Statistics 322, Statistical Computing, Hand Out 1

August 22, 1993

This is intended to get you started with using MATLAB in the Statistics Department. MATLAB runs on both the Department UNIX, and also on PC's (although it may not be loaded on too many departmental PC's. The difference between the UNIX and PC versions appears to be fairly minimal, but we may find out more as we proceed.

There will doubtless be some problems in this document, and in others. Please tell me (either personally, or by email) about all that you find, as the document is evolving (and you get "credit" for doing so).

First, here are some basics of computing that you need to understand. If any of these puzzle you, ask a classmate, or Pranab Mandal.

1. The concept of "files", and how they are organized into "directories".

2. How to manipulate files, e.g. copy and move around in directories. On the UNIX system, this can be done via "SUN Windows", or else by command line UNIX.

3. How to edit files. On the UNIX system, this can be done using the editor in "SUN Windows". A simple and easy to use UNIX editor is PINE. More powerful, but also much harder to use UNIX editors include VI and EMACS.

4. How to print files.

5. How to use a World Wide Web Browser, such as Netscape, and how to find the Statistics 322 "Web Page", e.g. from the Department's Page, then through Fall 96 Schedule Page.

When you are puzzled about various aspects of MATLAB, one approach is to look in "the manual". This is actually a set of books. The most important of these for you right now are:

A. The User's Guide. This contains basic ideas and concepts. A useful part (discussed more below), is a long "tutorial" in Chapter 2.

B. The Reference Guide. This is the fattest of the books that come with MATLAB, and is mostly a list of all the MATLAB basic commands (there are a lot!). This is useful when you forget exactly how to use a command (although online help is available for any command as noted below). When you are wondering "how do I do...", but don't know the command, the first section of the reference guide, which contains logically grouped lists of MATLAB commands are useful.

Next comes an introduction to some basic ideas about MATLAB, and how it works. MATLAB on the PC is started by clicking on the MATLAB Icon. In UNIX it is started by the UNIX command:

```
matlab
```

After the software loads, MATLAB will display a "prompt", which looks like:

```
>>
```

This says that MATLAB is waiting for you to enter a MATLAB command. The most important to know is:

```
>>  exit
```

(you can also use `>> quit`)

It is important to understand that MATLAB operates in an "interactive" or "interpreted" mode. This is quite different from the "batch" mode that you might be familiar with if you are accustomed to primitive languages such as FORTRAN and C. The main difference is that batch mode processors run a program, and then "forget" everything, while "interactive" processors store everything in memory (this can surprise you, and leave you rather puzzled on occasion, here is an example).

```
» clear
» a = 3 ;
» b = sqrt(16) ;
» b = sqrt(b) ;
» A = b ;
» a

a =

     3

» A

A =

     2
```

Notes:

a. the purpose of "=" is to assign values to variables (OK to have the same variable on each side).

b. to see the value of a variable, use its name as a command.

c. MATLAB is case "sensitive", i.e. the difference between capitals and lower case *matters*. This can cause headaches if you are not used to it. It is a good idea to avoid use of both cases in this way.

d. the apparently tricky answer of $\sqrt{\sqrt{16}} = 3$ is caused by MATLAB "remembering" the value of `a`. Similar effects are a common source of errors.

e. the semicolon at the end of lines suppresses a display of the result of a command:

When you are wondering what variables you have in memory, use:

```
» who

Your variables are:

A         a         b         c
```

More information about the variables in memory comes from:

```
» whos
      Name       Size      Elements    Bytes   Density   Complex

         A     1 by 1         1          8       Full       No
         a     1 by 1         1          8       Full       No
         b     1 by 1         1          8       Full       No
         c     1 by 1         1          8       Full       No

Grand total is 4 elements using 32 bytes
```

If you feel like the memory is a total mess, you can delete all variables with:

```
» clear
» who

Your variables are:
```

(note: nothing listed because variables have been cleared)

More commands in this spirit can be found on page 2 of the Reference Guide, and also via the online "help". Try these:

```
» help
» help help
```

MATLAB can do all the things that your hand calculator can do, and much more:

```
» c = a - 4

c =

    -1
```

Here are two simple advantages over the classical calculator:

1. You can still see and redo your last screenful of calculations. To "redo", use the up arrow key, and note that this steps through your previous. Here the up arrow key was used, then the lines were appropriately edited, to change the above calculation to $\sqrt{\sqrt{256}}$:

```
» x = sqrt(256) ;
» sqrt(x)

ans =

     4
```

2. MATLAB is much more accurate than your garden variety calculator:

```
» format long
» pi

ans =

   3.14159265358979
```

For today's purposes, we don't really need so many significant digits, the default is:

```
» format short
```

The power of using assignment statements comes from "building up" very long calculations, a piece at a time. Here is an example for finding the roots of a quadratic equation:

$$ax^2 + bx + c = 0,$$

for the values of `a=3, b=2, c=-1` that were put in memory by the above operations:

```
» sq = sqrt(b^2 - 4 * a * c) ;
» r1 = -b + sq ; r1 = r1 / (2*a) ;
» r2 = -b - sq ; r2 = r2 / (2*a) ;
» r1

r1 =

    0.3333
```

```
» r2

r2 =

    -1
```

But there are many things that MATLAB can do at a much higher level as well:

```
» roots([a b c])

ans =

   -1.0000
    0.3333
```

(this command found the roots of the polynomial given above, with much less programming effort). To really master MATLAB, you should take a careful look through pages 1-25 of the Reference Guide where all the commands are listed, (although this will be more productive when you have a firmer idea of the basics).

One of the most powerful aspects of MATLAB is that you can work directly, simply and efficiently with vectors and matrices. As in linear algebra, it is useful to think of vectors as matrices with just one row or one column, and of scalars as being 1×1 matrices.

Here is an example of the matrix structure of MATLAB makes it a really high level language: a least squares fit of a line to the points (1,4), (2,2), (3,0). First let's construct the familiar linear models "data vector" and "design matrix":

```
» y = [4 2 0]'

y =

     4
     2
     0

» X = [1 1 ; 1 2 ; 1 3]

X =

     1     1
     1     2
     1     3
```

A key concept of matlab is that matrices are defined by putting things (can be several types of thing, e.g. scalars, variables, or other matrices which then become submatrices, or even command outputs between []'s. Things separated by spaces are put into rows. Things separated by semicolons are treated as columns. Here is another way to construct the design matrix:

```
» x = [ones(3,1)   [1 2 3]']

x =
     1     1
     1     2
     1     3
```

Now do the familiar least squares fit matrix calculation:

```
» betahat = inv(x' * x) * x' * y

betahat =

    6.0000
   -2.0000
```

By a simple drawing, check that these are the least squares fit values of the slope and intercept. Think about how programming this one line calculation of `betahat` compares with the large amount of effort required to program the same thing in a primitive language like FORTRAN or C.

With computing in general, the best way to learn something is to actually do it. Hence I suggest that you really sit down and do each of the things that are discussed above, and watch what happens (especially if you do not feel 100% at home with any aspect of computing, or with MATLAB). This is also useful because you will learn about what happens when you make typing errors....

To learn more, I also suggest that you work through pages 2-1 through 2-24 of the "tutorial" in the User's Guide. Photocopies of this are being handed out for this purpose. This will put you more in touch with some of the ideas shown above. Most things in MATLAB (and indeed in computing in general) can be done in several ways. Above ONE way (not always the best way), was used. You are encouraged to try any variations that come to your mind as you work through.

While individual MATLAB commands are very powerful, and sometimes running them one at a time is useful, most serious work is doing by running collections of them all at once. This is done by putting them together into files called "script files". These always have the suffix ".m" (short for "matlab"). You create script files using a standard file editor, as described at 3 near the beginning of this document. The suffix of the script filename then works like a MATLAB command: type it at the prompt, and all the commands execute (unless there is some type of error!). In fact many of the MATLAB "commands" are actually of just scripts.

An important feature of script files is that lines starting with "`%`" are "comments", i.e. nothing is executed when MATLAB encounters such a line. This is useful in two ways:

(i) You can put in notes and reminders (classical programming "comments"), which say in words what is going on. This is useful in general, but particularly important if somebody else is going to look at your code, or when you look at it after a long time!

(ii) You can temporarily make lines not execute, by putting a "`%`" in front of them. This is very useful for debugging purposes.

For you to experiment with these ideas, in a hands on way, download the script file `322eg1.m` that is available on the Stat 322 Web Page, and run it with the matlab command

```
>> 322eg1
```

If you get the message:

```
???  Improper number or expression.
```

it is because MATLAB could not find the file, and instead was treating `322eg1` as a variable that had not been defined yet. A simple way for it to find the file (In UNIX, there is slightly different syntax), is to use a command like

```
>> path(path,'/home/myname/matlabdir')
```

This adds the quoted directory to the "MATLAB search path", i.e. the set of directories where it looks for .m files. You can configure MATLAB to automatically issue command of this type, at the beginning of your session, by creating a script `startup.m`, with such commands in it.

The program itself will explain what to do next, in particular how to edit it.

After this try Homework #1 (due August 29), which is downloadable as a .ps file. You can either print this, or just look at it with a viewer such as ghostview. It is a first taste of how numerical work with MATLAB can interact well with mathematics, and includes some "mathematical folklore".