Boolean algebra, conditional statements, loops.

Eugeni E. Mikhailov

The College of William & Mary

Lecture 03
Boolean algebra

Variable of boolean type can have only two values
- true
- false
Boolean algebra

Variable of boolean type can have only two values
- true (Matlab use 1 to indicate it, actually everything but zero)
- false
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)
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
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 ˜

  ¬true = false
  ¬false = true
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
- \( \neg \) - logic not, Matlab ~
  \( \neg true = false \)
  \( \neg false = true \)

- \( \wedge \) - logic and, Matlab &
  \[ A \wedge B = \begin{cases} 
    true, & \text{if } A=\text{true and } B=\text{true}, \\
    false, & \text{otherwise} 
  \end{cases} \]
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 ~

          ¬true = false
          ¬false = true

  • ∧ - logic and, Matlab &

          \[ A \land B = \begin{cases} 
              \text{true, if } A=\text{true and } B=\text{true}, \\
              \text{false, otherwise}
            \end{cases} \]

  • ∨ - logic or, Matlab |

          \[ A \lor B = \begin{cases} 
              \text{false, if } A=\text{false and } B=\text{false}, \\
              \text{true, otherwise}
            \end{cases} \]
Boolean operators precedence in Matlab

If $A = false$, $B = true$, $C = true$

$$A \mid \sim B \& C$$

Thus $A \mid \sim B \& C = false$

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

$\sim Z \& Z = false$
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 \( \land \), and then \( \lor \)

Thus \( A \lor \neg B \land C = \text{false} \)

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

\[ \neg Z \land Z = \text{false} \]
Boolean operators precedence in Matlab

If $A = false$, $B = true$, $C = true$

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

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

$$A | ((\sim B) \& C)$$
Boolean operators precedence in Matlab

If $A = false$, $B = true$, $C = true$

$$A \mid \sim B \& C$$

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

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

Thus

$$A \mid \sim B \& C = false$$
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”
If $A = \text{false}$, $B = \text{true}$, $C = \text{true}$

$A | \sim B \& C$

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

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

Thus

$A | \sim B \& C = \text{false}$

“Cat is an animal and cat is not an animal” is false statement
Boolean operators precedence in Matlab

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

\[ \sim Z \& Z = \]
Boolean operators precedence in Matlab

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

$$\sim Z\&Z = false$$
Boolean logic examples

There is an island, which is populated by two kind 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?”
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”. 

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?
There is an island, which is populated by two kind 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=

\[
\begin{bmatrix}
1 & 0 \\
0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
1.2231 & 0 \\
34.3430 & 12.0000
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 \\
1 & 1
\end{bmatrix}
\]

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

- $123.3 \& 12 = 1$
- $\sim 1232e-6 = 0$

```matlab
>> B=[1.22312, 0; 34.343, 12]
B =
    1.2231    0.0000
    34.3430    12.0000
>> ~B
ans =
    0.0000    1.0000
    0.0000    0.0000
>> B|~B
ans =
    1.0000    1.0000
    1.0000    1.0000
```
Matlab boolean logic examples

- $123.3 \& 12 = 1$
- $\sim 1232e^{-6} = 0$

```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"
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
Matlab boolean logic examples

- $123.3 \ & \ 12 = 1$
- $\sim 1232e^{-6} = 0$

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

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

- 123.3 \ & 12 = 1
- \sim 1232e^{-6} = 0

\[
\begin{bmatrix}
1.22312 & 0 \\
34.343 & 12
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix}
\]
Matlab boolean logic examples

\[ 123.3 \ & 12 = 1 \]
\[ \sim 1232e-6 = 0 \]

\[
\text{>> } B=[1.22312, 0; 34.343, 12] \\
B = \\
1.2231 \quad 0 \\
34.3430 \quad 12.0000
\]

\[
\sim B \\
\text{ans} = \\
0 \quad 1 \\
0 \quad 0
\]

\[
B | \sim B
\]
Matlab boolean logic examples

- \( 123.3 \ & \ 12 = 1 \)
- \( \sim 1232e-6 = 0 \)

```
>> 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]
B =
  1.2231     0
 34.3430   12.0000

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

B&A
```

Eugeniy Mikhailov  (W&M)
Practical Computing
Lecture 03
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
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

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
```
## Comparison operators

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

$x = [1, 2, 3, 4, 5]$

```matlab
x = [1, 2, 3, 4, 5]
x >= 3
ans = [0, 0, 1, 1, 1]
```

% chose such 'x' where x >= 3

```matlab
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>== double equal sign!</td>
</tr>
<tr>
<td>≠</td>
<td>~=</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>≤</td>
<td>&lt;=</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>≥</td>
<td>&gt;=</td>
</tr>
</tbody>
</table>

\[
x = [1, 2, 3, 4, 5]
\]

\[
x =
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 \\
\end{array}
\]
### Comparison operators

<table>
<thead>
<tr>
<th>Math</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>$=$</td>
<td>$==$ double equal sign!</td>
</tr>
<tr>
<td>$\neq$</td>
<td>$\sim=$</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 =

```
    1  2  3  4  5
```

$x \geq 3$
### Comparison operators

<table>
<thead>
<tr>
<th>Math</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>=</code></td>
<td><code>==</code> double equal sign!</td>
</tr>
<tr>
<td><code>≠</code></td>
<td><code>~=</code></td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td><code>&lt;</code></td>
</tr>
<tr>
<td><code>≤</code></td>
<td><code>&lt;=</code></td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td><code>&gt;</code></td>
</tr>
<tr>
<td><code>≥</code></td>
<td><code>&gt;=</code></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
```
## Comparison operators

<table>
<thead>
<tr>
<th>Math</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>$=$</td>
<td>== double equal sign!</td>
</tr>
<tr>
<td>$\neq$</td>
<td>$\sim=$</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>

```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)
```
Comparison operators

<table>
<thead>
<tr>
<th>Math</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>== double equal sign!</td>
</tr>
<tr>
<td>≠</td>
<td>~=</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>≤</td>
<td>&lt;=</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>≥</td>
<td>&gt;=</td>
</tr>
</tbody>
</table>

\[
x = [1, 2, 3, 4, 5]
\]

\[
x = \\
1 \quad 2 \quad 3 \quad 4 \quad 5
\]

\[
x \geq 3
\]

\[
n\text{ans} =
\]

\[
0 \quad 0 \quad 1 \quad 1 \quad 1
\]

\[
\text{% chose such 'x' where } x \geq 3
\]

\[
x(x \geq 3)
\]

\[
n\text{ans} =
\]

\[
3 \quad 4 \quad 5
\]
Comparison with matrices

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

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

\[
A(A \geq 2) = \begin{bmatrix}
3 \\
2 \\
4 \\
\end{bmatrix}
\]

\[
B(A \geq 2) = \begin{bmatrix}
53 \\
11 \\
42 \\
\end{bmatrix}
\]
Comparison with matrices

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 33 & 11 \\ 53 & 42 \end{bmatrix}$$

$$A \geq 2$$

$$B(A \geq 2) = \begin{bmatrix} 53 & 11 \\ 42 \end{bmatrix}$$
Comparison with matrices

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

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

\[
A \geq 2
\]

\[
\begin{bmatrix}
0 & 1 \\
1 & 1
\end{bmatrix}
\]

Choose such elements of \( B \) where elements of \( A \) \( \geq 2 \).
Comparison with matrices

```
>> A=[1,2;3,4]
A =
1 2
3 4

>> B=[33,11;53,42]
B =
22 11
53 42
```

```
A(A>=2)
ans =
3
2
4

B(A>=2)
ans =
53
11
42
```

Eugeniy Mikhailov (W&M)
Practical Computing
Lecture 03
Comparison with matrices

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

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

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

\[
A(A \geq 2)
\]

\[
\begin{bmatrix}
0 & 1 \\
1 & 1
\end{bmatrix}
\]

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

\[
A(A \geq 2)
\]

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

\[
B(A \geq 2)
\]

\[
\begin{bmatrix}
53 & 11 & 42
\end{bmatrix}
\]
Comparison with matrices

>> A=[1,2;3,4]
A =
1   2
3   4

>> B=[33,11;53,42]
B =
22  11
53  42

>> A>=2
ans =
0   1
1   1

A (A>=2)
ans =
3
2
4

B (A>=2)

Chose such elements of B where elements of A ≥ 2
Comparison with matrices

\[
\begin{align*}
\text{>> } A &= \begin{bmatrix} 1, 2; 3, 4 \end{bmatrix} \\
A &= \\
&= \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \\
\text{ans =} & \\
&= \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{>> } B &= \begin{bmatrix} 33, 11; 53, 42 \end{bmatrix} \\
B &= \\
&= \begin{bmatrix} 22 & 11 \\ 53 & 42 \end{bmatrix} \\
A (A \geq 2) &= \\
&= \begin{bmatrix} 3 \\ 2 \\ 4 \end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{B } (A \geq 2) &= \\
&= \begin{bmatrix} 53 \\ 11 \\ 42 \end{bmatrix}
\end{align*}
\]

Chose such elements of B where elements of \(A \geq 2\).
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
this part is executed only if **expression** is true
**else**
this part is executed only if **expression** is false
**end**

**if** hungry
buy some food
**else**
keep working
**end**
if-else-end statement

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

if *hungry*
buy some food
else
keep working
end

if (x>=0)
  y=sqrt(x);
else
  error('cannot do');
end
Common mistake in the 'if' statement

```matlab
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 (==).
Common mistake in the 'if' statement

```matlab
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 (==)
**Short form of ’if-end’ statement**

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

if-end statement

```
if won a million 
go party
end
```

```plaintext
if (deviation<=0)
exit;
end
```
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
```
Short form of 'if-end' statement

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

while expression
this part is executed
while expression is true
end
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 not updating the term leading to fulfillment of the while condition

Eugeniy Mikhailov (W&M)
Practical Computing
Lecture 03
The 'while' statement

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
The 'while' statement

While expression this part is executed while hungry keep eating end

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.
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
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

not updating the term leading to fulfillment of the while condition.
The '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.
The 'for' statement

```matlab
sum=0;
x=[1, 3, 5, 6]
for v=x
    sum=sum+v;
end
>> sum
sum =
15
```

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.
The '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.

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

\[ S = \sum_{i=1}^{100} i = 1 + 2 + 3 + 4 + \cdots + 99 + 100 \]
Example

\[ S = \sum_{i=1}^{100} i = 1 + 2 + 3 + 4 + \cdots + 99 + 100 \]

S=0; i=1;
while (i<=100)
    S=S+i;
    i=i+1;
end
Example

\[
S = \sum_{i=1}^{100} i = 1 + 2 + 3 + 4 + \cdots + 99 + 100
\]

```python
S=0; i=1;
while (i<=100)
    S=S+i;
    i=i+1;
end
```

```python
S=0;
for i=1:100
    S=S+i;
end
```
Example

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

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

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

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

\[ S=0; \quad k=1; \]
\[ \text{while}( \ (k\leq100) \ \& \ \ (k^{-k} \geq 1e^{-5}) \ ) \]
\[ \quad S=S+k^{-k}; \]
\[ \quad k=k+1; \]
\[ \text{end} \]

\[ S = 1.2913 \]
Example

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

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

\[
S=0; \quad k=1;
while( (k<=100) \& (k^-k \geq 1e-5) )
\]
\[
\quad S=S+k^-k;
\quad k=k+1;
end
\]

\[
>> S
\]
\[
S =
\]
\[
1.2913
\]
Example

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

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

S=0; k=1;
while( (k<=100) & (k^k >= 1e-5) )
    S=S+k^k;
    k=k+1;
end

>> S
S =
    1.2913
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; k=1;
while( (k<=100) & (k^-k >= 1e-5) )
    S=S+k^-k;
    k=k+1;
end

>> S
S =
    1.2913
Same example with ‘for’ loop and use of matrix ops

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

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

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

\[
\begin{align*}
    k &= 1:100; \\
    a_k &= k.^{-k}; \\
    S &= \text{sum}(a_k(a_k\geq1e-5))
\end{align*}
\]

\[ S = 1.2913 \]
Same example with 'for' loop and use of matrix ops

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

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

\[
\begin{align*}
S &= 0; \\
\text{for } & k=1:100 \\
& a_k=k^{-k}; \\
& \text{if (a_k < 1e-5)} \\
& \quad \text{break; } \\
& \quad \text{end} \\
& S=S+a_k; \\
\text{end}
\end{align*}
\]

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

\[
\begin{align*}
& k=1:100; \\
& a_k=k^{-k}; \\
& S=\text{sum}(a_k(a_k \geq 1e-5))
\end{align*}
\]

Note use of the choose elements construct built in sum function
Same example with 'for' loop and use of matrix ops

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

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

\[
\begin{align*}
S &= 0; \\
\textbf{for } k = 1:100 \\
&\quad a_k = k^{-k}; \\
&\quad \textbf{if } (a_k < 1e-5) \\
&\quad \quad \textbf{break}; \\
&\quad \textbf{end} \\
&\quad S = S + a_k; \\
\end{align*}
\]

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

\[
\begin{align*}
\texttt{k} &= 1:100; \\
\texttt{a}_k &= k.^{-k}; \\
\texttt{S} &= \texttt{sum}(a_k(a_k \geq 1e-5)) \\
S &= 1.2913
\end{align*}
\]

Note

- use of the \textit{choose elements} construct
- built in \texttt{sum} function
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

\[
M_1 = 1 \times (1 + x) = 1 \times (1 + .5) = 1.5
\]
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?

- one payment at the end of the year

\[ M_1 = 1 \times (1 + x) = 1 \times (1 + 0.5) = 1.5 \]

- interest payment every half a year

\[ M_2 = 1 \times (1 + x/2) \times (1 + x/2) = 1 \times (1 + 0.5/2)^2 = 1.5625 \]
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

\[ M_1 = 1 \times (1 + x) = 1 \times (1 + .5) = 1.5 \]

- interest payment every half a year

\[ M_2 = 1 \times (1 + x/2) \times (1 + x/2) = 1 \times (1 + .5/2)^2 = 1.5625 \]

- interest payment every month

\[ M_{12} = 1 \times (1 + x/12)^{12} = 1.6321 \]
Interest rate related example

Now let's find how your money growth \(M_N\) depends on the number of payments per year

\[
x = 0.5; \\
N_{\text{max}} = 100; \\
N = 1:N_{\text{max}}; \\
M = 0*(N); \quad \% \text{since } N \text{ is vector } M \text{ will be a vector too} \\
\text{for } i = N \\
\quad M(i) = (1+x/i)^i; \\
\text{end} \\
\text{plot}(N,M,\text{'-'}); \\
\text{xlabel}(\text{'N, number of payments per year'}); \\
\text{ylabel}(\text{'Money grows'}); \\
\text{title}(\text{'Money grows vs number of payments per year'});
\]

Of course we do not need computer to show that \(M_\infty = e^x = 1.6487\) but we need it to calculate something like

\[
M_{1001} - M_{1000} = 2.0572 \times 10^{-7}
\]
Interest rate related example