5 Week

FUNCTIONS

In addition to the basic operations, we can also call the internal functions to use, such as the trigonometry functions, logarithmic, exponential functions etc. A function is use as “functionname()”, where the input value to the function is given inside the parentheses.

```
>> sqrt(4)
ans =
 2
```

"sqrt()" is a function that takes the square root of the input variable.

Trigonometry Functions

Note, when you use trigonometric functions, such as sine and cosine, the input is an angle measured in radiance. If you know the angle measured in degrees, you can do it as

```
>> sin(30*pi/180)
ans =
  0.5000
```

Where pi (π) =3.1416 (built in constant). The above express converts the angle in degrees to radiances first and then evaluated by the sine function. Similarly you can do

```
>> cos(30*pi/180)
ans =
0.8660
>> tan(30*pi/180)
ans =
   0.5774
```

II. The *trigonometric* Function

sin (x)	Sine	csc (x)	Cosecant
sinh (x)	Hyperbolic sine	csch (x)	Hyperbolic cosecant
asin (x)	Inv. sine	acsc (x)	Inv. Cosecant
asinh (x)	Inv. hyper. Sine	acsch (x)	Inv. hyper. cosecant
cos (x)	Cosine	sec (x)	Secant
cosh (x)	Hyberbolic cosine	sech (x)	Hyberbolic Secant
acos (x)	Inv. cosine	asec (x)	Inv. Secant
acosh (x)	Inv. hyber. Cosine	asech (x)	Inv. hyber. Secant
tan (x)	Tangent	cot (x)	Cotangent
tanh (x)	Hyper. Tangent	coth (x)	Hyper. Cotangent
atan (x)	Inv. Tangent	acot (x)	Inv. Cotangent
atanh (x)	Inv. hyber. Tangent	acoth (x)	Inv. hyber. Cotangent

Tip : Trigonometric functions use radians but not degree; however, the angle in graphics (polar plot) views in degree.

EXAMPLE

Please find the value for the following function with a given variable x=45.

Function is $(\sin 4x)-(2\cos x)^3$

1. Way

```
» x = pi/3;
» sin(4*x)-(2*cos(x))^3
ans =
      -2.83
```

2. Way

```
» x = 45;
» sin(4*x*pi/180)-(2*cos(x*pi/180))^3
ans =
      -2.83
```

Exponential Functions

we use *exp(x)* to calculate the xth power to *e*.

e =2.718.

```
>> exp(1)
ans =
    2.7183
>> exp(2)
ans =
7.3891
```

Logaritmic Functions

For logarithms,

- the natural logarithm lnx in mathematics is written log(x) in MATLAB, and

lnx=logex (in mathematics)

For x variable

Lnx **in mathematics**

log(x) **in MATLAB**

Example

ln1	**ln10**	**ln2**
» log(1) ans = 0	log(10) ans = 2.3026	» log(2) ans = 0.6931

log ten of x in mathematics is log10(x) in MATLAB

Example

log1	**log10**	**log5**
» log10(1) ans = 0	» log10(10) ans = 1	» log10(5) ans = 0.6990

We know that *ln(1)* and *lg(1)* are both 0!!! İn mathematics

round, floor, ceiling. Rounding of number

round(number) - rounding to the closest integer

floor(number)-rounding towards lesser integer

ceil(number) -rounding towards greater integer

Example

math:round(45.50) -Will equal 46

math:floor(45.60) -Will equal 45

math:ceil(45.20) -Will equal 46

math:round(-4.5) -Will equal -4

math:floor(-4.6) -Will equal -5

math:ceil(-4.20) -Will equal -4

:fix(-4.20)=-4.0

The If Statement

Relational operators

Relational operators are used to specify the conditions for the for, elseif and while statements.The syntax of these statements is given in the following table:

Symbol	Relation
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

TABLE 1: **Relational Operators**

NOTE!!! **== is not the same as = ; MATLAB's treats them very differently.**
== compares two values, while = assigns a value to a variable.

Standard logic operators

"&" *(and)*

“¦” (or),

There are times when you want certain parts of your program to be executed only in limited circumstances. The way to do that is to put the code within an "if" statement.

The most basic structure for an "if" statement is the following:

```
if (condition statement)
(matlab commands)
end
```

More complicated structures are also possible including combinations like the following:

```
if (condition statement 1)
(matlab commands 1)
else
(matlab commands 2)
end
```

```
if (condition statement 1)
(matlab commands 1)
elseif (condition statement 2)
(matlab commands 2)
elseif (condition statement 3)
(matlab commands 3)
. . .
else (matlab commands )
end
```

EXAMPLE 1: For example, suppose that we had a program which checked the value of some variable, *a say, and if it's value was larger then 3 we wanted to consider half that value then* we would use the following matlab commands in a script M-file:

```
a=pi;
if a>3
   a=a/2
end
```

EXAMPLE 2: Whereas in the previous example, we only specified an outcome if the variable *a was bigger* than 3, this time we could specify outcomes depending on whether *a is smaller than 1, between 1* and three or if it was bigger than 3:

```
a=exp(1);
if a<1
   a=2*a
elseif 1<a<3
   a=a-1
else
   a=a/2
end
```

Since *a lay between 1 and 3, the elseif part of the loop was utilized*

a=

1.7183

EXAMPLE 3: For example to check to see if *a* is less than *b* and at the same time *b* is greater than or equal to *c* you would use the following commands:

```
if (a < b) | (b >= c)
Matlab commands
 else
Matlab commands
end
```

Switch – Case Construct

- Syntax :

```
SWITCH switch_expr
      CASE case_expr,
        statement, ..., statement
      CASE {case_expr1, case_expr2, case_expr3,...}
        statement, ..., statement
      ...
      OTHERWISE,
        statement, ..., statement
    END
```

Usage

- When there is one variable to execute one **and only one** of many options to be considered.

Eg : Say grading into A+,A,B+,B,…,Fail

The advantage of switch is that the above problem would take a complicated nested if structure but only one level switch-case structure.

Example

- Create a menu which asks the user to choose from options 1-5 and according to the choice either adds,subtracts, multiplies, finds maximum or finds minimum of 2 given numbers

Program

```
disp('1. Add the numbers');
disp('2. Find difference');
disp('3. Multiply');
disp('4. Find Maximum');
disp('5. Find Minimum');
disp(' ');
ch=input('Enter Your Choice : ');

x=input('Enter the first of the 2 numbers');
y=input('Enter the second of the 2 numbers');
switch (ch)
case 1 ,
   value=x+y
case 2 ,
   value=x-y
case 3 ,
   value=x*y
case 4,:
   value=max(x,y)
case 5,:
   value=min(x,y)
end
```

TASK 4: Use switch-case to solve the foll. problem

Input elements of a nxn matrix [A], if it is invertible find its inverse [B], print ½ B if det(A) is positive , if det. is negative print 2B.

If A is not invertible print A+A'. In all cases print the det of A

Remember, short is sweet

Solution to TASK 4

```
A=input('Enter a square matrix A');

d= det(A);

switch (sign(d))
case 1 , disp(['det. A is positive and is equal to ' num2str(d) ]);
   B=inv(A);
   0.5*B
case -1 , disp(['det. A is positive and is equal to ' num2str(d) ]);
   B=inv(A);
   2*B
case 0, disp(['det. A is zero']);
  A+A'
end
```

break command –exits for loop ; used in case of error.

continue command –ends one for loop iteration ; crudely equivalent to GOTO end

User-defined scripts & functions

- MATLAB is built around commands & functions: both sets are computer codes that accept input from the user and provide output. The latter does not use or save variables in the workspace.
- Functions and M-file command scripts allow you to do technical computing efficiently; well designed code helps you to tasks multiple times.
- It is necessary for you, as the programmer, to know exactly how a function performs its task; otherwise, how would you know that it is doing the task correctly.

Concept on function M-files

- User-defined functions are similar to the MATLAB pre-defined functions.
- They take a certain number of inputs, perform some operation, and give output(s).
- Just as with MATLAB built-in functions, we need to know what they are supposed to do, and know that it does it correctly.

Syntax for functions

- Calling a user-defined function:
  - my_function(x)
- Defining a user-defined function:
  - function y = my_function(x)
  - x is a value that is the input to my_function.
  - my_function performs a functional operation.
  - y is a value that is the output of my_function.
- Functions must be written in M-files. The M-file must have the same name as the function.

Notes on functions

- The convention for naming functions is the same as for variables. A function name must start with a letter, should be meaningful, should not use the name of an existing function, and should not be excessively long.
- It is important that you give meaningful variable names to variables inside a function that you write, so that you and others can understand what the function does.
- Comments become extremely important in functions. Comments help both you and anyone who might use the function to understand what it does.

Examples of function M-files

```
function volume = volume_sphere(radius)
    volume = (4/3)*pi.*(radius^3);

function perimeter = perimeter_square(side)
    perimeter = 4 .* side;

function root = square_root(x)
    root = x.^(1/2);
```

Using a function in the Command Window

- User-defined functions are accessed in the same way as any MATLAB built-in functions.
- The function M-file, in this case the perimeter_square function from the previous slide, must be saved as "perimeter_square.m" in your current directory (otherwise it must be in the path)*.
- You use it by typing in the command window:

```
perimeter_square(4)

ans=16
```

*Note: The directory that your function is saved in MUST be in the path; otherwise MATLAB will not be able to find and run the file.

Input/output & functions

- Several kinds of functions can be created with different combinations of input and/or output:
  - Functions with input and output arguments
  - Functions with input arguments and no output arguments
  - Functions with output arguments and no input arguments
  - Functions with neither input arguments nor output arguments

Function with multiple inputs

- User-defined functions can have a number of input parameters. They can also have a number of output parameters.
- Recall the constant acceleration equation of free fall:
  - $x = x_o + v_o t + (1/2)at^2$
  - This equation can be coded as a user-defined MATLAB function as follows:

```
function x = displacement(xo, vo, a, t)
   x = xo + vo.*t + (1/2).*a.*t.^2;
```

Function with multiple inputs and outputs

- The acceleration of a particle and the force acting on it are as follows:
  - a = (v2-v1)/(t2-t1) % An approximation!
  - F = ma % Newton's second law
  - A user-defined function can be created to perform both of the calculations.

```
function [a, F] =
     acceleration_calculation(v2,v1,t2,t1,m)
  a = (v2-v1)./(t2-t1);
  F = m.*a;
```

Hands on

- Use the acceleration_calculation function to calculate the acceleration and force on an object with the following properties:

v1=4 m/s, v2=7 m/s, m= 2 kg,

t1= 0 s, t2= 10 s

Since acceleration_calculation has two outputs, we must set it equal to two variables when we call it. The syntax is:
[acceleration, force]=acceleration_calculation(v2, v1, t2, t1, m).

The same method must be applied to all functions with multiple outputs in order for them to work properly.

Functions with no input

- If you need to access some constant value a number of times, instead of hard-coding the constant value every time it is needed, we can hard-code it once into a function that has no input.
- When the constant is needed, its value is called via the function.
- As an example, if we needed, for whatever reason, the mass of the earth (in kg) multiple times, we could write a function to store it as follows:

```
function mass_of_earth = moe()
  mass_of_earth = 5.976E24;
```

Functions with no output

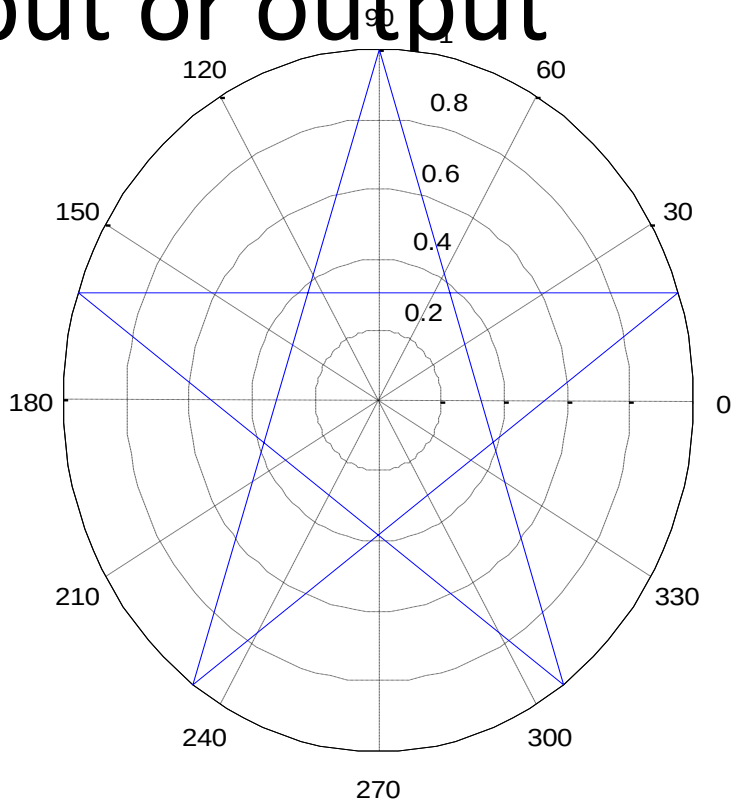
- Functions with no output can be used for many purposes, such as to plot figures from a set of input data or print input information to the command window (output here implies results from operations on input; no just displays).
- The built-in MATLAB function tic has no output. It starts a timer going for use with another built-in MATLAB function, toc.

Functions with no input or output

- A function with no input or output consists of some hard-coded information that is to be used in a specific procedure.
- An example of this is a function that plots a specific figure. Plotting the figure is the only thing the function does; it does not have any input or operations that lead to computed values for output.

A Function with no input or output

```
function [] = star()
  theta = pi/2 : 0.8 * pi :
          4.8 * pi;
  r = [1 1 1 1 1 1];
  polar(theta, r);
```



Although the star function generates a plot, in the strictest sense it does not have any output. If you try to evaluate the expression A=star in MATLAB, you will get an error, because star has no output.

Notes on variables

- Variables that are created inside of a user-defined function may only be accessed from inside of that function. They are referred to as local variables.
- After the function completes its operations, the local variables are deleted from memory.
- The only variable that appears in the workspace is the output, if any, of the function.
- Conversely, functions cannot access variables from the workspace (with the exception of any input parameters they might have or “global variables”---see MATLAB help on this matter).

The **`type`** command

- The type command followed by the name of an M-file prints the contents of the M-file onto the Command Window. This includes any user-defined or built-in functions and command scripts available in MATLAB. Using the type command instead of the editor to view the contents of M-files can prevent you from accidentally changing the contents of your files.
- The type command is an alternative to using the Help Menu or editor for looking up details associated with functions & commands.

Exercises

- Write a function that converts Celsius temperatures into Fahrenheit.
- Write a function that takes force, displacement, and angle (in degrees) as input and outputs work. Use MATLAB help to find a cosine function that uses degrees.

Summary

- Function concept and syntax
- Different kinds of functions
  - Functions with input and output
  - Functions with input only
  - Functions with output only
  - Functions with no input or output