

MATLAB Marina: Operations on Arrays

Student Learning Objectives

After completing this module, one should:

1. Be able to perform arithmetic and logic operations and apply built in functions on MATLAB 1D arrays.

Terms

array operation, element by element operation

MATLAB Functions, Keywords, and Operators

NA

Element by Element Arithmetic Operations

MATLAB supports two types of arithmetic operations: matrix (regular) operations and element by element (dot) operations. Element by element addition (+), subtraction (-), multiplication (*), division (/), and power (^) can be performed on 1D arrays of the same length or on a 1D array and a scalar. MATLAB does not have separate element by element operators for addition and subtraction as element by element addition and subtraction are defined the same as matrix addition and subtraction.

Figures 1a and 1b show some examples of element by element arithmetic operations.

```
>> vec1 = [1, 3, 5, 9, 13];
>> vec2 = [2, 1, 2, 4, 3];
>> vec1 + 1
ans = 2 4 6 10 14
>> vec3 = vec1 + vec2
vec3 = 3 4 7 13 16
>> vec4 = vec1 - vec2
vec4 = -1 2 3 5 10
```

Figure 1a. Element by Element Addition and Subtraction

```
>> vec1 = [1, 3, 5, 9, 13];
>> vec2 = [2, 1, 2, 4, 3];
>> 2*vec1
ans = 2 6 10 18 26
>> vec3 = vec1.*vec2
vec3 = 2 3 10 36 39
>> vec4 = vec1./vec2
vec4 = 0.5 3 2.5 2.25 4.33
>> vec5 = vec1.^2
vec5 = 1 9 25 81 169
```

Figure 1b. Element by Element Multiplication, Division, and Power

Note that for addition, subtraction, and multiplication by a scalar, the dot operation is not needed.

Be careful to use the matrix (regular) operations and the element by element operations correctly. Generally, when operations are performed on corresponding elements of two arrays producing an array of the same dimensions, element by element operations should be used.

Applying Functions to Arrays

Built-in MATLAB functions will generally accept either scalars or arrays as arguments. Generally, MATLAB functions perform their operation on each element in the argument and return a result the same size as the argument. Some of the more commonly used MATLAB functions are: `cos` (cosine in radians), `sin` (sine in radians), `cosd` (cosine in degrees), `sind` (sine in degrees), `sqrt` (square root), `pow` (power), `exp` (exponential), `log` (natural log), and `log10` (log base 10). Remember that MATLAB's help can be used to determine the arguments needed and the different variations of the built-in functions.

The MATLAB functions `sum`, `mean`, `min`, and `max` are commonly used to operate on arrays, but they do not return a result the same size as the array. The `sum` and `mean` functions return the sum of the elements in the array and the mean of the elements in the array. The `min` and `max` functions return the value of the minimum and maximum element in the vector as well as the location (index) of the minimum or maximum element in the vector.

```
>> data = [8, 4, 6, 9, 10, 8, 7, 2, 5];
>> meanOfData = mean(data)
meanOfData = 6.5556
>> minOfData = min(data)
minOfData = 2
>> [maxOfData, ind] = max(data)
maxOfData = 10
ind = 5
>> sumOfData = sum(data)
sumOfData = 59
```

Figure 2. MATLAB sum, mean, min, and max Functions

Logic and Relational Operations on Arrays

MATLAB supports the element by element relational operators: less than (`<`), greater than (`>`), less than or equal to (`<=`), greater than or equal to (`>=`), equality (`==`), not equal (`~=`) and the element by element logical operators not (`~`), and (`&`), or (`|`). There are additional logic operations such as exclusive or (`xor`) supported though built in functions. Element by element logical and relational operations can be performed on all the elements of an array as long as the operation involves two arrays of the same size or an array and a scalar.

There are short circuit versions of logic and (&&) and logic or (||). The short circuit versions of the logic operators must produce a scalar result and only use the second operand if the result cannot be determined from the first operand. In most cases, the element by element logic operators should be used.

Figure 3 shows some example of logic and relational operations on 1D arrays.

```
>> data1 = [1, 3, 5, 9, 13];
>> data2 = [2, 1, 2, 4, 3];
>> d1GT_log = data1 > data2
d1GT_log = 0 1 1 1 1
>> d1LT5_log = data1 < 5
d1LT5_log = 1 1 0 0 0
>> d1_log = (data1 > 4) & (data1 < 10)
d1_log = 0 0 1 1 0
```

Figure 3. MATLAB Logic and Relational Operations

MATLAB treats the value of 0 as false and the value of 1 as true (any nonzero value is treated as true).

Array indexing can be performed using arrays of logical values (true, 1 and false , 0) as long as the logic array is the same size as the array being indexed. When indexing with logic arrays, the elements corresponding to the true locations are returned.

```
>> data = [12, 5, 11, 7, 16];
>> d_log = (data > 6) & (data < 12)
d_log = 0 0 1 1 0
>> data2 = data(d_log)
data2 = 11 7
```

Figure 4. Indexing using a Logic Array

Table 1 illustrates how the result of the comparison relates to the array of data and the element location (index).

Element location (index)	1	2	3	4	5
data	12	5	11	7	16
d_log = (data > 4) & (data < 10)	0	0	1	1	0

Table 1. Element Location, Data Values, Comparison Result

find Function

The built-in `find` function returns the linear indices corresponding to the locations where a condition evaluates to true. If the condition is an array, then `find` returns the indices of the nonzero values. In effect, the `find` function converts a logic array of 0s and 1s to an array of indices corresponding to the places where the logic array is true.

```
>> data = [12, 5, 11, 7, 16];  
>> d_ind = find((data > 6) & (data < 12))  
d_ind = 3 4  
>> data2 = data(d_ind)  
data2 = 11 7
```

Figure 5. Using find to Convert Logic Array to Indices

Notice that the results after the comparison and indexing operations in the code of Figures 4 and 5 are the same. Either a logic array or array of indices can be used to index an array to extract the elements in the array matching the condition.


Naming Variables to Convey What they Contain

Variable names should convey the type of data they contain. This is especially important when analyzing collections of data and results may be data values, logicals, or indices. In the examples here, an underscore followed by log or ind is used to denote the result is logicals or indices resulting from a comparison.

```
data = [4, -2, 8, 6, 0, 12]; % data values  
d_log = data < 0; % logicals  
d_ind = data >= 0; % indices  
dataGT0 = data(d_ind); % data values
```

Figure 6. Variable Naming Convention

Last modified Friday, September 18, 2020

 [MATLAB Marina](https://creativecommons.org/licenses/by-nc-sa/4.0/) is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.