CSE 1570
Creating Vectors and Matrices

Instructor: Aijun An
Department of Computer Science and Engineering
York University
aan@cse.yorku.ca

http://www.cse.yorku.ca/course/1570

Outline

- What are vectors and matrices?
- Creating a vector
- Creating a matrix

Vectors and Matrices

Vectors and *matrices* are fundamental data structures that MATLAB uses to store and manipulate data.

A *vector* (also called *one-dimensional array*) is a list of numbers, placed in a row or a column.

- A row vector:

```
87 65 98 45 76
```

- A column vector :

```
87
65
98
45
76
```

The individual items in a vector are called its *elements*.

Vectors and Matrices

A *matrix* (also called two-dimensional array) is an arrangement of numbers into rows and columns.

```
 17 101  98  45 234
102  26  76 102  42
 80  65 267  87  96
```

The *size* of a matrix is **M** × **N**, where

- **M** is the number of rows
- **N** is the number of columns

The size of the above matrix is

3 × **5**

The individual items in a matrix are called its *elements*.

Vectors and Matrices

A vector is a matrix of

- size **1** × **n** (for row vector) or
- size **n** × **1** (for column vector),

where **n** is the number of elements in the vector

A single value (called *scalar value*) is a vector with one element and a matrix of size **1×1**.

Creating a Vector

A vector is created by assigning the elements of the vector to a variable, which can be done in several ways:

1. **Creating a vector from a known list of numbers**

```
variable_name = [ vector elements ]
```

- Row vector:

```
scores=[87 65 98 45 76]
```

or

```
scores=[87,65,98,45,76]
```

Separator is either ***space*** or a ***comma***

Creating a Vector

- Column vector:

```
scores=[87;65;98;45;76]
```

Separator is a ***semicolon***.

Output of the above command:

```
scores =
        87
        65
        98
        45
        76
```

Example of Using a Vector

If we have created a row or column vector as follows:

```
scores=[87 65 98 45 76]
```

or

```
scores=[87;65;98;45;76]
```

Can calculate the average score using

```
mean(scores)
```

Output from MATLAB:

```
ans =
   74.2000
```

Creating a Vector

- Can use a number, a math expression, a predefined variable or a function to specify an element:

```
x=10; y=5;
v=[12, sqrt(100)+2, x+y, log(5)]
```

Output:

```
v =

   12.0000   12.0000   15.0000    1.6094
```

Exercise 1

Define a row vector x that has the elements:

6, 8×3, 81, $e^{2.5}$, $\sqrt{65}$, π/3, $\log_{10}23.5$

Command:

```
x=[6, 8*3, exp(2.5), sqrt(65), pi/3, log10(23.5)]
```

Output:

```
x =
  6.0000 24.0000 12.1825  8.0623  1.0472  1.3711
```

Exercise 2

Define two variables x=0.85, y=12.5, and then use them to create **a column vector** z that has the following elements:

y, y^x, ln(y/x), y·x, and x+y

Commands:

```
x=0.85; y=12.5;
z=[y; y^x; log(y/x); y*x; x+y]
```

Output:

```
z =
   12.5000
    8.5580
    2.6882
   10.6250
   13.3500
```

Creating a Vector with Shorthand

2. Creating a vector with constant spacing by specifying the first element, the spacing and the last element:

- Constant spacing: the distance between adjacent elements is the same, such as:

```
1990 1992 1994 1996 1998
```

- Command:

```
variable_name = [m:q:n]
```

or

```
variable_name = m:q:n
```

where `m` is the first element, `q` is the spacing, `n` is the last element.

Creating a Vector with Shorthand

- Examples:

```
 >> yr=[1990:2:1998]
yr =

      1990    1992    1994    1996    1998


 >> x=[2.1:-0.2:1.5]
 x =

    2.1000     1.9000     1.7000     1.5000
```

Creating a Vector with Shorthand

- If **n** cannot be obtained by adding **q**'s to **m**, the last element will be the last number that does not exceed **n** (in the direction of change):

Examples:

```
>> x=[5:2:10]
x =

   5    7    9

 >> y=[19:-2: 10]
 y=
    19     17     15     13     11
```

Creating a Vector with Shorthand

- If only two numbers are typed, the spacing is omitted. In this case, the spacing is 1.

```
 >> z=[-3:2]
z =

     -3     -2     -1      0      1      2
```

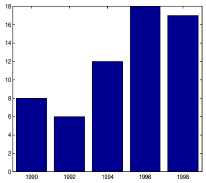
If spacing is omitted, the default is 1.

Example of Using Vectors

Year	1990	1992	1994	1996	1998
Sales	8	6	12	18	17

If we would like to plot the Sales amount against the year using a bar chart, use the following:

```
>> year=[1990:2:1998];
>> sales=[8, 6, 12, 18, 17];
>> bar(year,sales)
```



Exercise 3

Define a row vector y in which the first element is 0, and the last element is 24 with an increment of 3 between elements

Command:

```
y=[0:3:24]
```

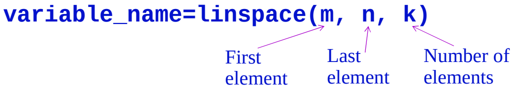
Output:

```
y =
     0     3     6     9    12    15    18    21    24
```

Creating a Vector with Shorthand

3. **Creating a vector with constant spacing by specifying the first and last elements and the number of elements:**

- Using the **`linspace`** function:



```
variable_name=linspace(m, n, k)
```

- MATLAB automatically determines the correct spacing by:

$$\frac{n-m}{k-1}$$

Creating a Vector with Shorthand

- Examples

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

>> va=linspace(20, 2, 4)
 va =
    20    14     8     2
```

Creating a Vector with Shorthand

- You can omit the number of elements in `linspace`. Its default value is 100.

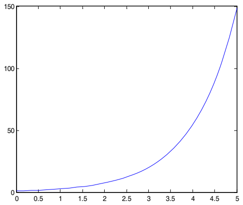
```
 >> u=linspace(0, 20)
u =
  Columns 1 through 7

      0  0.2020  0.4040  0.6061 0.8081  1.0101  1.2121
  Columns 8 through 14

 1.4141  1.6162  1.8182  2.0202 2.2222  2.4242  2.6263
   ……

  Columns 99 through 100

   19.7980   20.0000
```

Example of Using Vectors

Creating vectors using `linspace` is handy when plotting a function.

Plot *y=e^x^* over 0≤x ≤5

```
>> x=linspace(0,5,30);
>> y=exp(x);
>> plot(x,y)
```



The Transpose Operator

The transpose operator (**’**), when applied to a vector,

- converts a row vector into a column vector
- converts a column vector into a row vector

Example:

```
>> aa=[3 8 1]
aa =
     3   8   1
>> bb=aa’
bb =
         3
         8
         1
```

Exercise 4

Create a row vector z with 16 equally spaced elements in which the first element is 61 and the last element is 5.

Command:

```
z=linspace(61, 5, 16)
```

Output:

```
z =
     61.0000    57.2667    53.5333, …, 5.0000
```

What if we want to create a column vector of these elements?

Some Built-in Functions for Vectors

Assume A is a vector:

Function	Description	Example

`max(A)`	Returns the largest value in **A**	`>> A=[5 9 2 4]`
		`>> max(A)`
		`ans =`
		`9`

`min(A)`	Returns the smallest value in **A**	`>> A=[5 9 2 4]`
		`>> min(A)`
		`ans =`
		`2`

`sum(A)`	Returns the sum of the elements in **A**	`>> A=[5 9 2 4]`
		`>> sum(A)`
		`ans =`
		`20`

Some Built-in Functions for Vectors

Assume A is a vector:

Function	Description	Example

`mean(A)`	Returns the mean value of the elements in **A**	`>> A=[5 9 2 4]`
		`>> mean(A)`
		`ans =`
		`5`

`median(A)`	Returns the median value of elements in **A**	`>> A=[5 9 2 4]`
		`>> median(A)`
		`ans =`
		`4.5000`

`sort(A)`	Output the elements in A in value ascending order	`>> A=[5 9 2 4]`
		`>> sort(A)`
		`ans =`
		`2 4 5 9`

Some Built-in Functions for Vectors

Assume A is a vector:

Function	Description	Example

`length(A)`	Returns the number of elements in **A**	`>> A=[5 9 2 4]`
		`>> length(A)`
		`ans =`
		`4`

Exercise 5

Enter the following test scores into a vector s and calculate the highest, lowest and average score. Rank the score in ascending order:

87, 65, 98, 45, 76, 65, 77, 56, 82

Commands:

```
s=[87, 65, 98, 45, 76, 65, 77, 56, 82];
Highest=max(s), Lowest=min(s)
Average=mean(s), sort(s)
```

Output:

Outline

- What are vectors and matrices?
- Creating a vector
- *Creating a matrix*

Creating a Matrix

A matrix can be created by assigning the elements of the matrix to a variable. Can be done in several ways:

1. Using "**;**" to separate rows:

```
variable_name=[1st row elements; 2nd row
   elements ; 3rd row elements; …; last row
   elements]
```

Within a row, the elements can be separated by either space or comma.

Example:

```
>> a=[5 35 43; 4 76 81; 21 32 40]
a =

     5    35    43
     4    76    81
    21    32    40
```

Note that all the rows must have the same number of elements!

Creating a Matrix

2. Using "**Enter**" to separate rows:

```
variable_name=[1st row elements
      2nd row elements
      3rd row elements
       …
      last row elements]
```

Example:

```
>> a=[5 35 43
      4 76 81
      21 32 40]
a =
     5    35    43
     4    76    81
    21    32    40
```

Within a row, the elements can be separated by either space or comma.

Note that all the rows must have the same number of elements!

Creating a Matrix

3. Using the shorthand methods for creating vectors to create rows of the matrix.

Example:

```
>> a=[1:2:11; 0:5:25; linspace(10,60,6);
     67 2 43 68 4 13]

a =
     1     3     5     7     9    11
     0     5    10    15    20    25
    10    20    30    40    50    60
    67     2    43    68     4    13
```

Note that all the rows must have the same number of elements!

31

Creating a Matrix

Same as in creating vectors, the elements can be math expressions, and contains variables and functions.

Example:

```
>> a=2; b=3; c=4;
>> mat=[a^b, c+a*b, exp(a); linspace(3,9,3);
     1:3]

mat =

    8.0000   10.0000    7.3891
    3.0000    6.0000    9.0000
    1.0000    2.0000    3.0000
```

32

Concatenating Matrices

4. Matrices can be joined together to form a bigger matrix

Example:

```
>> F=[10 11 12];
>> G=[13 14 15];
>> H=[F;G]

H =

    10    11    12
    13    14    15
```

F and **G** must have the same number of columns

What is the result of `A=[H; F]`?

What about `B=[H; A]`?

33

Creating Special Matrices

The following built-in ***functions*** can be used to create special matrices:

- `zeros(m, n)` creates a `m×n` matrix of zeros

  Example:

```
>> z=zeros(3,4)
z =

     0     0     0     0
     0     0     0     0
     0     0     0     0
```

34

Creating Special Matrix

- `ones(m, n)` creates a `m×n` matrix of ones

  Example:

```
>> b=ones(4,3)

b =

     1     1     1
     1     1     1
     1     1     1
     1     1     1
```

35

Creating Special Matrices

- `eye(n)` can be used to create a `n×n` square matrix in which the *diagonal elements* are equal to 1 and the rest are 0. Such a matrix is called ***identity matrix***.

  Example:

```
>> idn=eye(3)

idn =

     1     0     0
     0     1     0
     0     0     1
```

36

Exercise 6

Use three ways to create a 4×5 matrix **M** in which the first two rows are 0's and the next two rows are 1's.

Solution 1:

```
M=[0 0 0 0 0; 0 0 0 0 0; 1 1 1 1 1; 1 1 1 1 1]
```

Solution 2:

```
A=zeros(2,5); B=ones(2,5); M=[A;B]
```

Solution 3:

```
M=[linspace(0, 0, 5); linspace(0,0,5);
  linspace(1,1,5); linspace(1,1,5)]
```

Some Built-in Matrix Functions

X is a matrix below:

Function	Description	Example

`size(X)`	Returns a row vector [m, n]	`>> X=[6, 2, 3, 4; 2, 3, 19, 2]`
		`X=`
		`6 2 3 4`
		`2 3 19 2`
		`>> size(X)`
		`ans =`
		`2 4`

`length(X)`	Returns the larger of its number of rows and columns	`>> X=[6, 2, 3, 4; 2, 3, 19, 2]`
		`X=`
		`6 2 3 4`
		`2 3 19 2`
		`>>length(X)`
		`ans =`
		`4`

Some Built-in Matrix Functions

X is a 2-dimensional matrix below:

Function	Description	Example

`max(X)`	Returns a vector in which each element is the largest number in the corresponding column of **X**	`>> X=[5 9;2 4;1 3];`
		`>> max(X)`
		`ans =`
		`5 9`

`min(X)`	Returns a vector in which each element is the smallest number in the corresponding column of **X**	`>> X=[5 9;2 4;1 3];`
		`>> min(X)`
		`ans =`
		`1 3`

`sum(X)`	Returns a vector in which each element is the sum of the values in the corresponding column of **X**	`>> X=[5 9;2 4;1 3];`
		`>> sum(X)`
		`ans =`
		`8 16`

Some Built-in Matrix Functions

X is a 2-dimensional matrix below:

Function	Description	Example

`mean(X)`	Returns a vector in which each element is the average of the values in the corresponding column of **X**	`>> X=[5 9;2 4;1 3];`
		`>> mean(X)`
		`ans =`
		`2.6667 5.3333`

`median(X)`	Returns a vector in which each element is the median value of the corresponding column of **X**	`>> X=[5 9;2 4;1 3];`
		`>> median(X)`
		`ans =`
		`2 4`

`sort(X)`	Sort each column of X in value ascending order	`>> X=[5 9;2 4;1 3];`
		`>> sort(X)`
		`ans = 1 3`
		`2 4`
		`5 9`

Home Exercise

Given two lists of test scores of a group of students:

Midterm score	Final score
76	80
69	75
90	67
82	87
38	78
67	92

Create a matrix to contain the scores, and then calculate the maximum, minimum and average scores for midterm and final.

Next Class

Location: AP Labs (TEL 2027&2032)

Topics:

- Addressing elements of matrices
- Script files