

Conditional Statements & Loops

Lecture: 9 (Dr. Ajeet Kumar)

Table of Contents

Conditional Statements.....	1
The if Statement.....	1
The else Statement.....	2
The elseif Statement.....	3
Loops.....	4
for Loops:.....	4
loop_variable is scalar.....	5
loop_variable is matrix.....	5
while Loops:.....	8
continue and break statement.....	9
switch Structure.....	10

Conditional Statements

Conditional statements enable the Matlab to write decision making programs along with relational and logical operators.

Conditional statements contain one or more of *if*, *else* and *elseif* statement.

The if Statement

Syntax of the if statement's is

 if *logical expression*

statements

 end

Problem: 1. Write a program to compute the square root of positive numbers and should not take any action for negative numbers.

```
% Prob 1: write a program to compute the square root of positive numbers and should
% not take any action for negative numbers
clear all
x=input('enter the no to calculate the square root: ');
if x>=0
y=sqrt(x);
disp(['square root of number x = ',num2str(x), ' is ', num2str(y)])
end
```

square root of number x = 5 is 2.2361

Problem: 2. Calculate the functional value $z = \sqrt{x} + \sqrt{y}$, for $x \geq 0, y \geq 0$

```
% Prob 2: logical expression is compound expression (x>=0 & y>=0) then calculate  
% sqrt(x)+sqrt(y)  
clear all  
x=2;y=4;          % check with other input x=-4,y=5  
if (x>=0) & (y>=0)  
    z=sqrt(x)+sqrt(y)  
end
```

$z = 3.4142$

The else Statement

When more than one action can occur as a result of a decision, we can use *else* and *elseif* statement along with if statement.

Syntax of the else statement's is

```
if logical expression  
    statements  
else  
    statements  
end
```

Problem: 3. Write a program to find out the functional value of function y

$$y = \begin{cases} \sqrt{x} & x \geq 0 \\ e^x - 1 & x < 0 \end{cases}$$

```
% Prob 3: y=sqrt(x) for x>=0 and y=exp(x)-1 for x<0  
clear all;  
x=-5;  
if x<0  
    y=exp(x)-1  
else  
    y=sqrt(x)  
end
```

$y = -0.9933$

Problem: 4. Write a program to calculate the square root of the array, if all elements are positive/zero number, else calculate the square of the number.

Input should be in form of row vector.

```
% Prob 4: y=sqrt(x) for x>=0, take x as an array having positive and negative numbers.  
% Test returns a value of TRUE (logical 1) if all the elements of the logical  
% expression are true else FALSE (logical 0)
```

```
clear all
x=[2 -4]; % check with another input [1 3 5]
if x>=0
    y=sqrt(x)
else
    y=x.^2
end
```

```
y = 1x2
    4    16
```

The elseif Statement

Syntax of the else statement's is

```
if logical expression_1
    statements_1
elseif logical expression_2
    statements_2
elseif logical expression_3
    statements_3
else
    statements_4
end
```

Problem: 5. Calculate functional value of function the y as per below mentioned condition

$$y = \begin{cases} \sqrt{x} & 0 \leq x < 5 \\ \ln x & x \geq 5 \end{cases}$$

```
% Prob 5: y=ln x for x>=5 and y=sqrt(x) for 0<=x<5 % One can use nested
% if loop or simple elseif statement.
x=4;
if (x>=0)&(x<5)
    y=sqrt(x)
elseif x>=5
    y=log(x)
end
```

```
y = 2
```

Problem: 6. Calculate functional value of function the y as per below mentioned condition

$$y = \begin{cases} \ln x & x \geq 10 \\ \sqrt{x} & 0 \leq x < 10 \\ e^x - 1 & x < 0 \end{cases}$$

```
% Prob 6: y=ln x for x>=10 and y=sqrt(x) for 0<=x<10, y=exp(x)-1 for x<0
x=10;
if x<0
    y=exp(x)-1
elseif (x>=0) & (x<10)
    y=sqrt(x)
else
    y=log(x)
end
```

y = 2.3026

Loops

Number of times, we came across the condition where you need to perform same operation more than one time. A Loop is a structure for solving this type of problem in simpler way.

MATLAB uses two types of explicit loops

- for loop
- while loop

for Loops:

for loop is used when it is well known that how many times this operation needs to perform.

Syntax

for loop_variable= str: inc: term

Statements

end

Loop variable may be scalar, vector or matrix, however, in most general case, it is a scalar.

loop_variable: This is the variable whose value will change during every loop. This variable will actually run the loop.

str: Referred as starting value of loop_variable. This is the first assigned value of loop_variable.

inc: Referred as increment in the loop_variable during each run of loop. Increment may be positive and negative.

term: Referred as terminating value of loop_variable.

Every **for loop** completes with one **end** statement.

One run of loop is referred as **PASS**

loop_variable is scaler

Problem: 7. Write a matlab script file to calculate the sum of first 20 terms of the series $3m^3 + 2m^2 - 6m, m = 1, 2, 3, \dots, 20$.

```
% Prob 7: Write a program to compute the sum of first 20 terms in the series
% 3m^3+2m^2-6m, m= 1,2,3....20
% loop_variable is scaler
clear all;
for m=1:1:20
    y(m)=3*m^3+2*m^2-6*m;
end
total_sum=sum(y)
```

```
total_sum = 136780
```

loop_variable is matrix

In each run, loop_variable will take column wise value. If A is a matrix of 3-by-3. In first run of for loop, first column of the matrix will be assigned with loop_variable. In second run, 2nd column of the matrix will be assigned with loop_variable and so on.

Problem: 8. (a) Write a script file where loop_variable is matrix. See the effect of operator during loop.

(b) There are five points P (2,-1), Q(-2, -2), R (3, 5), T (-3, -2) and U (2.5, -1.1). Find the coordinate of the points having minimum distance from the Centre O (0,0).

```
% Prob 8 (a): Define a matrix and assigned it as loop variable
% loop_variable is matrix. Take a 3-by-3 matrix
A=[1 2 3; 4 5 6; 7 8 9]
```

```
A = 3x3
     1     2     3
     4     5     6
     7     8     9
```

```
for i=A
    y=i.^2 % see the output, in each pass "i" takes values of the Matrix A columnwise
           % In first pass: first column, 2nd Pass:second column and so on
end
```

```
y = 3x1
     1
    16
    49
y = 3x1
     4
    25
    64
y = 3x1
     9
    36
    81
```

```

% Prob 8 (b)..... METHOD: 1 (SAVING OUTPUT USING NULL MATRIX)
% RUN PROGRAM BY WRITING A SCRIPT FILE AND CALCULATE COMPUTATION TIME
% tic, toc is used to calculate computation time.
clear all;
tic
A=[2 -2 3 -3 2.5; -1 -2 5 -2 -1.1]; % 1st row: x-coordinate, 2nd row: y-coordinate
distance=[]; % defined to save the output
for i=A
    distance=[distance i.^2];
end
distance=sqrt(sum(distance)); % CALCULATE DISTANCE OF EACH POINT FROM CENTRE "0"
[min_dist, index]=min(distance);
if index==1
    disp(['Point P is closest point from centre 0 at distance = ',num2str(min_dist),' unit'])
elseif index==2
    disp(['Point Q is closest point from centre 0 at distance = ',num2str(min_dist), ' unit'])
elseif index==3
    disp(['Point R is closest point from centre 0 at distance = ',num2str(min_dist), ' unit'])
elseif index==4
    disp(['Point T is closest point from centre 0 at distance = ',num2str(min_dist), ' unit'])
else
    disp(['Point U is closest point from centre 0 at distance = ',num2str(min_dist), ' unit'])
end

```

Point P is closest point from centre 0 at distance = 2.2361 unit

```

toc

```

Elapsed time is 0.113740 seconds.

```

% Prob 8 (b)..... METHOD: 2 (SAVING OUTPUT USING zeros/ones MATRIX)
% INORDER TO COMPARE THE ACTUAL RUN TIME WITH METHOD 1, EXECUTE THE PROG IN SCRIPT FILE
% tic, toc is used to calculate computation time.
clear all;
tic
A=[2 -2 3 -3 2.5; -1 -2 5 -2 -1.1]; % 1st row: x-coordinate, 2nd row: y-coordinate
distance=zeros(2,5); % defined to save the output
k=1;
for i=A
    distance(:,k)= i.^2; % IN EACH PASS, ZEROS OF ONE COLUMN IS REPLACED BY SQUARE VALUE
    k=k+1; % k=1, 1ST COLUMN, k=2, 2nd column and so on
end
distance=sqrt(sum(distance));
[min_dist, index]=min(distance);
if index==1

```

```

    disp(['Point P is closest point from centre O at distance = ',num2str(min_dist),' unit'])
elseif index==2
    disp(['Point Q is closest point from centre O at distance = ',num2str(min_dist), ' unit'])
elseif index==3
    disp(['Point R is closest point from centre O at distance = ',num2str(min_dist), ' unit'])
elseif index==4
    disp(['Point T is closest point from centre O at distance = ',num2str(min_dist), ' unit'])
else
    disp(['Point U is closest point from centre O at distance = ',num2str(min_dist), ' unit'])
end

```

Point P is closest point from centre O at distance = 2.2361 unit

toc

Elapsed time is 0.162986 seconds.

Problem: 9. Write a script file to plot the function:

$$y = \begin{cases} 15\sqrt{4x} + 10 & x \geq 9 \\ 10x + 10 & 0 \leq x < 9 \\ 10 & x < 0 \end{cases}$$

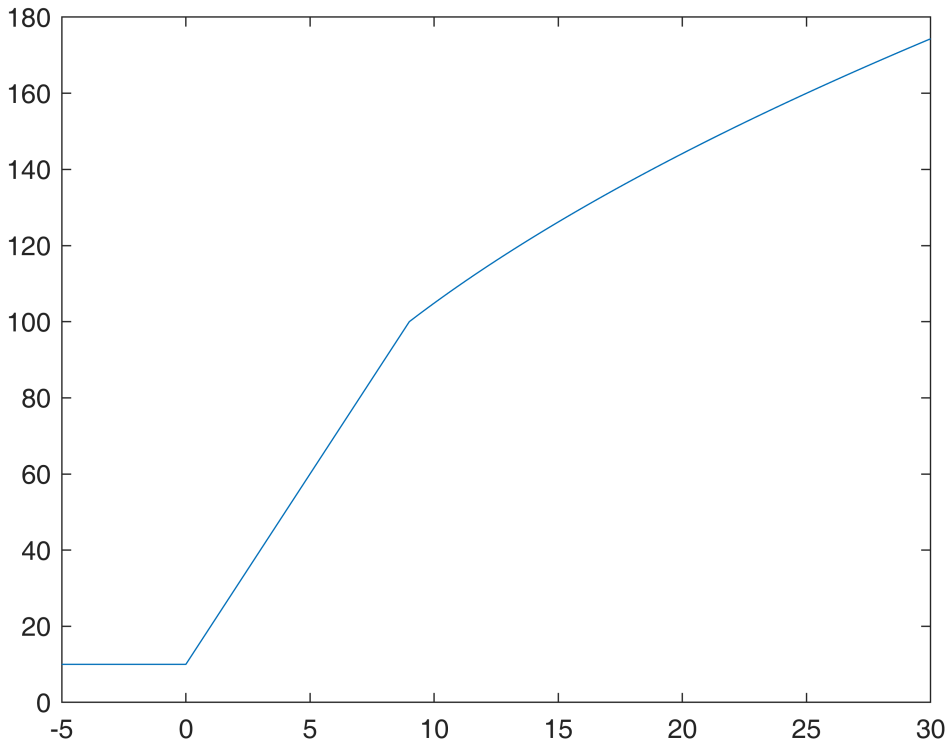
for $-5 \leq x \leq 30$

```

% prob 9: plot the function: y=15
x=-5:0.05:30;

for m=1:length(x)
    if x(m)<0
        y(m)=10;
    elseif (x(m)>=0)&(x(m)<9)
        y(m)=10*x(m)+10;
    else
        y(m)=15*sqrt(4*x(m))+10;
    end
end
plot(x,y)

```



while Loops:

while loop is used when a same operation needs to be done unless and until a given condition is satisfied.

Syntax

```
while condition
    Statements
end
```

Condition: 1. Loop variable must have a value before the while statement

Condition: 2. Loop variable must be changed somehow by the statements

Problem: 10

Write a script file to determine how many terms are required for the sum of the series $5k^2 - 2k$, $k=1,2,3,\dots$ to exceed 1500.

What is the sum for this many terms?

```
% Prob 10:
clear all
sum_ser=0.0;
k=0;
term=[];
while sum_ser<=1500
```



```

    k=k+1;
    sum_ser=sum_ser+(5*k^2-2*k);
    term=[term; k sum_ser];
end
term

```

```

term = 10x2
      1      3
      2     19
      3     58
      4    130
      5    245
      6    413
      7    644
      8    948
      9   1335
     10   1815

```

```

disp(['required terms = ',num2str(k)])

```

```

required terms = 10

```

```

disp(['sum of k = ' num2str(k), ' term is ',num2str(sum_ser) ])

```

```

sum of k = 10 term is 1815

```

continue and break statement

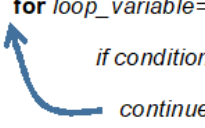
continue and *break* commands along with *if*-statement used to control *for* and *while* loops.

continue: This command along with *if*-statement, is used to pass the calculation just for the current *loop_variable* value and send the control for the next value of *loop_variable*. This command does not terminate the current loop.

```

for loop_variable= str: inc: term
    if condition
        continue
    else
        Statements
    end
end

```



break: This statement along with *if*-statement, break the current loop and stop processing of current loop.

for *loop_variable* = str: inc: term

if condition

break

else

Statements

end

end

Problem: 11. Write a program to print the following output.

1*1=1	1*1=1
1*2=2	1*2=2
1*4=4	1*3=3
1*5=5	1*4=4
2*1=2	1*5=5
2*2=4	2*1=2
2*4=8	2*2=4
2*5=10	2*3=6
3*1=3	2*4=8
3*2=6	2*5=10
3*4=12	4*1=4
3*5=15	4*2=8
4*1=4	4*3=12
4*2=8	4*4=16
4*4=16	4*5=20
4*5=20	5*1=5
5*1=5	5*2=10
5*2=10	5*3=15
5*4=20	5*4=20
5*5=25	5*5=25

%

switch Structure

This provides an alternative way to write a program without using if-else-elseif commands. In some cases, it looks better to use switch structure over if structure. However, switch-structure can be used along with if-structure.

Syntax:

switch case_expression

```
case value1
    statement 1
case value 2
    statement 2
case value 3
    statement 3
otherwise
    statement
```

end

- `case_expression` can be either numeric or string constants.
- `case_expression` is compared with each `case value`. If they are same, then the statement following that `case` will be executed.
- The statement under other cases are not executed. If none of the case value matches with `case_expression`, the block of the statement under **otherwise** section is executed. Using `end` statement complete the switch structure.
- `otherwise` section is an optional statement.

Problem: 12. Write a program which ask the user to enter two matrices and choice of array operation to perform between these matrices.

```
%
```