MATLAB Workshop 12 - Matrices (Arrays)

Objectives: Learn about matrix properties in MATLAB, methods to create matrices, mathematical functions with matrices, element-by-element matrix operations, and matrix algebra.

MATLAB Features:

vector/matrix variable properties

Property	Comment
`var_name`	user chooses names • name should represent *meaning* of variable ex: use **`radius`** rather than **`x`** or **`s`** for radius • letters, digits (0-9), underscore (_) allowed in name • must start with a letter • case sensitive (**`Radius`** is different than **`radius`**) • can be any length, but first 31 characters must be unique
`value`	all numerical values are • double precision ~ 16 significant figures (not decimal places) • and imaginary has both real and imaginary parts **`a ± bi`** (where **`i = (`**$\sqrt{-1}$**`)`**) both **`a`** and **`b`** are double precision if **`b = 0`**, MATLAB only displays real part, **`a`**
`memory location`	MATLAB assigns - you do not need to worry about this

order of precedence rules for vector/matrix arithmetic operators

Order of Precedence	Symbol	Meaning
1	**`( )`**	group together
2	**`'`**	transpose (exchange rows and columns)
	`.^`	raise each element to indicated power
	`^`	multiply matrix by itself indicated number of times
3	**`.*`**	element-by-element multiplication
	`./`	element-by-element right division
	`.\`	element-by-element left division
	`*`	multiply two matrices
	`/`	matrix right division
	`\`	matrix left division
4	**`+`**	addition
	`-`	subtraction

colon (`:`) operator - a special vector/matrix operator

Notation	Action	Result
`a:c`	creates vector with elements **`1`** apart	**`[a a+1 a+2 ... c]`**
`a:b:c`	creates vector with elements **`b`** apart	**`[a a+b a+2b ... (≤c)]`**
`A(m,:)`	selects m^{th} row of **`A`**	
`A(:,j)`	selects j^{th} column of **`A`**	

A(m:n,:)	selects m^{th} through n^{th} rows of **A**	
A(:,j:k)	selects j^{th} through k^{th} columns of **A**	
A(m:n,j:k)	selects m^{th} through n^{th} rows of **A** and j^{th} through k^{th} columns of **A**	

matrix functions

number of rows and columns	**[rows, cols] = size(x)**

- **Working with matrices (arrays) in MATLAB**

MATLAB is short for MATrix LABoratory. It was originally conceived as a powerful engineering software package that was directed toward economically solving commonly occurring matrix algebra problems by combining commonly used matrix and vector functions with an easily used interface. MATLAB has evolved considerably since its inception and has become a general purpose engineering problem solving tool. Although MATLAB has multidimensional array capabilities, we will focus on two-dimensional arrays (matrices).

(1) Creating a two-dimensional array (matrix) in MATLAB.

(a) Enter the following at the Command Line prompt

```
» Array1 = [3, 9, -4; 4, -3, 6; -2, 5, 8 ]
Array1 =
     3     9    -4
     4    -3     6
    -2     5     8
```

A 3x3 (rows x columns) matrix or array is created.

(b) Enter the following at the Command Line prompt

```
» Array2 = [3  12; -4, -3; 12, 5 ]
Array2 =
     3    12
    -4    -3
    12     5
```

A 3x2 (rows x columns) matrix or array is created.

(c) Enter the following at the Command Line prompt

```
» Array1(1,3)
ans =
    -4
```

The value of the element in the 1st row, 3rd column is displayed

Enter the following at the Command Line prompt

```
» Array2(3,2)
ans =
     5
```

The value of the element in the 3rd row, 2nd column is displayed

Notes on arrays and array elements
- The array name, **`Array_name`**, refers collectively to all elements in the array.
- Each element has an associated indices that uniquely identifies the element.
  - Index counting starts at 1
  - **`Array_name(1,2)`** refers to the element in the 1st row, 2nd column
  - **`Array_name(i,j)`** refers to the element in the i^{th} row, j^{th} column
- When creating arrays use square braces **`[ ]`**; when accessing elements use parentheses **`( )`**.
- Rows are separated by semicolons when creating arrays
- Elements in columns are separated by spaces or commas

(2) Transposing arrays in MATLAB.

Enter the following at the Command Line prompt

```
» Array2trans = Array2'
Array2trans =
     3    -4    12
    12    -3     5
```

Array2 has been "transposed", i.e., the columns and rows were interchanged so that the first column became the first row, etc.

(3) Extracting rows or columns from an array in MATLAB.

Enter the following at the Command Line prompt

```
» row2_Array1 = Array1(2,:)
row2_Array1 =
     4    -3     6
```

The second row of **Array1** has been extracted using the colon operator.

Enter the following at the Command Line prompt

```
» col1_Array2 = Array2(:,1)
col1_Array2 =
     3
    -4
    12
```

The first column of **Array2** has been extracted using the colon operator.

Enter the following at the Command Line prompt

```
» Array1sub = Array1(1:2,2:3)
Array1sub =
     9    -4
    -3     6
```

A subset array consisting of elements from the 1st and 2nd rows, 2nd and 3rd columns of **Array1** has been created.

- The colon operator is a powerful tool for extracting parts of an array for further use.

(4) Determining the size of a matrix.

Enter the following at the Command Line prompt

```
» size(Array1)
ans =
     3     3
```

size returns the number rows and columns in an array.

Enter the following at the Command Line prompt

```
» [A2rows, A2cols] = size(Array2)
A2rows =
     3
A2cols =
     2
```

We can assign the number of rows and columns in a matrix to variables for later use.

Enter the following at the Command Line prompt

```
» size(row2_Array1)
ans =
     1     3
```

The **size** of a row vector is (1,N) for 1 row, and N elements. What do you think the **size** of a column vector is? Contrast **size** with **length** for a vector.

- The **size** function returns the number of rows and columns in a matrix.

- **Function and element-by-element operations with matrices in MATLAB**

MATLAB processes mathematical functions and element-by-element operations for matrices in the same manner as it does for vectors. That is, application of a function, e.g.

```
B = exp(A)
```

where **A** is an array will raise every element in **A** to the e^{th} power to produce the corresponding elements of array **B**. Likewise, the element-by-element operators, **.^**, **.***, **./**, and **.**, are used to process matrices element-by-element.

(5) Functions of matrices in MATLAB.

Enter the following at the Command Line prompt

```
» Amat = [1 10; 100 1000]
Amat =
           1          10
         100        1000
» log10_Amat = log10(Amat)
log10_Amat =
         0    1.0000
    2.0000    3.0000
»
» Bmat = [0 pi/6 pi/3; pi/2 2*pi/3 5*pi/6]
Bmat =
         0    0.5236    1.0472
    1.5708    2.0944    2.6180
» sin_Bmat = sin(Bmat)
sin_Bmat =
         0    0.5000    0.8660
    1.0000    0.8660    0.5000
```

- Mathematical functions operate on a matrix element-by-element.

(6) Scalar-matrix operations in MATLAB.

Enter the following at the Command Line prompt

```
» const = 5;
» Cmat = Amat + const
Cmat =
           6          15
         105        1005
```

The constant value 5 was added to each element of **Cmat**.

Enter the following at the Command Line prompt

```
» Dmat = const + Amat
Dmat =
           6          15
         105        1005
```

Scalar-matrix addition is commutative.

Enter the following at the Command Line prompt

```
» Emat = Amat - const
Emat =
    -4     5
    95   995
```

The constant value 5 was subtracted from each element of **Cmat**.

Enter the following at the Command Line prompt

```
» Fmat = const - Amat
Fmat =
     4    -5
   -95  -995
```

The new matrix was created by subtracting each element of **Amat** from **const**. The answer is different than that for **Emat**. Scalar-matrix subtraction is not commutative.

Enter the following at the Command Line prompt

```
» Gmat = Amat*const
Gmat =
           5          50
         500        5000
```

Each element of **Amat** was multiplied by **const**.

Enter the following at the Command Line prompt

```
» Hmat = const*Amat
Hmat =
           5          50
         500        5000
```

Scalar-matrix addition is commutative.

What happens with scalar matrix-division? Try it. Remember that there are two division operators, `/` and `\`.

- Scalar-matrix operations produce a matrix of the same size.
- Scalar-matrix addition and multiplication are commutative.
- Scalar-matrix subtraction and division are not.

(7) Element-by-element arithmetic with matrices in MATLAB.

Enter the following at the Command Line prompt

```
» Gmat = Amat.*log10_Amat
Gmat =
  1.0e+003 *
         0    0.0100
    0.2000    3.0000
```

In addition to taking each element of **Amat** and multiplying it by the corresponding element of **log10_Amat** to produce the corresponding element of **Cmat**, note that the elements of

Cmat are displayed as **1.0e+003 ***. This means that each number that follows should be multiplied by 10^3 in order to produce the proper element value. MATLAB automatically moves to scientific notation type display when necessary.

Contrast the preceding with

```
» Hmat = Amat*log10_Amat
Hmat =
  1.0e+003 *
    0.0200    0.0310
    2.0000    3.1000
```

Using the multiplication operator, *****, instead of the element-by-element multiplication operator, **.***, instructed MATLAB to follow the rules of matrix multiplication (outlined below) and produces an entirely different answer.

- Confusing matrix algebra operators with element-by-element operators can result in big mistakes.

- **Matrix algebra**

Matrices are characterized by their dimension. For simplicity, $[A]_{MxN}$ will denote a matrix with M rows and N columns. Likewise, $a_{i,j}$ will denote the element value in the i^{th} row, j^{th} column of $[A]_{MxN}$. The row dimension will always come first and the column dimension second. The rules for basic matrix algebra operations are as follows.

Matrix addition/subtraction. Two matrices can be added or subtracted *if and only if* they have the same dimensions. The result is a matrix with the same dimensions. Thus,

$$[C]_{3x2} = [A]_{3x2} + [B]_{3x2}$$

is possible while

$$[A]_{3x2} + [D]_{2x3}$$

is not. Addition of two matrices is done element-by-element as

$$c_{i,j} = a_{i,j} + b_{i,j}$$

for each element in the matrices. Matrix subtraction is defined as

$$d_{i,j} = a_{i,j} - b_{i,j}$$

Note that matrix addition is commutative and matrix subtraction is not commutative.

Matrix multiplication. Two matrices can be multiplied *if and only if* the number of columns in the first matrix is the same as the number of rows in the second matrix. The result is a matrix with the same number of rows as the first matrix and the same number of columns as the second matrix. Thus

$$[C]_{PxS} = [A]_{PxQ}[B]_{QxS}$$

is possible while

$$[A]_{PxQ}[B]_{PxQ}$$

is not. Multiplication of two matrices is defined as

$$c_{i,j} = \sum_{k=1}^{k=Q} (a_{i,k})(b_{k,j})$$

for each element in the resulting matrix. Note that, in general, matrix multiplication *is not* commutative.

(8) Matrix algebra in MATLAB.

Define (enter) the following matrices at the MATLAB command prompt.

$$Amat = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 2 \\ -1 & 2 & 1 \end{bmatrix} \qquad Bmat = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Cmat = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Try performing the following mathematical operations

Amat + Bmat	*Amat + Cmat*	*Bmat – Amat*	*Cmat – Bmat*
*Amat * Bmat*	*Amat * Cmat*	*Bmat * Amat*	*Cmat * Amat*

What are the results? What error messages are generated?

Exercises: Perform the following operations using array arithmetic where appropriate.

1. Create the following three matrices:

$$A = \begin{bmatrix} 5 & -3 & 2 \\ 1 & 1 & -3 \\ 2 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 4 & -1 \\ 3 & 0 & 2 \\ 5 & -12 & 6 \end{bmatrix} \qquad C = \begin{bmatrix} -2 & 3 & -1 \\ 7 & 5 & 3 \\ 6 & 4 & -8 \end{bmatrix}$$

   a. Show that matrix addition is commutative by computing A+B and B+A.

   b. Show that matrix addition is associative by computing A+(B+C) and (A+B)+C.

   c. Show that scalar-matrix multiplication is distributive by computing 5(A+C) and 5A+5C.

   d. Show that matrix multiplication is distributive by computing A*(B+C) and A*B+B*C.

2. Use the matrices from exercise 1 to answer the following

   a. Does A*B = B*A?

   b. Does (A*B)*C = A*(B*C)?

   c. Does $(A*B)^t = A^t*B^t$ (t means transpose)?

   d. Does $(A+B)^t = A^t+B^t$?

   e. Does A*B = A.*B?

Recap: You should have learned

- How to declare row and column vectors in MATLAB
- How to declare a vector with evenly spaced elements (two methods)

- How to transpose a vector
- How to extract elements from a vector
- How to declare a matrix in MATLAB
- How to transpose a matrix
- How to extract elements from a matrix
- Arithmetic operations between a scalar and a vector
- Arithmetic operations between two vectors
- Simple function operations with a vector
- Arithmetic operations between functions of vectors
- Arithmetic operations with a matrix