Introduction

Matlab has many functions that make it extremely attractive to use. Some of these include

- Matlab can be used as a calculator performing algebraic operations such as addition, subtraction, root extraction, powers, etc.
- Matlab can be used to numerically evaluate many functions. These include familiar functions such as those found in ordinary calculators such as exponential, logarithmic and trigonometric functions.
- Matlab can be used to calculate functions which are normally not in any calculator. The built in functions are far more expansive than those in any calculator.
- Matlab is a programming language with all the components needed for programming. These include
  1. The use of symbolic constants to represent quantities
  2. A set of iterative operations
  3. A set of conditional and relational operations
  4. The use of data structures (vectors, matrices etc) to store data as well as array operations which can process these data structures in parallel.
  5. The ability to make user defined custom functions
  6. The ability to group commands sequentially using batch programs.
  7. Extensive graphics environment for both 2D and 3D applications.
  8. I/O compatibility.

MATLAB as Calculator

As a rule, text entered by the student at the Matlab prompt >> will be bold while the answer obtained will be in italics. Comments will be preceded by a % sign

>> 6+3 9
>> ((2+7*9)^3)/(42-2^(1/2)) 6.7665e+003
**MATLAB built-in functions and on-line help features.**

Matlab has an extensive set of functions and tools. To access the list and/or learn about the use of a particular internal function, Matlab has an easy to use on-line help capability

Some examples in this are given below.

```matlab
>> Help elfun %Lists Matlab's built in elementary functions
>> Help cos %Gives you information on the particular function cosine
```

**MATLAB Symbolic Constants**

With the ability to assign symbolic constants and perform operations on these constants, MATLAB becomes a programming environment. Some examples in this are given below.

```matlab
>> pi 3.1416
>> g=cos(pi/4) 0.7071
>> g 0.7071
```

Note that the value cos(pi/4) is stored in g and can be used again

```matlab
>> g1=acos(g);
```

Note that the (;) prevents the answer from appearing on the screen but the calculation was still performed.

**Question?** How do you see the value of g1?

After many operations using many different symbolic constants, you may forget and want to see all the symbolic constants. This is done with the Matlab **who** command

```matlab
>> who g g1
```
MATLAB as Data Structures

An important task for a computer environment is the ability to create and manipulate data structures which hold many data elements.

1. **Scalars:** Examples are 3, 4.1, pi, sin(32), exp(23), etc. We already know how to input scalars.

2. **Vectors:** A vector is a 1D data structure whose elements are scalars. A vector can be either vertical (column) or horizontal (row).

   \[
   \begin{bmatrix}
   2 \\
   4 \\
   1
   \end{bmatrix}
   \]

To input a column vector,

\[>> \text{vc}=[2;3;4] \quad \% \text{Note the semicolon between elements.}\]

To input a row vector,

\[>> \text{vr}=[1 \; 5 \; 3] \quad \% \text{Note no semicolon between elements.}\]

The semicolon acts as an indicator to Matlab to go to the next row.

**Question?** How do we change a row vector to a column vector and visa versa?

This operation is called taking the transpose (flipping the vector).

Matlab does this using the " ' " operator

\[\% \text{Column vector flips to row vector}\]
\[>> \text{wr}=\text{vc}'\]
\[\% \text{Row flips to column.}\]
\[>> \text{vrc}=\text{vr}'\]

**Question?** How do I get back to the original vector?

A. **Performing Arithmetic Operations on different Data Structures**

If we want to add or subtract two vectors, you must make sure that the lengths and type (row or column are compatible)

\[\% \text{Column vector flips to row vector}\]
\[>> \text{A}=[2 \; 3 \; 7]; \; \text{B}=[1 \; 3 \; 5 \; 7]; \; \text{A+B}\]
>> A=[2 3 7]; B=[1;3;5]; A+B

Question? What happened in these cases?

However, we are allowed to add a scalar to a vector (even though the sizes are different).

>> D=3+A;

Question? What happened?

If the vectors have the same orientation, they may be added.

>> B=B'; A+B % Now it works.

B. Array Operations on vectors

In this chapter, we will only consider array multiplication (defined below). There exist other definitions of vector multiplication that we will discuss in later lessons.

Array operations are defined as

\[
\begin{bmatrix}
A_1 \\
A_2 \\
\vdots \\
A_N
\end{bmatrix}
\text{(oper)}
\begin{bmatrix}
B_1 \\
B_2 \\
\vdots \\
B_N
\end{bmatrix} =
\begin{bmatrix}
A_1 (\text{oper}) B_1 \\
A_2 (\text{oper}) B_2 \\
\vdots \\
A_N (\text{oper}) B_N
\end{bmatrix}
\]

where (\text{oper}) can stand for multiplication (*), division (/) or power (^).

This allows for parallel processing of all elements of the vector and greatly speeds up the process. (Notice the "." before the operator).

>> A=[2;1;4]; B=[1;3;2];
>> C=A.*B
   2 3 8

We should note the same limits (equal lengths and orientation are needed).

Problem: Perform the requested operation and explain what happens in each case below.

>> D=A./B %Array division
>> E=A.^B %Array power
>> F=3.^A
>> G=A.^3

C. Array Operations on Matrices

A Matrix is a 2 Dimensional (2D) table of numbers. A matrix must be rectangular. For example \[
\begin{bmatrix}
1 & 6 & 2 \\
-1 & 3 & 1
\end{bmatrix}
\] is okay. The size is measured by defining the number of rows (m) and number of columns (n). The size is then expressed as m x n.

Question? What is the matrix size of the above matrix?

The matrix above is inputted into Matlab as

>>&A=[1 6 2;,-1 3 1];

The same array operations that were applicable to vectors can be applied to matrices. Obviously, the operands need to have the same size for array operations to work.

Problem: For the matrices below, calculate the following operations

>>&A=[1 6 2;,-1 3 1];
>>&B=[3 1 1;2 -1 4];

>>&C=A.*B %Array Multiplication
>>&D=A./B %Array division
>>&E=A.^B %Array power
>>&F=A’ %Transpose

D. MATLAB functions applied to different Data Structures

Matlab is extremely useful because most functions operate on vector and matrix inputs as well as scalars.

Problem: For the matrix A below, evaluate and comment

>>&A=[1 6 2;,-1 3 1];
>>&B=\exp(A)
>>&C=\log(B)

The same would obviously apply for vector inputs.
E. Addressing and Manipulation of Vectors and Matrices. Creation of Sub and Super Vectors and Matrices. (The colon (:) operator)

Suppose you want to access a particular element of a vector and place it into another variable

```matlab
>> vc=[3;1;6];
>> h=vc(2) 1
```

Note that h has the value of the second element of vc. This operation can be broken into two steps. The RHS evaluates the 2nd element of vc and then, this value is assigned to h.

Note: This addressing scheme is the same for row or column vectors. Suppose we want to take the first two elements of a vector and place them in another vector.

**Method 1**

```matlab
>>v1=[vc(1) vc(2)]; % A pain
```

**Method 2**

First we form a vector which stores the addresses we wish to access.

```matlab
>>address_vector=[1 2];
```

The vector name does not have to be this long. Only for instructional purposes.

Now, we extract the target elements in vc with the desired addresses and assign to v2.

```matlab
>>v2=[vc(address_vector)];
```

In many cases, the number of elements we want to extract are large and we would need to write all the addresses. However, the addresses we wish to access often form a pattern.

**Examples:**

Suppose we want to access the first 50 elements of a long vector (don’t worry how it was generated). It would be ridiculous to manually write

```matlab
>>address_vector=[1 2 3 4 5 . . . 50];
```

To help, Matlab has a wild card operator (:) which can generate large groups of numbers which follow a simple pattern.
Problem: Calculate and explain the following operations

```matlab
>> A = [1:10]
>> B = [1:3:10]
>> C = [2:2:11]
>> D = [3:-1:-5]
>> E = [1:pi:10]
```

Problem: If \( \text{hr} = [3 \ 6 \ 2 \ 1 \ 5 \ 8 \ 2 \ 4] \); Place the 4th element, the second and the 6th element of \( \text{hr} \) into the 2nd, the first and third positions of a new vector \( \text{h}1r \).

```
\[
\begin{bmatrix}
h1(1) \\
h1(2) \\
h1(3) \\
h1(4) \\
h1(5) \\
h1(6) \\
h1(7) \\
h1(8)
\end{bmatrix}
\]

h1r(1) \\
h1r(2) \\
h1r(3)
```

Hint: You need to make two address vectors, one for \( h1 \) and the second for \( h1r \).
Addressing a vector required one number (or index). Addressing a matrix requires 2 indices, one index for the row and one for the column. Just like for vectors, we may want to extract some matrix elements and place them in another data structure.

Problems: Perform the following operations on the matrix

```matlab
>> A = [1 \ 6 \ 2; -1 \ 3 \ 1]; 
>> B = A(2,3) 
>> C = A(1:2,3); 
>> D = A(:,3); 
>> E = A(2,:); 
>> F = A(1,1:2:3)
```

F. **Creation of Super Vectors and Matrices.**

Until now, we have been plucking sub sections of vectors and matrices but many times, we want to combine smaller vectors or matrices into larger ones.
Problem: Suppose A=[1 2; 3 4]; Calculate and explain what happens

```matlab
>> SuperA= [A A; A A]  
>> SuperA=[A;A]  
>> SuperA=[A A;A]  
```

G. MATLAB time saving functions for building vectors and matrices.

The zeros, ones, and eye (identity) functions are ways to generate matrices whose elements are connected in a trivial way.

Problem Calculate and comment on the following

```matlab
>> M=zeros(3)  
>> M=zeros(4,2)  
>> N=ones(3,3)  
>> N=ones(1,5)  
>> P=eye(3)  
>> P=eye(2,3)  
>> T=5*ones(4,3)
```

H. Generating function tables for a given function over a given domain in MATLAB. Polynomials, exponential, and sinusoidal, etc.

The data structures (vectors and matrices) have value besides being a convenient storage base. They are the natural way to represent functions and as we will see, to graph them.

Suppose we want to describe the function (and plot) the function

\[ y = x^2 \] on the interval \([1, 5]\)

In high school, you would take samples of \(x\) on the interval and calculate the corresponding \(y\) values. The generated table is the function table or function.

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
</tr>
</tbody>
</table>
It is obvious the finer the sampling of $x$ on the interval, the better the function table is since there is less chance of missing some particular behavior.

```matlab
>> x = [1:5]';  % Why the transpose?
>> y = x.^2
>> ftable = [x y];  % looks like the table
>> x = [1:.1:5]';
>> y = x.^2
>> ftable = [x y];  % Better table
>> x = [0:2*pi/100:2*pi]';  % 100 sample points on [0,2pi]
>> y = cos(x);  % Table for cos(x) on the interval
               % [0,2pi]

2D Graphs and Plotting

A. The plot command.
Function tables do not help (especially if they are long). In order to get a better sense of the function, we need to plot the function.

*Problem* Comment on the following procedure
```
x = [1:5]';
y = x.^2
plot(x, y, 'o')
```

*Question*? What is the purpose of the third argument?

While seeing the individual points of the function table are nice, it is often better to connect the points with lines to make a continuous plot.

*Problem* Comment on the following procedures

```matlab
>> x = [1:5]';
>> y = x.^2
>> plot(x, y)

>> x = [1:0.1:5]';
>> y = x.^2
>> plot(x, y)

>> x = [1:0.1:5]';
>> y = x.^2
>> plot(x, y, x, y, 'o')

>> x = [1:0.1:5]';
>> y = x.^2
>> y = y.^2
```
>>x1=[1:1:5]';
>>y1=x.^2
>>plot(x,y,x1,y1,'o')

B. Controlling the plot axis.

The axis command controls the range of the x and y axis. (See help axis)

Question? What does axis([1 2 0 4]) to your graph
Question? If I wanted to view the function between x=4 and x=5 only, what axis limits would you use.

C. Provide labeling to your graph.

We can label the graph along the x axis, y axis and title
>>xlabel('xaxis')
>>ylabel('yaxis')
>>title('Plotting Function y=x^2 on the interval [1,5]')
% Note that xlabel etc. has string input.

D. Input text into your graph. (The text command)

The text command has the form text(xpos,ypos,'string') and places the text in string onto the position marked by the first two arguments.

Example: Label the point (4, 16).
>>plot(x,y,'o')
>>text(4.05,16,'<- (4,16)')

Question?
Why is the label position at (4.05,16)?
What would happen if you placed it at (4,16)?

E. Plotting multiple plots on one graph. (The hold command)

Suppose we want to plot two functions simultaneously and place them on the same graph.

Example Plot y=x^2 and y=x^3 on the interval [0,1];

>>x1=[0:.01:1];
>>y1=x1.^2;
>>x2=[0:.01:1];
>>y2=x2.^3;
>>plot(x1,y1,x2,y2)

The first graph is blue, the second green (default setting).
In this example, the function domains are the same (same x) but they need not be.
Alternatively, we could use the hold command that is useful if we want to make many graphs at different times

>>plot(x1,y1)
>>hold on

Screen is frozen (but still adapts to fit all subsequent plots)

>>plot(x2,y2)
>>hold off  % screen is un frozen

Unfortunately both graphs are the same color. It would be nice to control the line style so we can differentiate the graphs if we decide to print the plots. Even if the color of the graphs are different on the screen, a black and white printer would output black traces. Therefore, it is important to have different line styles

**F. Different graphic line font / colors.**

Matlab plot supports several line fonts and several point fonts as well as different colors.
Type (help plot) for a complete list.

**Examples**

>> plot(x1,y1,'b-',x2,y2,'r--')  %what happens
>> plot(x1,y1,'k:',x2,y2,'ro')   %what happens

**G. Placing Multiple Graphs on a single figure window. The (subplot) command.**

The subplot command breaks up the figure window into a rectangular partition of cells. Each cell can support a graph. The syntax is subplot(m,n,p) where m is the number of rows, n is the number of columns and p is the particular cell you want the figure to appear in.
H. Loading graphs into other application programs.

Suppose you are writing a report (Lab etc.) in Microsoft Word and want to input MATLAB figures into the report. The following procedure below outlines the idea.

1. Select figure
2. Select Copy Figure from the edit menu
3. Open Microsoft Word document
4. Select Paste from the Edit Menu

I. Saving plots as postscript files.

We can either print a graph to the printer or print to a postscript file. If you type `print`, Matlab prints the figure to the default printer. If you type `print -dps filename`, the graph will be saved as a postscript file with name (filename.ps). This file can be printed.

**Loading and exporting data from MATLAB**

Most external data is in ASCII format. If the data is numeric and stored in equally sized columns, MATLAB can read the data using

```
>> load datafile.txt -ascii
```
If the data is in MATLAB form, use

```
>> load datafile
```
or

```
>> load datafile.mat
```
To save a variable x in ASCII format

```
>> save x.data x -ascii
```
To save a variable x in matlab format

```
>> save x x
```

* Do not do
```
>> save x
```
This causes every variable to be saved in x.mat