Introduction to Computing for Engineers

Matrices and Vectors

Dr. Nguyen Ngoc Truong Minh

Arrays: Vectors and Matrices

- The **array** is the fundamental form MATLAB uses to store data
- **Scalars** – one row and one column (special case)
- **Vectors**

  Row – one row and multiple columns

  Column – multiple rows and one column
- **Matrices** – multiple rows and multiple columns

Row Vector

nằm ở sau

[1 x n] matrix

$$A\left[a_1\, a_2, \ldots, a_n\right] = \left\{a_j\right\}$$

nằm ở đầu

Column Vector

[m x 1] matrix

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_m \end{bmatrix} = \{a_i\}$$

Matrix Review

A matrix is any doubly subscripted array of elements arranged in rows and columns. (rectangular array)

$$\mathbf{A} = \begin{bmatrix} a_{11} & ,\ldots, & a_{1n} \\ a_{21} & ,\ldots, & a_{2n} \\ \ldots & \ldots \quad \ldots & \ldots \\ a_{m1} & ,\ldots, & a_{mn} \end{bmatrix} = \{A_{ij}\}$$

Square Matrix

Same number of rows and columns

$$B = \begin{bmatrix} 5 & 4 & 7 \\ 3 & 6 & 1 \\ 2 & 1 & 3 \end{bmatrix}$$

Creating Matrices

A vector can be created in MATLAB by typing the elements (numbers) inside the square brackets []

Row vector: To create an array in a single row, separate the elements with either a space or a comma (,).

```
>> a = [1 2 3 4]   or  >> a = [1, 2, 3, 4]
a = 1×4
1     2     3     4
```

Column vector: To create an array in a single column, separate the elements with either a semicolon (;) or enter next line.

```
>> b = [2; 5; 6]        or  >> b = [2        or >> b = [2
 b = 3×1                            5                   5 ; 6]
     2                              6]
     5
     6
```

Creating Matrices

A matrix can be created in MATLAB by typing the elements (numbers) inside the square brackets []

Matrix: To create a matrix that has multiple rows, separate the rows with semicolons or enter next line.

```
>> c = [1 2 3; 4 5 6; 7 8 9]
```

or
```
>> c = [1 2 3
        4 5 6
        7 8 9]
```

or
```
>> c = [1 2 3
        4 5 6 ; 7 8 9]
```

```
c = 3×3
    1     2     3
    4     5     6
    7     8     9
```

Matrix Addition and Subtraction

A new matrix **C** may be defined as the additive combination of matrices **A** and **B** where:

$$\mathbf{C} = \mathbf{A} + \mathbf{B}$$

is defined by:

$$\{C_{ij}\} = \{A_{ij}\} + \{B_{ij}\}$$

Note: all three matrices are of the same dimension

Addition

If $\quad A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$

and $\quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$

then $\quad C = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$

Matrix Addition Example

$$A + B = \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 4 & 6 \\ 8 & 10 \end{bmatrix} = C$$

```
>> A = [3 4; 5 6]
A =
    3    4
    5    6
>> B = [1 2; 3 4]
B =
    1    2
    3    4
```

```
>> C = A + B

C =
    4    6
    8   10
```

Matrix Subtraction

C = A - B

is defined by

$$\{C_{ij}\} = \{A_{ij}\} - \{B_{ij}\}$$

Note: all three matrices are of the same dimension

Matrix Multiplication

Matrices **A** and **B** have these dimensions:

$$\underbrace{[r \times c] \text{ and } [s}_{c\,=\,s} \times d]$$

Matrix Multiplication

Matrices **A** and **B** *can be multiplied if*:

$$[r \times c] \text{ and } [s \times d]$$

$$c = s$$

The resulting matrix will have the dimensions:

$$[r \times c] \text{ and } [s \times d]$$

$$r \times d$$

Computation: A x B = C

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

[2 x 2]

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}$$

[2 x 3]

$$C = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} & a_{11}b_{13} + a_{12}b_{23} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} \end{bmatrix}$$

[2 x 3]

Computation: A x B = C

$$A = \begin{bmatrix} 2 & 3 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \end{bmatrix}$$

[3 x 2] [2 x 3]

column row

A and B can be multiplied

$$C = \begin{bmatrix} 2*1+3*1=5 & 2*1+3*0=2 & 2*1+3*2=8 \\ 1*1+1*1=2 & 1*1+1*0=1 & 1*1+1*2=3 \\ 1*1+0*1=1 & 1*1+0*0=1 & 1*1+0*2=1 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 8 \\ 2 & 1 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

[3 x 3]

The Transpose Operation

Rows become columns and columns become rows (A^T)

The transpose operation '

For a vector: Converts a row vector to a column vector, or vice versa.

Example for a vector:

```
>> a = [3 8 1]
a =
    3    8    1
>> b = a'
b =
    3
    8
    1
```

The Transpose Operation

For a matrix, interchanges the rows and columns

$$A' = \begin{bmatrix} a_{11} & a_{21} & ,\ldots, & a_{m1} \\ a_{12} & a_{22} & ,\ldots, & a_{m2} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{1n} & a_{2n} & ,\ldots, & a_{mn} \end{bmatrix}$$

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

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

Identity Matrix

Square matrix with ones on the diagonal and zeros elsewhere.

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
>> I = eye(4)
I = 1 0 0 0
    0 1 0 0
    0 0 1 0
    0 0 0 1
```

Matrix Inversion

Inverse: if A is a square matrix, then its inverse A^{-1} is a matrix of the same size.

>> A^-1 or inv(A)

```
>> A = [2 -1 0; -1 2 -1; 0 -1 2];
>> inv(A)
ans =
    0.7500    0.5000    0.2500
    0.5000    1.0000    0.5000
    0.2500    0.5000    0.7500
```

$$B^{-1}B = BB^{-1} = I$$

```
A*A^-1
ans =
    1.0000         0         0
   -0.0000    1.0000   -0.0000
   -0.0000   -0.0000    1.0000
```

Arrays of numbers are used in many applications

Examples:

Arrays of numbers can represent data:

Year	1984	1986	1988	1990	1992	1994	1996
Population	127	130	136	145	158	178	211

In MATLAB, a vector, or any list of numbers, can be entered in a horizontal (row) or vertical (column) vectors.

For example, the population data in the previous slide can be entered in rows:

$$[1984\ 1986\ 1988\ 1990\ 1992\ 1994\ 1996]$$

$$[127\ 130\ 136\ 145\ 158\ 178\ 211]$$

or in columns:

$$\begin{bmatrix} 1984 \\ 1986 \\ 1988 \\ 1990 \\ 1992 \\ 1994 \\ 1996 \end{bmatrix} \qquad \begin{bmatrix} 127 \\ 130 \\ 136 \\ 145 \\ 158 \\ 178 \\ 211 \end{bmatrix}$$

The position vector can be entered in a:

row: $[2\ 4\ 5]$

column: $\begin{bmatrix} 2 \\ 4 \\ 5 \end{bmatrix}$

Entering Vectors

CREATING VECTORS WITH CONSTANT SPACING

- Two common methods
  - Specify **first term: step size: last term**
  - **linspace** (first term, last term, number of terms)

Entering Vectors

In a vector with constant spacing the difference between the elements is the same, (e.g. v = 2 4 6 8 10 12).

A vector in which the first term is `m`, the spacing is `q` and the last term is `n` can be created by typing `[m:q:n]`.

```
>> x = [1:2:13]
x =
    1    3    5    7    9    11    13
```

```
>> x = [1.5:0.1:2.1]
x =
    1.5000    1.6000    1.7000    1.8000    1.9000    2.0000    2.1000
```

If spacing is omitted the default is 1

```
>> x = [-3:7]
x =
   -3   -2   -1    0    1    2    3    4    5    6    7
```

Entering Vectors using linspace() command

A vector in which the first term is **`xi`**, the last term is **`xf`**, and the number of terms is **`n`**, can be created by typing **`linspace(xi,xf,n)`**.

```
>> u = linspace(0,8,6)
u =
      0    1.6000    3.2000    4.8000    6.4000    8.0000
```

If the number of terms is omitted the default is 100

Type:

```
>> u = linspace(0,49.5)
```

press **Enter** and watch the response of the computer.

It should be:

```
u = 0    0.5000    1.0000    1.5000   …(100 terms)…  49.0000  49.5000
```

Example

- ***Create the following row vectors***
  - [2 3 4 5 6]

```
>> a = [2:1:6]
a =

     2     3     4     5     6
```

```
>> u = linspace(2,6,5)
u =

     2     3     4     5     6
```

  - [1.1000 1.3000 1.5000 1.7000 1.9000]

```
>> a = [1.1:0.2:1.9]
a =

    1.1000    1.3000    1.5000    1.7000    1.9000
```

```
>> u = linspace(1.1,1.9,5)
u =

    1.1000    1.3000    1.5000    1.7000    1.9000
```

  - [8 6 4 2 0]

```
>> a = [8:-2:0]
a =

     8     6     4     2     0
```

```
>> u = linspace(8,0,5)
u =

     8     6     4     2     0
```

- ***Create Vector***

Example

- ***Create a vector*** ***vec*** ***which consists of 20 equally spaced points in the range from –pi to +pi.***

  `>> vec = linspace(-pi,pi,20);`

- ***Write an expression using*** ***linspace*** ***that will result in the same as***

  [2: 0.2: 3]

  `>> linspace(2,3,6)`

Two Dimensional Arrays: Matrices

A matrix is a two dimensional array of numbers.

In a **square** matrix the number of rows and columns is equal:

$$\begin{matrix} 7 & 4 & 9 \\ 3 & 8 & 1 \\ 6 & 5 & 3 \end{matrix}$$

Three rows and three columns (3x3)

In general, the number of rows and columns can be **different**:

$$\begin{matrix} 31 & 26 & 14 & 18 & 5 & 30 \\ 3 & 51 & 20 & 11 & 43 & 65 \\ 28 & 6 & 15 & 61 & 34 & 22 \\ 14 & 58 & 6 & 36 & 93 & 7 \end{matrix}$$

Four rows and six columns (4x6)

(*m*x*n*) matrix has *m* rows and *n* columns

(*m*x*n*) is called the size of the matrix

Array Addressing (vectors)

The address of an element in a **vector** is its position in the row (or column). For vector `v`, `v(k)` refers to the element in position `k`. The first address or position in an array is 1.

```
>> v = [35 46 78 23 5 14 81 3 55]
v =
   35   46   78   23    5   14   81    3   55
```

V4

```
>> v(4)
ans =
   23
```

```
>> v(7)
ans =
   81
```

```
>> v(1)
ans =
   35
```

It is possible to change an element in a vector by entering a value to a specific address directly:

```
>> v(6)=273
v =
   35   46   78   23    5  273   81    3   55
```

Single elements can be used like variables in computations:

```
>> v(2)+v(8)
ans =
   49
```

```
>> v(5)^v(8)
ans =
   125
```

Array Addressing (matrices)

The address of an element in a **matrix** is its position, defined by the number of the row and the number of the column.
For matrix `m`, `m(k,p)` refers to the element in row `k` and column `p`.

```
>> m=[3 11 6 5; 4 7 10 2; 13 9 0 8]
m =
     3    11     6     5
     4     7    10     2
    13     9     0     8
```

Cột

```
>> m(1,1)
ans =
     3
```

```
>> m(2,3)
ans =
    10
```

Hàng

It is possible to change an element in a matrix by entering a value to a specific address directly:

```
>> m(3,1)=20
m =
     3    11     6     5
     4     7    10     2
    20     9     0     8
```

Single elements can be used like variables in computations:

```
>> m(2,4)-m(1,2)
ans =
    -9
```

Using the colon (:) in addressing vectors

A colon can be used to access a range of elements in a vector or a matrix.

v(:)	Represents all the elements of a vector (either a row vector or a column vector)
v(3:6)	Represents elements 3 through 6 (i.e. *v*(3), *v*(4), *v*(5), *v*(6)).

```
>> v = [4 15 8 12 34 2 50 23 11]
v =
    4    15    8    12    34    2    50    23    11
>> u = v(3:7)
u =
    8    12    34    2    50
```

Using the colon (:) in addressing matrices

$A(:,3)$	Refers to the elements in all the rows of column 3.
$A(2,:)$	Refers to the elements in all the columns of row 2.
$A(:,2:5)$	Refers to the elements in columns 2 through 5 in all the rows.
$A(2:4,:)$	Refers to the elements in rows 2 through 4 in all the columns.
$A(1:3,2:4)$	Refers to the elements in rows 1 through 3 and in columns 2 through 4.

Using the colon (:) in addressing matrices Examples

```
>> A = [1 3 5 7 9; 2 4 6 8 10;
3 6 9 12 15; 4 8 12 16 20;
5 10 15 20 25]
A =
     1     3     5     7     9
     2     4     6     8    10
     3     6     9    12    15
     4     8    12    16    20
     5    10    15    20    25
```

column

```
>> B = A(:,3)
B =
     5
     6
     9
    12
    15
```

row

```
>> C = A(2,:)
C =
     2     4     6     8    10
```

Using the colon (:) in addressing matrices
Examples

```
>> A = [1 3 5 7 9; 2 4 6 8 10;
3 6 9 12 15; 4 8 12 16 20;
5 10 15 20 25]
A =
     1     3     5     7     9
     2     4     6     8    10
     3     6     9    12    15
     4     8    12    16    20
     5    10    15    20    25
```

row column

```
>> E = A(2:4,:)
E =
     2     4     6     8    10
     3     6     9    12    15
     4     8    12    16    20
```

row column

```
>> D = A(:, 2:5)
D =
     3     5     7     9
     4     6     8    10
     6     9    12    15
     8    12    16    20
    10    15    20    25
```

row column

```
>> F = A(1:3,2:4)
F =
     3     5     7
     4     6     8
     6     9    12
```

Array Examples

```
>> a = 7
a =
     7

>> E = 3
E =
     3

>> d = [5 a+E 4 E^2]
d =
     5    10     4     9

>> g = [a a^2 13; a*E 1 a^E]
g =

     7    49    13
    21     1   343
```

```
>> who

Your variables are:

E  a  d  g

>> whos
  Name      Size         Bytes  Class

  E         1x1              8  double array
  a         1x1              8  double array
  d         1x4             32  double array
  g         2x3             48  double array

Grand total is 12 elements using 96 bytes
```

Example of Linear System

A system of linear equations is a set of linear equations that you usually want to solve at the same time; that is, simultaneously.

$$\begin{aligned} 2x + y &= 13 \\ x - 3y &= -18 \end{aligned}$$

Everyone solves this equation by hand (using matrices)

Example of Linear System

A system of linear equations is a set of linear equations that you usually want to solve at the same time; that is, simultaneously.

2x + y = 13
x – 3y = -18

Using matrix algebra

$$[A] \bullet [X] = [B]$$

$$\begin{bmatrix} 2 & 1 \\ 1 & -3 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 13 \\ -18 \end{bmatrix}$$

$$[X] = [A]^{-1}[B]$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & -3 \end{bmatrix}^{-1} \begin{bmatrix} 13 \\ -18 \end{bmatrix}$$

This matrix is called Inverse Matrix

Example of Linear System

In MATLAB

A = [2 1
      1 -3];

B = [13
    -18];

InvA = inv(A)
X=InvA * B

$$[A]\bullet[X]=[B]$$

$$[X]=[A]^{-1}[B]$$

$$\begin{bmatrix} 2 & 1 \\ 1 & -3 \end{bmatrix}\bullet\begin{bmatrix} x \\ y \end{bmatrix}=\begin{bmatrix} 13 \\ -18 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix}=\begin{bmatrix} 2 & 1 \\ 1 & -3 \end{bmatrix}^{-1}\begin{bmatrix} 13 \\ -18 \end{bmatrix}$$

Linear System of Simultaneous Equations

$$x_1 + x_2 = 6$$

$$2x_1 + x_2 = 9$$

Solution

$$\begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 9 \end{bmatrix}$$

Note: Inverse of $\begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$ is $\begin{bmatrix} -1 & 1 \\ 2 & -1 \end{bmatrix}$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

Arithmetic Operations with Arrays

<u>Array multiplication and division</u> $\quad * \quad \backslash \quad$ and $\quad /$

$A*B$ follows the multiplication rules of matrices. It is defined only if the number of columns in A is equal to the number of rows in B.

\ is the left division. It is used to solve a matrix equation.

If: $A*x = B$ (x, and B are column vectors, A is a matrix)

Then: $x = A\backslash B$

/ is the right division. It is used to solve a matrix equation.

If: $x*C = D$ (x, and D are row vectors, C is a matrix)

Then: $x = D/C$

Arithmetic Operations with Arrays

Element-by-element multiplication, division, and exponentiation .* ./ .\ and .^

Element-by-element operations between two vectors or matrices is done by typing a period (.) in front of the arithmetic operator. Both arrays or vectors must be of the same size.

Element-by-element operations for **vectors**:

If: $a = [a_1\ a_2\ a_3\ a_4]$ and $b = [b_1\ b_2\ b_3\ b_4]$

Then: $a\ .*\ b = [a_1 b_1\ \ a_2 b_2\ \ a_3 b_3\ \ a_4 b_4]$

$$a\ ./\ b = [a_1/b_1\ \ a_2/b_2\ \ a_3/b_3\ \ a_4/b_4]$$

$$a\ .\hat{}\ b = [a_1\hat{}b_1\ \ a_2\hat{}b_2\ \ a_3\hat{}b_3\ \ a_4\hat{}b_4]$$

Arithmetic Operations with Arrays

Element-by-element operations for **matrices**:

Given: $A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$ **and** $B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$

Then:

$$A\ .*\ B = \begin{bmatrix} A_{11}B_{11} & A_{12}B_{12} & A_{13}B_{13} \\ A_{21}B_{21} & A_{22}B_{22} & A_{23}B_{23} \\ A_{31}B_{31} & A_{32}B_{32} & A_{33}B_{33} \end{bmatrix} \qquad A\ ./\ B = \begin{bmatrix} A_{11}/B_{11} & A_{12}/B_{12} & A_{13}/B_{13} \\ A_{21}/B_{21} & A_{22}/B_{22} & A_{23}/B_{23} \\ A_{31}/B_{31} & A_{32}/B_{32} & A_{33}/B_{33} \end{bmatrix}$$

$$A\ .\wedge\ 2 = \begin{bmatrix} (A_{11})^2 & (A_{12})^2 & (A_{13})^2 \\ (A_{21})^2 & (A_{22})^2 & (A_{23})^2 \\ (A_{31})^2 & (A_{32})^2 & (A_{33})^2 \end{bmatrix}$$

Also:

$$6 * A = 6\ .*\ A = \begin{bmatrix} 6A_{11} & 6A_{12} & 6A_{13} \\ 6A_{21} & 6A_{22} & 6A_{23} \\ 6A_{31} & 6A_{32} & 6A_{33} \end{bmatrix}$$

Any number

Matrix Element-by-Element Examples

```
>> A = [2, 6, 3; 5, 8, 4]
A =
    2    6    3
    5    8    4

>> B=[1, 4, 10; 3, 2, 7]
B =
    1    4    10
    3    2    7
```

```
>> A .* B
ans =
    2    24    30
   15    16    28
```

```
>> A./ B
ans =
    2.0000    1.5000    0.3000
    1.6667    4.0000    0.5714
```

```
>> B .^ 3
ans =
        1        64        1000
```

Matrix Element-by-Element Examples

a =

4	2	3
1	7	6
1	2	4

max(a) – *(max in Column)*

4 7 6

max(a’) *(transpose then max)*

4 7 4

mean(a) *(mean in Column)*

2.0000 3.6667 4.3333

mean(a’) *(transpose then mean)*

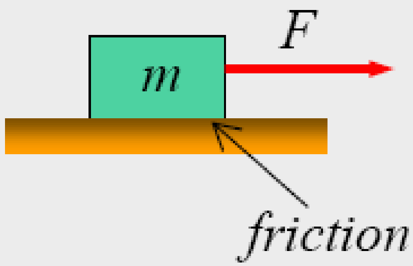
3.0000 4.6667 2.3333

Example of Matrix Operations

Element-by-element calculations are useful in processing data and in calculating the value of a mathematical function at many points.

EXAMPLE OF PROCESSING DATA

The coefficient of friction μ is determined by measuring the force F required to move a mass m by $\mu = F / (mg)$ (g = 9.81 m/s^2).
Results from measuring F in five tests are given in the table.
Determine the coefficient of friction in each test, and the average from all tests.



Mass m (kg)	2	4	5	10	20	50
Force F (N)	12.5	23.2	30	61	116	294

Example of Matrix Operations

```
>> mass = [2 4 5 10 20 50];
```

Create the **`mass`** vector.

```
>> force = [12.5 23.2 30 61 116 294];
```

Create the **`force`** vector.

```
>> mu = force./(9.81*mass)
mu =
    0.6371    0.5912    0.6116    0.6218    0.5912    0.5994
```

Calculate **`mu`** for each mass-force pair, using **element-by-element** calculations.

```
>> mu_ave = mean(mu)
meu_ave =
    0.6087
```

Determine the average of the elements in the vector **`mu`** by using the function **`mean()`**.

Example of Matrix Operations: Evaluation of Function

For the function:

$$y = \frac{z^3 + 5z}{4z^2 - 10}$$

calculate y for z = 1,3, 5, 7, 9, 11, 13, and 15.

SOLUTION USING MATLAB:

```
>> z = [1:2:15]

z =

     1     3     5     7     9    11    13    15

>> y = (z.^3+5*z)./(4*z.^2-10)

y =

  -1.0000    1.6154    1.6667    2.0323    2.4650    2.9241    3.3964
3.8764
```

Create a vector **z** with eight elements.

Vector **z** is used in element-by-element calculations of the elements of vector **y**.

Some Useful Notes about Variables

- All variables in MATLAB are **arrays**. A **scalar** is an array with one element, a **vector** is an array with one row or one column of elements, and a **matrix** is an array of rows and columns of elements.

- The variable type is defined by the input when the variable is created.

- A scalar, the elements in a vector, or the elements in a matrix can be real numbers, complex numbers, or expressions.

- The “who” command shows what variables are currently stored in the memory.

- The “whos” command lists the the variables currently stored in the memory, their type, and the amount of memory used by each.

Properties of Matrix operations

$$A + B = B + A, \tag{1}$$
$$(A + B) + C = A + (B + C), \tag{2}$$
$$A + 0 = A, \tag{3}$$
$$r(A + B) = rA + rB, \tag{4}$$
$$(r + s)A = rA + sA, \tag{5}$$
$$r(sA) = (rs)A; \tag{6}$$
$$A(BC) = (AB)C, \tag{7}$$
$$A(B + C) = AB + AC, \tag{8}$$
$$(B + C)A = BA + CA, \tag{9}$$

$$r(AB) = (rA)B = A(rB), \tag{10}$$
$$I_m A = A = A I_n; \tag{11}$$
$$(A^T)^T = A, \tag{12}$$
$$(A + B)^T = A^T + B^T, \tag{13}$$
$$(rA)^T = rA^T, \tag{14}$$
$$(AB)^T = B^T A^T, \tag{15}$$
$$(I_n)^T = I_n; \tag{16}$$
$$AA^{-1} = A^{-1}A = I_n, \tag{17}$$
$$(rA)^{-1} = r^{-1}A^{-1}, \ r \neq 0, \tag{18}$$
$$(AB)^{-1} = B^{-1}A^{-1}, \tag{19}$$
$$(I_n)^{-1} = I_n, \tag{20}$$
$$(A^T)^{-1} = (A^{-1})^T, \tag{21}$$
$$(A^{-1})^{-1} = A. \tag{22}$$