Today's Topics

- Using MATLAB as a calculator.
- Constants
- Arithmetic operators and expressions
- Elementary functions, help, `format`
- Variables, reserved words,
- Introduction to script files
- Input

MATLAB is a computer language/application designed for scientific computation. It can be used in two ways, interactively (like a calculator) or non-interactively (as a programming language).

Starting
Click on the Windows start symbol at the bottom left of the screen, click Programs from the menu that appears, maths and stats, then move to click Matlab. This opens the Command Window where you can use MATLAB like a calculator.

Prompt
When MATLAB is ready to accept a new instruction it shows a prompt `>>`.

Enter
Press the enter key when you have typed in your instruction. It does not matter if the cursor is not at the end of the line.

Arithmetic Operators
The arithmetic operators in *decreasing* order of precedence are:

arithmetic operator	operation performed
`^`	raise to a power
`/`	division
`*`	multiplication
`-`	subtraction
`+`	addition

You often need to use brackets to make sure that the operations are carried out in the correct order.

Arithmetic Expressions

```
>>2*3
ans=6
```

```
>>12/5
ans=2.400
```

```
>>2^3
ans=8
```

```
>>10-(1+3)
ans=6
```

Write in MATLAB

1. $3^2 + 5$
2. 3^{2+5}
3. $\frac{60}{2+3}$

4. $\frac{60+3}{2}$
5. $-2 \times 5 \times -7$
6. $\frac{12-3}{5+1}$
7. $\frac{1}{2^4}$
8. $(\frac{3}{4})^4$
9. 5π

Elementary Functions
Most of the usual mathematical functions are implemented in MATLAB.

```
>> cos(0)
>> 6*sin(pi/2)
>> exp(1)
>> log(exp(3))
>> sqrt(9)
```

Note:

- MATLAB uses radians, not degrees.
- The `log` is the natural log (often labelled `ln` on calculators). The log base 10 is `log10`.

Variables and Memory
A variable is a labelled piece of computer memory.

```
>> s=5.6
```

The number 5.6 is stored in the computer memory and labelled.

- Matlab is case sensitive. That means that the upper case and lower case of a letter are different. For example, the variable called `s` is different from the variable called `S`.
- Reserved words are variable names it is best not to use. This is often because they are names of built-in functions.
- A semicolon after a statement suppresses the output. For example, the statement `x=2*3` will set the value of variable `x` to 6 and also show the result on the screen. However, `x=2*3;` still sets the value of `x` to 6, but does not show the result of the multiplication on the screen.

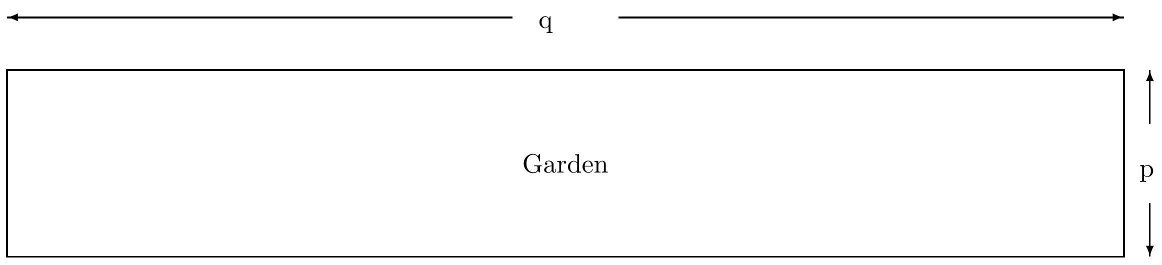
Example
Find the area of a circle with radius 5cm.

```
>> r=5
>> A=pi*r^2
```

Now let radius be 7cm.

```
>> r=7
>> A=pi*r^2
```

Example
A piece of wire netting to put a fence around a garden is 10 metres long.



If $p = 2$, what is q and what is the area enclosed?

```
>> p=2
>> q=5-p
>> A=p*q
```

Note that it is better not use "length" as a variable name as it a reserved word (it is already defined as a function we will learn about later).

Script Files
A *program* is a sequence of statements that are performed in order. They are stored in a *script file.*

A *file* is the basic unit that the computer stores on mass storage (such as hard disk or floppy disk). Files can store pictures, or programs (such as MATLAB script files) or other data (such as text files).

To *run* a program either type the name of its file in the Command Window or use the Debug menu in its window and choose Save and Run. Note that the Save and Run commands changes to Run if the file has been saved.

Examples

1. Write a script file to set a value for `d`, a distance in km, and calculate how long it takes to drive that distance at 80 kph.

   ```
   d=240
   t=d/80
   ```

2. Write a script file to set a value for `r`, the radius of a circle, and calculate the area of the circle and its circumference.

```
r=2
a=pi*r^2
c=2*pi*r
```

`input` statement
In the example above we put values directly into the Matlab script file. The `input` statement is another way to enter values for MATLAB. It includes a string which is used as a prompt so the user knows what to enter.

Example
```
x=input('Enter a positive number ');
```

When this statement is executed, the string
`Enter a positive number`
will appear in the command window (we will look in detail at strings later). The user then types in a value, and this value is assigned to the variable `x`. Note that just this statement will not check that the number entered is positive.

Today's Topics

- Strings and `disp`
- Boolean expressions
- `if` statements - flow control

Revision Exercises
Write Matlab expressions to calculate:

- $4^2 + 3\cos\frac{\pi}{2}$
- $\sqrt{1-\pi}$
- $5e^3$

- Write a Matlab script file to:
  - Prompt the user to enter the value of the variable `r`; and
  - Calculate the circumference of a circle of radius `r`.

Well Depth Formula

$$depth = (\frac{g}{k}) \times (time + \frac{e^{-k\times time} - 1}{k}).$$

g: acceleration due to gravity, $g \approx 9.81$.
k: friction constant, $k \approx 0.05$ for a stone (roughly).

Write Matlab to set the values of g and k and then calculate the depth for a given time.

```
g=9.81
k=0.05
t=input('How long did the stone take to reach the bottom of the well? ')
d=(g/k)*(t+(exp(-k*t)-1)/k)
```

Character Strings
Variables can be used to store numeric values (eg -6.5e3) or strings (eg 'Hello').
Note that the single quote marks are used for strings in MATLAB.

```
>> n=5;
>> y='Course MATHS 162';
```

The first character of string `y` above is `y(1)`.
The 7th to 13th characters inclusive are `y(7:13)`.

`length(y)` gives the length of string `y`.

Note that because `length` is the name of a function, it is not a good idea to use it as a variable name.

Concatenation
This is following one string by another.

```
>> x='I am studying'
x =
I am studying
>> y='Course MATHS 162'
y =
Course MATHS 162
>>[x y]
ans =
I am studyingCourse MATHS 162
```

`disp` **command**
`disp(x)` will display `x`, whether it is a number or a string.
If `x` and `y` are as above, then use

```
>> disp([x y])
```

Examples

disp command	output
`disp(5.7)`	`5.7000`
`disp('Hello')`	`Hello`
`disp(15*4)`	`60`
`disp('15*4')`	`15*4`

A script file example

```
greeting='Hello.';
question='What is your name?';
disp([greeting,' ',question])
```

Output
`Hello. What is your name?`
The `;` after a statement suppresses the output.

Well Depth Formula in a Script File

```
% Program to calculate depth of well from
% time a stone takes to reach the bottom.
% Use g=9.81, k=0.05
g=9.81;
k=0.05;
time=2;
depth=g*(time+(exp(-k*time)-1)/k)/k;
disp('Depth of well is ')
disp(depth)
disp('metres')
```

Output

```
Depth of well is
   18.9820
metres
```

Boolean Expressions

Boolean Expressions are comparisons which are either true or false.

In MATLAB: false is represented by 0
true is represented by 1
(or any other non-zero number)

Examples

Boolean	result	Boolean	result
`6>3`	true (or 1)	`3>6`	false (or 0)
`7==4`	false (or 0)	`7~=4`	true (or 1)

Booleans may be combined using logical operators. Use brackets to make sure of the order of evaluation.

`~`	logical NOT	changes to opposite truth value
`|`	logical OR	at least one relation is true
`&`	logical AND	both relations are true

Examples

expression	result
`(6>3) & (4>2)`	true (or 1)
`7==4 | 2<6`	true (or 1)
`~(2>4)`	true (or 1)
`(~(2>4)) | 2>1`	true (or 1)
`(2<0)&((1>-1)|(4==2))`	false(or 0)

Quick Quiz

```
 ( (5>0) | (5<10)) &(2==3)
```

```
(~(5>0)) | ((1+3==4) & (5<12))
```

Answers: First expression is false, second is true.

Introduction to `if` statements

Use an `if` statement to execute some commands only when a certain Boolean is true (i.e. when some condition is met). Use `end` to show the end of the statements to be executed. For example,

```
n=input('Enter the number now ')
if n<0
    disp('This number is negative')
end
```

We can also include the command `else` which executes the command if the Boolean is falue (i.e the condition is not met). For example

```
n=input('Enter the number now ')
if n<0
    disp('This number is negative')
else
    disp(s'Square root is')
    disp(sqrt(n))
end
```

Example
Write a script file to prompt the user to input a value for `d`, a distance in km, and calculate how long it takes to drive that distance at 80 kph. If it is greater than 10 hours, display a message that it will take more than one day.

```
d=input('Enter the distance ');
t=d/80;
if t>10
    disp('It will take more than one day')
end
```

More on IF statements
We can also use the command `if else` to add further commands which will execute if a Boolean is true. For example

```
if (x>2)|(x<0)
   y=2;
elseif x<1
   y=1;
else
   y=0;
end
```

Summary
We have the following three kinds `if` statements

- `if` *boolean expression*
  some statements
  `end`

- `if` *boolean expression*
  some statements
  `else`
  some more statements
  `end`

- `if` *boolean expression*
  some statements
  `elseif` *boolean expression*
  some more statements
  `else`
  some more statements
  `end`

Examples

1. No more than 10 drinks may be served to one person. If more than 10 are ordered, the number is reduced to 10 and a message printed out. Write a script file to prompt the user to enter the number of drinks ordered and display the appropriate messages.

   Sample Outputs
   ```
   You have ordered
   20
   drinks.
   Reduced to
   10


   You have ordered
   5
   drinks.
   ```

   ```
   drink=input('How many drinks? ');
   disp('You have ordered')
   disp(drink)
   disp('drinks.')
   if drinks>20
      drinks=10
      disp('Reduced to')
      disp(drinks)
   end
   ```

2. Write a script file to prompt the user to enter an integer, and then display whether the integer is zero, positive or negative.

   ```
   n=input('Enter an integer ');
   if n>0
      disp('Positive')
   elseif n<0
      disp('Negative')
   else
      disp('Zero')
   end
   ```

3. Grades are to be assigned as follows:

A	80% - 100%
B	65% - 79%
C	50% - 64%.

   Write a script file to prompt the user to input a mark and display the appropriate grade. If the user enters a number greater than 100 or less than zero, display a message that the mark is invalid.

```
mark=input('Enter the mark ');
if mark > 100 | mark < 0
   disp('Invalid mark')
elseif mark >= 80
   disp('A')
elseif mark >= 65
   disp('B')
elseif mark >= 50
   disp('C')
else
   disp('Fail')
end
```

MATHS 162F MATLAB lecture 3.

Today's Topics

- `for` statement
- `while` statement

Revision Examples

- We started with 15 kg of apples. Write a script file to:
  - Prompt the user to enter how many kg have been used today;
  - Display how many were used, and how many are now left.

  Remember that you cannot use more than you have left!

`for` **statement**

```
for variable= start :increment: finish
            some statements
end
```

Note that the `increment` can be omitted in which case it defaults to one or it can be negative in which case the loop counts dow.

Example
Write a script file to calculate the squares of the integers up to 20.

```
% A script file to print out the squares
% from 1 to 20.
for i=1:20
         disp(i)
         disp('squared is')
         disp(i^2)
end
```

We can also count up by twos (or threes etc.) as the following example shows.

Example
Now change the script file to calculate the squares of the odd numbers up to 21.

```
% A script file to print out the odd
% squares to 21.
for i=1:2:21
         disp(i)
         disp('squared is')
         disp(i^2)
end
```

We can also count down

```
% A script file to print out the odd
% squares backwards from 21.
for i=21:-2:1
         disp(i)
         disp('squared is')
         disp(i^2)
end
```

Example
Write a script file to calculate the sum of the integers up to 100.

```
total=0;
for n=1:100
           total=total+n;
end
disp('Sum up to 100 is')
disp(total)
```

`while` statement
The `while` statement is used to execute a loop while a given boolean expression is true.

The syntax is

```
while  boolean expression
      some statements
end
```

Example
Write a script file which prompts the user for a number M and then displays the integers and their squares while the square is less than M.

```
M=input('Enter the maximum number ');
i=1;  i2=i^2;
while i2 < M
   disp(i)
   disp('has square')
   disp(i2)
   i=i+1;
   i2=i^2;
end
```

Use of `for` and `while`
The `for` statement is used when we know in advance how many times we want to execute a loop. For example, to evaluate

$$\sum_{i=1}^{n} i^2$$

we know that the loop must be repeated n times even if we do not know the value of n before we run the program. With the `while` statement we can test for the desired condition each time we execute the loop. For example, to keep repeating some process until the user enters a zero.

Example
Write a script file to repeatedly prompt the user for a student mark and display the student mark along with the grade, until a negative mark is entered. Grades are:
A 80% - 100%, B 65% - 79%, C 50% - 64%.

We will need a `while` loop with a boolean expression to recognise the negative mark.

```
mark=input('Enter a mark ');
while mark > 0
   if mark > 100
      disp('Invalid mark')
   elseif mark >= 80
      disp('A')
   elseif mark >= 65
      disp('B')
   elseif mark >= 50
      disp('C')
   else
      disp('Fail')
   end
   mark=input('Enter a mark (or 0 if finished) ');
end
```

Example
Write a script file to keep a count of how many mobile phones are left. First prompt the user to enter the number available at the start. As they are sold, display each sale and update the number available. When all phones have been sold, display a message that all are sold.

```
phones_left=input('How many phones are there? ');
while phones_left > 0
   sale=input('How many phones are being sold? ')'
   if sale > phones_left
      disp('There are only'
      disp(phones_left)
      disp('Sale reduced to')
      disp(phones_left)
      sales=phones_left
   end
end
disp('All Sold')
```

Example
The following script file asks users to guess values of x for which $x - cos(x)$ will be zero. The variable `tol` is used to indicate how far from zero is acceptable. Variable `max_guess` is used to set the maximum number allowed. (Why?)

```
disp('* When is x-cos(x) is zero? *')
disp(' ')
tol=1e-2;  max_guess=8;
y= ______ ;
n=0;
while abs(y)>tol & ______ < max_guess
           x=input('Enter your guess ');
        y=x-cos(x);
        disp('Value is')
        disp(y)
        n=n+1;
end
if abs(y)<=______
          disp(x)
          disp('is close enough')
          disp('Value is')
          disp(y)
else
          disp('Your guess limit is reached.')
          disp('Bad luck!')
end
```

Suggested answers: Ist: verby=2*tol (to get the loop started). 2nd: `n` 3rd: `tol`

Today's Topics

- Functions

Revision Examples

1. Write a Matlab script file which prompts the user to enter a number and displays a message showing whether the number is less than or equal to 2, or greater than 2.
2. Write a Matlab script file which prompts the user to enter a number, K, and displays the value of the sum
$$s = \sum_{j=1}^{K} j^2.$$
3. Write a Matlab script file which prompts a user to enter a negative number, and keeps prompting until a negative number is entered.

Functions

We can use 'built-in' functions like `sin`, `cos` and `sqrt`. But we can also write our own functions.

Function Examples

Suppose we want to calculate the amount of soda in a can and the price of that soda in dollars/litre. we may want to do this for lots of different soda cans and soda can prices.

Since we're going to have to do the same calculation several times over, we choose to write a **function**. This will help keep the calculation tidy. To calculate the amount of soda we need to calculate the volume of a can. Let h be the height of the can in cm, r its radius in cm, and vol its volume in litres. The formula for the volume in litres is
$$vol = \pi h r^2/1000$$
since there are 1000 cubic centimetres in a litre.

Input:
Height of can
Radius of can
Price of can

Output:
Volume of can/soda
Price per litre of soda

2. Large jaffas
How much chocolate is there in a given number of jaffas of a given diameter?

Input:
Diameter of a jaffa
Number of jaffas

Output:
Volume of chocolate

Syntax of a Function Header

`function [` output `]=` funct-name (input)

(if only one output variable, you can omit the [ ]). There may also be no output variables.

For example,
```
function [y,z]= f(x)
```

The statements that follow the function header will calculate the output variables in terms of the input variables.

The function header and statements are typed into a file. Each function is in a separate file.

Example

```
function [vol,p_l]=can(h,r,p_c)
%returns a vector containing the volume
%vol (litres) and price/litre p_l of can.
%Inputs are height h (cm), radius r (cm),
%price p_c (dollars).
vol=pi*h*r^2/1000;
p_l=p_c/vol;
```

This will be typed into a file and saved as `can`. *It is important that the function name is the same as the file name.*

We can use the function by typing `[v,p]=can(12,3,1)`.

Use of Variables
A function communicates with the rest of the workspace only through the input and output variables.
Any other variables that you use inside the function are known only inside the function. They are not defined in the Command Window workspace for example.

Examples for Functions

1. Write a function file to calculate the volume of chocolate required for the jaffas.

   ```
   function V=choc(d,N)
   r=d/2;
   V=N*pi*r^2;
   ```

2. Write a function file with a student mark as the input variable, and the grade (as a character string) as the output variable.

   ```
   function s=grade(mark)
   if mark > 100 | mark < 0
        s='Invalid';
   elseif mark >= 80
        s='A';
   elseif mark >= 65
         s='B';
   elseif mark >= 50
         s='C';
   else
         s='F';
   end
   ```

3. Write a function file that returns the sum of the positive components and the sum of the negative components of the input vector.

   ```
   function [p,n]=sumpn(a)
   % to find sum of positive and sum of negative components of a
   p=0; n=0;
   for i=1:length(a)
      if a(i) > 0
          p=p+a(i);
       elseif a(i) < 0
          n=n+a(i);
       end
   end
   ```

4. Write a script file for the large jaffas that:
   (a) prompts the user to input the diameter
   (b) prompts the user to input the number
   (c) calls the function in Example 1
   (d) displays the amount of chocolate required.

```
diam=input('What is the diameter of the jaffas? ');
num=input('How many jaffas are to be made? ');
vol=choc(diam,num);
disp('Volume required is ')
disp(vol)
```

MATHS 162 MATLAB lecture 5.

Today's Topics

- Vectors
- Graphing
- Element by element operations

Revision

- Write a Matlab function file with input parameter n and output parameter r where

$$r = \sum_{i=1}^{n} i^3.$$

Vectors
A vector is a one dimensional array.
Some examples are:

```
[1,3,2,9,5],  [-1 3 8 0 -2 1],
1:100,
1:0.1:10,
linspace(0,10,6)
```

We can find how many elements in a vector by using the function `length`.
`length([1,3,2,9,5])` is 5.
`length(1:0.1:10)` is 91.

A vector of length 1 (ie just a number) is called a scalar. The vector `[]` is the null vector. It has length zero.

We can refer to the element of a vector by using its index. For example, if we use
`v=[1,3,2,9,5];`
then we can use `v(3)` to refer to the third element, which is 2.

We can use a vector for the index in order to refer to more than one element. For the vector above, `v(2:4)` refers to the second to fourth elements inclusive of `v`. This will be `[3,2,9]`.

Examples
Let `w=[9,2,10,-5,0,-2,4,1]`.

Write down:

1. `w(5)`
2. `2:3:8`
3. `1:2:8`
4. `w(2:3:8)`
5. `w(8:-3:2)`
6. `w(length(w):-1:1)`

Answers: 1. 0 2. [2, 5, 8] 3. [1, 3, 5, 7] 4. [2, 0, 1] 5. [1, 0, 2] 6. [1, 4, -2, 0, -5, 10, 2, 9].

Input statements, Vectors and Strings
An input statement can prompt the user to enter a vector or a string. The user must enter a vector in square brackets or a string in single quote marks. When the computer displays them, it does not use square brackets or quotes.

Some Vector Functions

`length` returns the length of a vector (i.e. how many elements or components it has).
`min` and `max` returns minimum and maximum elements respectively of the vector.
`sum` returns sum of the elements in vector.
`prod` returns the product of the elements in the vector.
`transpose` returns the transpose of the vector i.e. a row vector becomes a column vector and vice versa. We can also use `v'` instead of `transpose`

Examples

`v=[1 -3 56 -3 2 7];`

`max(v)`= 56
`min(v)`= -3
`min(abs(v))`= 1
`sum(v)`= 60
`sum(abs(v))`= 72
`prod(v)`= 7056
`v'`= `[1; -3; 56; -3; 2; 7]`

Some Vector Operations

You can add and subtract vectors, and multiply vectors by scalars in the usual way.

If you multiply vectors together, i.e. `v*w`, then it is matrix multiplication that you do. This means that you can only multiply a row vector by a column vector (and the result is a scalar), or multiply a column vector by a row vector (and the result is a matrix). All other multiplications will give an error.

`v=[1,3,5,7]; w=[2;4;5;6];`
Find:
`v*w` `w*v`

Answers: `v*w` is 81; `w*v` is $\begin{pmatrix} 2 & 6 & 10 & 14 \\ 4 & 12 & 20 & 28 \\ 5 & 15 & 25 & 35 \\ 6 & 18 & 30 & 42 \end{pmatrix}$.

Example
Write a function with parameters:

- input parameter is a vector, v;
- output parameters are:
  - w which is the first element of v
  - z which is the last element of v
  - s which is the sum of the elements of v

```
function [w, z, s] = xxx(v)
w=v(1);
z=v(length(v));
z=sum(v);
```

Element by Element Operations

Two arrays can be multiplied together element by element. For example, for two vectors, `v` and `w`, `v.*w` is a vector of the products of the corresponding elements of `v` and `w`. The two arrays multiplied will need to have the same size and shape. Also called componentwise operations.

We can also take powers of arrays element by element. Each element of the array is raised to that power to form the result.

Examples
If `v=[-5 0 2 1]; w=[9 1 5 2];` then

`v+w` is `[4 1 7 3]`.

`v-w` is `[-14 -1 -3 -1]`.

`v.*w` is `[-45 0 10 2]`.

`v.^2` is `[25 0 4 1]`.

`v.+w` and `v.-w` will give an error. Adding or subtracting arrays is done element by element anyway, so we do not need these operations.

`v*w` will give an error because we cannot do matrix multiplication with two row vectors.

`v^2` will give an error because we cannot square a row vector using matrix multiplication.

Plotting
When we plot a graph, we make a vector of x values for the horizontal axis and then a vector of y values that correspond to these x values. For example, suppose we want a rough graph of $y = x^2$ from $x = -5$ to $x = 5$.

We could use `x=[-5,-4,-3,-2,-1,0,1,2,3,4,5]`.

To make the `y` vector we need to square *each* of these `x` values. This means we want element by element squaring, not to multiply the vector `x` by itself in matrix multiplication.

So we write `y=x.^2`.

MATLAB makes plotting the vector of values in `x` against the vector of values in `y` very easy:

`plot(x,y)`.

It is also easy to plot multiple sets of data on the same axes, for instance:
`plot(x,x.^2,x,sin(x))`

Most of the built in functions in Matlab, such as `sin` and `cos`, can be used with vectors or matrices. The function is applied to each element. If we write our own functions that we might want to use for plotting we should make them suitable for vector input parameters.

```
>> x=linspace(0,2*pi,50);
>> plot(x,cos(x))
```

The first line sets up a variable called `x` which is 50 equally spaced numbers starting with 0 and ending with 2π. The second statement plots the values in `x` along the horizontal axis and the cos of each value up the vertical axis.

Adding Labels

```
>> xlabel('x')
>> ylabel('cos x')
>> title('A graph')
```

Multiple Graphs

```
>> plot(x,cos(x),x,sin(x))
```

Alternatively

```
>> x=linspace(0,2*pi,50);
>> y=x+2*cos(x);
>> plot(x,y)
```

If we write our own functions
that we might want to use for plotting
we should make them suitable for
vector input parameters.

Example
Write a Matlab function to evaluate the function

$$f(x) = \frac{2x+1}{x+3}e^{-x}.$$

Write a script file to plot a graph of f using 50 values of x from 0 to 20. Be sure to use the function that is written above!

The function file will be:

```
function y=f(x)
y=(2*x+1)./(x+3).*exp(-x);
```

The script file will be:

```
x=linspace(0,20,50);
y=f(x);
plot(x,y)
```

Today's Topics

- Matrices

Revision

- Write a Matlab function to evaluate the function

$$f(x) = \frac{\sin x + 1}{4x - 3}.$$

- Write a Matlab script file, using the above Matlab function, to plot a graph of f using values of x from 0 to 20 with a spacing of 0.1 between x values.

Introduction to 2-D Arrays - Matrices

Suppose we have some students' results from an assignment marked out of 10, a test out of 50 and an exam out of 100. In the final mark, the assignment is worth 20%, the test 30% and the exam 50%.

Name	Assignment	Test	Exam
John	8	36	81
Ying	7	44	75
Jenny	5	39	77
Alice	9	29	62

How do we calculate the final marks?
Multiply the assignment mark by 2
Multiply the test mark by 0.6
Multiply the exam mark by 0.5 then add them.

Writing the students' marks as a matrix and the factors above as a column vector, we get

$$\begin{pmatrix} 8 & 36 & 81 \\ 7 & 44 & 75 \\ 5 & 39 & 77 \\ 9 & 29 & 62 \end{pmatrix} \begin{pmatrix} 2.0 \\ 0.6 \\ 0.5 \end{pmatrix}$$

2-D Arrays - Matrices

We have studied 1-D arrays (vectors) which are like a list of numbers. The 2-D arrays go both across and down, like the matrices we study in mathematics.

Example

The matrix

$$\begin{pmatrix} 1 & -5 & -4 \\ -2 & 7 & 0 \\ 5 & 0 & -1 \end{pmatrix}$$

can be written in Matlab by
`A=[1 -5 -4; -2 7 0; 5 0 -1]` or

`A=[1, -5 ,-4; -2 ,7, 0; 5, 0, -1]`

The numbers in a row are separated by spaces or commas (as in a vector), and the end of the row is marked by a `;`.

If we want to refer to the element in the 3rd row and the 1st column then we use `A(3,1)`. Note that the row number is the first index and the column number is the second index.

If a matrix is multiplied by a vector in Matlab, then the result is the same as the usual matrix/vector multiplication.

If `v=[1; 4; -2]` find `A*v`.

$$Av = \begin{pmatrix} 1 & -5 & -4 \\ -2 & 7 & 0 \\ 5 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 4 \\ -2 \end{pmatrix}$$

Answer:

$$\begin{pmatrix} -11 \\ 26 \\ 7 \end{pmatrix}$$

Exercise
Write Matlab statements to calculate

$$\begin{pmatrix} 1 & 0 & 2 \\ 2 & 0 & 0 \\ 2 & 5 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ -2 \\ 7 \end{pmatrix}$$

Picking out rows and columns
If we want to pick out just the first row of matrix `A`, then we use `A(1,:)`. It picks out the components which have their first index as 1.
To pick out the second column use `A(:,2)`. This gives us a vector.

Examples

```
C=[25,14,0,29,21;19,12,10,28,18];
```

Find `C(1,:)` and `C(:,2)`.
Answers: `C(1,:)=[25,14, 0,29 ,21]` and `C(:,2)= [14;12]`.

Examples

1. Write a function file with input parameter matrix `A` and output parameter `v` which is the first row of `A` as a *column* vector.

   ```
   function v=firstrc(A)
   v=A(1,:)';
   ```

2. Write a function file to find the maximum element in a particular row of a matrix. Use the matrix and the number of the row as input parameters, and the set the output parameter to be the maximum element in that row.

   ```
   function m=max_in_row(A,p)
   v=max(A(p,:));
   ```

`whos`
This is the command to return the number and size of the variables in use.

Some Matrix Functions

`size` returns the dimensions of a matrix (i.e. how many rows and columns it has).

`sum` returns the sum over rows or columns.
`sum(A,1)` is a vector of the sums of the columns.
`sum(A,2)` is a vector of the sums of the rows.

`prod` as for `sum` but with products.

`transpose` returns the transpose of the matrix, i.e. a rows become columns and vice versa.

Some Matrix Operations

You can add and subtract matrices, and multiply matrices by scalars in the usual way.

If you multiply two matrices together then it is matrix multiplication and the number of columns of the first matrix must equal the number of rows of the second.

If you want the power of a matrix then the matrix must square, i.e. the number of rows equals the number of columns.

Matrices can also be multiplied together, or powers may be evaluated, element by element as for vectors.

Example
Write a function with input parameter a matrix `K`. The output parameter will be a column vector of the sum of the absolute values of the elements in each row.

```
function w=absmax(K)
w=sum(abs(K),2);
```

Example
Suppose we have a matrix of marks for 3 assignments for a group of 8 students, such as

$$\begin{pmatrix} 25 & 14 & 0 & 29 & 21 & 18 & 27 & 30 \\ 19 & 12 & 10 & 28 & 18 & 27 & 30 & 26 \\ 26 & 18 & 16 & 25 & 20 & 29 & 29 & 28 \end{pmatrix}$$

How will we enter this matrix into Matlab?

Write Matlab statements to find:

1. A row vector of the total marks for each student.

2. A column vector for the highest marks in each assignment.

3. How many got more than 20 marks for Assignment 1.

4. How many marks (over all assignments) were below 12.

5. A column vector of assignment means.

Answers:
Enter the matrix as
`M=[25 14 0 29 21 18 27 30;19 12 10 28 18 27 10 26;26 18 16 25 20 29 29 28]`

1. `sum(M,1)`
2. `[max(M(1,:));max(M(2,:));max(M(3,:))]`
3. `sum(sum(M<12))`
4. `mean(M,2)`

Example
Write a Matlab function file with

- input parameters A, a matrix, and n, an integer,
- output parameter p where $p = -1$ if there is no column n in A; otherwise p is the maximum absolute value in column n of A

```
function p=maxabs(A,n);
[q,r]=size(A);
if n>r | n<1
    p=-1;
else
    p=max(abs(A(:,n)));
end
```

Today's Topics

- Systems of Linear Equations
- `rand` and `ceil` functions

Revision Examples

- Write a Matlab script file which prompts the user to enter a matrix and then displays how many rows and columns in the matrix.

- During a given week, the maximum temperature is to be recorded every day at several different locations. Assume the data has been stored as a matrix where each row represents a location. Write a Matlab script file which prompts the user to enter the matrix of data then
  - displays the mean maximum temperature for the week at each location
  - for each day, displays which location has the highest maximum temperature

Systems of Linear Equations
A system of linear equations can be written in matrix form, and then be solved by Matlab.
For example,

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 32 \\ 77 \\ 68 \end{pmatrix}.$$

If A is the matrix on the left, x is the vector on the left and b is the vector on the right, then we can write this as

$$Ax = b.$$

The solution will be

$$x = A^{-1}b.$$

This can be written in Matlab as `A\b`.

Similarly the system

$$xA = b$$

has the solution

$$x = bA^{-1}$$

which is written in Matlab as `b/A`.

Example
Write Matlab to solve the above system of linear equations.

```
A=[1 2 3;4 5 6;7 8 0];
b=[32;77;68];
x=A\b
```

Example
Four friends want to buy some stationery. The prices and their orders are

Lecture Pad	\$3.20
Pen	\$2.50
Ruler	\$1.40

	Pad	Pen	Ruler
Victoria	3	2	1
Jun	4	3	1
Anne	2	1	2
Henare	3	1	0

We can write this data as a matrix, N, of the numbers wanted, and a vector, p, of prices.

$$N = \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 1 \\ 2 & 1 & 2 \\ 3 & 1 & 0 \end{bmatrix}, \qquad p = \begin{bmatrix} 3.20 \\ 2.50 \\ 1.40 \end{bmatrix}.$$

Let $q = Np$, (matrix vector multiplication).

$$q = \begin{bmatrix} 3 \times 3.20 + 2 \times 2.50 + 1 \times 1.40 \\ 4 \times 3.20 + 3 \times 2.50 + 1 \times 1.40 \\ 2 \times 3.20 + 1 \times 2.50 + 2 \times 1.40 \\ 3 \times 3.20 + 1 \times 2.50 + 0 \times 1.40 \end{bmatrix} = \begin{bmatrix} 16.00 \\ 21.70 \\ 11.70 \\ 12.10 \end{bmatrix}.$$

What does q represent?
Answer: The cost of each person's stationery.

Example
Write a Matlab script file that

- sets the value of the vector p of stationery prices above
- prompts the user to enter a matrix, N, of the orders (numbers wanted by each person)
- displays how many orders (people) there were
- calculates the total amount each person owes
- numbering the people $1, 2, 3, \ldots$, use a `for` loop to display for each person the number of that person and the amount they owe.

```
% To calculate the stationery amounts for each person
prices=[3.20;2.50;1.40];
orders=input('Enter the orders as a matrix, one row per person ');
[m,n]=size(orders);
disp('Number of orders is ')
disp(m)
totals=orders*prices;
for i=1:m
    disp([i,totals(i)])
end
```

Example
After the friends had ordered their stationery, they found the prices had changed. All we know is the amounts they owe are

Victoria	\$16.30
Jun	\$22.00
Anne	\$12.20
Henare	\$12.30

Write a Matlab script file to set the matrix of the orders and the vector of the amounts owed, and to use that information to find what the prices have changed to.

```
% To calculate the stationery prices
orders= [3 2 1;4 3 1;2 1 2;3 1 0];
totals=[16.30;22.00;12.20;12.30];
prices=orders\totals;
disp('The prices are now: ')
disp(prices)
```

`rand` **function**
The function `rand` produces a random number between 0 and 1.

`ceil` **function**
This function rounds to the next integer towards infinity. For example, `ceil(8.2)=9` and `ceil(-8.2)=-8`. This function can be used with `rand` to produce random integers. When we are simulating the tossing of dice, for example, we need random integers from 1 to 6. We can use `ceil(6*rand)`.

Example

Write a script file which simulates the sum of three six sided dice.

```
% To simulate the throwing of three dice
d=ceil(6*rand(3,1));
s=sum(d);
disp('Sum of the three dice is ')
disp(s)
```

Example

Write a script file to

- prompt the user to enter an integer `n`
- make a square matrix with `n` rows and columns of random numbers between 0 and 1
- use the above function to find the maximum element in the last row

```
% Find max abs element in last row of random marix
n=input('Please enter a positive integer ');
A=rand(n,n)
maxAn=maxabs(A,n);
disp('Max abs value in last row is')
disp(maxAn)
```

MATHS 162 MATLAB lecture 8.

Today's Topics

- Using `sprintf` for better looking output

Better Looking Output

Use function `sprintf` to produce strings which include numerical results.
For example, in the well depth program

```
disp(sprintf('Depth is %4.2f metres.',depth))
```

The output would then be

```
Depth is 18.98 metres.
```

The `%4.2f` indicates that a value is to be put in here. It will have 2 decimal places and there will be 4 digits or signs overall. There are other formats described below.

Example

Write a script file which prompts the user to enter a number and then displays the square of that number.

```
n=input('Enter a number ');
disp(sprintf('The square of %4.0f is %4.0f.',n,n^2))
```

The output from this will be

```
Enter a number 4
The square of    4 is   16.
```

Notice that there are extra spaces before the 4 and the 16 because we did not know in advance how many positions to allow for. One way to fix this is to use 0 for the total width. If the total width is too small to fit in the answer (as it will be if we use 0), then Matlab just uses as many positions as are necessary.

Conversion characters

Specifier	Description
`d`	Decimal notation - good for integers
`e`	Exponential notation
`f`	Fixed point notation
`g`	The more compact of `e` or `f`
`\n`	New line
`\t`	Tab

Examples

Command	Result
`sprintf('%d ',12^3)`	1728
`sprintf('%7.3e ',212.5^3)`	9.596e+06
`sprintf('%7.5e ',212.5^3)`	9.59570e+06
`sprintf('%4.4f ',exp(1))`	2.7183
`sprintf('%4.4f \n',exp(1),exp(2))`	2.7183 7.3891

Displaying Matrices and Vectors

When there are matrices or vectors to display, often we will not know when we write the Matlab program how big the matrix or vector will be. However we can take advantage of the fact that `sprintf` keeps using the specifiers over and over until all elements are displayed.

Example
Consider this script file.

```
% Displaying matrices
M=[exp(1:4); exp(0:-1:-3)];
disp(sprintf('%10.4f',M))
```

The output will be

```
2.7183    1.0000    7.3891    0.3679   20.0855    0.1353   54.5982    0.0498
```

Notes:

- The specifier `%10.4f` has been used for each element of the matrix.
- The matrix has been displayed as a vector, ordered by rows.
- The specifier allows the values to have width 10; the values less width so there are spaces between them.

Alternatively, we could display the matrix in rows as follows.

```
% Displaying matrices
M=[exp(1:4); exp(0:-1:-3)];
n=size(M,1);   % finds the number of rows in M
for i=1:n
disp(sprintf('%10.4f',M(i,:)))
end
```

The output will be

```
    2.7183    7.3891   20.0855   54.5982
    1.0000    0.3679    0.1353    0.0498
```

Tables
When displaying tables, we need to be careful about the widths we use in the specifiers.

Example

```
% Displaying tables using tab
% Display the square roots of the integers  to 3 d.p.
disp('Integer Square Root')
for i=1:40:200
    disp(sprintf('%d \t %4.3f',i,sqrt(i)))
end
```

produces the output

```
Integer Square Root
1        1.000
41       6.403
81       9.000
121      11.000
161      12.689
```

Note that the integers are all lined up on the left because using `%d` means that they only use as much space as they need. The tab has lined up the square roots. However the specifier `%4.3f'` only allows for a width of 4. This is suitable for the first three square roots but the last two require five so they finish one place further on. To make the decimal points line up in the square roots and to 'right justify' the integers we would use the following commands.

```
disp('Integer Square Root')
for i=1:40:200
    disp(sprintf('%3d \t %6.3f',i,sqrt(i)))
end
```

```
Integer Square Root
  1    1.000
 41    6.403
 81    9.000
121   11.000
161   12.689
```

Alternatively, we could produce a table by putting extra spaces in the string, or allowing for extra width.

```
disp('Integer  Square Root  Cube Root')
for i=1:40:200
    disp(sprintf('%3d %12.3f %12.3f',i,sqrt(i),i^(1/3)))
end
```

```
Integer  Square Root  Cube Root
  1        1.000        1.000
 41        6.403        3.448
 81        9.000        4.327
121       11.000        4.946
161       12.689        5.440
```

or

```
disp('Integer   Square Root  Cube Root')
for i=1:40:200
    disp(sprintf('%3d      %8.3f      %8.3f',i,sqrt(i),i^(1/3)))
end
```

```
Integer   Square Root  Cube Root
  1         1.000         1.000
 41         6.403         3.448
 81         9.000         4.327
121        11.000         4.946
161        12.689         5.440
```