EENG 212-M01
EENG 221-M01
Fall 2014

Tutorial 4
PROGRAMMING IN MATLAB
Loops, Control Flow & Conditional Statements

MATLAB has many commands for control-flow statements, such as conditional statements, and loops. Most of you have or are taking a high level language course, so you will recognize many of the programming structures described below.

1-Conditional Statements

1.1 If-Else-End Statement:

The simplest **if-else-end** statement is in its structured format is:

if *logical expression*
Statements
end

The statements can be a single command or several commands separated by commas, semicolons, or "returns". The statements are executed if the logical expression is true.
The above format can also be written as:

if *logical expression, statements,* **end**

Example 1:

```
burger= 10;                    % number of burgers
cost= burger* 5                % cost of pizzas
    if  burger > 10            % give discounts for larger purchases
       cost= (1- 5/10)*cost;
    end
```

Example 1b:

```
A= 3;
B= 5;
 If  A<B,
     C=6;
 end;
```

Example 2: This gives a first look at how a function is created

```
function [capital, interest] =compound( capital, years, rate, timescomp);
%  function to compute the compounded capital and the interest you get at the end of n years, at a flat
%  annual rate of r%. The interest is added to your account k times a year, and the principal amount you
%  invested is x0, then at the end of n years you would have  x= x0 (1 + r/k)^kn amount of money in your
%  account. This function computes  the interest (x-x0) for a given x, n , r and k.
%
x0 = capital; n = years; r = rate; k= timescomp;
if  r > 1
disp(' check your interest rate. For 8% enter .08, not 8.')
end
capital= x0*(I +r/k)^(k*n);
interest = capital –x0
```

In its more complex format, the ***if*** statement can be used with ***elseif*** and ***else***. The following highlights Its many structures.

```
if  logical expression
    statement 1
else
    statement 2
end
```

The statement 1 is executed if the logical expression is true otherwise it is the statement 2 which is executed. Let's look at another **if** structure:

```
if  logical expression 1
    statements 1
elseif  logical expression2
        statement 2
end
```

Here the statements 1 are executed if logical expression 1 is **true**, while the statement 2 is executed if logical expression 1 is **false** and logical expression 2 is **true**. Notice that the command **elseif** is attached and does not require an **end**, while the command **else** if require an **end**.

The **if** statements can also be nested as follows:

```
if    logical expression 1
      statements 1
elseif  logical expression 2
      statements 2
else
      statements 3
end
```

Example 3:

A 3 bit A/D converter, with an analog input x and digital output y has the following I/O relationship

```
y=0              x < -2.5
y=1     -2.5 <=  x < -1.5
y=2     -1.5 <=  x < -0.5
y=3     -0.5 <=  x
```

Let's write the conditional statements for the above I/O relationship. With input X_analog and output Y_dig , As respectively the input and the output of our A/D converter:

```
      if  X_analog < -2.5
   Y_dig= 0;
elseif  X_analog>=-2.5 &  X_analog < -1.5
      Y_dig =1;
 elseif  X_analog  >= -1.5 &  X_analog < -0.5
      Y_dig = 2
 else
      Y_dig = 3
 end                              % do not forget the end here
 Y_dig;
 end
```

Statements that puts any vector in place of the vector 1:n are legitimate:

Example 3b:

```
for j= [ 1, 2, 3, 4,5 ],
     <program>
end
```

1.2 Loops

For repeated execution of statements, Matlab uses the commands **for** and **while.**
The general format of a **for-loop** is :

```
for variable = expression
    Statements
end
```

Here the variable is the loop variable.

Example 4a:

```
for x= 1: 1000
    den = 1/(x+1)
end
```

Example 4b: Suppose we want to calculate the quantity six factorial (8! = 8 × 7 × 6 × 5 × 4 × 3 × 2 × 1) using MATLAB.
One way of doing this is:

```
fact = 1;
for i = 2 : 6
fact = fact * i;
```

Example 4c: Calculate the expression nCm for a variety of values of n and m.
The mathematical expression for it is:

$$^{n}C_{m} = \frac{n!}{m!(n-m)!}.$$

Since

$$n!/(n-m)! = n\times(n-1)\times(n-2)\times\cdots\times(n-m+1).$$

One way of doing this is:

```
% using the loop structure.

 prod = 1;
 nfact = 1;
for i = 0 : (m-1)
nfact = nfact * (i+1);
prod = prod * (n-i);
end
soln = prod/nfact;
```

Of course geometric series of the form

$\sum_{j=1}^{N} j^p$ or summing the first N integers at a power p can also be solved using Matlab script

One way of doing it, assuming that the value of N and p are entered through keyboard is:

```
% Summing series
N = input('Please enter the number of terms required ');
p = input('Please enter the power ');
sums = 0;
for j = 1:N
sums = sums + j^p;
end
disp(['Sum of the first ' int2str(N) ...' integers raised to the power ' ... int2str(p) ' is ' int2str(sums)])
```

Example 4d:
Let's write a Matlab program to solve the following geometric progression

$$\sum_{n=1}^{6} 2^n.$$

One way of doing it is:

```
geop = 0
for  n = 1:6
geop= geop + 2^n;
```

The following example illustrates how looping constructs may be nested.

Example 5a:

The following matrix A =[5 1 0 0 0
1 5 1 0 0
0 1 5 1 0
0 0 1 5 1
0 0 0 1 5]

can be created with Matlab as follows:

```
% A is a square matrix with  k rows and j columns
for  k= 1: 5
   for  j = 1: 5
       if  k == j;
           A( k, k ) = 5 ;
       elseif  abs (k-j) == 1
           A( k , j ) = 1;
       else
           A( k , j ) = 0;
       end
      end
end
```

Also

Example 5b:

```
for i=1 : 20;
       for j= 1: 20;
            A(i,j) = i*j;
       end
end
```

Example 5c:

% Create a Hilbert Matrix with a nested loop:

```
H = zeros(5);
for k=1:5
   for l=1:5
       H(k,l) = 1/(k+l-1);
   end
end

H
```

```
H =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
```

With **While Loops** we can execute a statement (or a group) an indefinite number of times until the condition specified by **while** is no longer satisfied.
The format is :

```
while condition is true
        statements
end
```

Example 6:

```
% find all powers of  4 below 100
 nb= 1;   i= 1;
while  nb < 100
      nb = 2^ i
      i= i + 1 ;  % increment i
end
```

2- The Break , Error and Return commands

The command **break** inside a **while** or **for** loop terminates the execution of the loop

Example 7:

```
% Determine Epsilon through iterations
  pcsteps =1   ;
  for  i = 1: 1000
       pcsteps = pcsteps/2 ;
       if pcsteps + 1 <= 1
            break
      end
   end
  pcsteps = pcsteps*2
```

The command **error('message')** inside a function aborts the execution and displays the error message

The command **return** the control to the invoking function

Example 8:

```
% do you want to see the plot
disp ('Do you want to see Plot')
ans = input( 'Enter 1 if Yes, 0 if NO ')
```

```
if ans == 0
    return
else
  plot ( x,….)
end
```

3-Random numbers

The generation of random numbers can be done by using the command **rand** . The initial value or **seed** Is set by default to zero but it can be changed with the **rand** command.

rand(n) returns an n by n matrix. Each value is a random number between 0 and 1.
rand(m,n) returns an m by n matrix. Each value is a random number between 0 and 1.
rand('seed', n) sets the value of the seed number
rand ('seed') returns the current value of the random number generator

Example 9:

```
%Generate random sequence
data_random= rand ( 1, 1000) * 3
```