Big Picture

Matlab can be run either as a calculator from the command screen, or as a programming language when series of commands are saved to a file that ends in .m and the filename is entered from the command line. Every modern graduate of an electrical engineering program is expected to be familiar with Matlab and the C or C++ programming languages. Unlike C or C++, programming is simplified since variables don’t have to be declared, and there exists several built-in and powerful graphing commands.

Online help is available from the Matlab prompt (a double arrow) For instance, if you want to learn about how the “sin” function, type:

`>> help sin`

and a help message on the sin function will be displayed. This is useful if, for instance, you forget if the inverse sin function is called invsin, arcsin, or asin; just type “help invsin” etc. for each possibility until you get help. Or, if you are smarter, just type “help sin” and find its inverse in the “see also” section.

Matlab As A Scientific Calculator

Matlab can be used as a standard scientific calculator. From the command prompt, try

```
4+10   add
4-10   subtract
4/10   divide
4*10   multiply
sqrt(4) square root
3^2    3 raised to the 2nd power
pi     same as typing 3.14159265...
sin(pi/2) sine of an argument given in radians. Also, cos, tan, sec, csc, etc.
sind(90) sine of an argument given in degrees. Also, cosd, tand.
asin(1) arc sin, returned in radians. Also acos, atan...
asind(1) arc sin, returned in degrees.
expl(2) natural exponent, i.e. e^x, or 2.1783^x. Same as Exp() on a TI-89.
log10(100) same as Log() on a TI-89; returns 2 for this example since 10^2 = 100.
log(100) same as ln() on a TI-89; returns 4.602 for this example since e^{4.602} = 100.
```

For the big and small numbers common in electrical engineering, Matlab supports a special syntax

```
3e9    same as 3x10^9 = 3,000,000,000. An engineer would write this simply as “3G” where G is “Giga”.
0.6e-6 same as 0.6x10^-6 = 0.000 000 6. An engineer would write this simply as “0.6u” where u is “micro”.
```

Like any good scientific calculator, Matlab can handle imaginary numbers with ease.

```
(4+2i)+(5+i) returns 9+3i
exp(pi*i) returns -1, Euler’s famous identity
```

Matlab can also associate a number with a name

```
x = 5 assign 5 to variable x
y = 12 assign 12 to variable y
sqrt(x^2+y^2) returns 13
d = sqrt(x^2+y^2) assigns 13 to the variable d
```

Matlab’s most recent computational result is stored as a variable called “ans” (short for “answer”). So you could type 3+4, and then ans**2 to compute (3+4)**2 = 14. Of course, you could also just use parenthesis and type (3+4)**2 directly.

Make a typing mistake? I do frequently. Press the up arrow on your keyboard to recall past commands.
Matlab as a Matrix Calculator

Matlab can work with groups of numbers arranged in rows and columns, called matricies. The simplest kind of matrix is made of a single row of numbers and is called a row matrix, or simply a vector.

Matrices can be made up of real or complex numbers. You can create a matrix in several ways.

One way is to explicitly name each component:

\[
\begin{bmatrix}
1 & 2 & 4 & -2i
\end{bmatrix}
\]

create a row matrix (vector) with the numbers 1, 2, 4, -2, \(\sqrt{-1}\). Notice columns are separated with a space.

\[
x = [0 1 \frac{1}{2} 0]
\]
create a vector and call it “x”

\[
x = [1 2; 3 4]
\]
create a 2 row x 2 column matrix. The top row has the numbers 1,2. The bottom row has the numbers 3, 4. Notice rows are separated with a semicolon.

You might write the matrix by hand like this:

\[
\begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix}
\]

You can also create a vector by using the : (the colon) operator to evenly-space numbers

\[
x = 0:5
\]
same as typing \(x=[0 1 2 3 4 5]\)

\[
z = 0:2:10
\]
steps by 2, so same as \(z = [0 2 4 6 8 10]\)

\[
jim1 = [0:100; 100:200]
\]
a matrix called jim1 with 2 rows and 100 columns

\[
w = 0:0.001:1000
\]
a vector with 1,000,001 numbers, from 0 to 1,000 in 0.001 increments

Note that in the last case, 1,000,001 numbers will scroll around your screen, which will take a long time and annoy you in the process. Use asemicolon at the end to tell Matlab not to print the answer.

\[
w = 0:0.001:1000;
\]
suppresses the output

There are also some commands that make special vectors or matricies.

You can use linspace to make a vector with evenly-spaced numbers from some start number to some end number.

\[
z = linspace(0,1,11)
\]
11 numbers from 0 to 1, same as \(z=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]\)

A random matrix or vector with \(N\) rows and \(M\) columns of random numbers between 0 and 1:

\[
\text{rand}(N,M)
\]
So, if you want a matrix of 4 rows and 5 columns of numbers between -10 and +10, \(\text{rand}(4,5)*20\-10\)

Other common matrix-forming commands:

\[
\text{eye}(6)
\]
a square matrix that is all zeros except for ones down its diagonal

\[
\text{zeros}(4,7)
\]
a 4 row by 7 column matrix filled with zeros

\[
\text{ones}(5,5)
\]
a 5 row by 5 column square matrix filled with ones

for example, \(7*\text{ones}(2,3)\) gives a 2 row by 3 column matrix filled with 7’s

Sometimes you will want to interchange rows and columns; this operation is called transposing a matrix and is done with the transpose operator, the apostrophe (‘). So,

\[
[1 2 3]' \text{ is the same as entering } [1; 2; 3] \text{ and represents the column vector }
\]
\[
\begin{bmatrix}
1 \\
2 \\
3
\end{bmatrix}
\]

Lastly, you can change specific values in a matrix using parenthesis. For instance, say \(A = [1 2;3 4]\);

To change the number in row 2 column 1 from 3 to 12, say \(A(2,1) = 12;\)

A Word on Matrix Mathematics

This section isn’t really about Matlab but rather about linear algebra, which deals with algebra on matrices. You should know this from your college algebra class; if not, you will learn it in bits and pieces as you progress through the ECE curriculum. You can add, subtract, multiply, and divide a matrix (and therefore a vector) with a scalar (a scalar is what we call a single number).

\[
x = [0 1 2]*2+1
\]
now \(x\) holds \([0*2+1 1*2+1 2*2+1] = [1 3 5]\)

In Matlab you can also take the sin, cos, log, power, etc. of a matrix; it will return a matrix of the same size

\[
sin([1 2 3]) \text{ returns a vector whose values are } [\sin(1) \sin(2) \sin(3)]
\]

There are 2 common meanings when we say we want to multiply a matrix by a matrix: we either want to multiply matrices using the laws of linear algebra (in which the number of columns of the first matrix must equal the number of rows in the second), or you want to multiply corresponding elements in two matrices of the same size.
The former is accomplished with the * operator, just as if you were multiplying two scalars together. The latter is accomplished with the .* operator (a period before the asterisk). So

\[
\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \ast \begin{bmatrix} 4 & 5 & 6 \end{bmatrix} = [4 \ 10 \ 18]
\]

(note the transpose operator) and \[ [1 \ 2 \ 3] \ast [4 \ 5 \ 6] \] is an error since the column/row relationship for matrix multiplication is not satisfied.

You’ll notice I haven’t mentioned division. In linear algebra, instead of saying \[ b \div A \] we usually say \[ A^{-1} b \]. This way we don’t have to define a “divided by” operation, just the inverse operation. In Matlab, inverse is called \[ \text{inv}() \]. So, \[ \text{inv}(4) = \frac{1}{4} \] since their product is 1, and, \[ \text{inv}([1 \ 0; -1 \ 1]) = [1 \ 0; -1 \ 0] \] since their product is the identity matrix \( [1 \ 0; 0 \ 1] = \text{eye}(2) \).

As a general rule, if Matlab gives you errors when working with vectors or matrices, especially if division is involved, try using ./ instead of / since you probably mean element-by-element division.

Here’s a useful example; solve:

\[
\begin{align*}
&x + 2y + z = 5 \\
&3x - y + 2z = 2 \\
&-x + y - 2z = -4 \\
\end{align*}
\]

\[
A = \begin{bmatrix} 1 & 2 & 1; -2 & -1 & 1 \end{bmatrix}; \text{ these are the coefficients to the left of the equals sign}
\]

\[
b = \begin{bmatrix} 5; 2; -4 \end{bmatrix}; \text{ these are the numbers to the right of the equals sign}
\]

\[
\text{inv}(A) \ast b \text{ gives you the solution, which is}
\]

\[
x = -1, y = 1.4, z = 3.2 \text{. Very fast! Matlab can solve a set of equations with 100 unknowns in less time than it takes to press the Enter key.}
\]

**Working With Matlab’s Windows**

In the upper left corner of the Matlab screen is a small window with tabs for “Current Directory” and “Workspace”. In the “Workspace” tab, all the currently-defined variables are listed. So, for example, if you type \( x = 4 \) you’ll see \( x \) listed in that window. When you close down Matlab, all the variables will be erased. If you want to erase some of them sooner (maybe you are running out of memory) you can type \( \text{clear} \)

and your \( x \) variable will disappear.

You can save your variables from one Matlab session to another by typing \[ \text{save} \]

which will save all variables to the file \( \text{MyVariables} \).mat in the current directory.
The variables can be reloaded in a future session by typing \[ \text{load} \]

The simplest way to save your work is to cut and paste command text into Word. To cut and paste a plot from a figure window, choose “copy figure” from the figure menu and then paste into Word.

**Plotting functions**

One of the most useful functions for Matlab is to plot data. For instance, to plot a triangle with vertices at \( \{0,0\}, \{1,0\}, \{\frac{1}{2}, 1\} \) you would type

\[
x = [0 \ 1 \ 0.5 \ 0];
\]

\[
y = [0 \ 0 \ 1 \ 0]; \text{ and finally}
\]

\[
\text{plot}(x,y)
\]

To learn more, type “help plot”.

To make your plot more readable, you can annotate it with the following commands:

- \[ \text{title} \quad \text{Graph title. E.g. title(’My first plot’)} \]
- \[ \text{xlabel} \quad \text{X-axis label. E.g. xlabel(’Frequency in Hz’)} \]
- \[ \text{ylabel} \quad \text{Y-axis label. E.g. ylabel(’Voltage’)} \]
- \[ \text{gtext} \quad \text{Mouse placement of text. E.g. gtext(’peak’) then click mouse to place.} \]

So, to plot an ellipse with a major horizontal axis of 5 and a minor vertical axis of 2,

\[
t = \text{inspace}(0, \pi, 100); \text{ creates 100 points evenly spaced from 0 to } \pi.
\]

\[
x = 5 \ast \cos(t); \ y = 2 \ast \sin(t); \text{ create the data points from the parametric definition of an ellipse.}
\]

\[
\text{plot}(x,y); \text{title(’My ellipse’); xlabel(’the x axis’); ylabel(’the y axis’)}
\]

Page 3