4. 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.

Trigonometric Functions

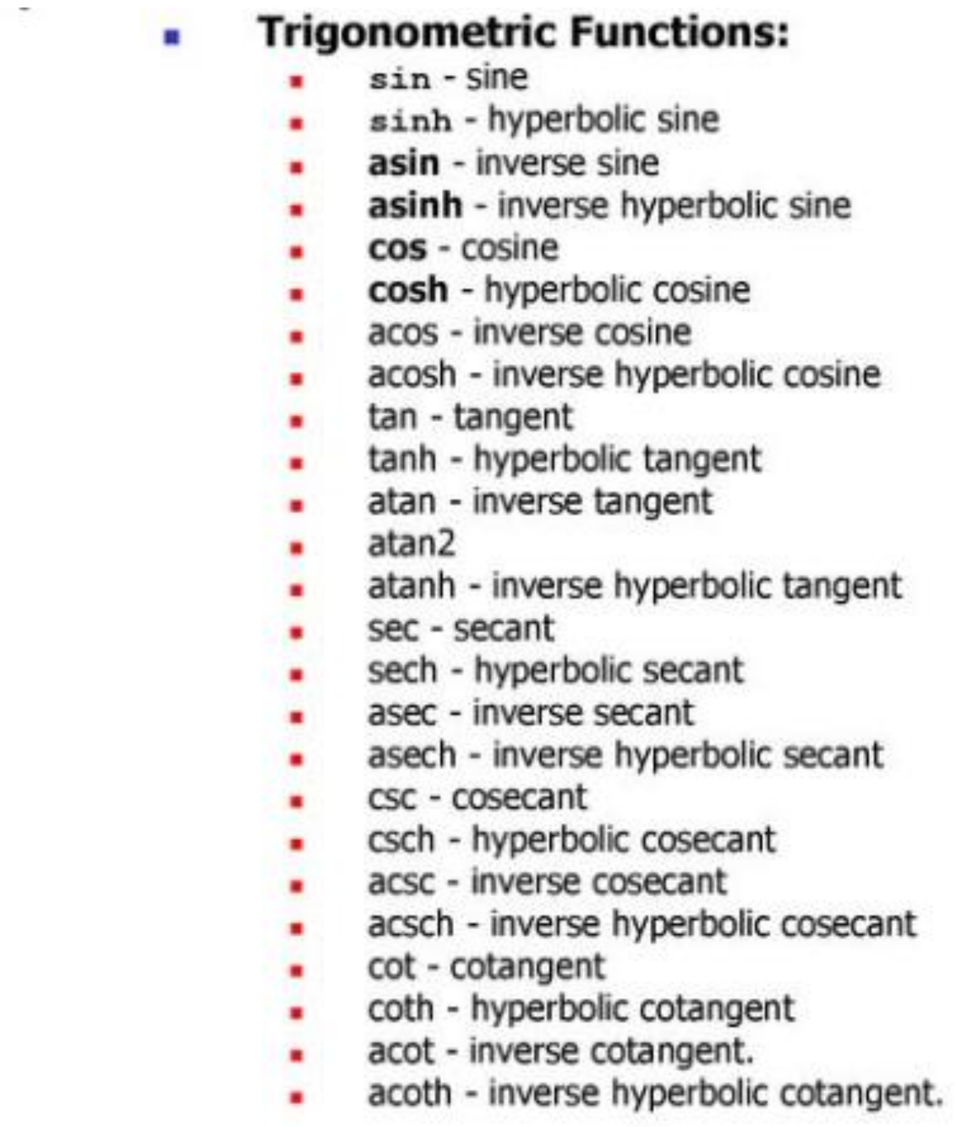
Note, when you use trigonometric functions, such as sine and cosine, the input is an angle measured in radiance. If you know the angle measured in degrees, you can do it as

```
>> 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
```

Trigonometric Functions



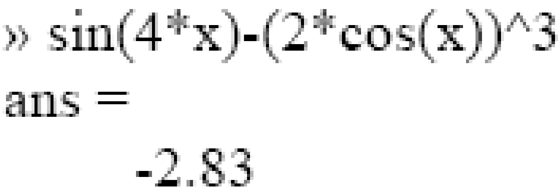
- **Trigonometric Functions:**
  - `sin` - sine
  - `sinh` - hyperbolic sine
  - **asin** - inverse sine
  - **asinh** - inverse hyperbolic sine
  - **cos** - cosine
  - **cosh** - hyperbolic cosine
  - acos - inverse cosine
  - acosh - inverse hyperbolic cosine
  - tan - tangent
  - tanh - hyperbolic tangent
  - atan - inverse tangent
  - atan2
  - atanh - inverse hyperbolic tangent
  - sec - secant
  - sech - hyperbolic secant
  - asec - inverse secant
  - asech - inverse hyperbolic secant
  - csc - cosecant
  - csch - hyperbolic cosecant
  - acsc - inverse cosecant
  - acsch - inverse hyperbolic cosecant
  - cot - cotangent
  - coth - hyperbolic cotangent
  - acot - inverse cotangent.
  - acoth - inverse hyperbolic cotangent.

EXAMPLE

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

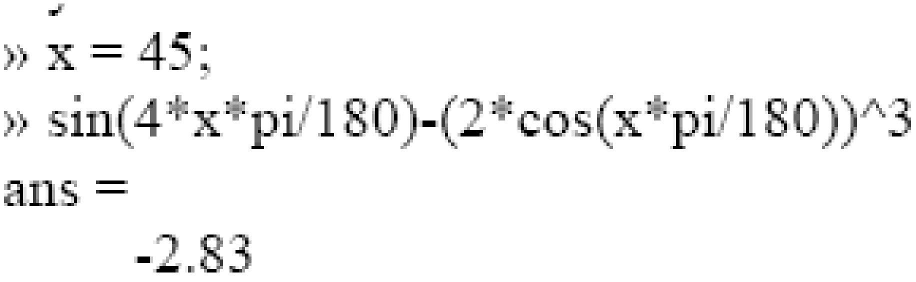
Function is $(\sin 4x)\text{-}(2\cos x)^3$: $x=45^\circ$

1. Way



```
» 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,

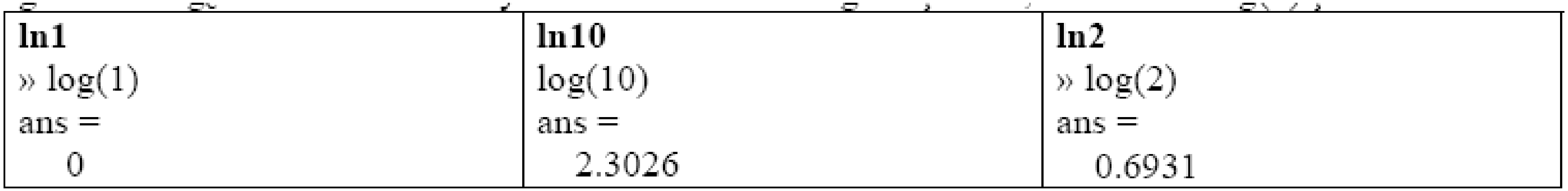
- the natural logarithm lnx in mathematics is written log(x) in MATLAB, and

lnx=log$_e$x (in mathematics)

For x variable

Lnx	**in mathematics**
log(x)	**in MATLAB**

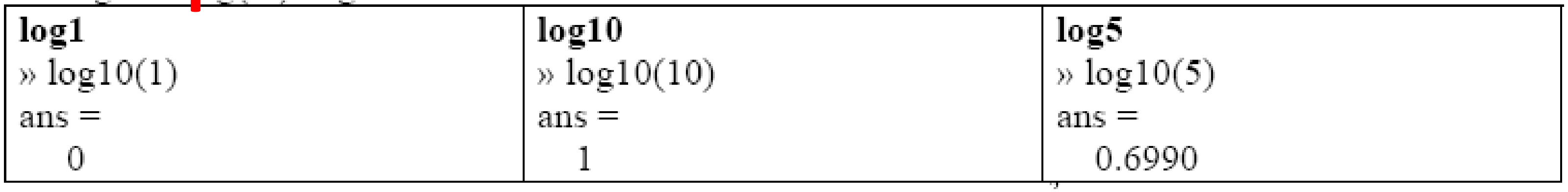
Example



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

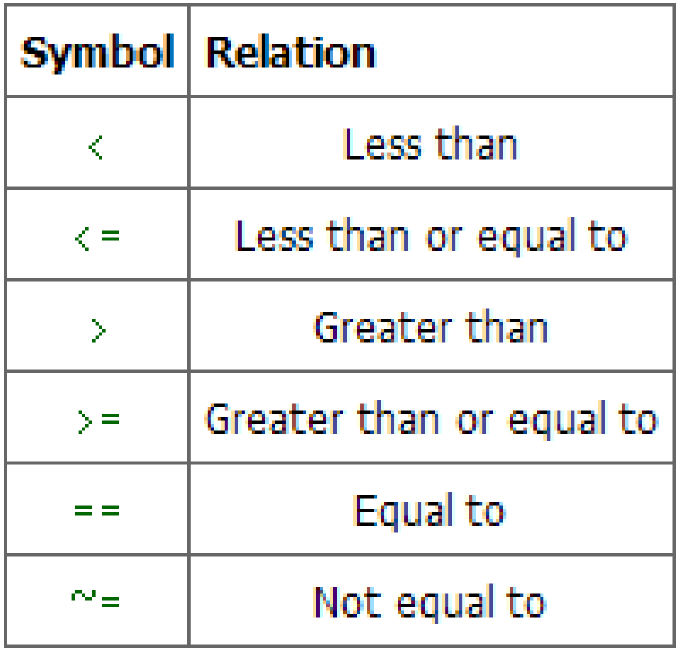
math:floor(-4.6) -Will equal -5

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

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),*

“~ *“(not).*

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:

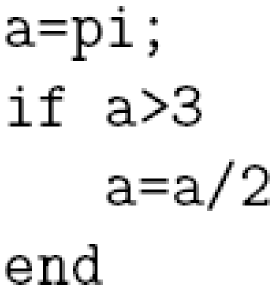
```
if (condition statement)
(matlab commands)
end
```

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

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

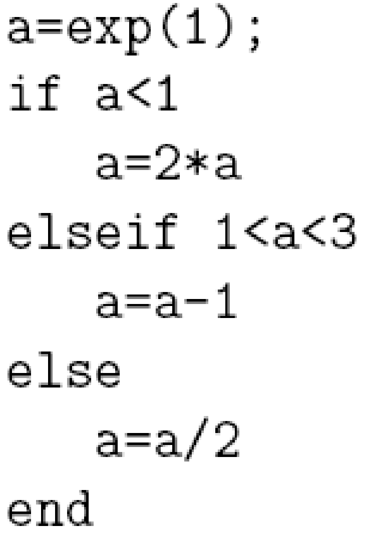
```
if (condition statement 1)
(matlab commands 1)
elseif (condition statement 2)
(matlab commands 2)
elseif (condition statement 3)
(matlab commands 3)
. . .
else (matlab commands )
end
```

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
```

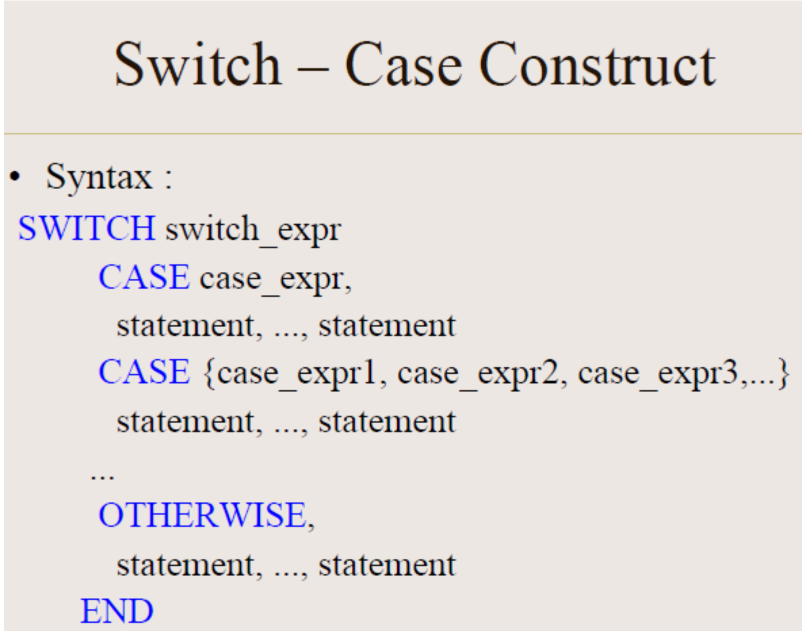
Since *a lay between 1 and 3, the elseif part of the loop was utilized*

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:

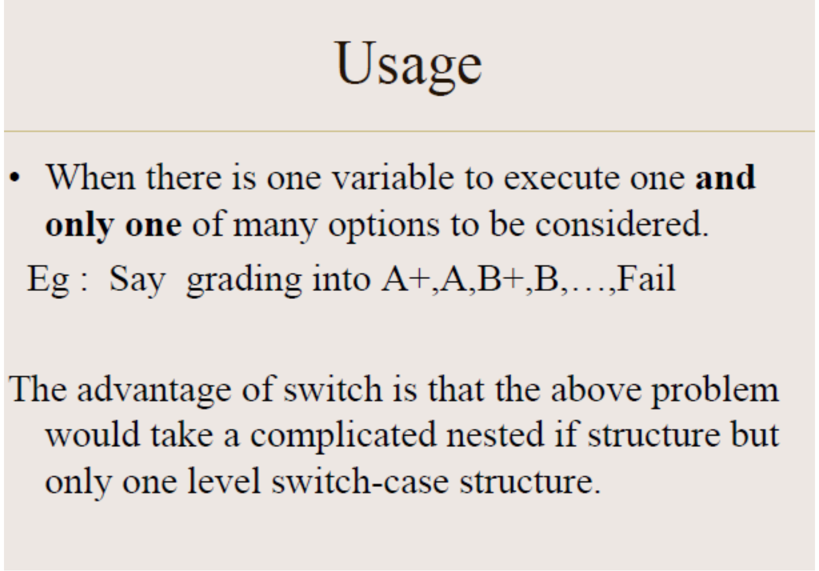
```
if (a < b) & (b >= c)
Matlab commands
end
```



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