

Introduction to Engineering MATLAB – 3 Arrays

Agenda

- Creating arrays of numbers
 - Vectors: 1-D Arrays
 - Arrays: 2-D Arrays
- Array Addressing
- Strings & String Variables

1

Note

- In MATLAB, all variables are stored as arrays
- If the value of a variable is a single number, an $|x|$ array is used.

2

Arrays of numbers are used in many applications.

Examples:

Arrays of numbers can represent data:

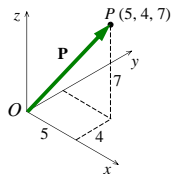
Year	1984	1986	1988	1990	1992	1994	1996
Population	127	130	136	145	158	178	211

Array of numbers can represent a vector.

An example is a position vector. The location of point P in a three dimensional space can be represented by the three Cartesian coordinates 5, 4, and 7.

A position vector that points to the location of point P relative to point O (the origin of the coordinate system) is defined by:

$$\mathbf{P} = 5\mathbf{i} + 4\mathbf{j} + 7\mathbf{k}$$



3

In MATLAB, a vector, or any list of numbers, can be entered in a horizontal (row) or vertical (column) vectors. A vector is a one-dimensional array

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:

```
[1984]
[1986]
[1988]
[1990]
[1992]
[1994]
[1996]

[127]
[130]
[136]
[145]
[158]
[178]
[211]
```

The position vector can be entered in a:

row: [5 4 7]

column: $\begin{bmatrix} 5 \\ 4 \\ 7 \end{bmatrix}$

4

CREATING A VECTOR IN MATLAB

A vector is created by typing the elements (numbers) inside square brackets [].

To create a **ROW VECTOR** type a space or a comma between the elements inside the square brackets.

```
>> yr=[1984 1986 1988 1990 1992 1994 1996]
yr =
    1984    1986    1988    1990    1992    1994    1996
```

```
>> cor = [5,4,7]
cor =
     5     4     7
```

NOTE: MATLAB is not "picky" about how the data is typed in. You can type spaces before and/or after the = sign. Between the elements you can have a space in addition to the comma, or you can type more than one space.

5

To create a **COLUMN VECTOR** type a left bracket [and then enter the elements with a semicolon between them, or press **Enter** after each element. Type a right bracket] after the last element.

```
>> pop = [127; 130; 136; 145; 158; 178; 211]
pop =
    127
    130
    136
    145
    158
    178
    211

>> cor = [5
          4
          7]
cor =
     5
     4
     7
```

6

CREATING A VECTOR WITH CONSTANT SPACING

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
```

7

CREATING A VECTOR BY SPECIFYING THE FIRST AND LAST TERMS, AND THE NUMBER OF TERMS

A vector in which the first term is x_i , the last term is x_f , and the number of equally-spaced terms is n , can be created by typing $\text{linspace}(x_i, x_f, n)$.

```
>> 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
```

8

TWO DIMENSIONAL ARRAY - MATRIX

A matrix is a two dimensional array of numbers.

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

```
7 4 9
3 8 1
6 5 3
```

Three rows and three columns (3x3)

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

```
31 26 14 18 5 30
3 51 20 11 43 65
28 6 15 61 34 22
14 58 6 36 93 7
```

Four rows and six columns (4x6)

$(m \times n)$ matrix has m rows and n columns

$(m \times n)$ is called the size of the matrix

9

CREATING A MATRIX IN MATLAB

A Matrix is created by typing the elements (numbers) row by row inside square brackets [].

Type the left bracket [, then type in the first row separating the elements with spaces or commas. To type the next row type a semicolon or press **Enter**. Type the right bracket] at the end of the last row.

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

Type and press **Enter**

```
a =
     1     2     3
     4     5     6
     7     8     9
```

Computer response

```
>> b = [11 12 13 14 15
16 17 18 19 20
21 22 23 24 25]
```

Type and press **Enter** after each row and after the].

```
b =
    11    12    13    14    15
    16    17    18    19    20
    21    22    23    24    25
```

Computer response

10

THE TRANSPOSE OPERATION

The transpose operation ⁴

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

For a matrix: Interchanges the rows and columns.

Example for a vector:

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

11

THE TRANSPOSE OPERATION

Example for a matrix:

```
>> c = [2 55 14 8; 21 5 32 11; 41 64 9 1]
c =
     2    55    14     8
    21     5    32    11
    41    64     9     1
>> d = c'
d =
     2    21    41
    55     5    64
    14    32     9
     8    11     1
```

12

ARRAY ADDRESSING (VECTOR)

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

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

```
>> 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
```

13

ARRAY ADDRESSING (MATRIX)

The address of an element in a Matrix is its position, defined by the number of row and the number of column.

For a matrix "m", m(k,p) refer 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
```

```
>> m(1,1)
ans =
     3
>> m(2,3)
ans =
    10
```

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

```
>> m(3,1)=20
m =
    20    11     6     5
     4     7    10     2
    13     9     0     8
```

Single elements can be used like variables in computations:

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

14

USING A COLON (:) IN ADDRESSING ARRAYS

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

For a vector:

v(:) Represents all the elements of a vector (either row vector or 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
```

15

USING A COLON (:) IN ADDRESSING ARRAYS

For a matrix:

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.

16

EXAMPLES OF USING A COLON (:) IN ADDRESSING ARRAYS

Define a matrix

```
>> 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
```

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

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

17

EXAMPLES OF USING A COLON (:) IN ADDRESSING ARRAYS (CONT.)

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
```

```
>> E = A(2:4,:)
```

```
E =
     2     4     6     8    10
     3     6     9    12    15
     4     8    12    16    20
```

```
>> 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
```

```
>> F = A(1:3,2:4)
```

```
F =
     3     5     7
     4     6     8
     6     9    12
```

18

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.
- The element (scalar) or elements (vector, matrix) of a variable can be numbers (real or complex), 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.

19

EXAMPLE

```
>> 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
```

20

STRINGS AND STRING VARIABLES

- ❖ Strings are characters enclosed in single quotes.
- ❖ A string can include letters, numbers, other symbols, and spaces.
- ❖ Examples of strings: 'ad ef ', '%fr2', '{edcba :21!'.
- ❖ Strings can be used to define variables.
- ❖ Strings are used in the input of some functions.

21

STRING VARIABLES

A variable can be defined as a string by typing:
Variable name = ' string '

```
>> a = 'ERty 8'
a =
ERty 8
```

```
>> B = ['My name is John Smith']
B =
My name is John Smith
```

- Strings are stored as row vectors in which every character, including spaces, is an element.
- In the variables above, a has 6 elements, and B has 21 elements.
- The elements can be addressed directly as in numerical vectors.
- In the variables above:

```
>> a(4)
ans =
y
```

```
>> B(12)
ans =
J
```

22

STRING VARIABLES

The string variable:

```
>> x = '536'
x =
536
```

is not the same as the number variable:

```
>> x = 536
x =
536
```

The number variable can be used in calculations while the string variable can not.

An important application of strings is in creating input prompts and output messages. This will be shown later when script files are discussed.

23

Assignment #3

- Do the problems below in the command window. Start each problem in a new (clear) window. The first two lines in each problem should be:
 - % First Last, CID
 - % MATLAB 3, Problem Number Page Number
- Submit the printout of the command window.
- Print each problem separately.
 - Problem 1, page 105 in the textbook.
 - Problem 3, page 105 in the textbook.
 - Problem 5, page 106 in the textbook.
 - Problem 6, page 106 in the textbook.

24