Lab #1 - Overview of MATLAB

0. Initial activities (all files are located at ehs.sph.berkeley.edu/eisenberg)

1. Download the file, startup.m, into the directory c:\MATLAB6p5\toolbox\local.
   (right click on filename and select "save target as")
2. Create a the directory PH252B in c:\MATLAB6p5\
3. Download the file PH252B.mdl into c:\MATLAB6p5\PH252B
4. Open Matlab 6.5 application

I. Introduction to MATLAB.

The following section provides an overview of the basic syntax of MATLAB (Parts of this document are taken from the MATLAB demonstration file).

MATLAB is a high performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning.

The command window is the main window in which you communicate with the MATLAB interpreter. The MATLAB displays a prompt (>>) indicating that is ready to accept commands from you.

Before beginning this exercise type the following

```
>> diary L1.txt
```

The command diary causes a copy of all subsequent terminal input and most of the resulting output to be written on the named file. DIARY OFF suspends it. DIARY ON turns it back on. DIARY, by itself, toggles the diary state. By typing diary followed by the name of an already existing file, input will be appended to that file.

Using this diary command you will be able to document your work.

Vectors

Vectors are one-dimensional arrays of numbers. First, let's create a simple vector with 9 elements called 'a'. Enter the following text at the command prompt and press the **Enter** or the **Return** key

```
>> a = [1 2 3 4 6 4 3 4 5]
```

MATLAB will respond by displaying the new vector 'a'.

This is called a row vector. The transpose of this is called a column vector. To create a new variable 'b' that is the transpose of 'a' type

```
    >> b = a'
```

Now let's add 2 to each element of our vector, 'a', and store the result in a new vector called 'b'.

```
    >> b = a + 2
```

Notice how MATLAB requires no special handling of vector or matrix math.

The Colon Operator
We can create some vectors using the colon operator. The following creates a 10 element vector with values from 1 to 10

```
   » a = 1:10
```

To obtain nonunit spacing, specify an increment. For example,

```
   >> a  = 50:10:100
```

increments by 10 from 50 to 100

```
   >> a  = 100:-7:50
```

decrements by 7 from 100 to 50

You can reassign a portion of the vector to another vector.

```
   >> b = a(1:5)
```

Creating graphs in MATLAB is as easy as one command. Let's plot the result of our vector assignment b. The plot command will create a new window that contains the figure.

```
    >> plot(b)
    >> grid
```

MATLAB can make other graph types as well, with axis labels.

```
    >> bar(b)
    >> xlabel('Sample #')
    >> ylabel('Pounds')
```

MATLAB can use symbols in plots as well. Here is an example using *'s to mark the points. MATLAB offers a variety of other symbols and line types

```
    >> plot(b,'*')
```

You can also zoom in and out by using the axis command. The syntax is :

axis([Xmin, Xmax, Ymin, Ymax])

```
    >> axis([0 3 90 100])
```

For a listing of different symbols and line types

```
    >> help plot
```

Matrices

Creating a matrix is as easy as making a vector, using semicolons (;) to separate the rows of a matrix.

```
   >> A = [1 2 0; 2 5 -1; 4 10 -1]
```

Note: It is convention to use a capital letter to represent a matrix and a lower case letter to represent a vector.

Other ways to create a matrix are

```
   >> zeros(2,4)    % 2 row, 4 column matrix of zeros
   >> ones(2,4)     % 2 row, 4 column matrix of ones
   >> 3*ones(2,4)   % 2 row, 4 column matrix of threes
```

Note: The character '%' denotes a comment.
The convention with all of these matrix functions is that the first argument represents the rows and the second the columns

Transpose

We can easily find the transpose of the matrix 'A'.

```
   >> B = A'
```

Sum

For vectors, SUM(x) is the sum of the elements of x.
For matrices, SUM(A,1) is a row vector with the sum over each column.

```
   >> sum(A,1)
```

Sum(A,2) results in a column vector of sums over each row

```
   >> sum(A,2)
```

sum(sum(A)) will sum up all of the elements of A

```
   >> sum(sum(A))
```

Matrix Multiplication

Now let's multiply these two matrices together.

```
   >> C = A * B
```

MATLAB knows when you are dealing with matrices and adjusts your calculations accordingly.

Instead of doing a matrix multiply, we can multiply the corresponding elements of two matrices or vectors using the .* operator.

```
>> C = A .* B
```

Preceding an operator with a dot will be used often in this class. In addition to **.***, **./** is an element by element division, and **.^** is an element by element exponentiation.

```
>> C = A .^ 2
```

Subscripts

We can access one element of the matrix. Back to using the matrix A, the following accesses the element located in the third row and first column.

```
>> A(3,1)
```

Note: Whenever addressing a given element remember the notation is row, column

Now we can assign the vector b the first row of A

```
>> b = A(1,:)
```

This previous statement can be interpreted as b is assigned all columns of the first row of A.
Or we can assign the vector b all rows of the first column of A

```
>> b = A(:,1)
```

Or we can assign the vector b the first two elements in the first row of A

```
>> b = A(1,1:2)
```

Let's find the inverse of a matrix.

```
>> X = inv(A)
```

... and then illustrate the fact that a matrix times its inverse is the identity matrix.

```
>> I = inv(A) * A
```

MATLAB has functions for nearly every type of common matrix calculation, such as eigenvalues, characteristic polynomial, etc. Most of these functions are beyond the scope of this class. Relevant functions will be introduced when needed. However, if there is any function that you would like to use, refer to either the user manual or the online help for details.

At any time, we can get a listing of the variables we have stored in memory using the "who" or "whos" command.

```
>> whos
```

You can get the value of a particular variable by typing its name.

```
>> A
```

Help

The help command allows you to find out information on the syntax of a particular function. Try typing some functions that you have already used such as **zeros**, **ones**, and **sum**.

History

To save on typing you can recall previous commands by using the ↑ key. By continuously typing the ↑ key you can scroll through previous commands. By typing the first letter of the command you want to recall followed by the ↑ key only the commands starting with that letter will appear. Try typing **z** followed by the ↑ key to recall the zero function.

Tasks

1. Using the functions **ones** and **zeros**, and the various operators including the colon ('**:**') operator, create the matrix

$$D = \begin{bmatrix} 3 & 3 & 3 & 3 \\ 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Hint: try typing [a_1; a_2; $a_{3,}$ a_4], where the a_i's are vectors created by using the ones and zeros functions, along with the colon operator

2. Extract from the matrix D the sub-matrix $\begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix}$ and assign it to d

3. Calculate the sum $\sum_{i=2}^{5} i^2$, in a single command. Hint: There are two ways to accomplish this summation: 1) use the SUM function; 2) use a dot product by multiplying a row vector by a column vector.

4. Type in the row vector, a = [2 4 7 11 16 22]. Then using a single command, create a row vector consisting of the differences of successive elements from vector a. Hint: Consider using the colon operator.

II. Data manipulation

We will next learn how to manipulate and analyze data in two applications: human population growth, and infectious disease epidemics.

In this first application we will be using the data and analysis from Westing (Westing 1981). The first step is to load in the human population data. The data comprises human population estimates from eight key dates, starting with the year 300,000 BC. This initial date is assumed to be the time in which *Homo sapiens* began to populate the Earth. Each date reflects a major transition that affected population growth. The data set contains two columns, date and population level. Note that BC dates are entered as negative numbers.

Population growth

Loading in and looking at the data

```
>> load popdat.txt
```

The data is retrieved from the current working directory and an 8 by 2 matrix is created called popdat.
Typing the following will list the variable and its dimensions.

```
>>  whos
```

Typing the following will display the data.

```
>> popdat
```

Since there is such a huge range of values in the data, MATLAB does not provide ideal formatting. One way to look at the data is to look at it one column at a time

```
>> popdat(:,1)
```

lists all rows of column 1

```
>> popdat(:,2)
```

lists all rows of column 2. Another way to look at the data is to plot it

```
>>plot(popdat(:,1), popdat(:,2))
```

To re-scale type

```
>>axis([-8000 2000 0 4.4e9])
```

or

```
>>axis([0 2000 0 4.4e9])
```

Assuming exponential growth, we can observe the shifts in *per capita* growth rates by plotting the log of the population data

```
>>plot(popdat(:,1), log(popdat(:,2)))
```

Or alternatively,

```
>>semilogy(popdat(:,1), popdat(:,2))
```

This can be best seen in two plots

```
>>axis([-300000 0 0 25])
```

shows the first three time periods and

```
>>axis([0 2000 19 22])
```

shows the last four time periods. Note: To see both time scales at once, first type

```
>>figure
```

This will provide you with a 2nd figure window. Then using the up-arrow key recall the plot command and replot. Then recall the axis command.

```
>>plot(popdat(:,1), log(popdat(:,2)))
>>axis([-300000 0 0 25])
```

Analyzing the data
In his article, Westing analyzed the population data by assuming Malthusian or exponential growth, in which the *per capita* growth rate changes for each time period (Paleolithic, Mesolithic, Agricutural (BC), Agricultural (AD), Literate, Industrial, Nuclear).

Tasks:
1. Calculate the growth rate for each time period and save it in a vector called r.
   Hint: There are seven time periods each governed by the equation
   $$P(t) = P_0 \cdot e^{r_i \cdot \Delta t}$$
   where *P(t)* is the current population level, P_0 is the initial population level for each time period, r_i is the per capita growth rate for the i^{th} time period and Δt is the time period. For each time period you have two data points, the initial and final population levels.
   You can use the approach from task 4 of the last section to calculate the growth rate.

2. From the growth rates calculate the doubling time (DT) for each time period, where DT = 0.69/r.
3. Graph the doubling time for the seven time periods.

Saving your results
If you want to save your results to use later either in MATLAB or another program like Excel you can use the following command

```
>>save 'filename' X -ascii -tab
```

where filename will be the name of the file you will save it to, X is the matrix you are saving, -ascii is an option to save data in a text format, and -tab states you want the data tab delimited.

Note: If you are saving your work to use in MATLAB you can ignore the -ascii and -tabs options. If you are saving your data to use in Excel then it is often best to save one matrix at a time, since saving two matrices of different dimensions the row/column structure is compromised.

To save figures, cut and paste them in a Word document. Within Word you can scale the size to fit more than a few graphs on a page.

Epidemics

The Reed-Frost model

In the 1930's Reed and Frost developed a model that described an epidemic pattern of an acute, contagious infection after its introduction into a closed population. The model assumptions are outlined by Abbey (Abbey 1952).

> The infection is spread directly from infected individuals to others by a certain kind of contact (adequate contact) and no other way. Any non-immune individual in the group, after such a contact with an infectious person in a given period, will develop the infection and will be infectious to others only within the following time period, after which he is wholly immune. Each individual has a fixed probability of coming into adequate contact with any other specified individual in the group within on time interval, and this probability is the same for every member of the group. The individuals are wholly segregated from others outside the group. These conditions remain constant during the epidemic.

The variables are:
1. S_t, S_{t+1}, the numbers of susceptibles during time interval t and t+1 respectively.
2. C_t, C_{t+1}, the numbers of cases during time interval t and t+1 respectively.

And the parameter $p = 1 - q$ is the probability that any two individuals come into "effective contact".
The probability that an individual comes into contact with none of the cases is q^{Ct}, and therefore the probability that an individual comes into contact with one or more cases is $1 - q^{Ct}$.
The model follows directly, i.e., the number of new cases at time t + 1 is

$$C_{t+1} = S_t \cdot (1 - q^{Ct})$$

The following are the steps needed to simulate this model

```
c=zeros(15,1);
s=zeros(15,1);
c(1)= 1;
s(1)= 100;
for i=1:15
  c(i+1) = s(i)*(1 - q^c(i));
  s(i+1) = s(i) - c(i+1);
end
```

The new instruction used in this application is the *for loop*. You can type these commands, one at a time in the command window. However, since repeating these steps every time you want to run a simulation can be tedious, MATLAB allows you to write *scripts*, which are lines of instructions that you can save in a file. To create a script you would open up a new M-file using the MATLAB editor (in the MATLAB command window select file, new, m-file). Then type in the instructions and save the file with the name ReedFrost. You can then run these instructions by simply typing in ReedFrost. Instead if typing in these commands you can download the file ReedFrost.m from the web site.

Note that you will have to assign a value for the variable *q*, within the workspace before running the script.

Task1:
For different values of q (0.99, 0.98, 0.97, etc.) and different initial number of susceptibles (100, 200, 500, etc), simulate and plot the number of susceptibles and number of cases in the same figure (hint: typing "hold" between two plots allows you to overlay curves).
Here are some suggested values of *q* and $S_{initial}$. To automate the process of obtaining these values try using the script ReedFrost1.m.

Why does the epidemic die out?

Task2:
Collect data on the q, S_{initial} and S_{final}, and create two representative plots showing how the number of susceptibles after the end of the epidemic depends on the contact rate and the initial number of susceptibles.

q	*S initial*	*S final*	C_{max}	T_{Cmax}	*q*	*S initial*	*S final*	C_{max}	T_{Cmax}
0.99	100				0.97	100			
0.99	200				0.97	200			
0.99	300				0.97	300			
0.99	400				0.97	400			
0.99	500				0.97	500			
0.98	100				0.96	100			
0.98	200				0.96	200			
0.98	300				0.96	300			
0.98	400				0.96	400			
0.98	500				0.96	500			

Optional section

Compare model prediction with measles epidemic data (Abbey, 1982).
A measles epidemic within a boarding school fits the assumptions of the model. The file Aycock.txt contains data from a measles epidemic in a New England boys' boarding school in 1934. The three columns in this data set contain the time interval (12 day periods corresponding to the incubation period), the reported number of cases, the reported number of susceptibles.

Optional Task3:
Find the value of *q* that looks to be the best fit of the data and plot the observed number of cases (using markers) with the expected number of cases (using a continuous line). To learn about how to designate marker and line types, type *help plot*.

Note:

1. You will need to take into account the fact that the model starts at t=1, whereas the data starts at t=0.
2. Don't forget to set the initial condition in your model.

Optional Task4: Assessing goodness-of-fit
Add a line to the existing script that assesses the goodness-of-fit using the following equation

$$\chi^2 = \sum_{t=1}^{n-1} \frac{[O(C_t) - E(C_t)]^2}{E(C_t)} + \sum_{t=1}^{n-1} \frac{[O(S_t) - E(S_t)]^2}{E(S_t)}$$

where O(·) are observed values and E(·) are expected or predicted values. Fine tune your best-fit estimate for *q*. Download ReedFrost2.m for this task.

References

Abbey, H. 1952. An examination of the Reed-Frost theory of epidemics. *Human Biology* 24: 201-233.
Westing, A. H. 1981. A note on how many humans that have ever lived. *Bioscience* 31: 523524.