EXERCISES

Introduction to MATLAB: Linear Algebra and Calculus

I. Class Materials

1. Download LA_Calculus.tar or LA_Calculus.zip

From a web browser:

Open your browser and go to http://web.mit.edu/acmath/matlab/IntroMATLAB.
Download the file **LA_Calculus.tar** to a local work directory.
On Windows, if you do not have WinZip, download **LA_Calculus.zip** instead.

Alternatively, on Athena:

Copy the file **LA_Calculus.tar** from the locker **acmath** to a local work directory.

```
add acmath
cp /afs/athena.mit.edu/astaff/project/acmath/matlab/IntroMATLAB/LA_Calculus.tar .
```

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

(Alternatively, you can download, or copy from the locker, the files one by one.)

On Athena** (or the UNIX shell on Mac OS X)**:

```
tar –xvf LA_Calculus.tar
```

On laptops:

Use your computer's utilities, such as double click or WinZip on Windows or StuffIt on Mac.
If you download **LA_Calculus.zip** on Windows, double click on it and select File->Extract All.

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

LA_Calculus

Exercise_One
example1.m

Exercise_Two
example2.m

Exercise_Three
example3.m
ode.m

You may place and rename directories and files any way you wish. For consistency, we shall refer to the directory **LA_Calculus** as the work directory for these exercises.

II. Start MATLAB

On Athena:

Go to the work directory **LA_Calculus** using the cd command:
athena% cd LA_Calculus

Then launch MATLAB from that directory by typing at the Athena prompt:
athena% add matlab
athena% matlab &

Start the MATLAB desktop interface by typing at the MATLAB prompt:
>> desktop

On laptops:

Launch MATLAB the same way you launch any software on your laptop. Then navigate to the work directory **LA_Calculus** either from the Current Directory window, or by using the cd command in the Command Window.

III. Exercise 1: Linear Algebra

Purpose

To practice the following in MATLAB:

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

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

We shall fit a cubic equation (i.e. a polynomial of the 3^{rd} order or $y=c_1x^3+ c_2x^2 +c_3x+c_4$) to a set of four points; therefore we are solving a determined system that has a unique solution.

1. Open Exercise_One/example1.m in the MATLAB Editor

>> edit example1.m
Lines that start with % are comments.
The rest are **MATLAB commands**.

2. Try commands from example1.m in the Command Window

- Type the commands in the Command Window (or use Copy and Paste to copy them).
- Press Enter after commands and see the results in the Command Window.
- Note how matrices can be created from vectors; for example:
  `>> A = [X.^3  X.^2  X.^1  X.^0]`
  creates a 4x4 matrix from a 4x1 vector.
- Note the use of the ones function; for example:
  `>> A = [X.^3  X.^2  X.^1  ones(4,1)]`
  creates a 4x4 matrix from a 4x1 vector, and the last column contains only 1s.
- In the above two examples, note the use of the . operator to force element-wise (i.e. scalar) exponentiation for the elements of the X vector.
- In lines 26-29 of example3.m, note that you can enter the elements of a matrix in a tabular form, i.e. every row on a new line: as long as you are inside square brackets [ ], MATLAB understands a new line to be a new row in the matrix (i.e. no need for ;).
- Note how the ^, *, and \ operators can be applied to matrices; for example:
  `>> Ainv = A^(-1)`
  `>> C = Ainv * Y`
  `>> C = A \ Y`
  Can you guess what these operators do?
  A \ Y means "A divides Y" and is particularly useful for matrix math, although it can also be used for scalar or vector-scalar computations.
- Note how built-in functions act on matrices; for example:
  `>> R = rref(AY)`
  performs row reduction on a matrix called AY.
  `>> Lambda = diag(lambda)`
  creates a diagonal 4x4 matrix from the elements of a 4x1 column vector.
- Note the use of the eig function to find eigenvalues:
  `>> lambda = eig(A)`
  and to find both eigenvalues and eigenvectors:
  `>> [S, Lambda] = eig(A)`

3. Execute the M-file in the Command Window

All commands in the M-file can be executed by running the file from the Command Window:

`>> example1`

IV. Exercise 2: Curve Fitting

Purpose

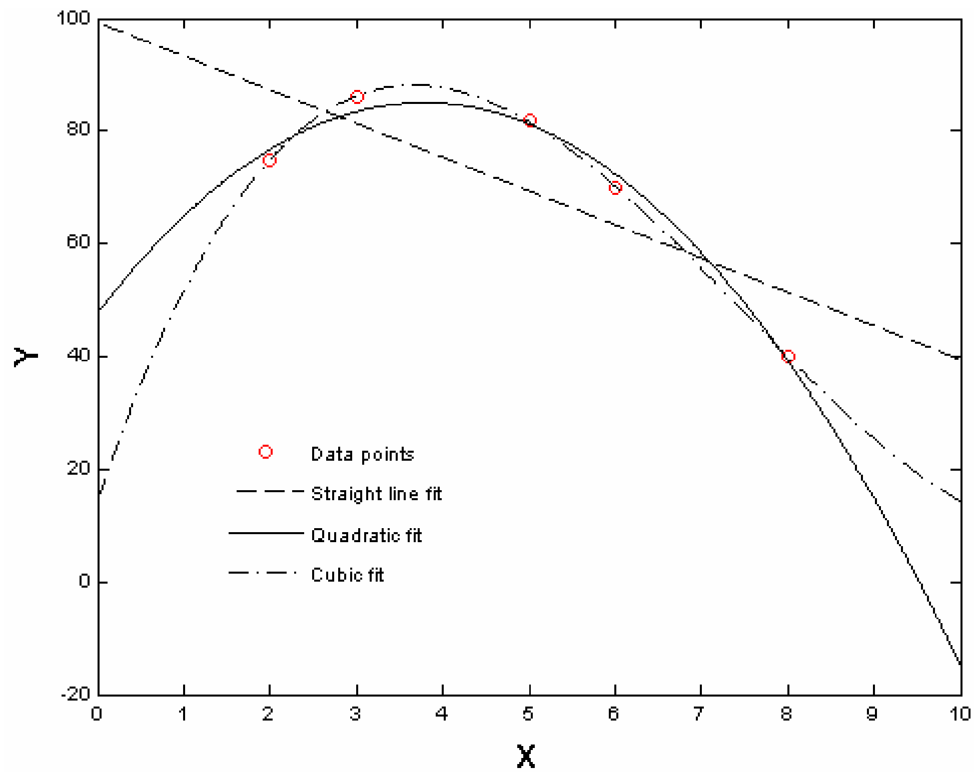
To practice the following in MATLAB:

- Fitting polynomials of a different order to a set of points.
- Using built-in functions such as polyfit and polyval for polynomials.

Background

In this example we shall fit a straight line, a quadratic, and a cubic to a set of five points (see figure); therefore this is an over-determined system. Also, we shall compute residuals (i.e. the difference between actual, y, and predicted values y_p) and compare them to determine which polynomial is the best fit. Residuals for five points are computed as follows:

$$res = \sum_{i=1}^{5} \left(y_{p,i} - y_i\right)^2$$



1. Open M-file Exercise_Two/example2.m

>> edit example2.m

2. Try commands from example2.m in the Command Window

- Type the commands in the Command Window (or use Copy and Paste to copy them).
- Press Enter after commands and see the results in the Command Window.
- Note again how matrices can be created from vectors, the use of the ones function, and the use of the . operator to force element-wise (scalar) operations; for example:
  >> A2 = [X.^2 X.^1 ones(5,1)]
- Note how the * and \ operators are applied to matrices; for example:
  >> C2 = A2 \ Y
  >> Y2 = A2 * C2
- Note the use of built-in functions for polynomials; for example:
  >> C2 = polyfit(X, Y, 2)
  >> Y2 = polyval(C2, X)
- At the end, you should be able to do a polynomial fit of any order. This requires only a one-line MATLAB command. Try fitting a 4th order polynomial to the same point set. To plot the curve, you need two more lines of MATLAB commands. Try that too.

3. Execute the M-file in the Command Window

```
>> example2
```

V. Exercise 3: RC Circuit

Purpose

To practice the following in MATLAB:

- Solving first order ordinary differential equations (ODE).
- Using built-in functions for differential equations such as ode45.
- Using **Function Handles** to pass functions as arguments to other functions.

To prepare for the Programming session by learning the following:

- Creating and using **function M-files**.
- Defining and using **global variables** and your own functions.

Background

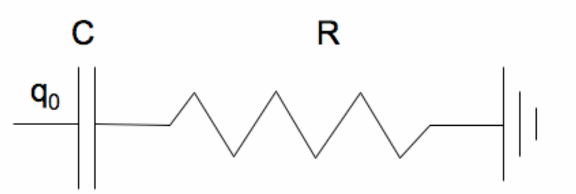
Ohm's law: V = IR, where V is voltage, I is current, and R is resistance.

In an RC Circuit (see figure), in which a capacitor with capacitance C and initial charge q_o drains through a resistor with resistance R, a 1st order ODE describes the capacitor's rate of discharge:

$$R\left(\frac{d}{dt}q(t)\right)+\left(\frac{q(t)}{C}\right)=0$$

Another way to write this is: $\frac{dq}{dt}=-\frac{1}{RC}q$

where q is the charge, and $\frac{dq}{dt}$ is its rate of change.



1. Open M-files Exercise_Three/example3.m and ode.m

```
>> edit example3.m
>> edit ode.m
```

2. Try commands from example3.m in the Command Window

- Type the commands in the Command Window (or use Copy and Paste to copy them).
- Press Enter after commands and see the results in the Command Window.
- Note that variables called from both files are defined as **global**:

```
>> global R C
```

- In ode.m, note how the first derivative dq/dt is represented as a variable Dq, which is computed with the function ode that takes two arguments t and q:
  >> function Dq = ode(t, q)
- In example3.m, note how the function ode is passed as an argument to the built-in function ode45, which is one of the MATLAB's solvers of differential equations:
  >> [t, q] = ode45(@ode, tspan, q0)
- In the above command, note also that the result from computing with ode45 is returned into two vectors t and q.
- Note the three arguments of ode45: the **Function Handle** @ode; the interval tspan over which to compute the numerical solution; and the initial condition q0.
- Note how ode is called to compute the current I (which is the first derivative of the charge q, i.e. dq/dt):
  >> I = ode(t, q)
- Note the use of a new graphics function semilogy to plot V and I on a semi-logarithmic scale (we shall do more exercises with 2d and 3d plotting in the session on Graphics).
- Note the use of functions such as xlabel, ylabel, title, and legend to create graphics annotations (more on that in the session on Graphics).

3. Execute the M-file in the Command Window

>> example3

4. MATLAB video tutorial

This example was based on the MATLAB Mastery Tutorial. You may watch a video tutorial on solving second order ODEs at: https://web.mit.edu/tm/matlab_mastery_I/setup/Start.htm (choose the module on Differential Equations).