Math 151A Week 1

These notes are partially styled after the excellent Matlab tutorial written by Tobin A. Driscoll.

A brief introduction to Matlab

The default layout of the Matlab window is divided into five main sections. The top bar contains various menus. The left side of the window displays your **current directory**. Matlab can only access files in the current directory and on your **path**. If you try to run a command that refers to a file that is not in either of these places, you will get a prompt to add the necessary file to your path. The right side of the window displays your **workspace**, which holds all variables currently defined in your Matlab session. If you have a Matlab file open, it will by default appear in the middle of your screen. Finally, the **command window** is at the bottom of the screen. Here, you can run snippets of Matlab code, inspect variables, and more.

Matlab basics

You can carry out basic arithmetic in the command line by typing out an expression and pressing "enter".

```
3+(2*4)^2
```

```
ans = 67
```

The output is stored in a variable called "ans" unless you specify otherwise.

```
x = pi*2;
```

To suppress output on the command line, end the line with a ";":

```
y = exp(2);
```

To print formatted text, you can use "fprintf":

```
fprintf("y = %f", y)
```

```
y = 7.389056
```

Matrix operations

One of the main strengths of Matlab is matrix operations. There are two main ways to construct a matrix: direct construction, and calling functions.

To construct a matrix directly, use square brackets to denote the beginning and end of the matrix, and semicolons to separate rows. Entries on a given row can either be separated by a space or by a comma.

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

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

```
v = [5; 6]
```

```
v = 2×1
     5
     6
```

```
b = A*v
```

```
b = 2×1
    17
    39
```

Matlab has many matrix operations built in. Here are some of them:

```
At = A'
```

```
At = 2×2
     1     3
     2     4
```

```
exp(A)
```

```
ans = 2×2
    2.7183    7.3891
   20.0855   54.5982
```

To perform entrywise operations, add a "." before the operator:

```
v = [1 2 3];
v .^ 2
```

```
ans = 1×3
     1     4     9
```

Matlab is a 1-indexed language, and indexing is performed using parentheses ()

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

```
ans = 2
```

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

```
ans = 6
```

You can access ranges/slices as well; in Matlab, endpoints are inclusive.

```
x(3:5)
```

```
ans = 1×3
     3     4     5
```

```
A(2:3, 1:3)
```

```
ans = 2×3
     4     5     6
     7     8     9
```

You can access the last element along a dimension using "end":

```
x(end)
```

```
ans = 5
```

You can append values to a vector using "end+1":

```
x(end+1) = 6;
size(x)
```

```
ans = 1×2
     1     6
```

```
x(6)
```

```
ans = 6
```

Special matrix/vector constructors

Matlab has several useful functions for constructing common matrices and vectors.

eye(n) returns the n by n identity matrix:

```
eye(5)
```

```
ans = 5×5
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

zeros(n) or ones(n) returns the all-zeros or all-ones n by n matrix. Alternatively, zeroes(m, n) or ones(m, n) returns the all-zeros or all-ones m by n matrix.

```
zeros(2)
```

```
ans = 2×2
     0     0
     0     0
```

```
ones(3, 4)
```

```
ans = 3×4
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

The linspace() function returns a vector with linearly spaced elements:

```
v = linspace(1, 5, 5)
```

```
v = 1×5
     1     2     3     4     5
```

Alternatively, you can use the v = start:step:stop syntax:

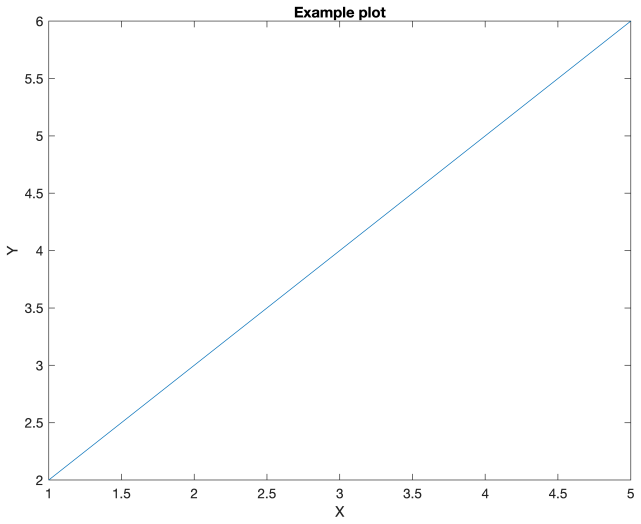
```
v = 1:1:5
```

```
v = 1×5
     1     2     3     4     5
```

Plotting functions

To make a new figure, use `figure` followed by a plotting function. The simplest plotting function is plot(X, Y), which plots the values in Y against the values in X. Specific functionality can vary depending on the dimensions of X and Y.

```
X = 1:1:5;
Y = 2:1:6;
figure;
plot(X, Y)
xlabel('X')
ylabel('Y')
title('Example plot')
```



Always label your axes!