Boolean algebra

Variable of boolean type can have only two values

- true
- false

There are three logical operators which are used in boolean algebra

- ¬ - logic not, Matlab ˜
  - true = false
  - false = true

- ∧ - logic and, Matlab &
  - \( A \land B = \begin{cases} true, & \text{if } A=true \text{ and } B=true \\ false, & \text{otherwise} \end{cases} \)

- ∨ - logic or, Matlab |
  - \( A \lor B = \begin{cases} false, & \text{if } A=false \text{ and } B=false \\ true, & \text{otherwise} \end{cases} \)
Boolean algebra

Variable of boolean type can have only two values
- true (Matlab use 1 to indicate it, actually everything but zero)
- false (Matlab uses 0)
There are three logical operators which are used in boolean algebra
- ¬ - logic not, Matlab
  \[
  \neg \text{true} = \text{false} \\
  \neg \text{false} = \text{true}
  \]
- ∧ - logic and, Matlab
  \[
  A \land B = \left\{ \\
  \begin{array}{ll}
    \text{true, if } A=\text{true and } B=\text{true}, \\
    \text{false, otherwise}
  \end{array}
  \right.
  \]
- ∨ - logic or, Matlab
  \[
  A \lor B = \left\{ \\
  \begin{array}{ll}
    \text{false, if } A=\text{false and } B=\text{false}, \\
    \text{true, otherwise}
  \end{array}
  \right.
  \]
Boolean operators precedence in Matlab

If \( A = \text{false}, B = \text{true}, C = \text{true} \)

\[ A \lor \neg B \land C \]

\( \neg \) has highest precedence, then \&, and then |

Thus 

\[ A \lor \neg B \land C = \text{false} \]

"Cat is an animal and cat is not an animal" is a false statement

\( \neg Z \land Z = \text{false} \)
If $A = false$, $B = true$, $C = true$

$$A|\sim B&C$$

$\sim$ has highest precedence, then $\&$, and then $|$

$$A|((\sim B)&C)$$

Thus

$$A|\sim B&C = false$$

“Cat is an animal and cat is not an animal” is false statement
There is an island, which is populated by two kinds of people: liars and truthlovers.
- Liars always lie and never speak a word of truth.
- Truthlovers always speak only truth.

Suppose, you are landed on this island and met a person. What will be the answer to your question “Who are you?”
- The answer always will be “Truthlover”.

Now you see a person who answers to your question. “I am a liar.” Is it possible?

This makes a paradox and should not ever happen on this island.
Matlab boolean logic examples

- 123.3 & 12 = 1
- ~ 1232e-6 = 0

```matlab
>> B = [1.22312, 0; 34.343, 12]
B =
    1.2231   0
    34.3430  12.0000
```

```
~B
ans =
    0   1
    0   0

B | ~B
```

"To be or not to be"

```matlab
ans =
    1   1
    1   1
```
Matlab boolean logic examples

123.3 & 12 = 1
~ 1232e-6 = 0

>> B=[1.22312, 0; 34.343, 12]
B =
  1.2231  0
  34.3430 12.0000

~B
ans =
  0  1
  0  0

B|~B

"To be or not to be"
Matlab boolean logic examples

```
>> B=[1.22312, 0; 34.343, 12]
B =
 1.2231 0
34.3430 12.0000

>> A=[56, 655; 0, 24.4]
A =
 56.0000 655.0000
 0 24.4000

B&A
ans =
 1 0
 0 1

A|~B
ans =
 1 1
 0 1
```
Matlab boolean logic examples

```matlab
>> B = [1.22312, 0; 34.343, 12.0000]
ans =
1.2231 0
34.3430 12.0000

>> A = [56, 655; 0, 24.4]
A =
56.0000 655.0000
0 24.4000

B & A
ans =
1 0
0 1

A | ~B
ans =
1 1
0 1
```

Comparison operators

<table>
<thead>
<tr>
<th>Math</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>==</td>
</tr>
<tr>
<td>~</td>
<td>~</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&lt;=</td>
<td>&lt;=</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>&gt;=</td>
<td>&gt;=</td>
</tr>
</tbody>
</table>

```matlab
x = [1, 2, 3, 4, 5]
x =
1 2 3 4 5

x >= 3
ans =
0 0 1 1 1

% chose such 'x' where x >= 3
x(x >= 3)
an = 
3 4 5
```
Comparison operators

<table>
<thead>
<tr>
<th>Math</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>==</td>
</tr>
<tr>
<td>\neq</td>
<td>\neq</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>\leq</td>
<td>\leq</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>\geq</td>
<td>\geq</td>
</tr>
</tbody>
</table>

\[
x = [1, 2, 3, 4, 5]
\]
\[
x = \begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix}
\]
\[
x \geq 3
\]
\[
\text{ans} =
\begin{bmatrix}
0 & 0 & 1 & 1 & 1
\end{bmatrix}
\]

\[
x = [1, 2, 3, 4, 5]
\]
\[
x \geq 3
\]
\[
\text{ans} =
\begin{bmatrix}
0 & 0 & 1 & 1 & 1
\end{bmatrix}
\]

\[
x = [1, 2, 3, 4, 5]
\]
\[
x \geq 3
\]
\[
\text{ans} =
\begin{bmatrix}
0 & 0 & 1 & 1 & 1
\end{bmatrix}
\]

Comparison with matrices

\[
A = [1, 2; 3, 4]
\]
\[
A =
\begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix}
\]
\[
B = [33, 11; 53, 42]
\]
\[
B =
\begin{bmatrix}
33 & 11 \\
53 & 42
\end{bmatrix}
\]
Comparison with matrices

\[
\begin{align*}
\text{A} & = \begin{bmatrix} 1, & 2; & 3, & 4 \end{bmatrix} \\
\text{B} & = \begin{bmatrix} 33, & 11; & 53, & 42 \end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]

\[
\begin{align*}
\text{A} & = \\
1 & 2 \\
3 & 4
\end{align*}
\]
Comparison with matrices

\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
33 & 11 \\
53 & 42 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
3 \\
2 \\
4 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
53 \\
11 \\
42 \\
\end{bmatrix}
\]

Chose such elements of B where elements of A \(\geq 2\)

if-else-end statement

if expression
  this part is executed only if expression is true
else
  this part is executed only if expression is false
end

if expression
  if hungry
    buy some food
  else
    keep working
end

if-else-end statement

if expression
  this part is executed only if expression is true
else
  this part is executed only if expression is false
end
**if-else-end statement**

if expression
   if hungry
      buy some food
   else
      keep working
end

if expression
   if \( x \geq 0 \)
      \( y = \sqrt{x} \);
   else
      error('cannot do');
end

**Common mistake in the 'if' statement**

if \( x = y \)
   D=4;
   Z=45;
   C=12;
else
   D=2;
end

the value of 'D' is always 4, except the case when \( y=0 \)

if \( x = y \)
   D=4;
   Z=45;
   C=12;
else
   D=2;
end

the value of 'D' is always 4, except the case when \( y=0 \)

someone used assignment operator (=) instead of comparison (==)

Notes

Notes

Notes
**Short form of 'if-end' statement**

```plaintext
if expression
  this part is executed
  only if expression is true
end

if won a million
  go party
end

if (deviation<=0)
  exit;
end
```

**The 'while' statement**

```plaintext
while expression
  this part is executed
  while expression is true
end

while hungry
  keep eating
end
```

```
i=1;
while (i<=10)
  c=a+b;
  z=c*4+5;
  i=i+2;
end
```

Notes: 

---

Eugeny Mikhailov (W&M) 
Practical Computing 
Lecture 03 11/19
The 'while' statement

while expression
  this part is executed
while expression is true
end

while hungry
  keep eating
end

while loop is extremely useful but they are not guaranteed to finish.
For a bit more complicated conditional statement and loop it is impossible to predict if the loop will finish.

Yet another common mistake is

i=1;
while (i<=10)
  c=a+b;
  z=c*4+5;
  i=i+2;
end

while loop is extremely useful but they are not guaranteed to finish.
For a bit more complicated conditional statement and loop it is impossible to predict if the loop will finish.
while statement

while expression
this part is executed while expression is true
end

while loop is extremely useful but they are not guaranteed to finish.
For a bit more complicated conditional statement and loop it is impossible to predict if the loop will finish.

Yet another common mistake is

definition of loop is not updating the term leading to fulfillment of the while condition

for statement

for variable = expression
do something
end
In this case variable is assigned consequently with columns of the expression, and then statements inside of the loop are executed

i=1;
while (i<=10)
c=a+b;
z=c*4+5;
i=i+2;
end

i=1;
while (i<=10)
c=a+b;
end

two for loops are guaranteed to complete after predictable number of iterations (the amount of columns in expression).
Example

\[ S = \sum_{i=1}^{100} i = 1 + 2 + 3 + \cdots + 99 + 100 \]

\[ S = 0; \quad i=1; \]
\[ \text{while}(i<=100) \]
\[ \quad S=S+i; \]
\[ \quad i=i+1; \]
\[ \text{end} \]

Example

\[ S = \sum_{k=1}^{100} a_k \]

Until \( k \leq 100 \) and \( a_k \geq 10^{-5} \), where \( a_k = k^{-k} \).

\[ S = 0; \quad k=1; \]
\[ \text{for} \quad i=1:100 \]
\[ \quad a_k = k^{-k}; \]
\[ \text{end} \]
Example

\[ S = \sum_{k=1}^{\infty} a_k \]

Until \( k=100 \) and \( a_k \geq 10^{-5} \), where \( a_k = k^{-k} \).

\[
S=0; \quad k=1;
\text{while} \ ( (k=100) \ & \ (k^{-k} \geq 1e-5) )
\begin{align*}
S &= S+k^{-k}; \\
& k=k+1;
\end{align*}
\]

\[
>> S \\
S &= 1.2913
\]

Notes
Same example with ‘for’ loop and use of matrix ops

\[ S = \sum_{k=1}^{\infty} a_k \]

Until \( k=100 \) and \( a_k \geq 10^{-5} \), where \( a_k = k^{-k} \).

\[
S=0;\\
for k=1:100\\
a_k=k^-k;\\
if (a_k < 1e-5)\\n\hspace{1cm}break;\\nend\\nS=S+a_k;\\
end
\]

\[
\text{S} = 1.2913
\]

Often it is more elegant to use built in Matlab matrix operators:

\[
k=1:100;\\na_k=k.^{-k};\\nS=\text{sum}(a_k(a_k\geq1e-5))\\n\text{S} = 1.2913
\]

Interest rate related example

Suppose bank gave you 50% interest rate (let’s call it ‘x’), and you put one dollar in.

How much would you get at the end of the year?

- one payment at the end of the year
  \[
  M_1 = 1 \times (1 + x) = 1 \times (1 + 0.5) = 1.5
  \]
Interest rate related example

Suppose bank gave you 50% interest rate (let’s call it ‘x’), and you put one dollar in.
How much would you get at the end of the year?

1. one payment at the end of the year
   \[ M_1 = 1 \times (1 + x) = 1 \times (1 + 0.5) = 1.5 \]

2. interest payment every half a year
   \[ M_2 = 1 \times (1 + x/2) \times (1 + x/2) = 1 \times (1 + 0.5/2)^2 = 1.5625 \]

3. interest payment every month
   \[ M_{12} = 1 \times (1 + x/12)^{12} = 1.6321 \]

Now let’s find how your money grows \((M_n)\) depends on the number of payments per year.

```matlab
x = 0.5;
N_max = 100;
N = 1:N_max;
M = 0*(N); % since N is vector M will be a vector too
for i=N
    M(i) = (1+x/i)^i;
end
plot(N,M,'-');
xlabel('N, number of payments per year');
ylabel('Money grows');
title('Money grows vs number of payments per year');
```

Of course we do not need computer to show that
\[ M_\infty = e^x = 1.6487 \]
but we need it to calculate something like
\[ M_{1001} - M_{1000} = 2.0572 \times 10^{-7} \]