THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE

ENG 101
Arrays in Matlab

Dr. Emil Jovanov
Electrical and Computer Engineering
University of Alabama in Huntsville
emil.jovanov@uah.edu
http://www.ece.uah.edu/~jovanov

THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE

Arrays (Vectors and Matrices)

- Used to store and manipulate data
- Simple array (One-Dimensional) is a list of numbers in a row or column: *Vectors*
- Two-Dimensional arrays consist of numbers in rows and columns: *Matrices*
- Square brackets "[ ]" are used to define arrays

Arrays (Vectors and Matrices)

One-Dimensional Arrays

A=[12 -4 7 0.58 1.56] (row vector (1X5))

B=[2; 4; -8] (column vector (3X1))

**We can turn a row vector into a column vector and vice-versa by using the transpose command

Two-Dimensional Arrays

X = [2.1 3.3; 5.1 4.2] — 2 X 2 matrix

A = [1, 2; 3, 4; 5, 6] — 3 X 2 matrix

Special Arrays

- zeros(2,3) is an array with 2 rows and 3 columns where every element is 0
- ones(4,5) is an array with 4 rows and 5 columns where every element is 1
- eye(5) unity

Go to MATLAB

- One-Dimensional Row Vector:
  variable name = [element_1 element_2 element_3…]
  - Create a row vector that contains the first 5 even numbers using the variable name *R*
  - Create Rt = transpose(R)
  - Note that Rt=R' has the same effect
- One-Dimensional Column Vector:
  variable name = [element_1; element_2; element_3;…]
  - Create a column vector that contains the first 4 odd numbers using the variable name *C*
  - Create Ct = transpose(C)
  - Note that Ct=C' has the same effect
- Two-Dimensional Arrays (Matrices):
  variable name = [row 1 elements; row_2 elements; row_3 elements;…]
- Create a (2X3) matrix with the numbers 1-6 (in order by rows) using the variable name *M*

Spaced Arrays

One-Dimensional vector with constant spacing of elements between a starting term and an ending term – **we choose the spacing**

variable name = [m : q : n]

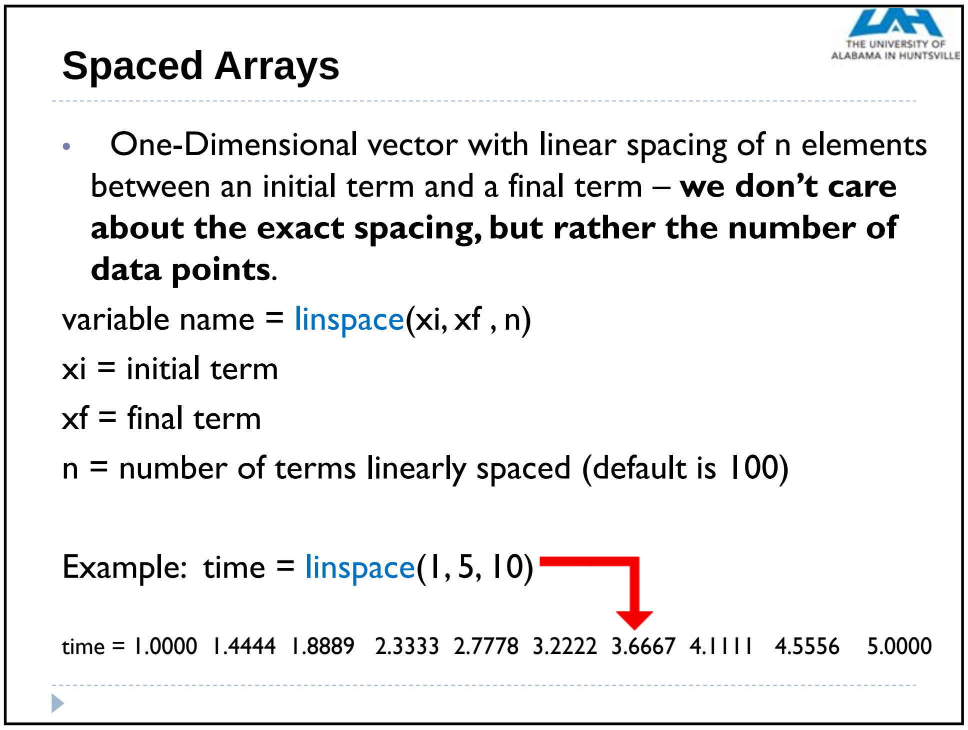
m = starting term

n = ending term

q = spacing

Example: x = [1 : 1: 10] $\rightarrow$
x = [1 2 3 4 5 6 7 8 9 10]

decades= [2000: 10: 2050] $\rightarrow$
decades = [2000 2010 2020 2030 2040 2050]



Spaced Arrays

- One-Dimensional vector with linear spacing of n elements between an initial term and a final term – **we don't care about the exact spacing, but rather the number of data points**.

variable name = linspace(xi, xf , n)

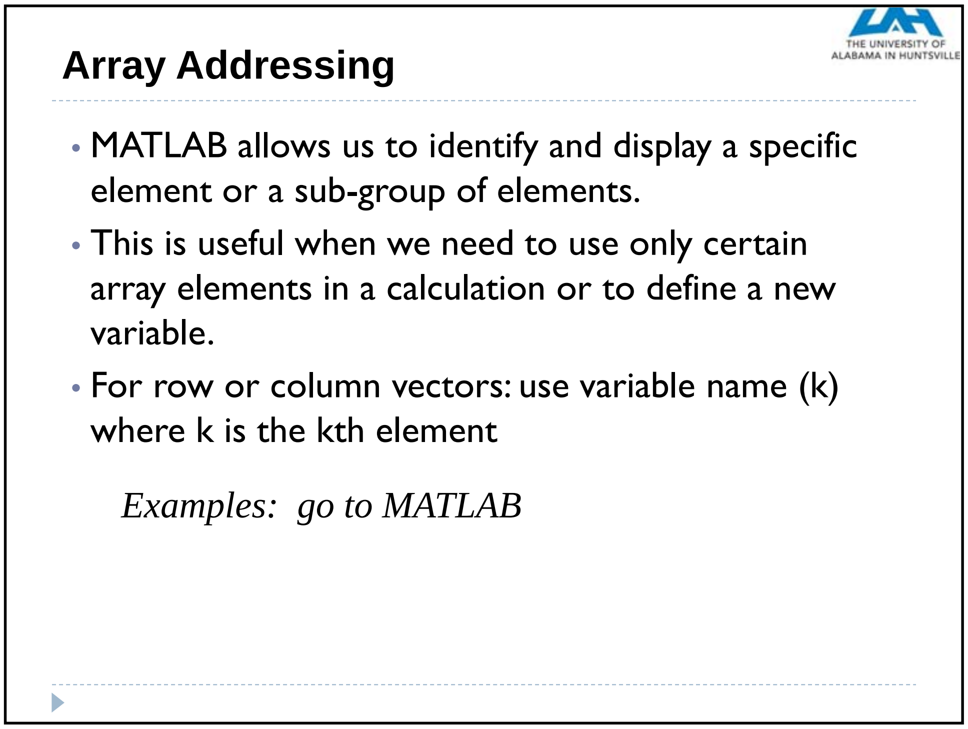
xi = initial term

xf = final term

n = number of terms linearly spaced (default is 100)

Example: time = linspace(1, 5, 10)

```
time = 1.0000  1.4444  1.8889   2.3333  2.7778  3.2222  3.6667  4.1111   4.5556    5.0000
```



Array Addressing

- MATLAB allows us to identify and display a specific element or a sub-group of elements.
- This is useful when we need to use only certain array elements in a calculation or to define a new variable.
- For row or column vectors: use variable name (k) where k is the kth element

Examples: go to MATLAB

Addressing vectors

- To call out the kth element in a row or column vector: variable_name(k)
- To call out the mth – nth elements in a row or column vector: variable_name(m:n)
- To call out the entire row or column vector: variable_name(:)

Addressing vectors

Example: Consider the row vector R and the column vector C we created earlier.

- Call out the 2nd element in vector R: R(2)
- Call out the 1st-3rd elements in vector C: C(1:3)
- Call out the 4th element in vector R and assign that value to the variable r_4: r_4 = R(4)
- Call out the 5th element in vector C and assign that value to the variable c_5: c_5 = C(5)
- Assign a new variable X that is equal to r_4/c_5: X = r_4 / c_5

Addressing matrices

- To call out the element in row k and column p of a matrix: variable_name(k,p)
- To call out the elements in column n of a matrix :
  variable_name(: , n)
- To call out the elements in row n of a matrix :
  variable_name(n , :)
- There are other ways to address matrices that are more advanced then what we need to do right now. You can find them in Chapter 2 of the book.

Example: Consider the matrix, M we created earlier.

Call out the element in row 2 and column 1 in matrix M:

M(2 , 1)

Call out the elements in column 2 of matrix M:

M(: , 2)

Call out the elements in row 3 of matrix M:

M(3 , :)

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

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

Column 1

```
MAT =   3     11      6      5
        4      7     10      2
       13      9      0      8
```

Element in row 3 and column 1

Row 3

```
>> MAT(3,1)
ans = 13
```

Assign new value to element in row 3 and column 1

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

Only this element changed

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Math functions with Arrays

- We can use arrays in computations/calculations in MATLAB.
- We will focus on using row/column vector arrays in calculations, ***mostly***.
- We will use matrices to see how to solve systems of equations in an easy way.

- Example: A capacitor has a voltage: Vc = 10e^{-5t}, where t is time in seconds

Create a row vector for time, t, which begins with 0 and ends at 1 seconds at intervals of 0.1 seconds

```
>> t = 0 :0.1 : 1
>>Vc = 10*exp(-5*t)
```

Other Built-In Functions for Arrays

- mean (A): Returns the mean value of the elements in vector A
- max (A): Returns the largest value of the elements in vector A
- min (A): Returns the smallest value of the elements in vector A
- [d,n]=max(A): Returns the largest value of the elements (d) and the position of the
- element (n) in vector A
- [d,n] = min(A): Returns the smallest value of the elements (d) and the position of the
- element (n) in vector A
- sum (A): Returns the sum of the elements in vector A
- sort (A): Arranges the elements in vector A in ascending order