Predifined size functions

	A=[1:4;linspace(1,8,4)]	**A=ones(1,3)**	
>>numel(A)	ans = 8	ans = 3	
>>length (A)	ans = 4	ans = 3	
>> R=size(A)	R = 2 4	R = 1 3	
>>[n,m]=size(A)	n = 2	m = 4	n = 1 | m = 3
>>size(A,1)	ans = 2	ans = 1	
>>size(A,2)	ans = 4	ans = 3	
>>ndims(A)	ans = 2	ans = 2	

Elements extraction/indexing

- variable_name(linear index)
- Variable_name(row,column)

~~A(0)~~	**A=[1:4;linspace(1,8,4)]** A = 1.00 2.00 3.00 4.00 1.00 3.33 5.66 8.00	**A=ones(1,3)** A=1 1 1
>>A(3)	ans = 2.00	ans = 1
>>A(2,3)	ans = 5.66	error
>>A(end)	ans = 8	ans= 1

Elements extraction /indexing

	A=[1:4;linspace(1,8,4)] A = 1.00 2.00 3.00 4.00 1.00 3.33 5.66 8.00		**A=ones(1,3)** A=1 1 1
>>A(:)	ans = 1.00 1.00 2.00	3.33 3.00 5.66 4.00 8.00	ans = 1 1 1
>>A(:,3)	ans = 3.00 5.66		1
>>A(1, [2 4]) >>A(1,2:2:4)	ans = 2.00 4.00		error
>> A(1:3)	ans = 1 1 3		1 1 1

Elements extraction
Logical search

Command Window

```
>> A=magic(4)
A =
    16.00     2.00     3.00    13.00
     5.00    11.00    10.00     8.00
     9.00     7.00     6.00    12.00
     4.00    14.00    15.00     1.00
>> find(A>5)
ans =
     1.00
     3.00
     6.00
     7.00
     8.00
    10.00
    11.00
    12.00
    13.00
    14.00
    15.00
fx >>
```

```
>> A(find(A>5))
ans =
    16.00
     9.00
    11.00
     7.00
    14.00
    10.00
     6.00
    15.00
    13.00
     8.00
    12.00
```

Elements extraction/ indexing

~~A(0)~~	**A=[1:4;linspace(1,8,4)]** A = 1.00 2.00 3.00 4.00 1.00 3.33 5.66 8.00	**A=ones(1,3)** A=1 1 1
>>A(3)=-5	A = 1.00 **-5** 3.00 4.00 1.00 3.33 5.66 8.00	A=1 1 -5
>>A(1,3)=-3	A = 1.00 **-5** -3 4.00 1.00 3.33 5.66 8.00	A=1 1 –3
A(end,end)=0	A = 1.00 **-5** -3 4.00 1.00 3.33 5.66 0	A=1 1 0

Elements extraction/ indexing

	A=[1:4;linspace(1,8,4)]	**A=ones(1,3)**
	A = 1.00 2.00 3.00 4.00 1.00 3.33 5.66 8.00	A=1 1 1
A(:)=5	A= 5 5 5 5 5 5 5 5	A= 5 5 5
A(:,3)=0	A = 1.00 2.00 0 4.00 1.00 3.33 0 8.00	0 0 0
A(1, 6)	error	error
A(1,6)=-1	1.00 2.00 0 4.00 0 -1.00 1.00 3.33 0 8.00 0 0	1 1 1 0 0 1

Delete operator

>>A=[]

>>A(row, column)=[]

A =

1.00	2.00	3.00	4.00
1.00	3.33	5.66	8.00

>> A(2)=[]

A =

1.00	2.00	3.33	3.00	5.66	4.00	8.00

>> A(2,3)=[] → error

>>A(:, 2) =[]→

A =

1.0000	3.0000	4.0000
1.0000	5.6667	8.0000

Functions /Operators

Arithmetic operators.

plus - Plus +
uplus - Unary plus +
minus - Minus -
uminus - Unary minus -
mtimes - Matrix multiply *
times - Array multiply .*
mpower - Matrix power ^
power - Array power .^
mldivide - Backslash or left matrix divide \
mrdivide - Slash or right matrix divide /
ldivide - Left array divide .\
rdivide - Right array divide ./
idivide - Integer division with rounding option.
transpose - Transpose .'
ctranspose - Complex conjugate transpose

Elementwise operations

a=1:4 → identical size of a and b

b=10:10:40

>> a+b → >> plus(a,b)

ans =

11.00 22.00 33.00 44.00

>> a-b

ans =

-9.00 -18.00 -27.00 -36.00

>> b.^a

ans =

10 400 27000 256

>> a.*b → >>times(a,b)

ans =

10.00 40.00 90.00 160.00

>> a./b → >>rdivide(a,b)

ans =

0.10 0.10 0.10 0.10

Algebria operations

- mtimes → size(A) $C(i,j) = \sum_{k=1}^{p} A(i,k)B(k,j)$

```
>> A = [1 2 3; 4 5 6]; B = [-1 2 -3; 1 2 3; -1 2 3];
>> C=A*B
C =
    -2    12    12
    -5    30    21
```

- mrdivide, mldivide → solves the system of linear equations : x.A=B , A.x=B (algebria)
- mpower → A^2 is equivalent to A*A.

  2/A (algebria)

Algebria operations

- mtranspose, transpose →

```
c =
   4.0000 - 1.0000i   5.0000 + 5.0000i
   5.0000 + 1.0000i   2.0000 - 2.0000i
  -2.0000 + 5.0000i  -3.0000 + 5.0000i
>> c'
ans =
   4.0000 + 1.0000i   5.0000 - 1.0000i  -2.0000 - 5.0000i
   5.0000 - 5.0000i   2.0000 + 2.0000i  -3.0000 - 5.0000i
>> c.'
ans =
   4.0000 - 1.0000i   5.0000 + 1.0000i  -2.0000 + 5.0000i
   5.0000 + 5.0000i   2.0000 - 2.0000i  -3.0000 + 5.0000i
```

Scalar/array operations

Command Window

```
>> A
A =
     1     2     3
     4     5     6
>> 2+A
ans =
     3     4     5
     6     7     8
>> 2.*A
ans =
     2     4     6
     8    10    12
>> 2*A
ans =
     2     4     6
     8    10    12
>> 2./A
ans =
    2.0000    1.0000    0.6667
    0.5000    0.4000    0.3333
>> A./2
ans =
    0.5000    1.0000    1.5000
    2.0000    2.5000    3.0000
>>
```

```
>> A/2
ans =
    0.5000    1.0000    1.5000
    2.0000    2.5000    3.0000
>> 2/A
Error using  /
Matrix dimensions must agree.
>> A*2
ans =
     2     4     6
     8    10    12
>> 2*A
ans =
     2     4     6
     8    10    12
>>
```

Functions /Operators

Relational operators.

eq	- Equal	==
ne	- Not equal	~=
lt	- Less than	<
gt	- Greater than	>
le	- Less than or equal	<=
ge	- Greater than or equal	>=

Logical operators.

and	- Element-wise logical AND	&
or	- Element-wise logical OR	|
not	- Logical NOT	~
xor	- Logical EXCLUSIVE OR	
any	- True if any element of vector is nonzero	
all	- True if all elements of vector are nonzero	

Functions /Operators

Relational operators.

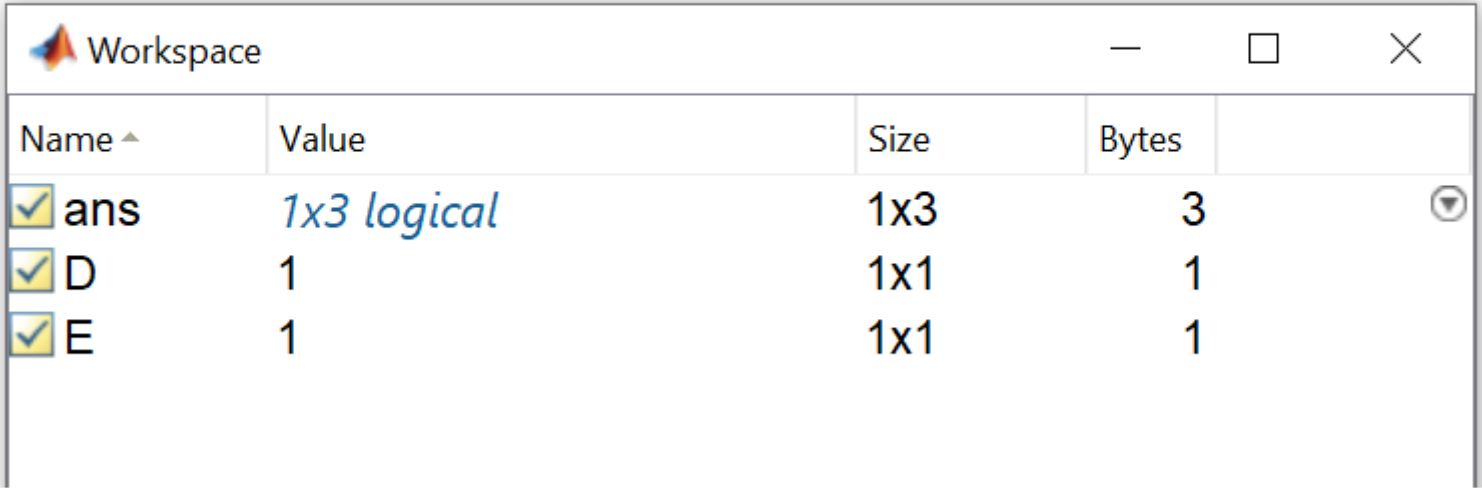
eq	- Equal	==
ne	- Not equal	~=
lt	- Less than	<
gt	- Greater than	>
le	- Less than or equal	<=
ge	- Greater than or equal	>=

>>help ops

Functions /Operators

```
>> 5<3
ans =
  logical
   0
>> 5>-1
ans =
  logical
   1
1:3==[1 2 4]
ans =
  1×3 logical array
   1   1   0
```

```
>> E= 3 & 7
E =
  logical
   1
>>R=  0 |0
R =
  logical
   0
>> xor([1 1 0 0],[1 0 1 0])
ans =
  1×4 logical array
   0   1   1   0
```


Workspace
Name
Value
Size
Bytes
ans
1x3 logical
1x3
3
D
1
1x1
1
E
1
1x1
1