EXERCISES

16.01/16.02: Introduction to MATLAB Tutorials

I. Class Materials

1. Download UEmatlab.tar or UEmatlab.zip

From a web browser:

Download the **UEmatlab.tar** from **http://web.mit.edu/acmath/matlab/unified** to a local directory. On Windows, if you do not have WinZip, download **UEmatlab.zip** instead.

Alternatively, on Athena:

```
athena% add acmath
athena% cp  /mit/acmath/matlab/unified/UEmatlab.tar .
```

2. Extract this session's sub-directories and files

On laptops:

Use your computer's utilities, such as double click, WinZip on Windows, or StuffIt on Mac.

Alternatively, on Athena *(or the UNIX shell of Mac OS X)**:***

```
tar –xvf UEmatlab.tar
```

Your local work directory **UEmatlab** should now contain the following directories and files:

UEmatlab

Exercise1	**Exercise2**
MatrixMath.m	veloctiyprogram.m
	orbitalvelocity.m

II. Start MATLAB

On Athena:

```
athena% cd UEmatlab
athena% add matlab
athena% matlab &
>> desktop
```

On laptops:

Launch MATLAB and navigate to the work directory **UEmatlab**.

IV. Exercise 1: Linear Algebra

Purpose

To practice the following in MATLAB:

- Using vectors and matrices to solve a system of linear equations.
- Fitting a polynomial equation through a set of points.
- Applying algebraic operators *, ^, / and \ to matrices and vectors.
- Using matrix-specific built-in functions such as ones, diag and eig.
- Finding eigenvalues and eigenvectors of matrices.
- Plotting points and lines and annotating figures.

Background

A polynomial of the n[th] order is defined as: $y = c_1 x^n + ... + c_{n-1} x^2 + c_n x + c_{n+1}$

A system of linear equations is: $\begin{bmatrix} y_1 \\ y_2 \\ ... \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_1^n & ... & x_1 & 1 \\ x_2^n & ... & x_2 & 1 \\ ... & ... & & \\ x_{n+1}^n & ... & x_{1+1} & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ ... \\ c_{n+1} \end{bmatrix}$ or in matrix form: Y = AC

Fit a cubic equation (a polynomial of the 3[rd] order i.e. $y=c_1x^3+ c_2x^2 +c_3x+c_4$) to four points by solving a determined system of linear equations (i. e. a system that has a unique solution).

Fit a quadratic equation (a polynomial of the 2[nd] order i.e. $y=b_1x^2 +b_2x+b_3$), and a straight line ($y=a_1x+a_2$) to the four points. Plot the four points, the cubic curve, the quadratic curve, and the straight line on the same plot.

1. Open the M-file in directory Exercise1 in the MATLAB Editor

```
>> edit MatrixMath.m
```

Lines that start with % are comments. The rest are **MATLAB commands**.

2. Follow instructions in MatrixMath.m

- Follow instructions in A1 – A3 to create vectors X and Y from the coordinates of four points, to create matrix A from X, and to solve Y=AC to find C.
- Follow instructions in B1 – B2 to compute the coefficients of a quadratic fit and a straight line fit to the four points, using function polyfit.
- Follow instructions in C1 – C2 to compute a cubic curve, a quadratic curve, and a straight line, and to plot them on the same graph with the points.
- Follow instructions in D1 to compute the eigenvectors and eigenvalues of the matrix A.
- After typing all commands in MatrixMath.m, save the M-file.

3. Execute the M-file in the Command Window

```
>> MatrixMath
```

Customize and save the graph that gets created in the Figure Window.

V. Exercise 2: Orbital Velocity Computation Program

Purpose

To practice the following in MATLAB:

- Creating new functions in function M-Files.
- Writing an interactive MATLAB program using script and function M-Files.
- Using functions such as input for program input from the Command Window.
- Writing if and switch statements for program flow control.
- Using relational (e.g. <=) and logical (e.g. ||) operators in a program.
- Working with strings and string-specific functions such as strcmp.

Background

The velocity V required for a rocket to move on a circular orbit at altitude H around a planet with mean radius R_e and gravitational constant g_0 can be computed with Johannes Kepler's formula[1]:

$$V = \sqrt{\frac{g_0 R_e^2}{R_e + h}}$$

Extend the interactive program, which takes input from the Command Line Window and computes the circular orbital velocity V. For Earth, Mars, and Moon, R_e and g_o are as follows:

	English Units		Metric Units	
	R_e [mi]	g_o [ft/sec^2]	R_e [km]	g_o [m/sec^2]
Earth	3963	32.2	6376	9.814
Moon	1079	5.30	1736	1.615
Mars	2111	12.1	3396	3.688

1. Open the M-files in directory Exercise2 in the MATLAB Editor

```
>> edit orbitalvelocity.m
>> edit velocityprogram.m
```

2. Read and understand the two M-Files

- The files form an interactive program that allows users to enter input in the Command Window at runtime. Read and understand the files before running the program.

[1] This example uses data from the web site http://exploration.grc.nasa.gov/education/rocket/rktrflght.html.

- velocityprogram.m is a **script M-File**
- orbitalvelocity.m is a **function M-file**, which defines a new function: orbitalvelocity.
  >> help orbitalvelocity
- Note the use of the built-in function input, which allows users to enter numerical or string input in the Command Window at runtime.
- Note the use of **relational operators** e.g. < ("less than") and **logical operators** e.g. ||.
- Note the construction of **if, elseif, else** statements and **switch, case** statements.
- Note the use of built-in functions for **strings**, e.g. strcmp.
- In the **script M-file**, note how arguments are passed to functions defined in other M-files.
- Write MATLAB code to add the Moon as an option for selection in the interactive program velocityprogram. Which file(s) do you need to modify?

3. Execute M-file velocityprogram.m in the Command Window

- Run velocityprogram and explain what happens in terms of specific command lines:
  >> velocityprogram
- Use the new function orbitalvelocity from the Command Window; for example:
  >> orbitalvelocity(1079, 5.3, 200, 'e')
- Run the program velocityprogram with the MATLAB Editor in **debugger** mode.