATLAB to generate and manipulate vectors (i.e., 1-dimensional arrays of numbers). For example, consider the following:

```
nn = 0:10;
mm = ones(1,70);
kk = -3:0.25:3;
```

   (a) How would you modify one of the above lines of MATLAB code to create a vector that steps from $-\pi$ to π in steps of 0.2π?

   (b) How would you modify one of the lines to create a vector containing seventy-seven eights?

2. You also learned that MATLAB has no problem handling complex numbers. Consider the following line of code:

   $$\texttt{yy} = -5 + 12\texttt{j};$$

   (a) How do you get MATLAB to return the magnitude-squared of the complex number `yy`?

   (b) How do you get MATLAB to return the phase of the complex number `yy`? What are the units of the phase?

3. In Section 1.5, you learned that multiple lines of MATLAB code can be stored in a file with an extension of `.m`. MATLAB then executes the code in the order that it appears in the file. Consider the following file, named `example.m`:

```
ff = 222;
tt =  0:1/(33*ff):0.09;
zz = exp(2i*pi*ff*tt);
subplot(211)
plot(real(zz))
title('Real part of exp(j*2*pi*222*tt)')
subplot(212)
plot(imag(zz)), grid on
title('Imaginary part of e^{j2\pi(222)t}','FontSize',14)
```

   (a) How do you execute the file from the MATLAB prompt?

   (b) Suppose the file were named `example.dog`. Would it run? How could you change it to make it work in MATLAB?

   (c) Assuming the M-file runs, what do you expect the plots to look like? If you're not sure type in the code and run it. Hint: recall Euler's formula.

Lab #1
EE-2200
Fall-1998
Instructor Verification Sheet

Staple this page to the end of your Lab Report.

Name: ____ Date of Lab: ___________

Part 1.4 Vector replacement using the colon operator:

Verified: ___________________

Part 1.5 Run the modified function `asdfg` from a file:

Verified: ___________________

Part 1.6 Use `sound` to play a 1.7 kHz tone in MATLAB:

Verified: ___________________

Part 1.7 Modify the MATLAB function `expand` to work without a `for` loop:

Verified: ___________________