**TRIGONOMETRIC FUNCTIONS**

* Perform trigonometric function
* The functions are sin, cos, tan, **asin, acos, atan, sinh, cosh, tanh**, **asinh, acosh, atanh,cot,sec,coth,sech** which take take radian inputs.i.e All angles are in radians.
* asin, acos, atan, asinh, acosh are inverse trig functions.
* They all follow the same syntax

Examples

Y=sin(X)

Y=asin(X) → sine inverse of X.

Where:

Y=output variable

X=input angle in radians

NB: You can write the same syntax for the other functions.

For Degree inputs

sind,cosd,tand,cotd,secd and asind,acosd,atand,acotd,asecd for inverse trig functions.All angles in degrees.

Examples

Type the following at the command prompt

1.>> a=60;

Convert a into radians

Find

a.cos of the result

b.sin of the result

c.tan of the result

d.find their inverse.

2.find

a.sind(a)

b.cosd(a)

c.tand(a)

d.find their inverse

3.compare the answers .

## Flow Control

## In this tutorial we will assume that you know how to create vectors,matrices,know how to index into them.

### Matlab has four kinds of statements you can use to control the flow through your code :

### if, elseif and else execute statements based on a logical test

### switch case and otherwise execute groups of statements based on a logical test

### while and end execute statements an indefinite number of times based on a logical test

### for and end execute statements a fixed number of times

***if, else, elseif***

if condition1

statements

elseif condition2

statements

elseif condition3

…

else

statements

end

the condition is an expression that is either 1 (true)or 0 (false).the statements between the if and end statements are executed if the condition is true. If the condition is false, the statements will be ignored and

execution will resume at the line after the end statement .the condition expression can be a vector or matrix. Further conditions can be made using the elseif and else statements.

The conditions are boolean statements and the standard comparisons can be made. Valid comparisons include "<" (less than), ">" (greater than), "<=" (less than or equal), ">=" (greater than or equal), "==" (equal - this is two equal signs with no spaces between them), and "~=" (not equal).

Note that “=” is used in assignments and “= =” is used in relations. Relations may be connected

(or quantified) by the following logical operators.

& → and

| → or

~ → not

For example, the following code will set the variable j to be -1:

E1

a = 2;

b = 3;

if (a<b)

j = -1;

end

Additional statements can be added for more refined decision making. The following code sets the variable j to be 2.

a = 4;

b = 3;

if (a<b)

j = -1;

elseif (a>b)

j = 2;

end

The *else* statement provides a catch all that will be executed if no other condition is met. The following code sets the variable j to be 3.

E2

a = 4;

b = 4;

if (a<b)

j = -1;

elseif (a>b)

j = 2;

else

j = 3

end

E3

>> t = rand(1);

>> if t > 0.75

s = 0;

elseif t < 0.25

s = 1;

else

s = 1-2\*(t-0.25);

end

E4

>> number=7;

>> remainder2 = rem(number,2);   
>> remainder3 = rem(number,3);

if remainder2==0 & remainder3==0

disp('Your number is divisible by both 2 and 3')

else

if remainder2==0

disp('Your number is divisble by 2 but not by 3')

else

if remainder3==0

disp('Your number is divisible by 3 but not by 2')

else

disp('Your number is not divisible by either 2 or 3')

end

end

end

NB: The two segments shown below produce identical results.

if A If A

x=a x=a

else elseif B

if B x=b

x=b elseif C

else x=c

if C else

x=c x=d

else end

x=d

end

end

end

**switch**

**The basic form of a switch statement is:**

switch test

**case result1,**

statements

**case** **result2**,

statements

...

**otherwise**,

statements

**end**

The respective statements are executed if the value of test is equal to the respective results.if none of the cases are true, the otherwise

Statements are done. Only the first matching case is carried out. If

You want the same statements to be done for different cases you can

enclose the several results in curly brackets:

example1

try this at the command prompt

selection = questdlg('Do you want to Exit Matlab?','Close Request','Yes','No','Yes');

switch selection

case 'Yes'

quit

case 'No'

return

end

example2

v=3;

switch v

case 1

j=1

case 2

j=2

otherwise

j=3

**LOOPS**

### 1. For Loops

* The *for loop* allows us to repeat certain commands. If you want to repeat some action in a predetermined way, you can use the *for loop*. All of the loop structures in matlab are started with a keyword such as "for", or "while" and they all end with the word "end".
* The *for loop* is written around some set of statements, and you must tell Matlab where to start and where to end. Basically, you give a vector in the "for" statement, and Matlab will loop through for each value in the vector:

The basic form of a for loop is:

for index=start: increment:stop

statements

end

For example, a simple loop will go around four times each time changing a loop variable, *j*:

for j=1:4

j

end

Another example, we define a vector and later change the entries.

v = [1:3:10]

for j=1:4

v(j) = j;

end

A better example, is one in which we want to perform operations on the rows of a matrix. If you want to start at the second row of a matrix and subtract the previous row of the matrix and then repeat this operation on the following rows, a *for loop* can do this in short order:

A = [ [1 2 3]' [3 2 1]' [2 1 3]']

>> B = A;

>> for j=2:3,

A(j,:) = A(j,:) - A(j-1,:)

end

EG2

for i=1:5

for j=1:5

A(i,j)=10\*i+j;

end

end

### 2.While Loops

If you don't like the *for loop*, you can also use a *while loop*. The *while loop* repeats a sequence of commands as long as some condition is met. This can make for a more efficient algorithm.

EXAMPLES

>> x=1;

>> while 1+x > 1

x = x/2;

end

>> x

x =

1.1102e-16

>> n=50;

>> p=1;

>> while (p < n)

    p = 2 \* p;

end

>> disp(p);       % displays 64

NB: The for loop is going to be more helpful to us.

You can nest multiple for loops

for m=1:5

for n=1:100

A(m,n)=1/(m+n-1);

end

end

Here is an example showing if, else, elseif and for loop.

k=5

for m = 1:k

for n = 1:k

if m == n

a(m,n) = 2;

elseif abs(m-n) == 2

a(m,n) = 1;

else

a(m,n) = 0;

end

end

end

a =

2 0 1 0 0

0 2 0 1 0

1 0 2 0 1

0 1 0 2 0

0 0 1 0 2

EXERCISE

In this exercise we going to import an excel file containing horizontal circle readings and distances.our objective is to reduce the H.C .R to angles by using the if statements and the for loops.

STEPS

1.Read the excel file into fn

>> A=xlsread(fn,2);

>> B=xlsread(fn,1);

d=A(:,1); %assigning 1st column to HCR in degrees

m=A(:,2); %assigning 2nd column to HCR in minutes

s=A(:,3); %assigning 3rd column to HCR in seconds

dist=A(:,4);% assigning 4th column to distances.

% REDUCTION OF ANGLES FROM FIELD BOOK

%.....................................

>> dms=[d m s];%deg,min,sec

>>deg=dms2degrees(dms);%converting deg,min,sec to decimal degrees.

n=(size(deg,1))/4 ;%size of file

% face left reductions

for i=1:n

FL(i,1)=deg((i+i+i+i)-2)-deg((i+i+i+i)-3);

if find(FL(i,1)<0)

FL(i,1)=FL(i,1)+360 ;

end

end

% face right reduction

for j=1:n

FR(j,1)=deg((j+j+j+j)-1)-deg((j+j+j+j));

if find(FR(j,1)<0)

FR(j,1)=FR(j,1)+360 ;

end

end

%COMPUTING DISTANCE FROM FIELD BOOK

% Back sight mean distance(Check)

for i=1:n

S(i,1)=(dist((4\*i)-3)+dist((4\*i)))/2;

end

BS=S(1,1); % distance between control points

S(1,:)=[]; % deleting 1st row,1st column of the matrice

s=[S;BS] ;% rearranging back sight distances

% Fore sight mean distance(original)

for j=1:n

D(j,1)=(dist((4\*j)-2)+dist((4\*j)-1))/2;

end

% mean distance

mean\_dist=(D+s)/2;

%mean included angle

mean\_angle=(FR+FL)/2;

% COMPUTING MISCLOSURE

% sum of measured angles

sum\_angles1=sum(mean\_angle);

dms\_sum\_angles1=round(degrees2dms(sum\_angles1));

D1=dms\_sum\_angles1(:,1);

M1=dms\_sum\_angles1(:,2);

S1=dms\_sum\_angles1(:,3);

% mathematical check for internal angles

sum\_angles2=((2\*n(:,1))-4)\*90;

dms\_sum\_angles2=degrees2dms(sum\_angles2);

D2=dms\_sum\_angles2(:,1);

M2=dms\_sum\_angles2(:,2);

S2=dms\_sum\_angles2(:,3);

% misclose

misclose=dms2degrees(dms\_sum\_angles1)-dms2degrees(dms\_sum\_angles2);

misclose=degrees2dms(misclose);

D3=misclose(:,1);

M3=misclose(:,2);

S3=misclose(:,3);

%correction

corrtn=dms2degrees(dms\_sum\_angles2)-dms2degrees(dms\_sum\_angles1);

%correction per station

corrn=corrtn/n;

corrn=degrees2dms(corrn);

% computing initial bearing and distance from coordinates

fnc=B(1,1); %assigning 1st row,1st column initial northing coordinates

fec=B(1,2); %assigning 1st row,1st column initial easting coordinates

tnc=B(2,1); %assigning 2nd row,1st column final northing coordinates

tec=B(2,2); %assigning 2nd row,2nd column final easting coordinates

dn=tnc-fnc; % change in northing coordinates

de=tec-fec; % change in easting coordinates

if dn==0 && de>0

bearing=90;

end

if dn<0 && de==0

bearing=180;

end

if dn==0 && de<0

bearing=270;

end

if dn>0 && de>0

bearing=atand(de/dn);

end

if dn<0 && de>0

bearing=(atand(de/dn))+180;

end

if dn<0 && de<0

bearing=(atand(de/dn))+180;

end

if dn>0 && de<0

bearing=(atand(de/dn))+360;

end

%computing back bearing

if find(bearing>180)

back\_bearing=bearing-180; % initial back bearing

else

back\_bearing=bearing+180; % initial back bearing

end

bearing\_DMS=round(degrees2dms(bearing));