

MATLAB Marina: Creating and Modifying Arrays

Student Learning Objectives

After completing this module, one should:

1. Be able to create and use MATLAB 1D arrays.
2. Be able to index MATLAB 1D arrays.

Terms

scalar, 1D array, vector, index, indexing (extracting, slicing), colon operator, colon notation, concatenation

MATLAB Functions, Keywords, and Operators

;, length, size, numel, zeros, ones, min, max, mean, sum, cumsum, find, end, (), []

MATLAB Arrays

A scalar is a single element. A 1D array or vector is a one-dimensional collection of data of the same data type. For example, a 1D array $v = [v_1, v_2, \dots, v_{10}]$ of ten integers could either be a row (1 by 10) or column (10 by 1) of ten integer values.

Colon Operator

The MATLAB colon operator allows one to create a range of values without using a loop structure; $K:L$ is the same as $[K, K+1, K+2, \dots, L]$ and $K:D:L$ is the same as $[K, K+D, K+2D, \dots, L]$; the default increment is one. Figure 1 shows two examples of using the colon operator to create ranges of numbers.

```
>> 1:1:8
ans = 1 2 3 4 5 6 7 8
>> 0.0:0.25:2.5
ans = 0 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00 2.25 2.50
>> 2.5:-0.5:-1.5
ans = 2.50 2.00 1.50 1.00 0.50 0 -0.50 -1.00 -1.50
```

Figure 1. Creating Ranges of Numbers using the Colon Operator

Creating 1D Arrays

There are several ways to create 1D arrays in MATLAB: entering the values directly enclosed by square brackets (row elements separated by commas or spaces and elements in columns separated by semicolons), using the colon operator, using built in MATLAB functions such as `linspace`, `zeros`, `ones`, and `rand`, and created as the result of operations on 1D arrays.

Figure 2a shows examples of creating 1D arrays by directly entering the values and using the colon operator. Figure 2b shows examples of creating 1D arrays using built in MATLAB functions. The `linspace` function takes three arguments (start value, end value, and number of points) and creates a row array with the specified number of values equally spaced from the start value to the end value.

```

>> emptyVector = []
emptyVector = []
>> vecDirect = [3, 7, -1, 2]
vecDirect = 3 7 -1 2
>> vecColon = 0:2:20
vecColon = 0 2 4 6 8 10 12 14 16 18 20

```

Figure 2a. Creating 1D Arrays Using Direct Entry and the Colon Operator

```

>> vecLinspace = linspace(0,5,10);
vecLinspace = 0 0.5556 1.1111 1.6667 2.2222 2.7778 3.3333
3.8889 4.4444 5.0000
>> rowVecZeros = zeros(1,5);
rowVecZeros = 0 0 0 0 0
>> colVecOnes = ones(7,1);
>> vecRand = rand(1,100);

```

Figure 2b. Creating 1D Arrays Using Built in Functions

Multiple 1D arrays can be concatenated to create larger 1D arrays. Concatenation joins the series of values in the provided order. The values to the right are appended to the left end of the first set of values. MATLAB's 1D array concatenation syntax is similar to creating arrays using direct entry. Row arrays are concatenated by providing the list of 1D row arrays enclosed in square brackets separated by spaces or commas. Column arrays are concatenated by providing the list of 1D column arrays enclosed in square brackets separated by semicolons.

```

>> row1 = [2, 4, -6];
>> row2 = [1, 3, 5];
>> row = [row1 , row2]
row = 2 4 -6 1 3 5
>> col1 = [2; 4; -6];
>> col2 = [1; 3; 5];
>> col = [col1 ; col2];

```

Figure 2c. 1D Array Concatenation

Indexing 1D Arrays

The individual items in a 1D array are called elements and the position in the 1D array is called the index. For example, the 1 by 6 array `vec = [13, 7, -5, 2, 63, 8]` contains the element 13 at index 1, the element 7 at index 2, and the element 8 at index 6. Individual elements of a 1D array can be indexed (accessed) using the array name and the index (position) enclosed in parentheses. Multiple elements that are successive can be indexed by providing a range of indexes. Indexing arrays is also called slicing, accessing, and extracting.

Figure 3 illustrates how to index and save the fourth element and the third through sixth elements of the 1D array.

```
>> vec = [13, 7, -5, 2, 63, 8];
>> v4 = vec(4)
v4 = 2
>> v3to6 = vec(3:1:6)
v3to6 = -5  2  63  8
```

Figure 3. Indexing 1D Arrays

Modifying and Removing Elements of Arrays

A portion of an array can be modified by specifying the range to modify and providing the appropriate number of new values; i.e. index the places in the array to be modified and assign new values to those places. Elements can be added to the beginning or end of a 1D array using concatenation. Elements can be removed from an array by specifying the range to remove and assigning the empty vector to the specified elements; i.e. index the places to be removed and assign the empty vector to those places.

```
>> vec = [13, 7, -5, 2, 63, 8];
>> vec(2) = 0
vec = 13  0  -5  2  63  8
>> vec(4:end) = zeros(1,3)
vec = 13  0  -5  0  0  0
>> vec(3) = []
vec = 13  0  0  0  0
```

Figure 4. Modifying and Removing Elements of Arrays

Useful Reserved Words and Built in Functions for Arrays

The `numel` function returns the number of elements in an array. The `length` and `size` functions provide information about the dimensions of a variable. The `length` function returns the value of the largest dimension. For a 1d array, the length is the same as the number of elements. The `size` function returns the dimensions of the variable.


The `end` reserved word, when used in an indexing expression, is equivalent to the length of the dimension it is being used to index, i.e. the last index along that dimension. Using the colon operator by itself when indexing is equivalent to using `1:1:end` along that dimension.

```
>> vec = [13, 7, -5, 2, 63, 8];
>> len = length(vec)
len = 6
>> vec4toEnd = vec(4:end)
vec4toEnd = 2  63  8
>> vecCopy = vec(:)
vecCopy = 13  7  -5  2  63  8
```

Figure 5. Examples of Using `length` function, `end` keyword, and Colon Operator

Indexing the array `vec` with the indices `4:end` extracts elements 4, 5, and 6; and indexing the array `vec` using the `:` extracts the entire array.

Last modified Friday, September 18, 2020

 [MATLAB Marina](#) is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.