Getting started with MATLAB

To get MATLAB, you can go:

`https://terpware.umd.edu/Windows/List/232`

There should be installation instructions on the website.

After opening MATLAB, one can see the "Command Window" where you can enter basic stuff in the line $fx >>$. You can type in some basic math operations and hit ENTER to get the output. For example, typing in 3/4 and hitting ENTER will yield 0.7500.

Of course, there may be times when you want to do multiple operations and stuff at once, where it will be annoying to have to input one thing at a time and hit ENTER each time. So instead, we look above at the EDITOR window. Create a $.m$ file here and save it on your computer (e.g. assignment1.m). Now this script allows you to enter multiple things until you choose to RUN it. For instance, one can put it $3 + 4$, hit ENTER, $4 + 5$, hit ENTER (hitting enter is just creating a new line of code that will be compiled later). Nothing is evaluated here. Now up at the top, click on the RUN button. The outputs of both the sums should be in the Command Window.

Here are some very basic tips when typing in code in the EDITOR window.

1. If you put a semicolon (;) after a line of code, it will suppress the output. For example, $3 + 4$; would not output 7 (try it!).

2. If you put a percent sign (%), the following line of code will be treated as text (commenting; this can occur when doing multiple step computations where you may have to remind yourself that this equation found volume with disc method, another equation was using shell method, etc)

3. If you put double percent signs (%%), followed by a space, then text (Problem X) and then hit ENTER, it will separate the script into sections. **You MUST do this to separate different problems and parts in the assignment. After the double percent signs, put another percent sign, followed by "Problem X", to distinguish what problem you are doing in each section.**

4. Asterisks ($*$) must be used for products on numbers/functions i.e. if you put (2)(3), MATLAB will not know you want to multiply. Similarly, if you put $2x$, it will not know you are multiplying x by 2. You have to input as $2 * x$.

5. $\sin(3), \cos(5), \pi, e^3$ are inputted as

```
sin(3)
cos(5)
```

```
pi
exp(3)
```

6. Matlab does operations often in terms of vectors, **which must be in bracket form**. For example, you can define a vector by typing in the following (**you can also put commas between entries**)

```
a=[4  2  5];
b=[6,  3,  1];
a+b
dot(a,b)
cross(a,b)
norm(a)
```

   will output the sum, dot product, and cross product of the two vectors, and the magnitude of the first vector (magnitude/length is given by *norm*). Notice the semicolons for the first two lines are suppressed so they don't get outputted.

7. MATLAB has values approximated automatically (e.g. π), but one can make them "exact" with a symbolic representation. This is also used to define variables. For example, when entering pi, the output is 3.1416. To keep keep it as a symbol, we use *sym* or *syms* (the latter can always be used, and is more handy for multiple things you define as symbolic)

```
syms pi;
3*pi
```

   will yield 3*pi, treating pi as a symbol instead of outputting 9.4248. **To remove this symbolic definition (in general for any terms you make symbolic)**, one uses 'clear all'

```
syms pi;
3*pi
clear all
3*pi
```

   will output 3*pi and then 9.4248 after interpreting 'clear all'

8. We use the *syms* command to define variables, allowing us to create functions, differentiate, and integrate. We declare functions in this manner.

```
syms f(x);
f(x) = x+3+sin(x);
diff(f(x),x)
int(f(x))
int(f(x),0,pi)
```

will create the function $f(x) = x + 3 + \sin(x)$, differentiate it with respect to x, find the indefinite integral of $f(x)$, and find the definite integral from $x = 0$ to $x = \pi$. Note that

```
diff(f(x),3)
```

would take the third derivative with respect to x. The x from line 3 above isn't required, but is useful in the future when taking partial derivatives.

Typing *clear all* will remove all the defined functions and those previously used notation for functions can be associated to new ones.

The following are basic commands for matrices.

9. We can create a matrix using brackets. A semicolon will begin a new row. For example,

```
A=[1 2 3; 4 5 6; 7 8 9]
```

will output $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$.

To determine a specific entry, say in row 2 column 3, input

```
A(2,3)
```

which will ouput 6.

To determine a specific row, say row 2, we use a colon (:), input

```
A(2,:)
```

which will output

$$4 \quad 5 \quad 6$$

To determine a specific column, say column 3, the idea is similar, input

```
A(:,3)
```

which will output

$$3$$
$$6$$
$$9$$

To interchange rows, say swap rows 2 and 3, input

```
A([2 3],:) = A([3 2],:)
```

which will output $\begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \end{bmatrix}$. **This will be defined as the new matrix A now!**

10. We can **apply row operations** to update the matrix. So suppose we first define a matrix A as follows

```
A=[1 2 3; 4 5 6; 7 8 9]
```

If we want to get in RREF, we want to multiply row 1 by -4 and add it to row 2 i.e. -4R1+R2. We input

```
A(2,:)=-4*A(1,:)+A(2,:)
```

Now the matrix A is updated with this elementary row operation i.e. if you type in A in the script following this, it outputs $\begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 7 & 8 & 9 \end{bmatrix}$.

11. Applying row operations can still be super tedious. Instead, Matlab can determine the **RREF of a matrix immediately** using the `rref` command. For example, if we input

```
A=[1 2 3; 4 5 6; 7 8 9]; % suppressing the matrix output
rref(A)
```

will output $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix}$.

12. To **extract columns** from a matrix, if $A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$,

```
A=[1 2 3 4; 5 6 7 8; 9 10 11 12];
A(:,[1 3 4])
A(:,2:4)
```

will extract columns 1,3,4, and columns 2 through 4, respectively. So the outputs would be $\begin{bmatrix} 1 & 3 & 4 \\ 5 & 7 & 8 \\ 9 & 11 & 12 \end{bmatrix}$ and $\begin{bmatrix} 2 & 3 & 4 \\ 6 & 7 & 8 \\ 10 & 11 & 12 \end{bmatrix}$

13. Matrix operations in Matlab are done in a similar manner as with numbers. **Make sure to include asterisk (*) when multiplying matrices**. So

```
A+B
A*B
```

```
A^4
4*A
```

will add, multiply, take the 4th power of A, and scale the matrix A by 4, respectively.

14. The **rank** of a matrix A can be computed by the command `rank(A)`.

15. The **transpose** of a real matrix A can be computed with either notation:

```
A=[1 2 3; 4 5 6]
A' %this finds the transpose
transpose(A) %this also finds the transpose
```

The `A'` actually computes the **conjugate transpose** or **adjoint** of A. It will take the complex conjugate of all entries of A^T (but there is obviously no effect if all entries are real).

16. The **eigenvalues** of a square matrix A can be computed by the command `eig(A)`.

17. More efficiently, if we can **diagonalize** the matrix as $A = PDP^{-1}$ (where D is the diagonal matrix whose diagonal are the eigenvalues of A, and the columns of P are the corresponding eigenvectors),then the matrix D and P can be obtained by the command `[P,D]=eig(A)`.