INTRODUCTION

Matlab is an interactive software system for doing scientific calculations. It has many built in mathematical functions and plotting capabilities that allow you easily to carry out computations that arise in Physics. It can be used rather like a calculator, or for more complex problems, it can be used to write programmes.

The purpose of these notes is to help you begin to use MATLAB to solve some physics problems. They can best be used hands-on. You should work at the computer as you read the notes and freely experiment with examples.

At the end of the course, you should have developed enough skills to allow you to use MATLAB on your own, to tackle problems that would be too difficult or tedious to do "by hand".
MATLAB: Key Ideas

Commands
To do work in MATLAB, you type commands at the >> prompt, followed by the return key. Often these commands will look like standard arithmetic, or function calls similar to many other computer languages.

Later, you will learn how to store a sequence of commands in a file, and to execute them automatically, just as if they had been entered from the keyboard. Such files are called script files in MATLAB. It is also possible to extend the commands that MATLAB knows about by writing function files. For example you might write a function file that calculates the volume of a sphere, given the value of the radius.

The Basic Objects: Matrices and Variables
Matlab works with essentially only one kind of object, a matrix (hence its name; MATrix LABoratory). A matrix is simply a rectangular collection of real or complex numbers, and is said to be of size $m \times n$ if there are $m$ rows and $n$ columns. In this course we will find that many physical problems of interest can be expressed in terms of matrices.

In some situations, MATLAB attaches special meaning to 1-by-1 matrices, which are scalars, and to matrices with only one row or one column, which are vectors.

Scalars
Let's start with the simplest example, a scalar, which we'll call $x$ (and which might represent the distance of a ball from the ground). MATLAB doesn't know about $x$ until we tell it. We can do this by assigning a value to $x$, by typing at the prompt

$$x = 5$$

Note the response. $x$ is called a variable, and we can store any numerical value in it.

Conventional decimal notation, with optional decimal point and leading minus sign, is used for numbers. A power-of-ten scale factor can be included as a suffix. Here are some examples of legal numbers:

3
9.6397238
-99
0.0001
1.60210e-20 (which represents $1.60210 \times 10^{-20}$)
6.0225e23 (which represents $6.0225 \times 10^{23}$)
At the simplest level you can use MATLAB just like a calculator, using the following arithmetic operators.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
</tr>
<tr>
<td>−</td>
<td>subtraction</td>
</tr>
<tr>
<td>*</td>
<td>multiplication</td>
</tr>
<tr>
<td>^</td>
<td>raise to a power</td>
</tr>
<tr>
<td>/</td>
<td>division</td>
</tr>
</tbody>
</table>

**Exercise:** Try \( x = 5 \times 4 \), and some other examples that occur to you.

There are also a number of built in functions available to you in MATLAB, such as square root. You can see how this works by typing

\[ \text{sqrt}(x) \]

A complete list of built in functions is given in the manual, and later in these notes.

**STATEMENTS, EXPRESSIONS AND VARIABLES**

Now that you have broken the ice with the basic ideas, it is worthwhile to describe the structure of MATLAB a little more formally.

MATLAB is an expression language; the expressions you type are interpreted and evaluated once you press the return key. MATLAB statements are usually of the form

\[ \text{variable} = \text{expression} \]

or simply

\[ \text{expression} \]

Expressions are usually composed from operators, functions, and variable names. Evaluation of the expression produces a matrix, which is then displayed on the screen and assigned to the variable for future use. If the variable name and = sign are omitted, a variable \texttt{ans} (for answer) is automatically created to which the result is assigned.

A statement is normally terminated with the return key. However, a statement can be continued to the next line with three or more full stops followed by the return key. On the other hand, several statements can be placed on a single line if separated by semicolons.

If the last character of a statement is a semicolon, you don't see the result, but the assignment is carried out. This is essential in suppressing unwanted printing of intermediate results.

**Exercise:** Try some examples.
Evaluating Formulas

You probably remember from elementary kinematics the equation for the distance $y$ that an object travels under constant acceleration $a$ in a time $t$, starting with an initial velocity $v_0$

$$y = v_0 t + \frac{1}{2} a t^2$$  \hspace{1cm} (1)

This equation can be entered in MATLAB by typing

$$y = v_0 \times t + (1/2) \times a \times t^2$$  \hspace{1cm} (2)

I have used $v_0$ to represent $v_0$, but you may choose any name that begins with an alphabetical character, and can include digits or underscores up to a total of 19 characters.

Try it. What did you get? The problem is, that MATLAB doesn't yet know about the variables $v_0$, $a$, and $t$; you must first assign some numerical values to them. Do this choosing some values that appeal to you. Now you can re-enter equation (2). (You can save yourself a lot of re-typing by using the up arrow to recall a previous line to the command line, and then edit it as required.)

Now MATLAB returns the value of $y$. Is it correct? Did you make a good choice for the assigned values of the variables $v_0$, $a$ and $t$, in order to allow you to reliably check that you had entered the formula correctly?

Suppose we want to evaluate the distance $y$ for a sequence of times, say $t = 0, 1, 2, 3, 4$. We could repeat the above procedure for each value of $t$, and it’s probably worth doing this once. Write down your answers for later reference. But now you are about to glimpse some of the real power of MATLAB. The sequence of times can be written as a matrix, which we can visualise as

$$t = (0 \ 1 \ 2 \ 3 \ 4)$$  \hspace{1cm} (3)

which has one row and five columns. All variables in MATLAB actually represent matrices, and the easiest way to assign a matrix of values to the variable (for a small matrix such as above) is by an explicit list of elements separated by blanks or commas, and surrounded by the square brackets [ and ]. So to enter the matrix from line (3) into MATLAB we type

$$t = [0 \ 1 \ 2 \ 3 \ 4]$$

If there is more than one row, you can indicate the end of each row by a semicolon ; or the return key. For example the matrix

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$  \hspace{1cm} (4)

is entered by either of the statements

$$A = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9]$$
or

\[
\mathbf{A} = [ \\
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
] 
\]

Try it. Notice that if you used the lower case \( a \) instead of \( A \) you will have overwritten your earlier assignment to the variable representing acceleration in equation (2).

MATLAB is case sensitive and distinguishes between lower case and upper case.

If you haven't already entered the matrix of line (3) into your variable \( \mathbf{t} \), do so now. Notice this is a one row matrix, which we sometimes call a vector in MATLAB. It is possible to get MATLAB to calculate the sequence of \( y \) values corresponding to the sequence of \( t \) values using a very similar expression to line (2). But first it may be helpful to recall a few important ideas about matrices.

**MATRIX CONCEPTS**

- A matrix is a two dimensional array of numbers (or symbols), arranged in a rectangular form with (say) \( m \) rows and \( n \) columns. If \( m = n \), we call the matrix square. Two matrices, \( \mathbf{B} \) and \( \mathbf{C} \), can be multiplied together in the order \( \mathbf{B} \mathbf{C} \)

  by the rule we can colloquially describe as "diving the rows of \( \mathbf{B} \) into columns of \( \mathbf{C} \)". You should remember this from your algebra courses. However a necessary requirement is that the number of columns of \( \mathbf{B} \) is equal to the number of rows of \( \mathbf{C} \).

- We can denote the element from the \( i \)th row and \( j \)th column of matrix \( \mathbf{A} \) by using subscripts \( A_{ij} \) or parentheses, \( A(i,j) \). In line (4) for example \( A_{23} = 6 \).

If you are rusty on basic matrix concepts, spend some time reviewing them before your next session.

**MATRIX OPERATIONS**

MATLAB is set up to do matrix operations, so if you have assigned values (or elements) to the variables \( \mathbf{B} \) and \( \mathbf{C} \), the expression

\[
\mathbf{B} \times \mathbf{C}
\]

multiplies them according to the rule discussed above. The operation is defined whenever the "inner dimensions" of the two matrices are the same.

*Exercise:* Explore some examples.
A scalar can also multiply, or be multiplied by a matrix e.g. Try

$$5 \times A$$

Addition and subtraction of matrices are denoted by + and – and are simply element-by-element addition or subtraction of the corresponding entries. The operations are defined whenever the matrices have the same dimensions.

**Exercise:** Enter a matrix $D$, where

$$D = \begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \end{pmatrix}$$

and then enter

$$F = A + D$$

Do you understand the result you get?

Addition and subtraction are also defined if one of the operands is a scalar, that is a 1-by-1 matrix. In this case, the scalar is added to or subtracted from all the elements of the other operand. For example try

$$z = 1 + D$$

There are other matrix operations; matrix "division", matrix powers, that you will discover as you explore MATLAB.

**ARRAY OPERATIONS** (ELEMENT-WISE OPERATIONS)

We will often want to perform an operation on each element of a vector while doing a computation. Addition (+) and subtraction (–) already work this way. For example, if $w = [1 \ 2 \ 3]$ and $u = [5 \ 6 \ 2]$, then $w + u$ will return $[6 \ 8 \ 5]$. Matrix multiplication is different; rows "dive into" columns; similarly matrix division and matrix power operations have specifically defined linear algebraic meaning. But suppose we wanted to multiply the first element of $w$ by the first element of $u$, and the second element of $w$ by the second element of $u$, and so on. (We will shortly give a physical example where this is done). We achieve this in MATLAB by preceding the * operator by the fullstop ., as in

$$w.*u$$

**Exercise:** Try it, as well as $w*u$

Similarly / or ^ can be preceded by a fullstop to specify elementwise division or power operations. In MATLAB, these are called "array operations".

**Exercise:** Explore the array operation using $w$ and $u$.

There is one subtlety you should be careful with. It is of course possible to take an element by element power using a scalar base and matrix exponent. Let's put $s = 2$ and type
Make sure you understand the result that is returned. We could have done this more directly by typing

\[ s \cdot \_w \]

but now the space between the digit 2 and the fullstop . is important. If it was not there, the fullstop would be interpreted as a decimal point associated with the 2.

MATLAB would then see only the isolated ^ and would attempt to calculate a matrix power, which in this case would result in an error message because the exponent matrix is non-square. An alternative to the space is to use parentheses, forcing MATLAB to interpret the base as a number before it carries out the array operation, \((2) \cdot ^w\)

Now let's return to our problem of an object travelling under constant acceleration, where we were wishing to evaluate the distance \(y\) for a sequence of times. First type \(t\) to be sure that your variable \(t\) is the vector given in equation (3).

Now re-enter equation (2) (or use the clipboard) but with \(t^2\) defined as an array operation

\[ y = v0 \cdot t + (1/2) \cdot a \cdot t \cdot ^2 \]

Note that because \(v0\) and \(a\) are scalars, we didn't need to use a fullstop with their multiplications (although you could do so – try it).

**Plotting Results**

A very powerful technique in physics is to graphically display results so they can be visualised, and MATLAB has very convenient plotting capabilities. We can make a graph of \(y\) versus \(t\) by the command

\[ \text{plot}(t, y) \]

Do you understand the graph? It is simple to redo the graph with a different value of one of the parameters. Change the value of the acceleration, re-evaluate the expression for \(y\), and plot the new results. Now change \(v0\). Then try a different sequence of times.

It is easy also to add a title and axes label to your graph, by entering (after the plot command)

\[
\text{title}(\text{'distance versus time'}) \\
\text{xlabel}(\text{'time'}) \\
\text{ylabel}(\text{'distance'})
\]

Notice that MATLAB refers to the vertical axis as \(y\), and the horizontal axis as \(x\), in the normal way. The single quotes around the characters you are putting on the graph are necessary to indicate they are to be treated as just a string of characters (and not variables, etc)
**MORE MATRIX STUFF**

So far we have dealt mainly with one-dimensional matrices but it is possible to use two-dimensional matrices, and calculate \( y \) distances for a sequence of times, and a range of accelerations, in *one calculation*. To do this we will need to know about the matrix transpose operation.

**MATRIX TRANSPOSE**

Formally, the transpose of a matrix \( A \), which we write \( A' \), is another matrix with elements defined as

\[
A'_{ij} = A_{ji}
\]

You can think of this as "rotating" \( A \) to make a new matrix \( A' \) whose columns are the rows of \( A \). MATLAB does this, naturally enough, with the apostrophe symbol. Type

\[
A
\]

to recall the matrix you entered earlier in the session (see Eq.(4)). Now type

\[
A'
\]

**Exercise:** Try it also with the row vectors you have previously entered. What is the transpose of a column vector?

The row vectors (or \( 1 \times 3 \) matrices) \( w = [ 1 \ 2 \ 3 ] \), \( u = [ 5 \ 6 \ 2 ] \) that you entered earlier can not be matrix multiplied. Try \( w*u \). Why does it fail? However \( w*u' \) gives an answer. This in fact is the inner product of the two vectors. Do you understand why? Now try \( u'*w \). Do you see what is happening? Let's return to our kinematic problem, and calculate the \( y \) distances for the sequence of times \( t = (0, 1, 2, 3) \) seconds for the cases where the acceleration \( a = 2 \text{ m/s}^2 \) and also \( a = 4 \text{ m/s}^2 \). Let's begin by assuming \( v_0 = 0 \), and using \( y = \frac{1}{2} at^2 \). Just as we represented the sequence of times as a vector, stored in the variable \( t \), we can represent the range of accelerations as a vector stored in the variable \( a \)

\[
a = [2 \ 4]
\]

In scalar arithmetic, the order of factors in a multiplication is unimportant. In matrix multiplication the order is crucial. Typing

\[
y = 0.5*(t.^2)'*a
\]

produces a \( 4 \times 2 \) matrix whose first column gives the \( y \) distances for \( a = 2 \) and the second column gives the \( y \) distances for \( a = 4 \). Make sure you understand why (you don't really need the brackets, but they make it clear what you are doing). Now include the contribution of the initial velocity. any success? You need the \( v_0/t \) part to also be a \( 4 \times 2 \) matrix, where column one applies for \( a = 2 \) and column 2 for \( a = 4 \). There are many ways to do this, and here is one

\[
y = v0*t'*[1 1] + 0.5*t.^2'*a
\]

By the way, it is worthwhile to know that in MATLAB you can easily make up an \( m \times n \) matrix with every element equal 1 by the command \texttt{ones(m,n)}. Other simple matrices (e.g. the identity), are equally simple to make; see the reference manual.
You can plot the distances for each acceleration together on one plot by typing

```matlab
plot(t, y)
```

which causes each column of \( y \) to be plotted successively against the vector \( t \).

**Exercise:** Repeat the distance calculations above, but this time producing the \( y \) distances in rows rather than columns. Plot them out.