Chapter 7: Programming in MATLAB
7.1 Relational and Logical Operators

- == Equal to
- ~= Not equal to
- < less than
- > greater than
- <= Smaller than or equal to
- >= Greater than equal to
Notes:

• If the statement is true, it is assigned a value to 1.
• If the statement is false, it is assigned a value to 0.

• If two scalars are compared, the result is a scalar 1 or 0.

• If two arrays (same size) are compared, the comparison is done element by element and the result is a logical array of the same size with 1’s and 0’s according to the outcome of the comparison.

• If a scalar is compared with an array, the scalar is compared with every element of the array, and the result is a logical array of the same size with 1’s and 0’s according to the outcome of the comparison of each element.
```matlab
>> 5>8
ans =
    0
>> a=5<10
a =
    1
>> y=(6<10)+(7>8)+(5*3==60/4)
y =
    2
>> b = [15 6 9 4 11 7 14];
>> c = [8 20 9 2 19 7 10];
>> d = c>=b
d =
    0  1  1  0  1  1  1  0
>> b==c
ans =
    0  0  1  0  0  0  1  0
>> b~=c
ans =
    1  1  0  1  1  1  0  1
>> f=b-c>0
f =
    1  0  0  1  0  0  0  1
```
Vector \( t \) consists of elements of \( r \) in positions where \( s \) has 1's.
Logical vectors and arrays:

- Differ from numerical vectors and arrays in which they can be used for addressing.
- They can be used in arithmetic operations. Once a logical vector or array is used in arithmetic operation it is changed to a numerical vector or array.

Order of precedence in a mathematical expression that includes relational and mathematical operation:

- The arithmetic operation (+, -, *, /, \) have precedence over relational operations.
- The relational operators themselves have equal precedence and are evaluated from left to right.
```matlab
>> 3+4<16/2
ans =
    1
>> 3+(4<16)/2
ans =
    3.5000
>>  
```
### Logical Operators

<table>
<thead>
<tr>
<th>Logical operator</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| &                | AND  | A&B → A and B  
• If **both** are true (nonzero), the result is true (1),  
• otherwise the result is false (0).  |
| | OR   | A|B → A or B  
• If **either one, or both are true**, the result is true (1),  
• otherwise (both are false) the result is false (0)  |
| ~                | NOT  | ~ A → not A  
• Operates in one operand. Gives the opposite of the operand.  
• True (1) if the **operand is false** and false (0) if the operand is true.  |
Logical operators (like relational operators) are used:

- As arithmetic operators within a mathematical expression.
- In addressing arrays.
- To control the flow of the program.
```plaintext
>> 3 & 7
ans = 
    1

>> a = 5 | 0
a = 
    1

>> ~25
ans = 
    0

>> x = [9 3 0 11 0 15];
>> y = [2 0 13 -11 0 4];
>> x & y
ans =
    1   0   0   1   0   1

>> z = x | y
z =
    1   1   1   1   1   0   1

>> ~(x+y)
ans =
    0   0   0   1   1   0
```
## Order of precedence

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (highest)</td>
<td>Parentheses</td>
</tr>
<tr>
<td>2</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>3</td>
<td>Not (~)</td>
</tr>
<tr>
<td>4</td>
<td>Multiplication, Division</td>
</tr>
<tr>
<td>5</td>
<td>Addition, subtraction</td>
</tr>
<tr>
<td>6</td>
<td>Relational operators (&gt;, &lt;,&lt;=, &gt;=, ==, ~=)</td>
</tr>
<tr>
<td>7</td>
<td>and (&amp;)</td>
</tr>
<tr>
<td>8 (lowest)</td>
<td>Or (</td>
</tr>
</tbody>
</table>
```matlab
>> x = -2; y = 5;
>> -5 < x < -1
ans =
    0
>> -5<x & x<-1
ans =
    1
>> ~(y<7)
ans =
    0
>> ~y<7
ans =
    1
>> ~(y>=8) | (x<-1))
ans =
    0
>> ~(y>8) | (x<-1)
ans =
    1
```
Built-in logical functions

• MATLAB has built functions that are equivalent to the logical operators.
  • `and(A,B)` equivalent to `A&B`.
  • `or(A,B)` equivalent to `A|B`.
  • `not(A)` equivalent to `~A`.

• MATLAB has other logical built-in functions. See the next table.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| xor(a, b) | True (1) if one operand is true and the other is false. | >> xor(7,0)  
an = 1  
>> xor(7,-5)  
an = 0 |
| all(A) | • True (1) if all elements in a vector A are true (nonzero)  
• False (0) if one or more elements are false (zero)  
• If A is a matrix, treats columns of A as vectors, returns a vector with 1’s and 0’s. | >> A = [6 2 15 9 7 11];  
>> all(A)  
an = 1  
>> B = [6 2 15 9 0 11];  
>> all(B)  
an = 0 |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>any(A)</td>
<td>• True (1) if any element in a vector A is true (nonzero).</td>
<td>&gt;&gt; A = [6 0 15 0 0 11];</td>
</tr>
<tr>
<td></td>
<td>• False (0) if all elements are false (zero).</td>
<td>&gt;&gt; any(A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ans = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;&gt; B = [0 0 0 0 0 0];</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;&gt; any(B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ans = 0</td>
</tr>
<tr>
<td>find(A)</td>
<td>• If A is a vector, returns the indices of the nonzero elements.</td>
<td>&gt;&gt; A = [0 9 4 3 7 0 0 1 8];</td>
</tr>
<tr>
<td>find(A&gt;d)</td>
<td>• If A is a vector, returns the address of the elements that are larger than d (any relational operator can be used).</td>
<td>&gt;&gt; find(A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ans = 2 3 4 5 8 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;&gt; find(A&gt;d)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ans = 2 5 9</td>
</tr>
</tbody>
</table>
• See sample problem 7-1 p 199
7.2 Conditional statements

• Allow MATLAB to make a decision of whether to execute or skip the commands follow the conditional statement.

If statement

• Conditional expression consists of relational and/or logical operation.
• Must have and end statement.

Examples:
• If a < b
• If c >= 5
• If a == b
• If a~ b
• If (d<h) & (x>7)
• If (x~ =13) | (y<0)
The **if-end** structure

```
if conditional statement
    MATLAB commands
end
```

- **See sample problem 7-2 p 201**
The **if-else-end** structure

```
.......... MATLAB program
.......... MATLAB commands 1
if conditional statement
.......... MATLAB commands 2
else
.......... MATLAB program
end
.......... MATLAB commands
.......... MATLAB program
```

- **See sample problem 7-3 p 203**
The **if-elseif-else-end** structure

```
................... MATLAB program
if conditional statement
................... MATLAB commands 1
elseif conditional statement
................... MATLAB commands 2
else
................... MATLAB commands 3
end
................... MATLAB program
```
Example (Problem 7.9)

```matlab
% this file to find the real roots of a quadratic equation:
% ax^2+bx+c=0
clc
a = input('Enter the constant a = ');
b = input('Enter the constant b = ');
c = input('Enter the constant c = ');

D = b*b - 4*a*c

if D > 0
    disp('The equation has two roots: ');
    r1 = (-b + D)/(2*a);
    r2 = (-b - D)/(2*a);
    fprintf('The roots of equation are: \n ', r1,r2);
elseif D == 0
    disp('The equation has one root: ');
    r = -b /(2*a);
    fprintf('The root of equation is: %f \n ',r);
else
    disp('The equation has no real roots ');
end
```
Enter the constant \( a = -2 \)
Enter the constant \( b = 16 \)
Enter the constant \( c = -32 \)
\( D = 0 \)
The equation has one root:
The root of equation is: \( 4.000000 \)

Enter the constant \( a = 8 \)
Enter the constant \( b = 9 \)
Enter the constant \( c = 3 \)
\( D = -15 \)
The equation has no real roots

Enter the constant \( a = 3 \)
Enter the constant \( b = 5 \)
Enter the constant \( c = -6 \)
\( D = 97 \)
The equation has two roots:
The roots of equation are:
  \( \text{Root1} = 15.333333 \)
  \( \text{Root2} = -17.000000 \)
7.3 The **switch-case** statement

```matlab
switch switch expression
    case value1
        MATLAB commands 1
    case value2
        MATLAB commands 2
    case value3
        MATLAB commands 3
otherwise
    MATLAB commands 4
end
```

MATLAB program
Notes:

1. The switch expression is a variable that is:
   – a scalar,
   – a string,
   – or a mathematical expression that includes preassigned variables and can be evaluated.

2. There are one or several case commands, each has a value (scalar or string)

3. How does the switch case statement work?
   • The value of the switch expression in the switch command is compared with values that are next to the case statement.
   • Only one group of commands is executed, which follow the match case statement.
   • If there is more than one match, only the first matching case is executed.
Example

```matlab
clc
Ein=input('Enter the value of the energy to be covered:');
EinUnits=input('Enter the current unit (J, ft-lb, cal, or eV): ','s');
EoutUnits=input('Enter the new units (J, ft-lb, cal, or eV): ','s');
error=0;
switch EinUnits
    case 'J'
        EJ = Ein;
    case 'ft-lb'
        EJ = Ein/0.738;
    case 'cal'
        EJ = Ein/0.239;
    case 'eV'
        EJ = Ein/6.24e18;
    otherwise
        error = 1;
end
```
switch EoutUnits
  case 'J'
    Eout = EJ;
  case 'ft-lb'
    Eout = EJ*0.738;
  case 'cal'
    Eout = EJ*0.239;
  case 'eV'
    Eout = EJ*6.24e18;
  otherwise
    error = 1;
end

if error
  disp('Error current or new units are typed incorrectly.')
else
  fprintf('E = %g %s \n',Eout,EoutUnits)
end
Enter the value of the energy to be covered: 2800
Enter the current unit (J, ft-lb, cal, or eV): cal
Enter the new units (J, ft-lb, cal, or eV): J
E = 11715.5 J
>> |
7.4 Loops

• In a loop, the commands are repeated several times. Every execution or round is called a **pass**.

• In each pass, all variables that are defined within the loop are assigned new values.
7.4.1 for-end loops

Loop index variable: for k = f : s : t
Increment in k after each pass
Value of k in the first pass
Value of k in the last pass

end
Notes:

• The loop index variable can have any variable name.
• The increment can be negative (i.e. k=10:-2:4)
• If the increment value is omitted, the value is 1.
• If t = f, the loop is executed once.
• If f>t and s>0, the loop is not executed.
• If f<t and s<0, the loop is not executed.
• If the values of s, s, t are such k cannot be equal to t:
  i.e k = 8:10:50 \Rightarrow k = 8, 18, 28, 38, 48 (largest value that is smaller than t)
  k = 20:-5:3 \Rightarrow k = 20, 15, 10, 5 (smallest value that is larger than t)
• The values of k cannot be predefined as a vector. For k = [2 5 7 9 8].
• Each for loop must have an end.
• When the loop ends, the loop index variable (k) has the value that was last assigned to it.
for k=1:3:10
    x = k^2
end

x =
    1
x =
    16
x =
    49
x =
    100
Sample problem 7-5, p 210

a) Calculate the sum of the first $n$ terms of the series:

$$\sum_{k=1}^{n} \frac{(-1)^k k}{2^k}$$

```matlab
1 - n = input('Enter the number of terms: ');
2 - s = 0;
3 - for k = 1:n
4 -     s = s + (-1)^k * k / (2^k);
5 - end
6 - fprintf('The sum of the series is: %f \n', s)
```

Enter the number of terms: 4
The sum of the series is: -0.125000
```
b) Calculate \( \sin(x) \) by using the Taylor series:

\[
\sin x = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)!}
\]

```matlab
function y = Tsin(x,n)
    % Tsin calculates the sin using Taylor series formula
    % Input arguments:
    % x: the angle in degrees, n: number of terms
    x = x * pi/180;
    y = 0;
    for k = 1:n
        y = y + (-1)^k * x^(2*k+1)/factorial(2*k+1);
    end
```

```matlab
>> Tsin(150,3)
ans =
    -2.1330
>> Tsin(150,7)
ans =
    -2.1180
```
Sample problem 7-6, p 212

```matlab
v = [5 17 -3 8 0 -7 12 15 20 -6 6 4 -2 16];
n = length(v);
for k = 1:n
    if v(k)>0 & (rem(v(k),3)==0 | rem(v(k),5)==0)
        v(k)=2*v(k);
    elseif v(k)<0 & v(k)>-5
        v(k) = v(k)^3;
    end
end
v
```

```
Column 1 through 11
10  17  -27   8  0  -7  24  30  40  -6  12
Columns 12 through 14
4   -8   16
```
7.4.2 **while – end** loop

The **while – end** loops are used when looping is needed but the number of passes is not known.

The looping process continues until a stated condition is false.

```matlab
1 - x = 1
2 - while x <= 15
    x = 2*x
3 - end

x =
1
2
4
8
16
```
Sample problem 7-7, p 215

- Calculate $f(x) = e^x$ represented in Taylor series:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

- Stop the loop when the absolute value of the term added is smaller than 0.0001 or $n \geq 30$.

```matlab
1 - x = input('Enter x ');  
2 -     n = 1;  
3 -     an=1; % assign the value 1 to an(1>0.0001) to enter while loop  
4 -     S=an; % the variable s should be preassigned  
5 -     while abs(an)>=0.0001 && n<=30  
6 -         an = x^n/factorial(n); % added term  
7 -         S = S + an;  
8 -         n = n+1;  
9 -     end  
10 -     if n>=30  
11 -         disp('More than 30 terms are needed')  
12 -     else  
13 -         fprintf('exp(%f) = %f \n',x,S)  
14 -         fprintf('The number of terms used is: %d \n',n)  
15 -     end
```
Enter x 2
\[ \exp(2.000000) = 7.389046 \]
The number of terms used is: 12
Enter x -4
\[ \exp(-4.000000) = 0.018307 \]
The number of terms used is: 18
Enter x 21
More than 30 terms are needed

>>
7.5 Nested loops and nested conditional statements

Every time k increasing by 1, the nested loop executed m times.

Overall, the group of commands are executed nxm times.
Sample problem 7-8, p 217

```
clc
n = input('Enter the number of rows: ');
m = input('Enter the number of columns: ');
A = [];

for k=1:n
    for h=1:m
        if k==1
            A(k,h)=h;
        elseif h==1
            A(k,h)=k;
        else
            A(k,h)=A(k,h-1)+A(k-1,h);
        end
    end
end
A
```
Enter the number of rows: 4
Enter the number of columns: 5

\[ A = \]
\[
\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
2 & 4 & 7 & 11 & 16 \\
3 & 7 & 14 & 25 & 41 \\
4 & 11 & 25 & 50 & 91 \\
\end{array}
\]

\[ \gg \]
7.6 the **break** and **continue** command

1. **The break command**
   - Inside *for* or *while* loop, the **break** command terminates the execution of the whole loop.
     (MATLAB jumps to the **end** command and continue with the next command).
   - Inside a nested loop, only the nested loop is terminated.
   - Outside a loop in a script or function file, it terminates the execution of file.

2. **The continue command**
   - Used inside the loop to stop the present pass and start the next pass in the loop.
1. The continue command
2. Used inside the loop to stop the present pass and start the next pass in the loop.
Problem 7.15

The required number is 42

>>