

GEORGIA INSTITUTE OF TECHNOLOGY  
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

**ECE 2025 Fall 2000**  
**Lab #1: Introduction to MATLAB**

Date: 28–31 Aug 2000

---

Lab is held in Van Leer, room 252, which has NT workstations.

⇒ Please verify that your ECE account exists by going to VL-252 and logging in before your first lab. The login should be your “gtxxxx” number and the *initial* password your complete nine-digit student number.

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 **during your assigned Lab time** 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 signing on the **Instructor Verification** line. When you have completed a step that requires verification, simply raise your hand and demonstrate the step to the TA or instructor.

It is only necessary to turn in Sections 2 and 3 as this week’s lab report: Section 2 with graphs and explanations, and Section 3 with answers to the review questions. 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-Sept. at the start of your lab**. If you have lab on Monday, turn in the report on 11-Sept.

---

**PRINTING BUDGET:** For the printers in the ECE labs, you have a quota. Please limit your printing to essential items for the labs. If you need to print lecture slides and other large documents, use the central (OIT) printing facilities.

---

## 1 Warm-up

### 1.1 Overview

MATLAB will be used extensively in all the labs. The primary goal of this lab is to familiarize yourself 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, including the help system.
2. Write and edit your own script files in MATLAB, and run them as commands.
3. Learn a little about advanced programming techniques for MATLAB, i.e., vectorization.

### 1.2 Movies: MATLAB Tutorials

On the Web-CT course page, there are a large number of Real-media movies on basic topics in MATLAB, e.g., colon operator, indexing, functions, etc. Look for the link with the movie film icon.

<http://classweb.gatech.edu:8080/public/ECE2025/index.html>

### 1.3 Preliminaries: NT Account Activation

For each student in ECE-2025 a computer account has been created. The username is your “gtxxxx” number and the password is your student number. *You must change your password very soon; otherwise, your account will be de-activated in a week or so.* Send mail to help@ece.gatech.edu if you encounter problems, or ask your TA for help.

### 1.4 Getting Started

After logging in, you can start MATLAB by double-clicking on a MATLAB icon, typing matlab in a terminal window, or by selecting MATLAB from a menu such as the START menu under Windows-95/98/NT. The following steps will introduce 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      %<--- a VERY IMPORTANT notation
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. Especially helpful is the link entitled **Getting Started**. The MATLAB preferences can be set to use Netscape or Internet Explorer as the browser for help.
- (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, w = -3 + 4j
real(z), imag(z)
abs([z,w])      %<-- Vector constructor
conj(z+w)
angle(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: plot imag vs. real
```

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

When unsure about a command, use `help`.

## 1.5 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 * [ 0:0.1:2 ];
```

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

```
xx = [ zeros(1,3), linspace(0,1,5), ones(1,4) ]
xx(4:6)
size(xx)
length(xx)
xx(2:2:length(xx))
```

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

- (c) Observe the result of the following assignments:

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

Now write a statement that will take the vector `xx` defined in part (b) and replace the even indexed elements (i.e., `xx(2)`, `xx(4)`, etc) with the constant  $\pi^\pi$ . Use a *vector replacement*, not a loop.

**Instructor Verification** (separate page)

## 1.6 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
xk = cos( pi*kset/4 )      %<---comment: compute cosines
```

Explain how the different values of cosine are stored in the vector `xk`. What is `xk(1)`? Is `xk(0)` defined?

NOTES: the semicolon at the end of a statement will suppress the echo to the screen. The text following the `%` is a comment; it may be omitted.

- (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 **always avoid loops** and use the colon notation instead. The following

code has a loop that computes values of the cosine function. (The index of `yy( )` must start at 1.) Rewrite this computation without using the loop (follow the style in the previous part).

```
yy = [ ]; %<--- initialize the yy vector to be empty
for k=-5:5
    yy(k+6) = cos( k*pi/3 )
end
YY
```

- (c) Use the built-in MATLAB editor (on Windows-95/98/NT), or an external one such as EMACS on UNIX/LINUX, to create a script file called `lab166.m` containing the following lines:

```
tt = -2 : 0.05 : 3;
xx = cos( 2*pi*1.5*tt );
plot( tt, xx ), grid on %<--- plot a sinusoid
title('TEST PLOT of a SINUSOID')
xlabel('TIME (sec)')
```

Note: *Do not save* this file or any of your MATLAB files to the local hard disk. Your computer account contains a private networked directory where you can store your own files. Use the MATLAB command `addpath( )` to allow MATLAB to “see” your personal directory.

- (d) Run your script from MATLAB. To run the file `lab166` that you created previously, try

```
lab166 %<---will run the commands in the file
type lab166 %<---will type out the contents of
% lab166.m to the screen
```

**Instructor Verification** (separate page)

## 1.7 MATLAB Sound

The exercises in this section involve sound signals, so you should bring headphones to the lab for listening.

- (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 `soundsc( )` command.<sup>1</sup> Refer back to part 1.6(c) for some code that creates values of a sinusoid. The frequency of your sinusoidal tone should be 1300 Hz and its duration should be 0.9 sec. Use a sampling rate (`fs`) equal to 11025 samples/sec. The sampling rate dictates the time interval between time points, so the time-vector should 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 both `sound( )` and `soundsc( )` to get more information on using this command. What is the length of your `tt` vector?

**Instructor Verification** (separate page)

<sup>1</sup>The `soundsc(xx, fs)` function requires **two** arguments: the first one (`xx`) contains the vector of data to be played, the second argument (`fs`) is the rate for playing the samples. In addition, `soundsc(xx, fs)` does automatic scaling and then calls `sound(xx, fs)` to actually play the signal.

## 2 Laboratory: Manipulating Sinusoids with MATLAB

Now you're on your own. **Include a short summary of this Section with plots in your Lab report.**

Write a MATLAB script file to do steps (a) through (d) below. Include a listing of the script file with your report.

- (a) Generate a time vector (`tt`) to cover a range of  $t$  that will exhibit approximately 3 cycles of the 1300 Hz sinusoids defined in the next part, part (b). Use a definition for `tt` similar to parts 1.6(c) and 1.7(b). If we use  $T$  to denote the period of the sinusoids, define the starting time of the vector `tt` to be equal to  $-T$ , and the ending time as  $2T$ . Then the three cycles will include  $t = 0$ . **Finally, make sure that you have at least 20 samples per period of the sinusoidal wave.**
- (b) Generate two 1300 Hz sinusoids with arbitrary amplitude and phase.

$$x_1(t) = A_1 \cos(2\pi(1300)t + \phi_1) \quad x_2(t) = A_2 \cos(2\pi(1300)t + \phi_2)$$

Select the value of the amplitudes and phases as follows: Let  $A_1$  be equal to your age and set  $A_2 = 16.9$ . For the phases, use the month of your birthday for  $\phi_1$  (in radians), let  $\phi_2 = 13.25\pi$  radians.

NOTE: when doing computations, make sure to convert degrees to radians!

Make a plot of both signals over the range of  $-T \leq t \leq 2T$ . For your final printed output in part (d) below, use `subplot(3,1,1)` and `subplot(3,1,2)` to make a three-panel subplot that puts both of these plots in the same figure window. See `help subplot`.

- (c) 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 values of each sinusoid. Make a plot of  $x_3(t)$  over the same range of time as used in the plots of part (b). Include this as the third panel in the plot by using `subplot(3,1,3)`.
- (d) Before printing the three plots, put a title on each subplot, and include your name in one of the titles. See `help title`, `help print` and `help orient`, especially `orient tall`.

### 2.1 Theoretical Calculations

Remember that the phase of a sinusoid can be calculated after measuring the time location of a positive peak,<sup>2</sup> if we know the frequency.

- (a) Make measurements of the “time-location of a positive peak” and the amplitude from the plots of  $x_1(t)$  and  $x_2(t)$ , and write those values for  $A_i$  and  $t_{m_i}$  directly on the plots. Then verify (by hand) that the phases of the two signals,  $x_1(t)$  and  $x_2(t)$ , are correct by converting each time-shift  $t_{m_i}$  to phase. Write the calculated phases  $\phi_i$  directly on the plots.
- (b) Measure the amplitude and time-shift of  $x_3(t)$  directly from the plot and then calculate the phase ( $\phi_3$ ) by hand. Write these values directly on the plot to show how the amplitude and time-shift were measured, and how the phase was calculated.
- (c) Now use the phasor addition theorem. Carry out a phasor addition of complex amplitudes for  $x_1(t)$  and  $x_2(t)$  to determine the complex amplitude for  $x_3(t)$ . Use the complex amplitude for  $x_3(t)$  to justify that your measurements of  $A_3$  and  $\phi_3$  were consistent.

---

<sup>2</sup>Usually we say time-delay or time-shift instead of the “time location of a positive peak.”

### 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. Answer the questions below in your lab report as an assessment of 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`). Also, consult Appendix B of *DSP First* as a reference source.

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*(0:44);  
mm = zeros(1,44);  
kk = 0:pi/44:pi;
```

- (a) Is the length of `kk` equal to 44 or 45? Explain.
  - (b) Which one of the lines above will produce forty-four zeros?
  - (c) How would you modify one of the above lines of MATLAB code to create a vector containing forty-five sevens?
2. You also learned that MATLAB has no problem handling complex numbers. Consider the following line of code:

```
yy = sqrt(3) - j;
```

- (a) How do you get MATLAB to return the magnitude 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?
  - (c) Use the relationship  $|z|^2 = (z^*)z$  to write a line of MATLAB code that returns the magnitude-squared of the complex number `yy`.
3. In Section 1.6, 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 `lab1test.m`:

```
ff = 22050;  
tt = 0:1/(20*ff):0.0002;  
zz = exp(2j*pi*ff*tt);  
subplot(2,1,1)  
plot(real(zz))  
axis tight  
title('Real part of exp(j*2*pi*22050*tt)')  
subplot(2,1,2)  
plot(tt, imag(j*zz))  
grid on, axis tight  
title('Imaginary part of je^{j2\pi(22050)t}', 'FontSize', 14)
```

- (a) How do you execute the file from the MATLAB prompt?
- (b) Explain how to use the `which` command to find the directory containing `lab1test.m`.
- (c) Suppose the file were renamed as `lab1test.lab`. Would it run? If not, how could you change it to make it work in MATLAB?

**Lab #1**  
**ECE-2025**  
**Fall-2000**  
**INSTRUCTOR VERIFICATION SHEET**

Turn this page in to your grading TA.

Name: \_\_\_\_\_ Date of Lab: \_\_\_\_\_

Part 1.5 Vector replacement using the colon operator:

Verified: \_\_\_\_\_ Date/Time: \_\_\_\_\_

Part 1.6 Run the modified function lab166 from a file:

Verified: \_\_\_\_\_ Date/Time: \_\_\_\_\_

Part 1.7 Use `soundsc()` to play a 2 kHz tone in MATLAB:

Verified: \_\_\_\_\_ Date/Time: \_\_\_\_\_