ECE 330: Discussion #2

Some notes on MATLAB

Philosophy of MATLAB:
1) Do most of the computations using matrix-vector form.
MatLab = Matrix Laboratory.
2) Program-oriented software; i.e. you need to write programs for doing what you
want, or type the commands and functions at the command line.

How to get help: There are different ways to get help:
1) The most straightforward is to type help at the Matlab prompt. A list of categories
and toolboxes containing different kinds of functions will show up. If you need help
on any of those categories, type help <category-name>, and if you need help on a
certain function, type help <function-name>.
Very useful if also the function lookfor <word>, which looks for the <word> in
all Matlab functions. For example lookfor Fourier, will show up all the functions
where the word “Fourier” is present in their help instructions. This might be useful
when you know what you want to do but don’t know the name of the function which
accomplishes that specific task.
2) Ask for the Matlab manual at the CAE consultant office or at the library.
3) Ask me.

General comments and elementary commands/functions:
1) Extension of programs: Programs to be run in Matlab have extension .m
2) Editing a function: edit <function-name>. This opens up the Matlab editor
3) Writing a comment: Use the percentage symbol, %.
4) Displaying the content of a variable: Just type in the name of the variable at the
command line (or even within a program). You can also use the function disp or
fprintf for more sophisticated displays. Use the help command to know how they
work.
5) Use of “;”: The use of this operator at the end of the line prevents Matlab from
showing the output of the operation. Ex: x = 2 + 2 will show ans = 4 in the Matlab
window, while x = 2 + 2; won’t show the output.
6) Operators used for defining and manipulating matrices and vectors: [] ( ) ; ’
7) Defining a vector: There are several ways to do it:
a) Explicit way, using the [ ] operators. Ex: v = [1 2.1 6 3]; or v = [1,2.1,6,3];
defines the row vector with components [1,2.1,6,3]. Note the optional use of the
“,” for separating the elements of the vector.
An empty vector is defined by v = [];
b) Implicit way, using “:”. Ex: v = 1:2:5; defines the row vector of components
[1,3,5], that is starting at 1 and ending up at 5 in steps of 2. The step can be any
real number.
c) By specifying each element, using ( ). Ex: v(1) = 1; v(2) = 2.1; v(3) = 6;
v(4) = 3; defines the same row vector as in (a).
Very important observation: The vectors in Matlab as defined above are treated as row vectors by default, since they are considered a special case of matrices.

Suggestion: Define the vectors as column vectors if you want to be consistent with the notation used in linear algebra; for example writing \(v^\top v\) for an inner product (instead of \(v^\top v\) in case \(v\) was defined as a row vector). (\(\top\) is the transpose operator, see below)

7) **Defining a matrix:** The different ways of doing so are essentially the same as the ones described in (6). The additional feature that you need to know is the use of the operator “;” to separate from different rows, in the case you are using methods (6a) or (6b). Example:

\[
M = [1;2:5; 3\ 0\ -1];\text{ defines the matrix: } M = \begin{bmatrix} 1 & 3 & 5 \\ 3 & 0 & -1 \end{bmatrix}
\]

Note that the number of columns of each section separated by “;” must be the same.

In the case of method (6c), specify first the row index and then the column index.

Ex: \(M(1,1) = 1; M(1,2) = 2.1; M(1,3) = 6; M(1,4) = 3;\) defines the 1x4 matrix whose first row is the \(v\) vector defined in (6c).

Higher order “matrices” (example of dimensions 3x2x6) are also handled by Matlab.

The way to reference their elements is by referencing the first dimension first, then the second and so on. The use of the “;” operator might be useful when using the methods described in (6a) and (6b) (see below or type **help colon**).

8) **Transpose operator:** The transpose operator is given by ‘\(\top\)’. Example: \(M^\top\);

9) **Important use of the operator “;”:** This operator is used in matrices or vectors to reference a whole row or column. Example: to reference the first row of the \(M\) matrix defined in (7), use: \(M(1,:),\) where “:” means “take all the columns in the matrix”. The output is the row vector \([1,3,5]\). To reference the second column of the matrix \(M\), use \(M(:,2),\) whose output is the column vector \([1,3]^\top\), etc.

An useful use of “;” is when defining a matrix by its rows. Ex:

\[
M(:,1) = 1:2:5;\\nM(:,2) = [3\ 0\ -1];
\]

gives the same matrix as in (7).

10) **Plotting:** 1-dimensional: use the command **plot**. See help and examples below.

2-dimensional: for plotting surfaces use the command **mesh**. See help.

- In order to plot several graphs in the same plot, use the command **hold on** after the first **plot** command and then keep using **plot** for the successive graphs.
- In order to draw several plots in the same window, use **subplot**.
11) **Complex numbers:** To right complex numbers you can use the letters i or j. If they are not being already used as variables, they represent the imaginary unit. Ex: \( z = 1 + i*2 \) is the complex number \( 1 + i2 \).

   a) *Real and imaginary part:* \texttt{real} and \texttt{imag}.
   
   b) *Absolute value and phase:* \texttt{abs} and \texttt{angle}.

**Note:** If you need help or want to learn more uses of the operators [ , ( , : , etc.. type help paren and help colon.

**Miscellaneous commands:**

- **who / whos:** Shows the variables defined in the Matlab workspace; \texttt{whos} gives a more detailed information.
- **cd:** Changes the working directory.
- **dir:** (Self explaining…)
- **pause:** Produces a pause in the execution of a program.

You can get similar type commands by typing \texttt{help general}.

**Some examples**

1) Plotting the sine function, between \(-\pi\) and \(\pi\):

\[
\begin{align*}
&\texttt{>> t = -pi:2*pi/99:pi;}\quad &\text{\% Divide by 99, so that } t \text{ is a vector of length 100.} \\
&\texttt{>> plot( t, \sin(t) );}
\end{align*}
\]

(\texttt{pi} is an internally defined variable in Matlab)

**Note** that the function defined (\(\sin(t)\)) is a *discrete* function, but it *looks* continuous in the plot. Special care has to be used in selecting the step given to the independent variable (\(t\)), in order not to lose information of the function (i.e. the sampling period of the continuous function has to be small enough in order to have a representative plot of the underlying continuous function).

For example try doing the following:

\[
\begin{align*}
&\texttt{>> t = -pi:2*pi/5:pi;} \\
&\texttt{>> plot( t, \sin(t) );}
\end{align*}
\]

In order to choose the appropriate step, we need to have a prior estimate of the rate of variation of the function to be plotted.

2) Plotting the function \( y(t) = \sin(t)/t \) on the interval \([-10\pi, 10\pi]\):

\[
\begin{align*}
&\texttt{>> t = -10*pi:20*pi/499:10*pi;} \\
&\texttt{>> y = \sin(t)/t;}\quad &\text{\% Note the dot before the division operation.} \\
&\texttt{>> plot( t, y, 'r' );} &\text{\% 'r' means use red color.}
\end{align*}
\]