MATLAB Project 1

Introduction to MATLAB

1 Introduction

Linear algebra is used in every branch of mathematics and is thus a crucial part of it. Mathematics, however, is not the only place linear algebra can be found; linear algebra has applications in practically every field of natural and social sciences. Physics, chemistry, biology, computer science, economics, and sociology have all benefited from the advent of linear algebra. Through the following 5 MATLAB projects you will see a few applications of linear algebra to some of these areas.
The problems that arise in these fields involve raw data, and working with it by hand, without the aid of computers, would be a tedious and painstaking task. Computer programs such as MATLAB help in performing calculations error-free, so that you can spend less time crunching numbers and more time working on the big picture. It turns out that MATLAB is especially well suited for working with matrices and performing various algorithmic routines that come up in linear algebra. In this initial project we will start to get acquainted with MATLAB and learn how it handles matrices.

2 Obligatory Tasks

Obviously, before you can begin this project you will need to run MATLAB, so be sure to do that now (the program may be hidden in a folder titled "MathWorks", since this is the publisher of MATLAB). Additionally, you will need to open your favorite word processor (like Microsoft Word) so that you can complete the Exercises listed in this project. Be sure to include your full name at the top of your document!

You are welcome to submit your project via email; however, if you choose to do so, you **MUST** use the following convention for the filename:

`Lastname_Firstname_MATLAB_Project1.docx`

(or whatever file extension you chose to use).

Remark: Some Exercises will ask you to copy MATLAB output and paste it into your document to be submitted. There is no need to manually copy the output from one program to the other, simply use your favorite method of performing Copy/Paste on text (if you don't have a favorite method, or don't know how to Copy/Paste text, Google it).

3 Exercise Key

The several icons will appear next to each Exercise you are asked to complete. Use the below table to determine what information should be included in your submitted document to fully complete the exercise.

Icon:	Item:	Comments:
	Commands entered in MATLAB & resulting output	You should copy relevant input and output from MATLAB and paste it into your document. You need only include commands that worked.
	Plots & Graphs	Include all graphs generated in an exercise unless the problem specifically tells you which/how many to include.
	Full sentence response	Each exercise contains a question that you should use at least one or two complete sentences to answer. Even if you're stuck, write down any reasoning or ideas you've had.
	Requires work by hand	Do scratch work by hand. Leave space in your document and write your scratch work directly on the assignment to turn in.

4 Some Basics

The command prompt in MATLAB looks like `>>`. Simply enter your command in the command window next to the `>>` symbol and hit the Enter key – the command will then be executed.

MATLAB will ignore anything you enter after a % sign. In all of your projects there will often be comments explaining the commands you are asked to enter (you do note have to type these comments as you work through the exercises!).

If you ever wish to interrupt MATLAB because it is taking too long to execute the current command, press the Control key and the C key together (Ctrl-C).

MATLAB's Product Help is extremely detailed and can be very useful – you can access it from the menu bar. If there is something specific you would like to know about, for instance, the command `plot`, simply type

```
doc plot
```

at a command prompt.

When MATLAB is closed, all variables (more on these next) are lost. However, the command

```
save filename
```

writes all of the variables to the file `filename.mat`. At the start of your next session, the command

```
load filename
```

restores all the variables that were set in the previous session.

5 Variables in MATLAB

People like to think of MATLAB as a very powerful calculator you can use on your computer. For our purpose, that is not far from the mark.

To define variables, simply type in the variable you wish to define, then an equal sign, and finally the value you wish to assign to that variable. So, if we want to assign the value of 5 to the variable x, we type

```
>> x = 5  %Now press Enter
 x =
     5
```

Unless we later redefine x, every time we type "`x`" MATLAB will replace it with 5 when doing a calculation.

For example, we can define the following variables by their place in the alphabet:

```
>> l=12; i=9; n=14; e=5; a=1; r=18; g=7; b=2;
```

We have just assigned specific values to the given letters. (Note: the semicolons in this command suppress the output. If they were not there, once you hit Enter MATLAB would list the eight variables and their new values.) It is also possible to assign values to more than just single letters using MATLAB. It is sometimes easier to define something with a specific name rather than a single letter. For example, we can define

```
>> LinearAlg=313

LinearAlg =
     313
```

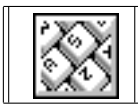
since our course is numbered as Math 313. Notice that there are no spaces in variable names. (We cannot define a variable called "`Linear Alg`".) Also, MATLAB is case-sensitive, so if we accidentally type "`linearalg`" and press Enter, we will receive an error message:

```
>> linearalg
     ??? Undefined function or variable 'linearalg'.
```

Note, whenever you see this "`Undefined function or variable`" error, it most likely means you made a typo when entering your command.

Using the variables we have previously defined, we can do some calculations and save them as specific names:

```
>> linearalgebra = l+i+n+e+a+r+a+l+g+e+b+r+a

linearalgebra =
     105

>> ringingbell = r+i+n+g+i+n+g+b+e+l+l

ringingbell =
     109

>> largebill = l+a+r+g+e+b+i+l+l

largebill =
     78
```



Exercise 1. Define the letters of your first and last name to be the number corresponding to their place in the alphabet. Then set your name, without any spaces, equal to the sum of the letters.

Note, if we perform a computation without assigning a variable name to it, MATLAB automatically assigns the variable `ans` to the output. The value of `ans` will change every time you perform a new computation without an assigned name:

```
>> 3-2^4

ans =
     -13

>> ans*5

ans =
     -65
```

One should also be aware that a $*$ must be used to perform multiplication involving variables:

```
>> two=2

two =

     2
```

```
>> two*3

ans =

     6

>> two3

Undefined function or variable 'two3'.
```

Without the $*$ MATLAB things `two3` is some new variable that it doesn't know about!

Finally, we can erase variables from MATLAB's memory by using the `clear` command:

```
>> clear  %Clears ALL varibles from MATLAB's memory

>> clear VariableName  %Clears ONLY the variable ``VariableName'' from memory
```

6 A Useful Trick

You will want to become very familiar with the use of the up-arrow key. When your cursor is at a command prompt, tapping the up-arrow key will cycle through your previously entered commands (you can then use your left- and right-arrow keys to position the cursor in the command in order to edit it). This is a great and very efficient way to correct a command that had an error in it.

Exercise 2. Suppose we want to evaluate

$$25 - \left(100 - 6e^{\left(5+\cos\left(\frac{\pi}{3}\right)\right)}\right).$$

Enter the command

```
>> z = 25-(100-7exp(5+cos(pi/3))
```

You will receive an error message. Use the up-arrow key to bring this line back and then correct the error(s). Include the wrong command, the error message, and your fix in your homework. (Hint: Good things to check are multiplication (you must use *) and parenthesis.)

7 Symbolic Computations

In the grand scheme of things MATLAB is meant to be used for large numeric computations rather than for manipulating symbols, e.g., factoring a polynomial (like programs such as Maple or Mathematica are designed to do). However, this does not mean that MATLAB isn't capable of doing so.

As one example, MATLAB can solve equations. The command `solve('eqn','var')` solves the equation `eqn` with respect to the variable `var`. Note, we must apostrophes to enclose these equations and variables – these apostrophes tell MATLAB that the expressions are *symbolic* and not *numeric*.

```
>> solve('x^2-2*x+10','x')

ans =
     1 + 3*i
     1 - 3*i
```

```
>> solve('x^2=a^2','x')     %MATLAB can solve equations in terms of parameters too!

ans =
     a
     -a

>> f = solve('a*x^2+b*x+c','x')     %The solution to this should look familiar

f =
     -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
     -(b - (b^2 - 4*a*c)^(1/2))/(2*a)

>> pretty(f)     %The pretty() command displays the answer in a nicer format
/             2          \
|   b + sqrt(b  - 4 a c) |
| - -------------------- |
|            2 a         |
|                        |
|             2          |
|   b - sqrt(b  - 4 a c) |
| - -------------------- |
\            2 a         /
```

Special Note: Depending on your installed version, MATLAB may or may not return an enlightening solution when you use the command `solve()`. MATLAB has a built in setting called "MaxDegree" which stipulates the maximum degree of polynomials for which the solver tries to return explicit solutions. (In MATLAB version 2015a MaxDegree=2 by default – this means only quadratic equations will be automatically solved). To remedy this, use the following syntax: `solve(eqn,var,'MaxDegree',degree)` where `degree` is the degree of the polynomial you are trying to solve. For instance:

```
>> solve('x^4-2*x+1','x')     %Doesn't give a good answer

ans =
     1
     RootOf(z^3 + z^2 + z - 1, z)[1]
     RootOf(z^3 + z^2 + z - 1, z)[2]
     RootOf(z^3 + z^2 + z - 1, z)[3]

>> solve('x^4-2*x+1','x','MaxDegree',4)    %A messy, but exact, answer!

ans =
 1
 ((11^(1/2)*27^(1/2))/27 + 17/27)^(1/3) - 2/(9*((11^(1/2)*27^(1/2))/27 + 17/27)^(1/3)) - 1/3
 1/(9*((11^(1/2)*27^(1/2))/27 + 17/27)^(1/3)) - (3^(1/2)*(2/(9*((11^(1/2)*27^(1/2))/27
     + 17/27)^(1/3)) + ((11^(1/2)*27^(1/2))/27 + 17/27)^(1/3))*1i)/2 - ((11^(1/2)*27^(1/2))/27
     + 17/27)^(1/3)/2 - 1/3
 (3^(1/2)*(2/(9*((11^(1/2)*27^(1/2))/27
     + 17/27)^(1/3)) + ((11^(1/2)*27^(1/2))/27 + 17/27)^(1/3))*1i)/2
     + 1/(9*((11^(1/2)*27^(1/2))/27 + 17/27)^(1/3)) - ((11^(1/2)*27^(1/2))/27
     + 17/27)^(1/3)/2 - 1/3
```

```
>> pretty(ans)     %An nicer way to see the answer
/          1         \
|                     |
|           2    1    |
|    #2 - ---- - -    |
|         9 #2   3    |
|                     |
|   1         #2   1  |
| ---- - #1 - -- - -  |
| 9 #2         2   3  |
|                     |
|        1    #2   1  |
| #1 + ---- - -- - -  |
\      9 #2    2   3 /
```

where

$$\#1 == \frac{\left(\frac{2}{9\,\#2} + \#2\right)^{\sqrt{3}}\,1i}{2}$$

$$\#2 == \left(\frac{\sqrt{11}\,\sqrt{27}}{27} + \frac{17}{27}\right)^{1/3}$$

Exercise 3. Find the solutions to the following equation:

$$x^3 - 2x + 5 = 0.$$

Make sure your solutions are displayed in a "nice" way!

While it's convenient to use apostrophes for one or two commands, if you have numerous commands all involving symbolic expressions it is easier to simply tell MATLAB to treat specific variables as symbols for the remainder of your computations. This is accomplished through the `syms` command:

```
>> syms x y  %This tells MATLAB to treat both x and y as symbols
```

Now if you execute the command

```
>> a=(x+y)^3

a =
     (x + y)^3

>> expand(a)

ans =
     x^3 + 3*x^2*y + 3*x*y^2 + y^3

>> factor(x^3 + 3*x^2*y + 3*x*y^2 + y^3)

ans =
     (x+y)^3
```

MATLAB is not limited to only symbolic algebraic manipulations, it can also perform operations such as differentiation and integration:

```
>> diff(sin(x))

ans =
     cos(x)

>> diff(sin(x+3*y))

ans =
     cos(x + 3*y)
```

Note here that MATLAB by default takes the derivative with respect to x. If you wanted to take the derivative with respect to y, we would type

```
>> diff(sin(x+3*y),'y')
```

Exercise 4. Compute the derivative of $\log(\cos(t) - 3s)$ with respect to t and then with respect to s. Include the input and output in your document.

8 Matrices in MATLAB

To define a matrix using MATLAB, we use the square brackets"[ ]". "[" tells MATLAB we are starting to create a matrix and "]" tells MATLAB that we have finished with our construction. The entries of the matrix should be entered in rows from left to right. To separate entries we can either hit the space bar or insert a comma ",". Once we finish with a given row and want to move to the next, we can either press the return key or use a semicolon ";". (This is not the same as using the semicolon to suppress output.)

So, to construct the following matrix

$$A = \begin{pmatrix} 2 & 1 \\ 4 & 3 \end{pmatrix}$$

there are possible ways to go about it:

```
>> A=[2 1;4 3]

>> A=[2,1  %Press Enter

4,3]
```

Both will yield the desired matrix:

```
A =
     2     1
     4     3
```

Exercise 5. Input the following matrix into MATLAB:

$$\text{Fibonacci} = \begin{pmatrix} 1 & 1 & 2 & 3 \\ 5 & 8 & 13 & 21 \\ 34 & 55 & 89 & 144 \\ 233 & 377 & 610 & 987 \end{pmatrix}$$

9 Matrix Operations and Manipulation

Sometimes we want to take a matrix that we defined and have MATLAB tell us information about it. For example, maybe we want to know what number is in the 3rd row and 4th column; or maybe we want to view the whole 5th row. This is done with regular parentheses, "(" and ")". For example, to see the (1,2) entry of the matrix A above (i.e. the entry in the first row and second column), we use the command

```
>> A(1,2)
```

We can also use the colon ":" to mean 'all', as in the command:

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

which gives us the entire second row of the matrix A. The colon can also be used to represent a range of rows or columns. The command:

```
>> Fibonacci(2:4,1)
```

will give us the entries of `Fibonacci` from the second through fourth rows in the first column.

Exercise 6. Using the commands introduced above, construct a new matrix (you may name it whatever you woudl like) from `Fibonacci` that consists of the *last two rows* and the *middle two columns* of `Fibonacci` (Your matrix should end up being a 2×2 matrix). Be sure to include the command you used and the resulting output in your document!