



# COMPUTER SKILLS

## LESSON 4

---

Valeria Cardellini  
cardellini@ing.uniroma2.it  
A.Y. 2015/16

# Objectives of this lesson

We'll discuss:

- Vector and matrix introduction
- How to create vectors
- How to refer to and modify elements of vectors
- Vector and matrix dimension

# Concept: data collections

Two very common ways to group and store set of values of the same type: **vectors** and **matrices**

**Data abstraction** (i.e., the abstraction of data structures) allows us to refer to groups of data collectively, e.g.,:

- “All the temperature readings for September”
- “All the purchases from Walmart”

We can also perform mathematical or logical operations on these groups, e.g.

- average, maximum, or minimum temperatures for a month

A **homogeneous data collection** is constrained to accept only items of the same data type

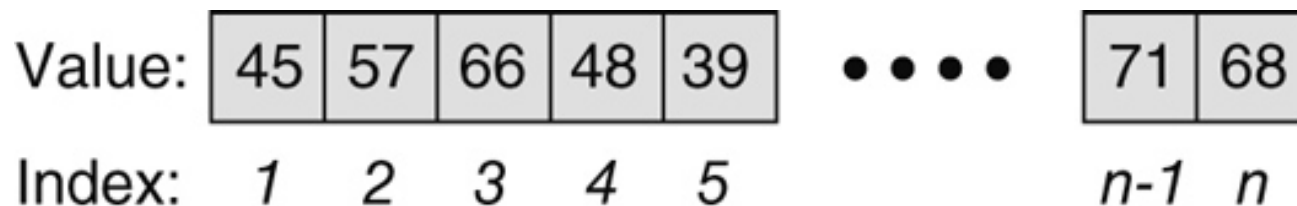
- In this case numbers

# Vectors

Individual items in a vector are usually referred to as its **elements**. Each vector element has two separate and distinct attributes that make it unique in a specific vector:

- Its *numerical value* and
- Its *position (index)* in the vector

Example: the individual number 66 is the third element in this vector. Its value is 66 and its index is 3. There may be other items in the vector with the value of 66, but no other item will be located in this vector at position 3.



# Matrices

A matrix is a table of values. The dimensions of a matrix are  $m \times n$ , where  $m$  is the number of rows and  $n$  is the number of columns.

$$A_{(m \times n)} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

# Types of vectors

- **Row vector**

- One row, any number of columns
- Dimension:  $1 \times n$

- **Column vector**

- One column, any number of rows
- Dimension:  $m \times 1$

- **Scalar** (one element)

- Dimension:  $1 \times 1$

- Vectors and scalars are just special types of matrices

- Vector: a matrix with only a single row or a single column

- In other programming languages:

- Vector: one-dimensional array
- Matrix: two-dimensional array

column vector

|     |
|-----|
| 144 |
| 13  |
| 25  |
| 108 |
| 96  |
| 61  |
| 73  |
| 60  |
| 48  |
| 109 |

row vector

|   |     |     |   |   |     |     |    |    |     |
|---|-----|-----|---|---|-----|-----|----|----|-----|
| 3 | 141 | 140 | 6 | 7 | 137 | 136 | 10 | 11 | 133 |
|---|-----|-----|---|---|-----|-----|----|----|-----|

# Vector manipulation

We consider the following basic operations on vectors in MATLAB:

- Create a vector
- Determine the size (dimension) of a vector
- Extract data from a vector by indexing
- Shorten a vector
- Mathematical and logical operations on vectors

# Creating a vector

- We will examine several ways to create row vector variables in MATLAB
  1. Constant values using the assignment statement
  2. Values as a range using the colon operator
  3. Values as evenly distributed in a linear space using the `linspace` function
  4. Filled with 0s, 1s, or random values

# 1. Creating a row vector: constant values

- Entering the values directly: in MATLAB, you create a vector by enclosing the elements in square brackets (commas are optional)

```
>> A = [2 5 7 1 3]; %values separated by space
```

```
>> A = [2, 5, 7, 1, 3]; %values separated by comma
```

- In the next slides you'll see how to enter the values automatically:
  2. in a range of numbers
  3. evenly distributed in a linear space
  4. as random, zeros, ones

## 2. Creating a row vector: entering the values as a range

$$B = [v_1, v_2, \dots, v_n] =$$

$$\{v_i \mid v_{\min} \leq v_i \leq v_{\max}, v_i = v_{i-1} + K, v_1 = v_{\min}, i = 2, \dots, n\}$$

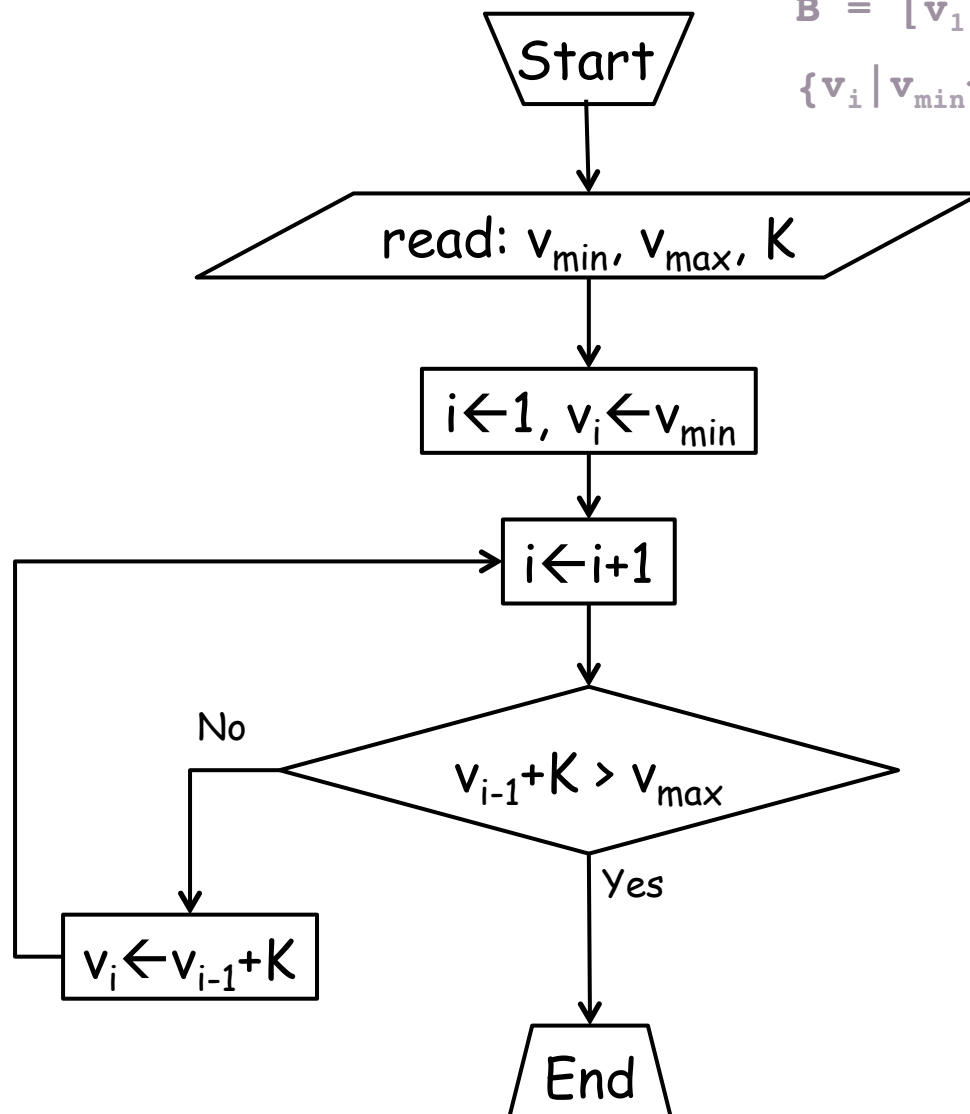
K is the **step value** (integer constant value)

Example:  $K=3$ ,  $v_{\min}=1$  and  $v_{\max}=20$

$$B = [1, 4, 7, 10, 13, 16, 19]$$

Let's define an algorithm to create vector B

## 2. Flow diagram



$$B = [v_1, v_2, \dots, v_n] =$$

$$\{v_i \mid v_{\min} \leq v_i \leq v_{\max}, v_i = v_{i-1} + K, v_1 = v_{\min}, i = 2, \dots, n\}$$

In the next lectures you will learn how to implement such algorithm in MATLAB

## 2. Creating a row vector: entering the values as a range

- How to do in MATLAB? Use the **colon operator** :

$$B = \{v_i \mid v_{\min} \leq v_i \leq v_{\max}, v_i = v_{i-1} + K, v_1 = v_{\min}, i=2, \dots\}$$

$$B = v_{\min} : K : v_{\max}$$

Example

```
>> B = 1:3:20
```

```
B = 1 4 7 10 13 16 19
```

This syntax is an implementation of the algorithm we have previously defined

### 3. Creating a vector: as evenly distributed in a linear space

$C = [v_1, v_2, \dots, v_N] = \{N \text{ evenly spaced points between } x_{\min} \text{ and } x_{\max}\}$

Example:  $N=11$ ,  $x_{\min}=0$  and  $x_{\max}=20$

$C = [0 \ 2 \ 4 \ 6 \ 8 \ 10 \ 12 \ 14 \ 16 \ 18 \ 20]$

- MATLAB offers the **linspace** built-in function:

$C = \text{linspace}(x_{\min}, x_{\max}, N)$

Example

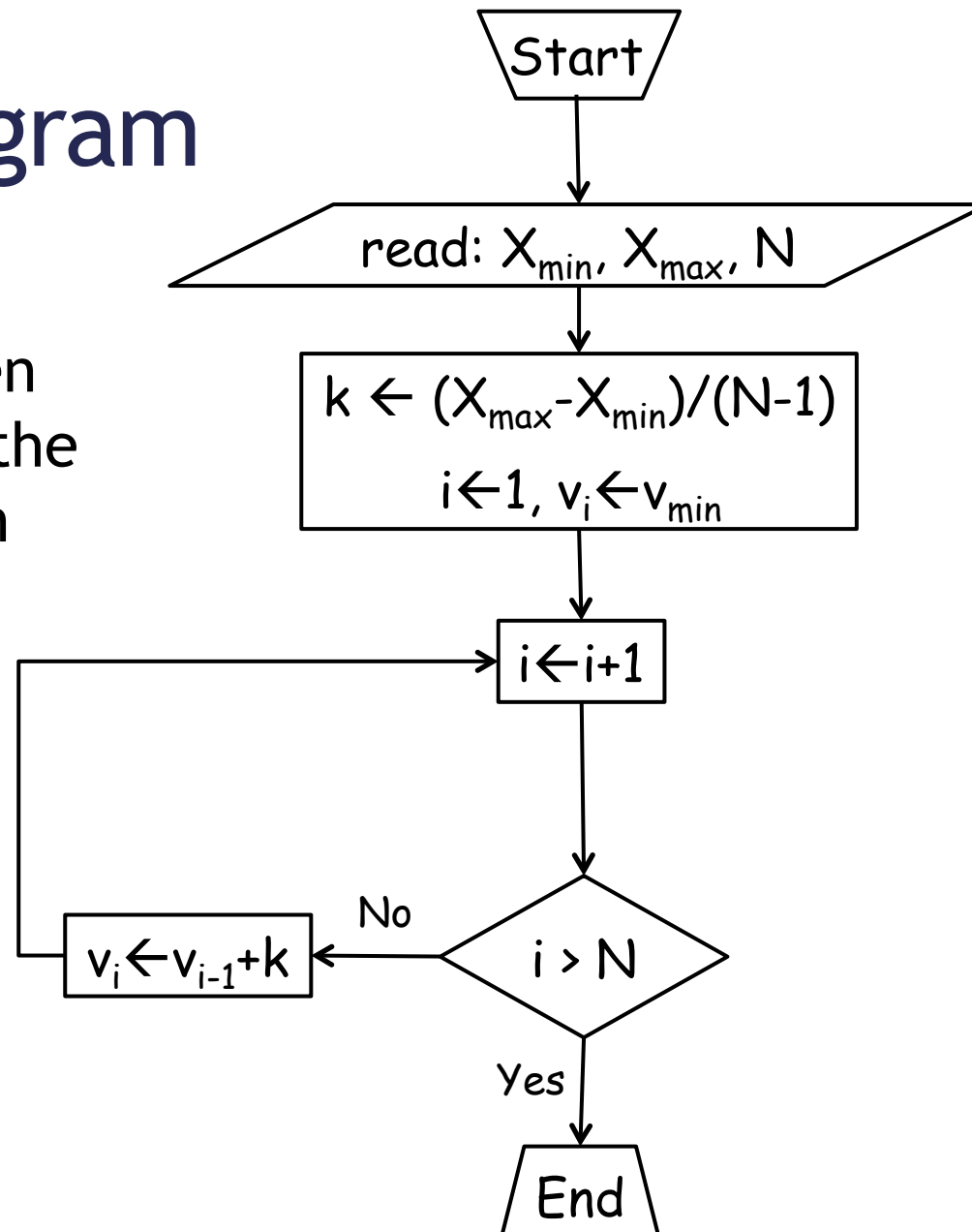
```
>> C = linspace(0,20,11)
```

```
C = 0 2 4 6 8 10 12 14 16 18 20
```

- **linspace** is similar to the colon operator but gives direct control over the number of points and always includes the endpoints

### 3. Flow diagram

- $k$  holds the distance between  $x_i$  and  $x_{i+1}$  (i.e., the spacing between the points)



## 4. Creating a vector: filled with 0, 1, or random values

$B = [v_1, v_2, \dots, v_N];$

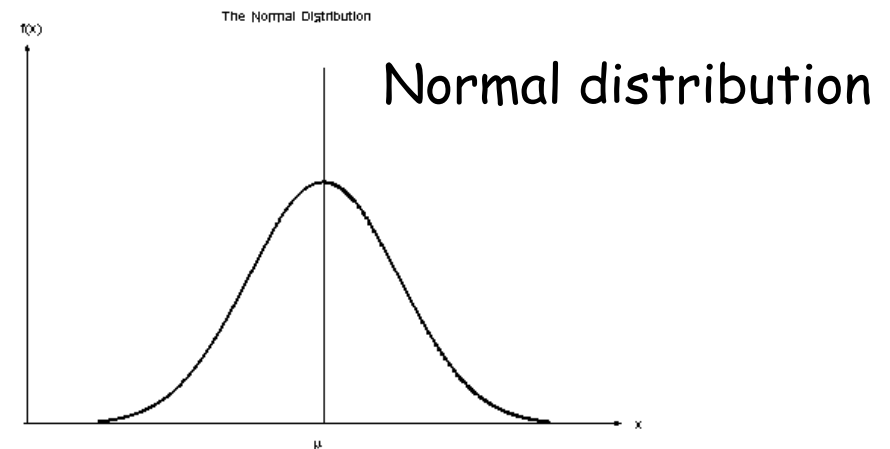
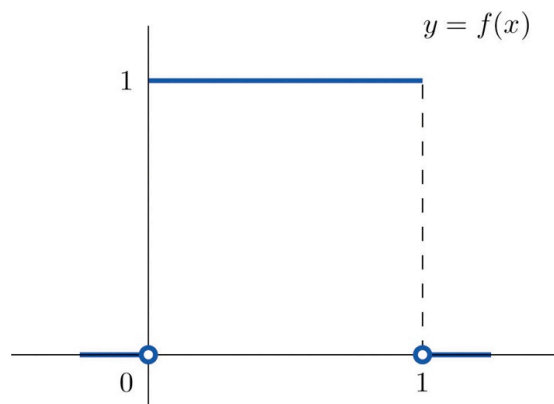
$v_i = 0$  for  $i = 1, \dots, N$

$v_i = 1$  for  $i = 1, \dots, N$

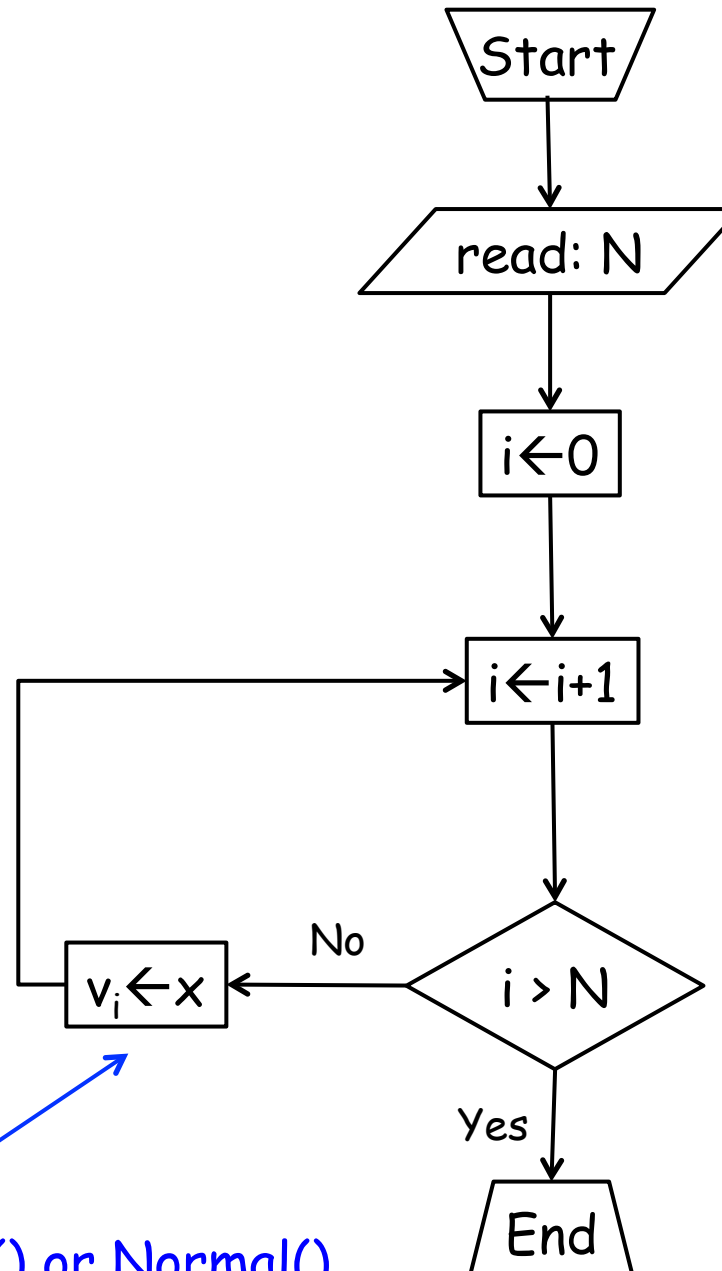
$v_i = x_k$  where  $x_k$  is a pseudorandom value  
uniformly or normally distributed

**All the cases can use the same algorithm**

Uniform distribution



## 4. Flow diagram



$x$  can be 0, 1, Uniform() or Normal()

## 4. Creating a vector: filled with 0, 1, or random values

MATLAB offers built-in functions to implement the previous algorithm: **zeros**, **ones**, **rand**, or **randn**

```
%B = [v1, v2, ..., vN];  
B=zeros(1, N); %vi=0 for i=1,...,N  
B=ones(1, N); %vi=1 for i=1,...,N  
B=rand(1, N); %vi=xk where xk is a  
               %pseudorandom value  
               %uniformly distributed  
B=randn(1, N); %vi=xk where xk is a  
               %pseudorandom value  
               %normally distributed
```

# Size of vectors and matrices

Given a vector (matrix), we want to know the number of its elements (i.e., its size)

MATLAB provides two built-in functions to determine the size of arrays in general:

- **size(A)** when applied to array A returns a vector containing two quantities: the number of rows and the number of columns
- **length(A)** returns the maximum value of the array size
  - If A is a vector, it corresponds to its length
  - If A is a matrix, it corresponds the number of rows or columns, whichever is greater

# Indexing a vector

- The process of referring to and modifying elements in a vector
- The vector elements are sequentially numbered, in MATLAB starting from 1
- Syntax:
  - `v(index)` returns the element(s) at the location(s) specified by the vector *index*.
  - `v(index)=value` modifies the elements at the location(s) specified by the vector *index*.
- The **index vector** may contain either numerical or logical values

```
>> A=1:3:12
A =
     1     4     7    10
>> A(1) %read first element
ans =
     1
>> B=[ 2, 4 ]
B =
     2     4
>> A(B)
ans =
     4    10
```

**B: index vector**

# An example of index vector

```
>> A=randn(1,10) %create A filled with normal values
A =
    Columns 1 through 7
    8.8840e-01   -1.1471e+00   -1.0689e+00   -8.0950e-01   -2.9443e+00
    1.4384e+00    3.2519e-01
    Columns 8 through 10
   -7.5493e-01    1.3703e+00   -1.7115e+00
>> A=round(A)
A =
     1     -1     -1     -1     -3      1      0     -1      1     -2
>> iv=2:2:10
iv =
     2      4      6      8     10
>> A(iv)
ans =
    -1     -1      1     -1     -2
>> A(2:2:10)
ans =
    -1     -1      1     -1     -2
```

# Numerical indexing

- The index vector may be of any length
- It should contain integer (non-fractional) numbers
- The values in the index vector are constrained by the following rules:
  - For reading elements, all index values must be  
 $1 \leq \text{element} \leq \text{length}(\text{vector})$
  - For modifying elements, all index values must be  
 $1 \leq \text{element}$

# Shortening an array

- Not necessary: it is better to extract what you want by indexing rather than removing what you do not want
  - Possible problems when the length of a vector changes
- Anyway, it can be achieved by assigning the **empty vector** (a vector that stores no value and is created using `[]`) to elements of a vector, or to entire rows or columns of a matrix

```
>> vec = 3:5
```

```
vec =
```

```
    3    4    5
```

```
>> vec(2)=[ ] %remove the second element
```

```
vec =
```

```
    3    5
```

## Creating a column vector

- Enter directly the values in square brackets but separated by semicolons (rather than commas or spaces as for row vectors)

```
>> c = [2; 5; 7; 1]
```

```
c =
```

```
2
```

```
5
```

```
7
```

```
1
```

- No direct way to use the colon operator
- However, you can create a row vector and then **transpose** it

```
>> r=1:3; % create a row vector
```

```
>> c = r' % for transposing use the apostrophe
```

```
c =
```

```
1
```

```
2
```

```
3
```